

Entanglement Generation in Quantum Networks
Towards a universal and scalable quantum internet

Dahlberg, E.A.

DOI

[10.4233/uuid:f4ba6396-f4c0-4f8c-95be-2620d62e4387](https://doi.org/10.4233/uuid:f4ba6396-f4c0-4f8c-95be-2620d62e4387)

Publication date

2021

Document Version

Final published version

Citation (APA)

Dahlberg, E. A. (2021). *Entanglement Generation in Quantum Networks: Towards a universal and scalable quantum internet*. [Dissertation (TU Delft), Delft University of Technology].
<https://doi.org/10.4233/uuid:f4ba6396-f4c0-4f8c-95be-2620d62e4387>

Important note

To cite this publication, please use the final published version (if applicable).
Please check the document version above.

Copyright

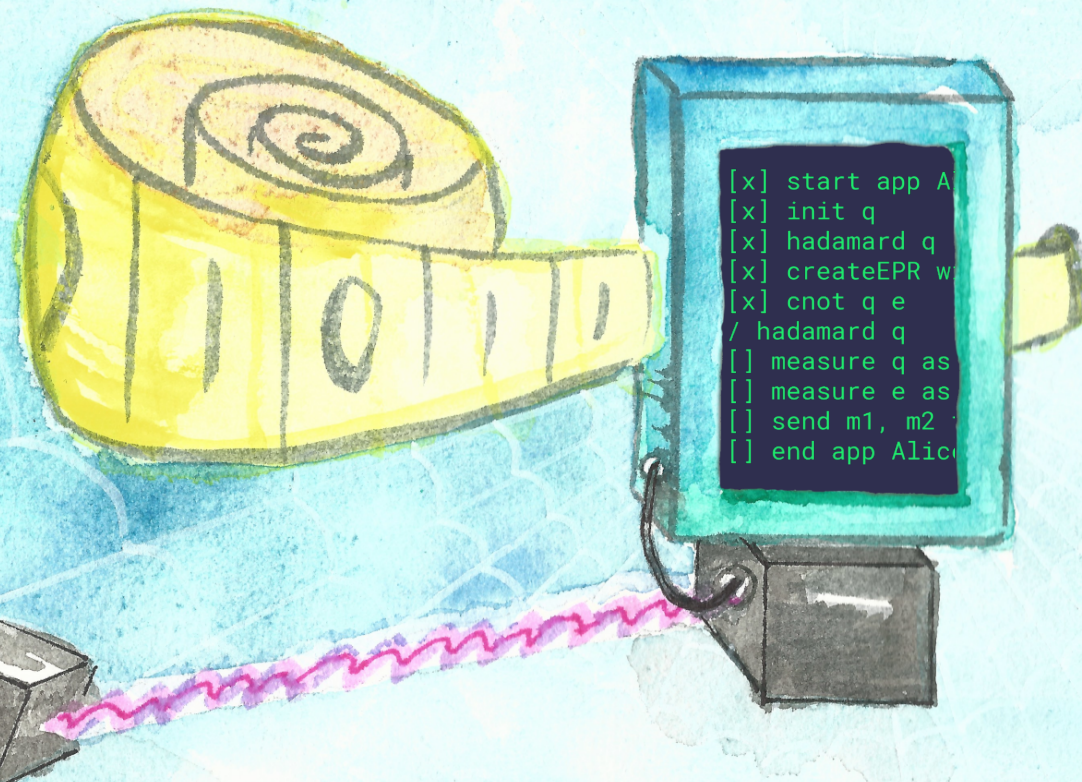
Other than for strictly personal use, it is not permitted to download, forward or distribute the text or part of it, without the consent of the author(s) and/or copyright holder(s), unless the work is under an open content license such as Creative Commons.

Takedown policy

Please contact us and provide details if you believe this document breaches copyrights.
We will remove access to the work immediately and investigate your claim.

Entanglement Generation in Quantum Networks

Towards a universal and scalable
quantum internet



```
[x] start app A
[x] init q
[x] hadamard q
[x] createEPR w
[x] cnot q e
/ hadamard q
[] measure q as
[] measure e as
[] send m1, m2
[] end app Alic
```


Entanglement Generation in Quantum Networks

Towards a universal and scalable quantum internet

Proefschrift

ter verkrijging van de graad van doctor
aan de Technische Universiteit Delft,
op gezag van de Rector Magnificus Prof.dr.ir. T.H.J.J. Hagen,
voorzitter van het College voor Promoties,
in het openbaar te verdedigen op dinsdag 12 januari 2021 om 12:30 uur

door

Axel Dahlberg

Master of science in Physics,
ETH, Zürich, Switzerland,
geboren te Onsala, Sweden.

Dit proefschrift is goedgekeurd door de
promotor: prof. dr. S. D. C. Wehner

Samenstelling promotiecommissie:

Rector Magnificus, voorzitter
Prof. dr. S. D. C. Wehner,
Technische Universiteit Delft

Onafhankelijke leden:

Prof. dr. W. Tittel, Technische Universiteit Delft
Prof. dr. D. Bruß, Universität Düsseldorf
Prof. dr. A. van Dursen,
Technische Universiteit Delft
Prof. dr. T. Northup, Universität Innsbruck
Dr. M. Walter, Universiteit van Amsterdam



Keywords: quantum networks, quantum internet, software stack, network stack, graph states, entanglement, complexity, simulation

Printed by: Gildeprint - www.gildeprint.nl

Front & Back: Anna Elisabeth Ekman Dahlberg

Copyright © 2020 by A. Dahlberg

ISBN 978-94-6384-187-0

An electronic version of this dissertation is available at
<http://repository.tudelft.nl/>.

*It is not the mountain we conquer
but ourselves.*

Edmund Hillary

Curriculum Vitæ

Axel Dahlberg

06-02-1991 Born in Onsala, Sweden.

Education

2007–2010 High school
Elof Lindälvs Gymnasium, Kungsbacka, Sweden

2011–2014 Undergraduate in Physics
Chalmers University of Technology, Göteborg, Sweden

2014–2016 Masters in Theoretical Physics
ETH Zürich, Switzerland

2016–2020 PhD in Quantum Information
Delft University of Technology, Delft, The Netherlands
Thesis: Entanglement Generation in Quantum Networks
Promotor: Prof. dr. S. D. C. Wehner

List of Articles

14. **A. Dahlberg**, B. van der Vecht, C. Delle Donne, M. Skrzypczyk, W. Kozłowski, S. Wehner, *NetQASM - A low-level instruction set architecture for hybrid quantum-classical programs in a quantum internet*, In preparation
13. T. Coopmans, R. Knegjens, **A. Dahlberg**, D. Maier, L. Nijsten, J. Oliveira, M. Papendrecht, J. Rabbie, F. Rozpędek, M. Skrzypczyk, L. Wubben, W. de Jong, D. Podareanu, A. Torres Knoop, D. Elkouss, S. Wehner, *NetSquid, a discrete event simulation platform for quantum networks*, [arXiv preprint arXiv:2010.12535](#)
12. **A. Dahlberg**, J. Helsen, S. Wehner, *Transforming graph states to Bell-pairs is NP-complete*, *Quantum* **4**, 348 (2020).
11. **A. Dahlberg** *QuAlg - A symbolic algebra package for quantum information*, [arXiv preprint arXiv:2008.06467](#)
10. W. Kozłowski, **A. Dahlberg**, S. Wehner, *Designing a Quantum Network Protocol*, [arXiv preprint arXiv:2010.02575](#)
9. **A. Dahlberg**, J. Helsen, S. Wehner, *How to transform graph states using single-qubit operations: computational complexity and algorithms*, *Quantum Sci. Technol.* **5**, 045016 (2020).
8. Y. Lee, E. Bersin, **A. Dahlberg**, S. Wehner, D. Englund, *A Quantum Router Architecture for High-Fidelity Entanglement Flows in Multi-User Quantum Networks*, [arXiv preprint arXiv:2005.01852](#) (2020).
7. **A. Dahlberg**, J. Helsen, S. Wehner, *Counting single-qubit Clifford equivalent graph states is #P-Complete*, *Journal of Mathematical Physics* **61**, 022202 (2020).
6. J. C. Adcock, S. Morley-Short, **A. Dahlberg**, J. W. Silverstone, *Mapping graph state orbits under local complementation*, *Quantum* **4**, 305 (2020).
5. **A. Dahlberg**, M. Skrzypczyk, T. Coopmans, L. Wubben, F. Rozpędek, M. Pompili, A. Stolk, P. Pawełczak, R. Knegjens, J. de Oliveira Filho, R. Hanson, S. Wehner, *A link layer protocol for quantum networks*, *SIGCOMM '19: Proceedings of the ACM Special Interest Group on Data Communication*, 159-173 (2019).
4. K. Chakraborty, F. Rozpędek, **A. Dahlberg**, S. Wehner, *Distributed Routing in a Quantum Internet*, [arXiv preprint arXiv:1907.11630](#) (2019).
3. **A. Dahlberg**, J. Helsen, S. Wehner, *The complexity of the vertex-minor problem*, [arXiv preprint arXiv:1906.05689](#) (2019).
2. **A. Dahlberg**, S. Wehner, *SimulaQron – a simulator for developing quantum internet software*, *Quantum Science and Technology* **4**, 015001 (2018).

1. **A. Dahlberg**, S. Wehner, *Transforming graph states using single-qubit operations*, *Phil. Trans. R. Soc. A.* **376** 20170325.

Summary

Quantum mechanics shows that if one is able to generate and manipulate *entanglement* over a distance, one is able to perform certain tasks which are impossible using only classical communication. Classical communication refers to what is used in the Internet of today. A *quantum internet* would therefore bring new capabilities to our highly connected world. These capabilities both involve (1) the ability to perform tasks which are provably impossible in the current Internet, such as unconditionally secure communication, and (2) the ability to perform certain tasks much more efficiently, such as distributed (quantum) computing or extending the baseline of telescopes.

To be able to build a quantum internet, two main components are needed: (i) hardware that can store, manipulate and entangle qubits and (ii) a software stack to control the hardware. The core task of both of these is to generate entanglement to be used by applications. In this thesis we focus on the latter, i.e. the development of software and protocols that enable entanglement generation using capable hardware.

To enable a certain application, one can certainly, in theory, manually specify each operation the hardware should perform, involving micro-wave pulses, lasers etc. However, in practice this is not feasible, if not to say impossible, due to the complexity of the operations needed, especially in a distributed system such as a quantum network. What is needed is a software stack, which can help with abstracting complexity away in multiple layers. This allows for someone to program a protocol in one layer without knowing all the details of the lower layers. In particular, one can abstract away the hardware details, in order to make higher-layer protocols and applications hardware-agnostic. Therefore, to be able to build a universal, efficient and scalable quantum internet, a software stack is crucial.

In chapter 2 we start discussing the networking part of a software stack. Namely, we introduce a network stack for a quantum internet, drawing parallels to the IP/TCP-suite of the classical Internet. We continue with proposing a service and interface of the lowest layer of the network stack: the link layer. The link layer is here responsible for generating entanglement between nodes in a quantum network which are directly connected by a quantum link, i.e. a fiber cable.

When developing a protocol or application it is very useful to be able to run it. Both to see if the intended ideas make sense and also to check that the implementation is actually correct. Currently we do not have quantum hardware that exposes a full-fledged API that can be used to execute applications. For this reason, it is very useful to be able to instead simulate the hardware in a way that exposes the same API as the hardware being developed. In chapter 3 we introduce SimulaQron for this exact purpose.

Any application of a quantum internet will need entanglement in one way or another. However, entanglement is generally hard to generate and is usually the bottleneck when executing an application. We would therefore like to make use of the generated entanglement in the most optimal way. To be able to do this we need to understand how entanglement can be transformed and distributed in a quantum network. We study the entanglement of a particular class of states called *graph states* in chapters 4 to 9 and how these states can be transformed in a quantum network.

Samenvatting

Kwantummechanica laat zien dat zodra de generatie en manipulatie van *verstrengeling* over afstanden mogelijk is, dat het dan mogelijk is om bepaalde taken uit te voeren die niet mogelijk zijn met enkel klassieke communicatie. Klassieke communicatie refereert naar dat wat gebruikt wordt in het Internet van vandaag de dag. Een *kwantuminternet* zou daardoor nieuwe mogelijkheden brengen aan ons nu al sterk verbonden samenleving. Deze mogelijkheden zijn (1) het vermogen om taken uit te voeren waarvan te bewijzen valt dat deze onmogelijk zijn uit te voeren met het huidige internet, zoals onvoorwaardelijk veilige communicatie, en (2) het vermogen om bepaalde taken veel efficiënter uit te voeren, zoals gedistribueerde kwantumcomputatie of het effectief vergroten van de straal van een telescoop.

Om een kwantuminternet te bouwen zijn twee hoofdcomponenten nodig: (i) *hardware* die het mogelijk maakt om qubits op te slaan, manipuleren en te verstrengelen, en (ii) een *software stack* die de hardware aanstuurt. De hoofdtaak van beide is het maken van verstrengeling, welke vervolgens gebruikt kan worden door applicaties. In deze thesis leggen we de focus op het laatste, i.e. de ontwikkeling van software en protocollen die het mogelijk maakt om verstrengeling te maken, mits de hardware toereikend daarvoor is.

Als een applicatie uitgevoerd dient te worden is het altijd mogelijk, in theorie, om handmatig de individuele operaties van de hardware te specificeren, zoals de microgolf pulsen, lasers, enzovoort. In de praktijk is dit niet haalbaar, dan wel onmogelijk, vanwege de complexiteit van de benodigde operaties, vooral in een gedistribueerde systeem zoals een kwantumnetwerk. Een software stack lost dit probleem op door het abstraheren van de complexiteit in meerdere lagen. Dit maakt het mogelijk voor iemand om een protocol te programmeren in een bepaalde laag, zonder de details te kennen van alle details van de onderliggende lagen. Het is in het bijzonder mogelijk om details van de hardware te abstraheren, waardoor protocollen op hogere lagen en applicaties agnostisch van de hardware kunnen zijn. Voor deze redenen is een software stack cruciaal voor het maken van een universele, efficiënte en schaalbaar kwantuminternet.

In hoofdstuk 2 bespreken we het netwerk gedeelte van een software stack. Namelijk, we introduceren een netwerk stack voor een kwantuminternet, waarbij we parallellen trekken tussen onze software stack en de IP/TCP-suite van het klassieke internet. Vervolgens maken we een voorstel voor de service en interface voor de laagste laag van de netwerk stack: de linklaag. De linklaag is verantwoordelijk voor het genereren van verstrengeling tussen *nodes* in een kwantumnetwerk die direct verbonden zijn met een kwantumlink, i.e. een glasvezel kabel.

Tijdens het ontwikkelen van een protocol of applicatie is het praktisch om deze uit te kunnen voeren, om te verifiëren dat zowel de bedoelde ideeën logisch zijn en dat de implementatie daadwerkelijk correct is. Momenteel is de hardware nog

niet zover dat een volledig API gebruikt kan worden om applicaties uit te voeren. Het zou daarom dus van praktische waarde zijn om een simulator te hebben die dezelfde interface heeft met de API die later gebruikt zal worden. In hoofdstuk 3 introduceren we SimulaQron voor deze reden.

Elke toepassing van een kwantuminternet zal verstrengeling nodig hebben. Verstrengeling is moeilijk om te genereren, en is de voornaamste *bottleneck* bij het uitvoeren van applicaties. Het is dus belangrijk om verstrengeling die al bestaat in een kwantumnetwerk optimaal te gebruiken. Om dit te doen, is het belangrijk om te begrijpen hoe verstrengeling getransformeerd en gedistribueerd kan worden in een kwantumnetwerk. We bestuderen de verstrengeling van een bepaalde klasse van toestanden genaamd *graaf toestanden* in hoofdstukken chapters 4 to 9 en hoe deze toestanden getransformeerd kunnen worden in een kwantumnetwerk.

Acknowledgements

This thesis would not have been possible without some incredible people that have been in my life during these past four years. I would like to use this section to thank these people.

First of all, **Stephanie**, I am so grateful to have had you as a supervisor. I really admire your reasoning, focus and knowledge. You have pushed me to accomplish things that I would not have imagined that I could be part of. Thank you for all the fun, interesting and fruitful discussions and conversations.

I would like to thank all the people at QuTech who have made the time there very enjoyable and rewarding. In particular, the blueprint and NetSquid team: **Tim, Guus, David, Julian, Francisco, Hana, Rob, Julio, Loek, Leon, Martijn, Ariana, Walter**, the QNodeOS team, **Matthew, Carlo, Wojciech, Bart, Ingmar, Önder, Przemyslaw**, and furthermore, **Jonas, Kenneth, Bas, Filip, Victoria, Kaushik, Sebastian, Glaucia, Mark, Constantijn, Liam, Arian, Matteo**. I wish you all the best and hope to stay in touch to hear about all the amazing things you are developing.

To write a thesis, you have to be able to sometimes let go and to not think about writing a thesis. I do this by going climbing. I would like to thank everyone at Delftsbleau that have made my spare time in the gym and in the bar simply fantastic. In particular, **Thomas, Tim, Casper, Tirza, Derek, Jelle, Tije, Joris, Aiko, Tabitha, Benjamin, Gert, Isabel, Kwinten, Mariette, Martijn, Robert, Ron, Yvette, Valentijn, Sherin**. Keep crushing those crimps! Furthermore, for longer climbing trips to Yosemite, France, Italy, one can not have a better climbing parter than **Lukas**.

A year into the PhD, me and **Niklas** drove for one month around Europe on our motorcycles. This is one of the best experiences I have had and was a great way to reset the mind after an intense first year. Thank you for this trip and your friendship which I value greatly! I also want to thank **Axel, Erik, Ludvig, Rebecka, Tobias, Niklas, Alexander** for being just amazing friends.

Doing a PhD, and perhaps in particular abroad, is an incredible experience. It can also be demanding from time to time and it is easy to loose focus on other things. Whenever I go home to Sweden and visit my family I feel comfort and can find myself again if I am lost. I want to thank my mother **Kristina** for showing me the beauty of mathematics, my dad **Kjell** for showing me the beauty of nature and my two older siblings **Anders** and **Anna** for showing me the how to grow and how to enjoy life.

Last, but far from least, **Maria**, my beautiful, talented, cheerful, caring, loving and wonderful fiancée. You motivate me when I am down, you laugh with me when I am happy, you hug me when I am sad, you belay me when I am climbing. I am so grateful to have you in my life and cannot wait to see where it takes us.

Contents

| | |
|---|-------------|
| Curriculum Vitæ | v |
| List of Articles | vii |
| Summary | ix |
| Samenvatting | xi |
| Acknowledgements | xiii |
| 1 Introduction | 1 |
| 1.1 Software stack for a quantum internet | 2 |
| 1.2 Transforming graph states | 4 |
| 1.3 Other work | 5 |
| 1.4 Quantum prelude | 6 |
| 1.4.1 Qubits and states | 6 |
| 1.4.2 Entangled states | 7 |
| 1.4.3 Remote state preparation | 8 |
| 1.4.4 Fidelity and QBER | 8 |
| 1.4.5 Decoherence. | 10 |
| References. | 12 |
| 2 A Link Layer Protocol for Quantum Networks | 13 |
| 2.1 Introduction | 15 |
| 2.2 Related Work | 17 |
| 2.3 Design Considerations for Quantum Network Architectures | 18 |
| 2.3.1 Qubits and Entanglement. | 18 |
| 2.3.2 Quantum Network Devices | 19 |
| 2.3.3 Use Cases | 21 |
| 2.3.4 Network Stack | 22 |
| 2.4 Design Considerations for quantum link layer | 23 |
| 2.4.1 Desired Service | 23 |
| 2.4.2 Performance Metrics | 26 |
| 2.4.3 Error Detection | 27 |
| 2.4.4 Physical Entanglement Generation | 28 |
| 2.4.5 Hardware Considerations | 29 |
| 2.5 Protocols | 30 |
| 2.5.1 Physical Layer MHP | 30 |
| 2.5.2 Link Layer QEGP | 32 |

| | | |
|----------|--|-----------|
| 2.6 | Evaluation | 35 |
| 2.6.1 | Robustness | 37 |
| 2.6.2 | Performance Metrics | 38 |
| 2.6.3 | Scheduling. | 39 |
| 2.7 | Conclusion | 42 |
| | References. | 42 |
| 3 | SimulaQron - A simulator for developing quantum internet software | 51 |
| 3.1 | Introduction | 53 |
| 3.1.1 | What SimulaQron does | 53 |
| 3.1.2 | What SimulaQron does not | 55 |
| 3.1.3 | Related Work | 57 |
| 3.1.4 | Overview | 58 |
| 3.2 | Backend | 58 |
| 3.2.1 | Challenges. | 59 |
| 3.2.2 | Design overview. | 60 |
| 3.2.3 | Virtual Qubits. | 61 |
| 3.2.4 | Register merges. | 62 |
| 3.3 | CQC. | 63 |
| 3.3.1 | Message types. | 64 |
| 3.3.2 | Possible commands | 65 |
| 3.4 | Examples | 66 |
| 3.4.1 | Sending BB84 States | 67 |
| 3.4.2 | Teleporting a qubit | 69 |
| 3.4.3 | Performance | 71 |
| 3.5 | Conclusion | 73 |
| | References. | 74 |
| 4 | Graph states and single-qubit operations | 77 |
| 4.1 | Introduction | 78 |
| 4.1.1 | Previous work | 79 |
| 4.2 | Notation and definitions. | 80 |
| 4.3 | Stabilizer states. | 83 |
| 4.4 | Graph states | 83 |
| 4.4.1 | Local Clifford operations | 85 |
| 4.4.2 | Local Pauli measurements | 88 |
| 4.5 | Vertex-minors | 90 |
| 4.6 | Rank-width | 93 |
| 4.7 | Circle graphs | 94 |
| 4.7.1 | Double occurrence words | 95 |
| 4.7.2 | Eulerian tours on 4-regular multi-graphs | 96 |
| 4.7.3 | Local complementations on circle graphs | 97 |
| 4.7.4 | Vertex-deletion on circle graphs | 98 |
| 4.7.5 | Vertex-minors of circle graphs | 98 |
| 4.7.6 | Semi-Ordered Eulerian tours. | 100 |

| | | |
|----------|---|------------|
| 4.8 | Leaves, twins and axils | 102 |
| 4.8.1 | Distance-hereditary graphs | 105 |
| | References | 109 |
| 5 | Transforming graph states using single-qubit operations | 113 |
| 5.1 | Introduction | 115 |
| 5.1.1 | Results | 115 |
| 5.1.2 | Overview | 117 |
| 5.2 | A non-efficient but general algorithm | 117 |
| 5.3 | Constant time transformation | 119 |
| 5.4 | Efficient algorithm based on theorem by Courcelle | 120 |
| 5.4.1 | Monadic second-order logic | 122 |
| 5.4.2 | MS problems and complexity | 123 |
| 5.4.3 | Vertex-minor as C_2 MS formula | 125 |
| 5.4.4 | Finding the sequence of operations | 129 |
| 5.5 | Discussion | 130 |
| | References | 131 |
| 6 | How to transform graph states using single-qubit operations: computational complexity and algorithms | 133 |
| 6.1 | Introduction | 135 |
| 6.1.1 | Results and proof techniques | 135 |
| 6.1.2 | Overview | 138 |
| 6.2 | Complexity | 138 |
| 6.2.1 | VertexMinor is in NP | 138 |
| 6.2.2 | VertexMinor is NP-Complete | 139 |
| 6.3 | Algorithms | 160 |
| 6.3.1 | Star graph as vertex-minor of a distance-hereditary graph 160 | |
| 6.3.2 | Fixed-parameter tractable algorithm for unbounded rank- width | 183 |
| 6.4 | Connected vertex-minor on three vertices or less | 192 |
| 6.5 | Conclusion | 194 |
| | References | 195 |
| 7 | The complexity of the vertex-minor problem | 197 |
| 7.1 | Introduction | 198 |
| 7.2 | Preliminaries | 199 |
| 7.2.1 | Vertex-minors | 199 |
| 7.2.2 | Semi-Ordered Eulerian tours | 199 |
| 7.3 | NP-completeness of Iso-VertexMinor | 199 |
| 7.3.1 | ISO-SOET is NP-hard | 199 |
| 7.3.2 | ISO-VERTEXMINOR is NP-Hard | 204 |
| 7.3.3 | ISO-VERTEXMINOR is in NP | 204 |
| | References | 205 |

| | | |
|-----------|--|------------|
| 8 | Transforming graph states to Bell-pairs is NP-Complete | 207 |
| 8.1 | Introduction | 208 |
| 8.2 | Bell vertex-minors | 209 |
| 8.3 | The Edge-disjoint path problem | 210 |
| 8.4 | BellVM is NP-Complete | 212 |
| 8.5 | Conclusion | 215 |
| | References | 216 |
| 9 | Counting single-qubit Clifford equivalent graph states is #P-Complete | 219 |
| 9.1 | Introduction | 220 |
| 9.2 | Graph states | 221 |
| 9.3 | Isotropic systems | 222 |
| 9.3.1 | Finite fields and Pauli groups | 222 |
| 9.3.2 | Isotropic systems | 223 |
| 9.3.3 | Complete and Eulerian vectors | 223 |
| 9.3.4 | Fundamental graphs | 224 |
| 9.3.5 | Graphic systems | 225 |
| 9.3.6 | Eulerian decompositions | 226 |
| 9.3.7 | Number of locally equivalent graphs | 227 |
| 9.4 | Complexity | 228 |
| 9.5 | Counting the number of locally equivalent graphs is #P-Complete | 229 |
| 9.5.1 | Reducing # of Eulerian tours to # of local equivalent graphs | 229 |
| 9.6 | Conclusion | 232 |
| | References | 232 |
| 10 | Conclusion | 235 |
| 10.1 | Summary of results | 236 |
| 10.2 | Future work | 236 |
| | References | 238 |
| A | A Link Layer Protocol for Quantum Networks | 239 |
| A.1 | Testing | 240 |
| A.2 | Simulation and modeling | 242 |
| A.2.1 | Validation of simulation | 243 |
| A.2.2 | Simulation data | 243 |
| A.3 | Under the hood | 258 |
| A.3.1 | The simulated network | 258 |
| A.3.2 | Qubits on the NV Platform | 261 |
| A.3.3 | Gates and their noise | 262 |
| A.3.4 | Physical Entanglement Generation and Noise | 264 |
| A.3.5 | Heralding station | 267 |
| A.3.6 | Classical communication | 276 |

| | | |
|----------|---|------------|
| A.4 | Protocols | 278 |
| A.4.1 | Distributed Queue Protocol | 278 |
| A.4.2 | Midpoint Heralding Protocol | 281 |
| A.4.3 | Entanglement Generation Protocol | 286 |
| | References | 298 |
| B | Transforming graph states using single-qubit operations | 301 |
| B.1 | Measuring without disconnections | 302 |
| B.2 | Corrections from sequence of Pauli Z measurement | 306 |
| B.3 | Vertex-minor formula | 308 |
| B.4 | Local complementations from Eulerian vector | 309 |
| | References | 310 |
| C | How to transform graph states using single-qubit operations: computational complexity and algorithms | 311 |
| C.1 | Circle graphs induced by Eulerian tours on triangular-expanded graphs are not distance-hereditary | 312 |
| | References | 320 |
| D | Transforming graph states to Bell-pairs is NP-Complete | 321 |
| D.1 | The 4-regular EDPDT problem is NP-Complete | 322 |
| | References | 328 |

1

Introduction

Axel Dahlberg

This thesis is about quantum networks, and in particular how to enable the executing of arbitrary applications in a scalable design, forming a universal quantum internet. In a large-scale quantum internet, all applications require the generation of *entanglement*. Entanglement is a phenomena in quantum mechanics where a system of two or more particles are in a state which cannot be described by considering each particle individually. Rather, to completely specify the system, one must describe the combined state of the particles. The non-quantum reader can find a gentle introduction about qubits and entanglement in section 1.4.

Most applications in a quantum internet require the generation of entanglement explicitly. Some *prepare-and-measure* applications simply need to be able to prepare, transmit and measure qubits [1]. However, long-range qubit-transmission can only be done by consuming entanglement through the use of *quantum teleportation* [2], at least with any foreseeable hardware. For this reason, the core functionality of a quantum internet is to generate entanglement between qubits held by remote nodes. These entangled qubits can then be used to transmit other qubit states by the use of teleportation [2]. This is in contrast to the classical Internet where the core functionality is to send classical data between remote nodes.

The main question we therefore try to solve in this thesis is how to **efficiently** generate entanglement in a quantum internet, using a **robust** and **scalable** design. We approach this question from two angles, which form the two main research directions of this thesis:

Software stack for a quantum internet: Here we take a computer-science approach to the question of entanglement generation and focus more on the **robustness** and to find a **scalable** design of a software stack.

Transforming graph states: Here on the other hand we take a mathematical-physics approach to study different forms of entanglement and how these can be transformed in a quantum network in order to distribute these to applications in an **efficient** manner.

In the next two sections we detail these research directions further. In section 1.3 we additionally mention some research done during this PhD which is not part of this thesis. Finally, we then provide a condensed quantum introduction, section 1.4, for anyone not familiar with the core concepts needed throughout the thesis.

1.1. Software stack for a quantum internet

In 1969, the first computers were connected in the ARPANET, forming the first wide-area packet-switching network [3]. This network is the foundation to the network we all use today. The key point to be able to expand this network in a scalable way to what we see today, was the development of the TCP/IP-suite [4]. The protocols in this suite are used by most people everyday, when sending an email, login to a bank or while streaming videos. The TCP/IP-suite is a network stack of multiple layers, each with its own responsibility which together allow the transmission of messages across the globe. The layered structure allow a protocol in one layer to

| Application | |
|------------------|---------------------------------|
| Transport | Qubit transmission |
| Network | Long distance entanglement |
| Link | Robust entanglement generation |
| Physical | Attempt entanglement generation |

Figure 1.1: Functional allocation in a quantum network stack. Entanglement forms an inherent connection already at the physical layer, which contrasts with classical networking where shared state is typically only established at much higher layers.

not have to know all the details of the lower layers, only what service and interface it provides.

To have a scalable design for a quantum internet it is crucial that a network stack tailored for quantum networks is developed. There are many similarities between a classical network and a quantum network which can be used when it comes to naming, routing, scheduling and more. However, the different tasks of sending messages and establishing entanglement clearly brings some different demands to the network stacks.

Possibly the biggest difference between classical networks and quantum networks, is where the notion of a *connection* happens. When a connection is established between two nodes in a network, both nodes are aware of the connection by holding some state indicating whether the connection is up or not. In the classical Internet this happens only in the transmission layer, which is the second to highest layer in the network stack. However, in a quantum network this happens in some way already at the lower layers when entanglement is established. Since entanglement requires the cooperation between the nodes part of the entanglement to keep it alive, it is in some sense already a connection. This puts further requirements on the protocols in the stack handling the entanglement. Concretely, more state needs to be held at the lower layers of the stack. In contrast to the lower layers of the classical network stack, which can forget about a message directly after it being sent, the same is not true in a quantum internet where the established entanglement needs to be kept alive.

The first half of the thesis, chapters 2 to 3, focuses on research related to a software stack for a quantum internet. Chapter 2 proposes a functional allocation of a quantum network stack and introduces a protocol providing the service of the link layer. This protocol is the world's first link layer protocol for a quantum network. As discussed in the related works section of the chapter, initial proposals have previously been outlined, however no explicit protocols have been worked out and benchmarked before our work.

In chapter 3 we introduce SimulaQron, which is a simulator to enable software-development of quantum Internet applications in the absence of available quan-

tum hardware. Similar simulators for quantum computing already exists, see section 3.1.3, however, SimulaQron was the first simulator for quantum networks that exposes an API which is intended to be used on actual hardware.

1.2. Transforming graph states

As mentioned, the core functionality of a quantum network is to establish entanglement between remote nodes. The process of establishing entanglement is difficult and, with current hardware, usually the bottleneck of any application. We would therefore like to make the most optimal use of the entanglement established in the network and not waste any of it. To do so, we need to be able to understand what forms entanglement can take and how these can be transformed into each other.

Graph states is a certain subclass of all quantum states. This subclass is in fact much smaller¹ that the class of all quantum states. One can show that for quantum computers, the class of graph states, together with operations that preserve the class, provide no computational advantage over classical computers. For quantum networks on the other hand, the class of graph states do provide an advantage over classical networks. This is because these states contain various form of entanglement which can be exploited. In fact, most, if not all, applications for a quantum internet can be performed efficiently by preparing graph states and performing local operations on these.

The second half of the thesis, chapters 4 to 9, focus on research related to transforming graph states in a quantum network. Graph states and its superclass *stabilizer states* have previously been studied extensively, for example in the context of quantum error-correcting codes [5]. Van den Nest et al. showed in [6] that the action of certain single-qubit operations on graph states can be completely characterized by a set of operations on simple graphs. Interestingly, these exact graph operations have already been studied in graph theory by for example Bouchet [7].

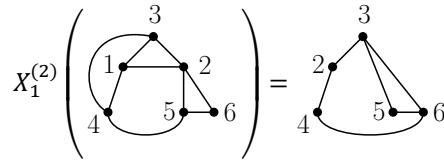


Figure 1.2: Example of measuring a graph state at qubit 1 in the X -basis.

We make use of this relation between graph states and graph theory to answer question related to transforming graph states in a quantum network. In particular we consider the problem of deciding if a graph state can be reached from another by using only single-qubit Clifford operations, single-qubit Pauli measurement and classical communication (LC + LPM + CC). We show that this problem is in general NP-Complete in chapter 6 but also provide efficient algorithms in certain restricted cases in chapters 5 and 6. It turns out that this work on graph states also answers

¹Graph states are countable many as opposed to all quantum states

a open questions in graph theory, see chapter 7. Furthermore, in chapter 8 we consider the complexity of a similar problem, namely that of transforming graph states to bipartite entangled states under LC + LPM + CC, which are the most common states in a quantum network. Finally, in chapter 9 we consider the complexity of counting graph states equivalent under single-qubit Cliffords, which can be related to the number of equivalent quantum error-correction codes of a certain type.

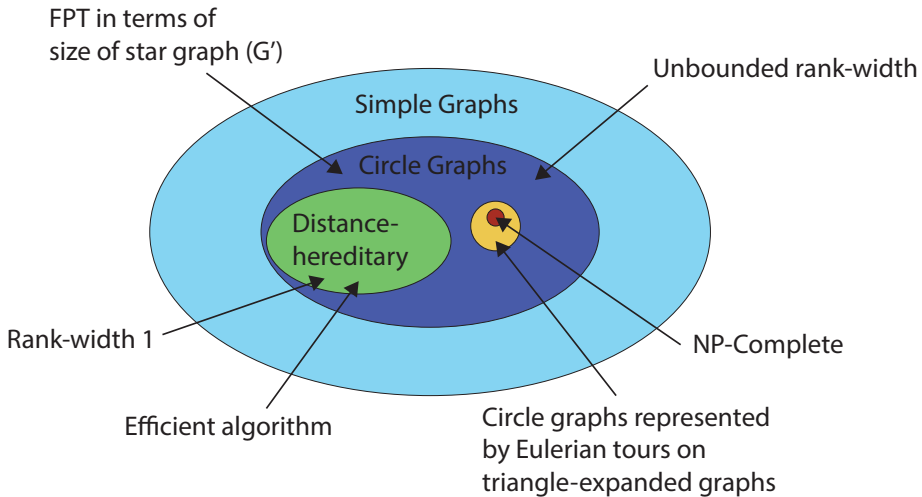


Figure 1.3: An overview of the graph classes discussed in the second half of this thesis and what the computational complexities of solving the problem of deciding if a graph state in one of these classes can be transformed into a GHZ-state using LC + LPM + CC. The sizes of the sets in the figure are not exact, however their intersections and non-intersections are.

1.3. Other work

In this section we list research projects conducted during this PhD but are not included in this thesis:

NetQASM: Additional to networking and entanglement generation, a software stack also need to be able to perform local gates, manage qubits, do scheduling etc. All these things need to be accessible to someone implementing an application for a quantum internet. The capabilities of the hardware need to be utilized in an efficient way, while at the same time not expose too much details and complexity of the underlying hardware to the user. What information is exposed is defined by the API or instruction set of the underlying architecture. In [8] we introduce such an instruction set for quantum networks called NetQASM, which replaces CQC as defined in section 3.3.

QuAlg: QuAlg is an open-source symbolic algebra package for quantum information. In [9] we introduce QuAlg. The source code and documentation can be found at <https://github.com/Acks1D/QuAlg>.

1.4. Quantum prelude

This section provides a very short introduction to quantum information and in particular quantum states, entanglement, fidelity and decoherence. For a deeper introduction to quantum information, see for example [10]. The content of this introduction is intended for someone without background in quantum information and the explanations are of a more informal nature.

1.4.1. Qubits and states

A quantum bit (*qubit*) is a two-level system, where the two levels are usually denoted $|0\rangle$ and $|1\rangle$ respectively ("ket"-notation) and called the *basis states* of the qubit. These levels can for example be two energy levels of an electron spin or - when considering transmitting qubits - vertical and horizontal polarization of a photon, presence or absence of a photon, or a time-bin of early and late. Compared to a "classical" bit $|0\rangle$ or $|1\rangle$, a qubit can be in *superpositions* thereof. Mathematically, a state $|\phi\rangle$ of a qubit is written as

$$|\phi\rangle = \alpha |0\rangle + \beta |1\rangle \quad (1.1)$$

where α and β are arbitrary complex numbers with the constraint that $|\alpha|^2 + |\beta|^2 = 1$, and

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}. \quad (1.2)$$

Note that $|0\rangle$ and $|1\rangle$ form a basis for \mathbb{C}^2 . Some common states are

$$|X, 0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \quad |X, 1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \quad (1.3)$$

$$|Y, 0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + i|1\rangle) \quad |Y, 1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - i|1\rangle) \quad (1.4)$$

$$|Z, 0\rangle = |0\rangle \quad |Z, 1\rangle = |1\rangle \quad (1.5)$$

corresponding to a '0' or '1' in the three different bases labeled X , Y , and Z . The label Z also refers to the *standard basis*. We also use $\langle\phi| = (|\phi\rangle^*)^T$ to denote the conjugate transpose of $|\phi\rangle$.

Measuring a qubit in the standard (Z) basis ($|0\rangle$, $|1\rangle$), gives measurement outcomes '0' (i.e. $|0\rangle$) or '1' (i.e. $|1\rangle$). Measuring a qubit which is in the state $|\phi\rangle$ as in equation 1.1 in the *standard basis*, yields the outcomes 0 or 1 with the following probabilities

$$P[\text{"measuring 0"}|Z\text{-basis}] = |\alpha|^2, \quad P[\text{"measuring 1"}|Z\text{-basis}] = |\beta|^2. \quad (1.6)$$

which is why $|\phi\rangle$ needs to be normalized. Measuring a qubit in the standard basis collapses it to $|0\rangle$ or $|1\rangle$. Measuring a qubit in the X - or Y -basis yields outcomes with probabilities

$$P[\text{"measuring 0"}|X/Y\text{-basis}] = |\langle X/Y, 0|\psi\rangle|^2 \quad (1.7)$$

$$P[\text{"measuring 1"}|X/Y\text{-basis}] = |\langle X/Y, 1|\psi\rangle|^2 \quad (1.8)$$

where $\langle \cdot | \cdot \rangle$ is the inner product.

Three useful single-qubit gates are the bit flip $X|x\rangle = |x+1 \pmod{2}\rangle$, phase flip $Z|x\rangle = (-1)^x|x\rangle$ and $Y|x\rangle = (-1)^x i|x+1 \pmod{2}\rangle$, given as

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}. \quad (1.9)$$

These, so called Pauli gates, are special cases of more general single-qubit operations, defined as

$$R_X(\theta) = \exp\left(-\frac{\theta}{2}iX\right) = \begin{pmatrix} \cos\left(\frac{\theta}{2}\right) & -i\sin\left(\frac{\theta}{2}\right) \\ -i\sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) \end{pmatrix} \quad (1.10)$$

$$R_Y(\theta) = \exp\left(-\frac{\theta}{2}iY\right) = \begin{pmatrix} \exp\left(-i\frac{\theta}{2}\right) & 0 \\ 0 & \exp\left(i\frac{\theta}{2}\right) \end{pmatrix} \quad (1.11)$$

$$R_Z(\theta) = \exp\left(-\frac{\theta}{2}iZ\right) = \begin{pmatrix} \cos\left(\frac{\theta}{2}\right) & -\sin\left(\frac{\theta}{2}\right) \\ -\sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) \end{pmatrix} \quad (1.12)$$

which performs basis rotations with angle θ radians around the X -, Y - and Z -axis respectively.

1.4.2. Entangled states

If qubit A is in a state $|\phi_1\rangle$ and qubit B is in the state $|\phi_2\rangle$, then their joint state (at possibly remote nodes) is given by the *tensor product* of the individual states $|\phi_1\rangle_A$ and $|\phi_2\rangle_B$, i.e. as

$$|\text{"joint state"}\rangle = |\phi_1\rangle_A \otimes |\phi_2\rangle_B. \quad (1.13)$$

Importantly, for the discussion here is that not all joint states can be factorized into single qubit states $|\phi_1\rangle_A$ and $|\phi_2\rangle_B$ in this way. These are called *entangled states*. For example, consider the state

$$|\Phi^+\rangle = \frac{1}{\sqrt{2}}(|0\rangle_A \otimes |0\rangle_B + |1\rangle_A \otimes |1\rangle_B), \quad (1.14)$$

which is a superposition of (1) both qubits being in the state $|0\rangle$ and (2) both qubits being in the state $|1\rangle$. This is an entangled state, i.e., it cannot be factorized into two individual states, giving rise to genuinely quantum correlations between A and B that have no classical analogue. The state $|\Phi^+\rangle$ is one of the so called *Bell states*. These are entangled states, where the other three are given as

$$|\Phi^-\rangle = \frac{1}{\sqrt{2}}(|0\rangle_A \otimes |0\rangle_B - |1\rangle_A \otimes |1\rangle_B), \quad (1.15)$$

$$|\Psi^+\rangle = \frac{1}{\sqrt{2}}(|0\rangle_A \otimes |1\rangle_B + |1\rangle_A \otimes |0\rangle_B), \quad (1.16)$$

$$|\Psi^-\rangle = \frac{1}{\sqrt{2}}(|0\rangle_A \otimes |1\rangle_B - |1\rangle_A \otimes |0\rangle_B). \quad (1.17)$$

Measurement outcomes of measuring the two qubits in any of the Bell-states in the bases X , Z and Y are either perfectly correlated or perfectly anti-correlated. For example, for $|\Phi^+\rangle$ the measurement outcomes are perfectly correlated in the X and Z bases but perfectly anti-correlated in the Y basis. On the other hand, for $|\Psi^-\rangle$ the measurement outcomes are perfectly anti-correlated in all three bases.

Relevant to understand the generation of bipartite entanglement is that all the Bell-states can be transformed to one another by only applying *local* quantum gates to one of the qubits. Applying the gates from equation (1.9) on qubit A (at node A only) allows one to transform:

$$|\Phi^-\rangle = Z^A |\Phi^+\rangle, \quad |\Psi^+\rangle = X^A |\Phi^+\rangle, \quad |\Psi^-\rangle = Z^A X^A |\Phi^+\rangle, \quad (1.18)$$

where we added the index A to emphasize the gates are applied to qubit A . We could also apply such gates to qubit B to have the same effect.

In the heralded entanglement generation, see chapter 2, we can obtain either failure, or else success. In the case of success, an additional bit indicates whether we produced the state $|\Psi^+\rangle$ or $|\Psi^-\rangle$. From equation (1.18), these two states can be transformed between each other by simply applying a Z -gate to one of the qubits.

Additionally to bipartite entanglement as above, multiple qubits can also be entangled in various forms. In chapters 4 to 9 we consider one class of multipartite entangled states called *graph states*.

1.4.3. Remote state preparation

Entanglement can be used to teleport qubits between remote nodes [2]. Another basic application of entanglement, which showcase its use in a quantum network, is that of *remote state preparation*.

A node A sharing an entangled state (for example $|\Psi^-\rangle$) with another node B can prepare a qubit state at B by the use of remote state preparation [11]. A chooses an arbitrary basis $\{|b_0\rangle, |b_1\rangle\}$ and the state prepared at B will either be $|b_0\rangle$ or $|b_1\rangle$ with equal probability. B will learn whether the first or second state was produced but not in which basis. To understand how this works, lets for simplicity assume that A and B share the state $|\Psi^-\rangle$. What is special about this state is that it is invariant under basis change, i.e. for an arbitrary basis

$$\{|b_0\rangle = \alpha |0\rangle + \beta |1\rangle, |b_1\rangle = \beta^* |0\rangle - \alpha^* |1\rangle\} \quad (1.19)$$

the state can be written

$$|\Psi^-\rangle = \frac{1}{\sqrt{2}}(|b_0\rangle_A \otimes |b_1\rangle_B - |b_1\rangle_A \otimes |b_0\rangle_B). \quad (1.20)$$

One can then see that if A measures in the basis $\{|b_0\rangle, |b_1\rangle\}$ and receives outcome 0 (1), the state at B will be $|b_1\rangle$ ($|b_0\rangle$).

1.4.4. Fidelity and QBER

In any real implementation of a quantum network, the generated entangled states will always differ from the perfect Bell states above due to noise in the system.

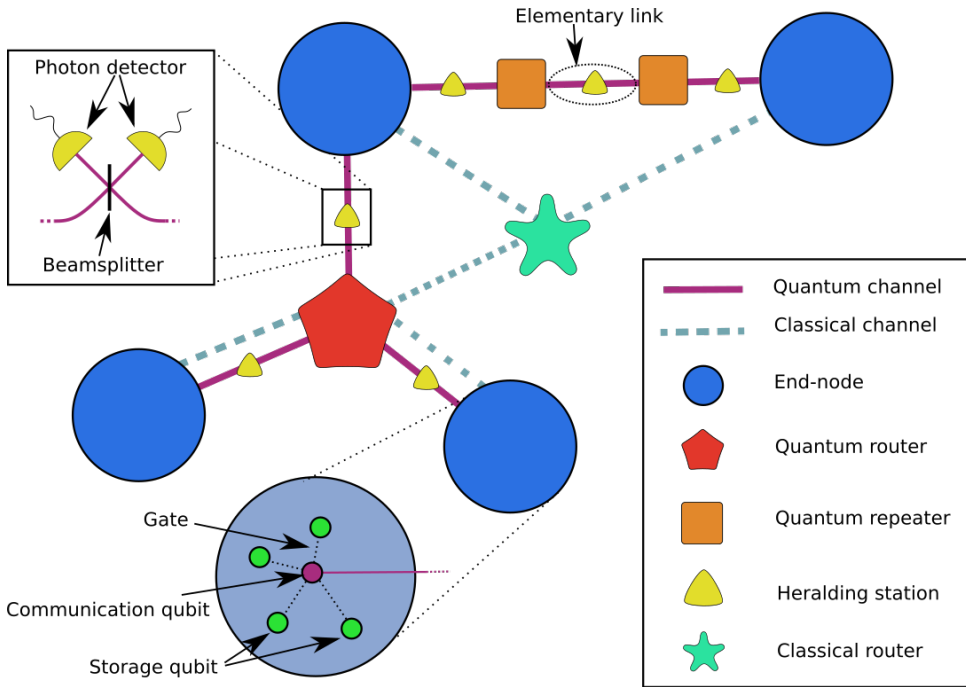


Figure 1.4: Abstract model of a quantum network and its components. *End-nodes* handle application code by users, which require end-to-end entanglement. Long-distance entanglement is generated in the network using the *quantum routers* and *repeaters* by consuming entanglement generated over *elementary links*. An elementary link may consist of quantum channels connected to a beam-splitter with two detectors where photons interfere, for more details see section 2.4.4. End-nodes hold two types of qubits: (1) *communication qubits* which can be used to generated entanglement with remote nodes and (2) *memory qubits* which can be used to store quantum states and apply operations. The qubits within an end-node can interact through quantum gates and their state can be measured.

When writing noisy states, it is convenient to express the state as a *density matrix*. For a perfectly prepared state $|\Psi^-\rangle$, the density matrix is $\rho = |\Psi^-\rangle\langle\Psi^-|$. This allows one to express noise. For example, the analogue of applying a classical bit flip error X with some probability p_{err} can be written as

$$\rho_{\text{noisy}} = (1 - p_{\text{err}})\rho + p_{\text{err}}X\rho X. \quad (1.21)$$

The *fidelity* F measures how close a realized state ρ is to an ideal target state $|\Psi^-\rangle$. The fidelity of a state ρ with the target state $|\Psi^-\rangle$ can be written as

$$F[|\Psi^-\rangle] = \langle\Psi^-|\rho|\Psi^-\rangle \quad (1.22)$$

where $F = 1$ if ρ is identical to the target state. We have $0 \leq F \leq 1$, where a larger value of F means we are closer to the target state.

It is important to note that one cannot measure the fidelity of a single instance of a quantum state. However, if we produce the same state many times in succession, we can estimate its fidelity. One way to do this, for bipartite entanglement, is

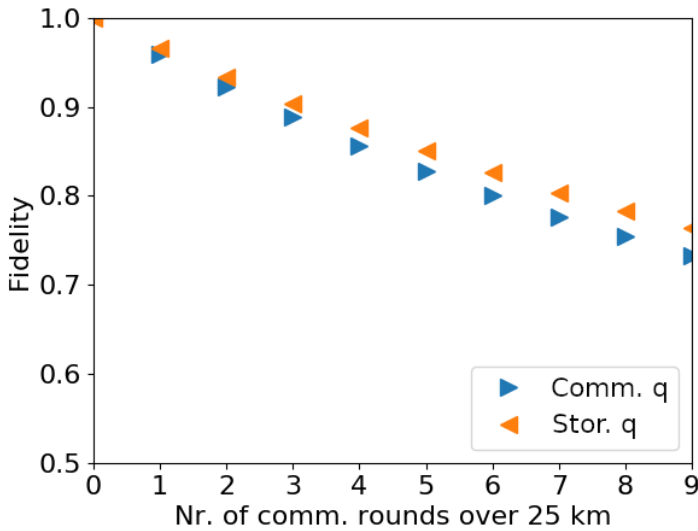
to measure the qubit-error-rate (*QBER*). Consider $|\Psi^-\rangle$ above and recall that measurement outcomes in the X , Z and Y bases are always perfectly anti-correlated in this case. I.e. we always get different measurement outcome for qubit A and for qubit B . In case the state is noisy, this is no longer the case. For a fixed basis (say Z) the QBER (here QBER_Z) is the probability of receiving equal² measurement outcomes, when measuring qubit A and qubit B in the Z basis. Similarly, we can define QBER_X and QBER_Y for measurements in the X and Y bases. One can show that the fidelity and QBER of the Bell state state $|\Psi^-\rangle$ are related as

$$F[|\Psi^-\rangle] = 1 - \frac{\text{QBER}_X + \text{QBER}_Y + \text{QBER}_Z}{2}. \quad (1.23)$$

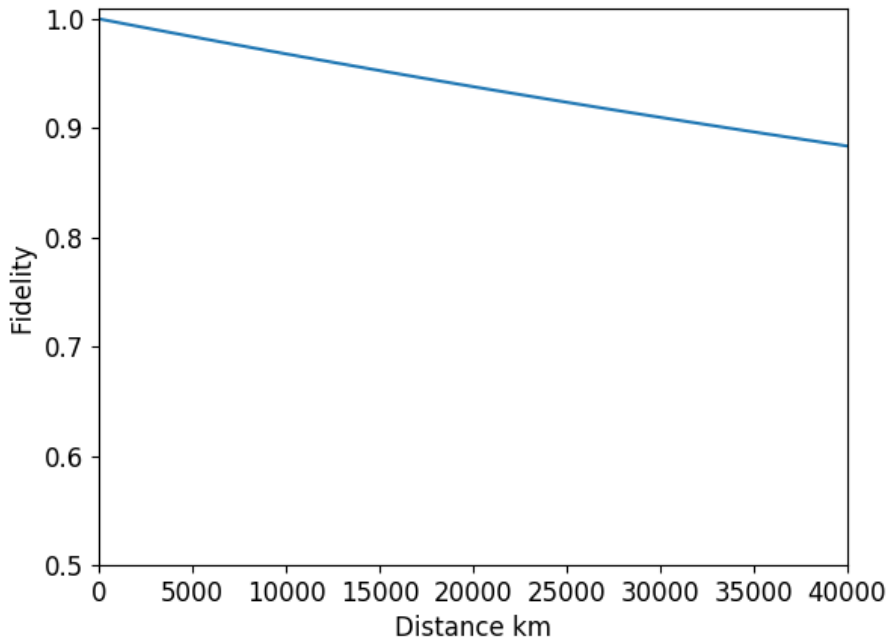
1.4.5. Decoherence

Quantum memories are inherently noisy and the amount of noise a qubit experiences depends on how long it stays in the memory. How long a qubit state is preserved is usually captured by the two numbers T_1 (energy/thermal relaxation time) and T_2 (dephasing time) of the qubit [10], as well as free-induction decay T_2^* (see e.g. [12]). In Figure 1.5a we illustrate how fidelity behaves as a function of time, in the presence of noise. To highlight the actual effect of limited memory lifetimes we show the timescales in terms of kilometers in fiber, where $c = 206753$ km/s is the speed of light in fiber. What is shown in the figure is the fidelity of an entangled state stored in two qubits with a coherence time (T_2) of 1.46 s as a function of the time it takes to communicate over a certain distance.

²QBER for the other Bell states is defined in a similar manner, taking into account that measurement outcomes are always equal in some bases for the other ideal Bell states.



(a) Reduction in fidelity F when storing a perfect entangled state $|\Psi^+\rangle$ in the communication (blue, left triangles) and memory (orange, right triangles) qubit in terms of the number of communication rounds between nodes separated by 25 km. Noise parameters used are: $T_1 = 2.68$ ms and $T_2 = 1.00$ ms for communication and $T_1 = \infty$ and $T_2 = 3.5$ ms for memory qubit.



(b) Illustration of an improved communication qubit with $T_2 = 1.46$ s ($T_1 = \infty$). Such improvements can be achieved by for example a technique called *dynamical decoupling*, see appendix A.3. If such a qubit was used in a platform connected to a network, the qubit could be kept alive while waiting for classical control communication over long distances.

Figure 1.5

References

- [1] S. Wehner, D. Elkouss, and R. Hanson, *Quantum internet: A vision for the road ahead*, *Science* **362**, eaam9288 (2018).
- [2] C. H. Bennett, G. Brassard, C. Crépeau, R. Jozsa, A. Peres, and W. K. Wootters, *Teleporting an unknown quantum state via dual classical and einstein-podolsky-rosen channels*, *Physical Review Letters* **70**, 1895 (1993).
- [3] H. Bidgoli, *The Internet Encyclopedia* (John Wiley & Sons, 2004).
- [4] V. Cerf and R. Kahn, *A protocol for packet network intercommunication*, *IEEE Transactions on Communications* **22**, 637 (1974).
- [5] D. Gottesman, *PhD Theses*, [Ph.D. thesis](#), California Institute of Technology (2004), [arXiv:9705052 \[quant-ph\]](#) .
- [6] M. Van den Nest, J. Dehaene, and B. De Moor, *Graphical description of the action of local clifford transformations on graph states*, *Physical Review A* **69**, 022316 (2004).
- [7] A. Bouchet, *An efficient algorithm to recognize locally equivalent graphs*, *Combinatorica* **11**, 315 (1991), [arXiv:0702057v2 \[cs\]](#) .
- [8] A. Dahlberg, B. van der Vecht, C. Delle Donne, M. Skrzypczyk, W. Kozłowski, and S. Wehner, *Netqasm - a low-level instruction set architecture for hybrid quantum-classical programs in a quantum internet*, (2020), in preparation.
- [9] A. Dahlberg, *Qualg - a symbolic algebra package for quantum information*, [arXiv preprint arXiv:2008.06467](#) (2020).
- [10] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*, 10th ed. (Cambridge University Press, Cambridge, 2010).
- [11] C. H. Bennett, D. P. DiVincenzo, P. W. Shor, J. A. Smolin, B. M. Terhal, and W. K. Wootters, *Remote state preparation*, *Phys. Rev. Lett.* **87**, 077902 (2001).
- [12] N. Kalb, A. A. Reiserer, P. C. Humphreys, J. J. W. Bakermans, S. J. Kamberling, N. H. Nickerson, S. C. Benjamin, D. J. Twitchen, M. Markham, and R. Hanson, *Entanglement distillation between solid-state quantum network nodes*, *Science* **356**, 928 (2017), [arXiv:1703.03244](#) .

2

A Link Layer Protocol for Quantum Networks

Axel Dahlberg, Matthew Skrzypczyk, Tim Coopmans, Leon Wubben, Filip Rozpędek, Matteo Pompili, Arian Stolk, Przemysław Pawełczak, Robert Knegjens, Julio de Oliveira Filho, Ronald Hanson, Stephanie Wehner

Quantum communication brings radically new capabilities that are provably impossible to attain in any classical network. Here, we take the first step from a physics experiment to a quantum internet system. We propose a functional allocation of a quantum network stack, and construct the first physical and link layer protocols that turn ad-hoc physics experiments producing heralded entanglement between quantum processors into a well-defined and robust service. This lays the groundwork for designing and implementing scalable control and application protocols in platform-independent software. To design our protocol, we identify use cases, as well as fundamental and technological design considerations of quantum network hardware, illustrated by considering the state-of-the-art quantum processor platform available to us (Nitrogen-Vacancy (NV) centers in diamond). Using a purpose built discrete-event simulator for quantum networks, we examine the robustness and performance of our protocol using extensive simulations on a supercomputing cluster. We perform a full implementation of our protocol in our simulator,

Parts of this chapter have been published in proceedings of ACM SIGCOMM 2019 Conference [1].

where we successfully validate the physical simulation model against data gathered from the NV hardware. We first observe that our protocol is robust even in a regime of exaggerated losses of classical control messages with only little impact on the performance of the system. We proceed to study the performance of our protocols for 169 distinct simulation scenarios, including trade-offs between traditional performance metrics such as throughput, and the quality of entanglement. Finally, we initiate the study of quantum network scheduling strategies to optimize protocol performance for different use cases.

2.1. Introduction

Quantum communication enables the transmission of quantum bits (qubits) in order to achieve novel capabilities that are provably impossible using classical communication. As with any radically new technology, it is hard to predict all uses of a future Quantum Internet [4, 5], but several major applications have already been identified depending on the stage of quantum network development [4]. These range from cryptography [6, 7], sensing and metrology [8, 9], distributed systems [10, 11], to secure quantum cloud computing [12, 13].

Qubits are fundamentally different from classical bits, which brings significant challenges both to the physical implementation of quantum networks, as well as the design of quantum network architectures. Qubits cannot be copied, ruling out signal amplification or repetition to overcome transmission losses to bridge great distances. Two qubits can share a special relation known as *entanglement*, even if these two qubits are stored at distant network nodes. Such entanglement is central not only to enable novel applications, but also provides a means to realize a quantum repeater, which enables quantum communication over long-distances (Figure 2.1).

At present, short-lived entanglement has been produced probabilistically over short distances (≈ 100 km) on the ground by sending photons over standard telecom fiber (see e.g. [14, 15]), as well as from space over 1203 km from a satellite [16]. Such systems can allow the realization of applications in the prepare-and-measure stage [4] of quantum networks on point-to-point links, i.e. the stage in where end nodes can only prepare and measure single qubits. However, they cannot by themselves be concatenated to allow the transmission of qubits over longer distances. Using such technology, secure communication links have been realized over short distances on the ground, individually or in chains of trusted nodes [4] - see e.g. [17–19]). In a chain of trusted nodes, a separate key is produced between each pair of nodes along the chain, and hence compromising any of those nodes leads to a break in security. Importantly, trusted nodes do not enable the end-to-end transmission of qubits.

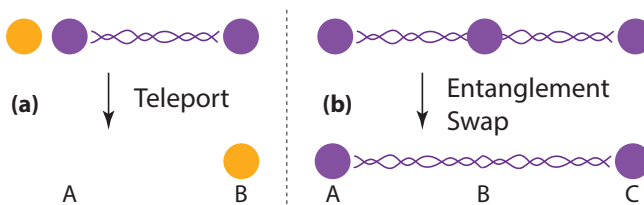


Figure 2.1: Entanglement enables long-distance quantum communication: (a) once two qubits (purple/dark) are confirmed to be entangled (threaded links between qubits), a data qubit (yellow/light) can be sent deterministically using teleportation [2], consuming the entangled pair; (b) long-distance entanglement can be built from shorter segments: If node *A* is entangled with *B* (repeater), and *B* with *C*, then *B* can perform *entanglement swapping* [3] to create long-distance entanglement between the qubits at *A* and *C*.

In order to enable long-distance quantum communication and the execution of complex quantum applications, we would like to produce long-lived entanglement between two quantum nodes that are capable of storing and manipulating qubits. To do so efficiently (Section 2.3.1), we need to confirm entanglement generation by performing *heralded* entanglement generation. This means that there is a *heralding signal* that can be sent to the two nodes to indicate that entanglement has been successfully generated. The generation of a specific entangled pair is not heralded by default, since it requires the ability to generate such a signal without collapsing the quantum state of the entangled qubits (see e.g. Section 2.4.4 for a method that achieves this).

The current world distance record for producing heralded entanglement is 1.3 km, which has been achieved using a solid state platform known as Nitrogen-Vacancy (NV) centers in diamond [20]. Intuitively, this platform is a few qubit (as of now 8 [21]) quantum computer capable of executing arbitrary quantum gates and measurements, with an optical interface to connect to other nodes for entanglement generation. Key capabilities of the NV platform have already been demonstrated, including qubit lifetimes of 1.46 s [22], entanglement production faster than it is lost [23], and sending qubits over entanglement using deterministic quantum teleportation [24]. Other hardware platforms exist that are identical on an abstract level (quantum computer with an optical interface), and on which heralded long-lived entanglement generation has been demonstrated (e.g. Ion Traps [25], and Neutral Atoms [26]). Theoretical proposals and early stage demonstrations of individual components also exist for other physical platforms (e.g. quantum dots [27], rare earth ion-doped crystals [28], atomic gases [29, 30], and superconducting qubits [31]), but their performance is not yet good enough to generate entanglement faster than it is lost.

Up to now, the generation of long-lived entanglement has been the domain of highly sophisticated, but arguably ad-hoc physics experiments. We are now on the verge of seeing early stage quantum networks becoming a reality, entering a new phase of development which will require a *joint effort* across physics, computer science and engineering to overcome the many challenges in scaling such networks. In this chapter, we take the first step from a physics experiment to a fully-fledged quantum communication *system*.

Design considerations and use cases: We identify general design considerations for quantum networks based on fundamental properties of entanglement, and technological limitations of near-term quantum hardware, illustrated with the example of our NV platform. For the first time, we identify systematic use cases, and employ them to guide the design of our stack and protocols.

Functional allocation quantum network stack: We propose a functional allocation of a quantum network stack, and define the service desired from its link layer to satisfy use case requirements and design considerations. In analogy to classical networking, the quantum link layer is responsible for producing entanglement between two nodes that share a direct physical connection (e.g. optical fiber).

First physical and link layer entanglement generation protocols: We proceed to construct the world's first physical and link layer protocols for a quan-

tum network stack that turn ad-hoc physics experiments producing heralded entanglement into a well defined service. This lays the groundwork for designing and implementing control and application protocols in platform-independent software in order to build and scale quantum networks. At the physical layer, we focus primarily on the quantum hardware available to us (NV platform), but the same protocol could be realized directly using Ion Traps or Neutral Atoms, as well as—with minor changes—other means of producing physical entanglement [32]. Our link layer protocol takes into account the intricacies of the NV platform, but is in itself already platform independent.

Simulation validated against quantum hardware: Using a purpose built discrete-event simulator for quantum networks, we examine the robustness and performance of our protocol using more than 169 scenarios totaling 94244 h wall time and 707 h simulated time on a supercomputing cluster. To this end, we perform a complete implementation of our protocols and let them use simulated quantum hardware and communication links. To illustrate their performance, we consider two concrete short and long-distance scenarios based on the NV platform: (1) Lab where the nodes A and B are 2 m apart. Since this setup has already been realized, we can use it to compare the performance of the entanglement generation implemented on real quantum hardware against the simulation to validate its physical model, and (2) a planned implementation of QL2020 where A and B are in two Dutch cities separated by ≈ 25 km over telecom fiber. Next to investigating trade-offs between traditional performance metrics (e.g. throughput or latency) and genuinely quantum ones (fidelity, Section 2.4.2), we take a first step in examining different quantum network scheduling strategies to optimize performance for different use cases.

2.2. Related Work

At present there is no quantum network stack connected to quantum hardware, no link layer protocols have been defined to produce long-lived entanglement, and no quantum networks capable of end-to-end qubit transmission or entanglement production have been realized (see [4] and references therein). Also, we are not aware of any other systematic investigation on use cases informing requirements for such an architecture.

A functional allocation of a stack for quantum repeaters and protocols controlling entanglement distillation (a process of correcting errors in entanglement) has been outlined in [33–36], which is complementary to this work. This is very useful to ultimately realize entanglement distillation, even though no complete control protocols or connection to a hardware system were yet given. We remark that here we do not draw layers from specific protocols like entanglement distillation, but focus on the service that these layers should provide (a layer protocol may of course choose distillation as a means to realize requirements). An outline of a quantum network stack was also put forward in [37], including an appealing high level quantum information theory protocol transforming multi-partite entanglement. However, this high level protocol does not yet consider failure modes, hardware imperfections, nor the requirements on entanglement generation protocols and the impact of classical

control. Plans to realize the physical layer of a quantum network from a systems view were put forward in [38], however development has taken a different route.

In the domain of single-use point-to-point links for quantum key distribution (QKD), software has been developed for trusted repeater networks [4] to make use of such key in e.g. VoIP [39]. However, these do not allow end-to-end transmission of qubits or generation of entanglement, and rely on trust in the intermediary nodes who can eavesdrop on the communication. Control using software defined networks (SDN) to assist trusted repeater nodes has been proposed, e.g. [40, 41]. These QKD-centric protocols however do not address control problems in true quantum networks aimed at end-to-end delivery of qubits, and the generation of long-lived entanglement.

In contrast, classical networking knows a vast literature on designing and analyzing network protocols. Some ideas can indeed be borrowed from classical networking such as scheduling methods, but fundamental properties of quantum entanglement, as well as technological considerations of quantum hardware capabilities (Section 2.4.5) call for new protocols and methods of network control and management. Naturally, there is a continuous flow of systems papers proposing new networking architectures, e.g. for SDN [42], data center networks [43], content delivery networks [44] or cloud computing [45], to name a few. Yet, we are unaware of any system-level papers proposing a quantum network stack including protocols for concrete hardware implementations.

2.3. Design Considerations for Quantum Network Architectures

We first discuss design considerations of quantum networks themselves, followed by considerations specific to the quantum physical and link layers (Section 2.4). These can be roughly subdivided into three categories: (i) fundamental considerations due to quantum entanglement, (ii) technological limitations of near-term quantum hardware, and (iii) requirements of quantum protocols themselves.

2.3.1. Qubits and Entanglement

We focus on properties of entanglement as relevant for usage and control (see section 1.4, and [46, 47]). Teleportation [2] allows entanglement to be used to send qubits (see Figure 2.1). We will hence also call two entangled qubits an *entangled link* or *entangled pair*. Teleportation consumes the entangled link, and requires two additional classical bits to be transmitted per qubit teleported. Already at the level of qubit transmission we hence observe the need for a close integration between quantum and classical communication. Specifically, we will need to match quantum data stored in quantum devices with classical control information that is sent over a separate physical medium, akin to optical control plane architectures for classical optical networks [48]. To create long-distance entanglement, we can first attempt to produce short-distance entangled links, and then connect them to form longer distance ones [49, 50] via an operation known as entanglement swapping (see Figure 2.1). This procedure can be used iteratively to create entanglement along

long chains, where we remark that the swapping operations can in principle be performed in parallel. From a resource perspective, we note that to store entanglement, both nodes need to store one qubit per entangled link. Proposals for enabling quantum communication by forward communication using quantum error correction also exist, which avoid entanglement swapping [51]. However, these have arguably much more stringent requirements in terms of hardware, putting them in a technologically more distant future: they require the ability to create entangled states consisting of a large number of photons (only ten realized today [52]) and densely placed repeater stations performing near perfect operations [53].

Producing heralded entanglement does however allow long-distance quantum communication without the need to create entanglement consisting of many qubits. Here, the heralding signal (see Figure 2.2) provides a confirmation that an entanglement generation attempt has succeeded. Such heralding - i.e. confirmed entanglement - allows techniques using entanglement swapping to enable long-distance quantum communication without exponential overheads [49], and without the need for more complex resources [54, 55]. Creating long-distance links between two controllable nodes by means of entanglement swapping (Section 2.3.2), and executing complex applications requires both nodes to know the state of their entangled links (which qubits belong to which entangled link, and who holds the other qubit of the entangled pair). As illustrated in Figure 2.1, remote nodes ("B" in the figure) can change the state of such entangled links ("A" and "C" in the figure). Entanglement is an inherently connected element already at the lowest physical level, whereas classical communication typically proceeds by forward communication that does not require information at both the sender and receiver to be used.

2.3.2. Quantum Network Devices

We focus on a high level summary of devices in a quantum network without delving into detailed physics (for more details, see [4, 32, 56] and Section 2.4.4). Qubits can be sent optically through standard telecom fiber using a variety of possible encodings, such as polarization [6, 57], time-bin [58], or absence and presence of a photon [55]. Such qubits can be emitted from quantum nodes [59–61], but in principle also transferred [61–63] from fiber into the node's local quantum memory. Present day quantum memories have very limited lifetimes, making it highly desirable to avoid the exchange of additional control information before the entanglement can be used.

We distinguish two classes of quantum nodes. One, which we will call a *controllable quantum node*, offers the possibility to perform controllable quantum operations as well as storing qubits. Specifically, these nodes enable decision making, e.g. which nodes to connect by entanglement swapping. Such nodes can act as quantum repeaters and decision making routers in the network (e.g. NV platform or other quantum memories combined with auxiliary optics), and—if they support the execution of gates and measurements—function as *end nodes* [4] on which we run applications (e.g. NV centers in diamond or Ion Traps). Others, which we call *automated quantum nodes*, are typically only timing controlled, i.e. they perform the same pre-programmed action in each time step. Such nodes can also support

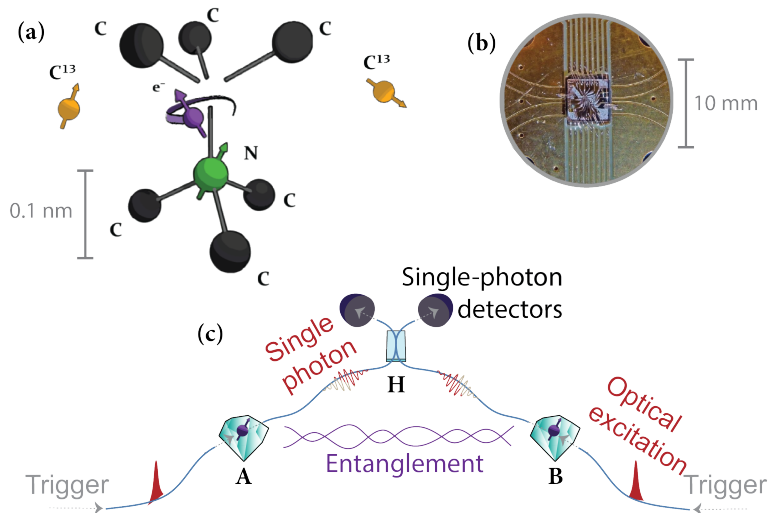


Figure 2.2: Heralded entanglement generation on the NV platform. (a) NV centers are point defects in diamond with an electronic spin as a communication qubit (purple) and carbon-13 nuclear spins as memory qubits (yellow), realized in custom chips (b). (c) A trigger produces entanglement between the communication qubits of *A* and *B* (diamonds) and two qubits (photons) traveling over fiber to the heralding station *H*. *H* measures the photons by observing clicks in the left or right detector giving the *heralding signal s*: [failure] (none or both click), [success, $|\Psi^+\rangle$] (left clicks), [success, $|\Psi^-\rangle$] (right clicks). Success confirms one of two types of entangled pairs $|\Psi^+\rangle$ or $|\Psi^-\rangle$ (wiggly purple line). *H* sends *s* to *A* and *B* (not pictured).

a limited set of quantum operations and measurements, but only those necessary to perform their pre-programmed tasks. Automated nodes are still very useful, for example, to establish entanglement along a chain of quantum repeaters performing the entanglement swapping operations [49, 50] (see again Figure 2.1). In Section 2.4.4 we give a concrete example of such a timing controlled element.

2.3.3. Use Cases

We distinguish five use cases of the stack: one related to producing long-distance entanglement, and four that come from application demands. Since no quantum network has been realized to date, we cannot gain insights from actual usage behavior. Instead we must resort to properties of application protocols known today. Looking into the future, we desire flexibility to serve all use cases, including supporting multiple applications at the same time.

Measure Directly (MD) Use Case: The first application use case comes from application protocols that produce many ($\geq 10^4$) pairs of entangled qubits sequentially, where both qubits are immediately measured to produce classical correlations. As such, no quantum memory is needed to store the entanglement and it is not necessary to produce all entangled pairs at the same time. It follows that applications making use of this use case may tolerate fluctuating delays in entanglement generation. Additionally, it is not essential to deliver error free correlations obtained from entanglement to the application. Such applications will thus already anticipate error fluctuation across the many pairs. This contrasts with classical networking where errors are often corrected before the application layer. Examples of such applications are QKD [7], secure identification [64] and other two-party cryptographic protocols [65–69] at the prepare-and-measure network stage [4], and device-independent protocols at the entanglement network stage [4].

Create and Keep (CK) Use Case: The second application use case stems from protocols that require genuine entanglement, possibly even multiple entangled pairs to exist simultaneously. Here, we may wish to perform joint operations on multiple qubits, and perform quantum gates that depend on back and forth communication between two nodes while keeping the qubits in local quantum storage. While more applications can be realized with more qubits, this use case differs substantially in that we want to create relatively few (even just one) entangled pairs, but want to store this entanglement. Since we typically want these pairs to be available at the same time, and memory lifetimes are short, we want to avoid delay between producing consecutive pairs, which is superficially similar to constraints in real time classical traffic. Also for CK, many applications can perform well with noisy entangled links and the amount of noise forms a performance metric (fidelity, Section 2.4.2). Examples of such protocols lie in the domain of sensing [8], metrology [9], and distributed systems [10, 11] which are in the quantum memory network stage and above [4].

Remote State Preparation (RSP) Use Case: For certain application protocols (for example, secure delegated quantum computation [12, 13]), an interpolation between the CK and MD use case can be considered. Here, one of the two qubits is immediately measured as in the MD use case, but the other is stored as in the

CK use case. Due to the similarity to the CK use case, we will only distinguish the RSP case in the appendix A.

Send Qubit (SQ) Use Case: While many application protocols known to date consume entanglement itself, some — such as distributed quantum computing applications — ask for the transmission of (unknown) qubits. This can be realized using teleportation over any distance as long as entanglement is confirmed between the sender and the receiver. For the quantum link layer, this again does not differ from CK, where we want to produce one entangled pair per qubit to be sent.

Network Layer (NL) Use Case: In analogy to the classical notion of a link layer, we take the quantum link layer to refer to producing entanglement between neighboring nodes (see Section 2.3.4). The network layer will be responsible for producing entanglement between more distant ones. While usage behavior of quantum networks is unknown, it is expected (due to technological limitations) that routing decisions, i.e. how to form long-distance links from pairwise links, will not be entirely dynamic. One potential approach would be to first determine a path, and reserve it for some amount of time such that pairwise entanglement can be produced. Producing pairwise entanglement concurrently enables simultaneous entanglement swapping along the entire path with minimal delay to combat limited memory lifetimes. For this, the network layer needs to be capable of prioritizing entanglement production between neighboring nodes.

2.3.4. Network Stack

Based on these considerations, we propose an initial functional allocation of a quantum network stack (see Figure 1.1). In analogy to classical networking, we refer to the lowest element of the stack as the physical layer. This layer is realized by the actual quantum hardware devices and physical connections such as fibers. We take the physical layer to contain no decision making elements and keep no state about the production of entanglement (or the transmissions of qubits). The hardware at the physical layer is responsible for timing synchronization and other synchronization, such as laser phase stabilization [23], required to make attempts to produce heralded entanglement (Section 2.4.4). A typical realization of the physical layer involves two controllable quantum nodes, linked by an (chain of) automated quantum node that attempt entanglement production in well-defined time slots.

The task of the quantum link layer is then to turn the physical layer making entanglement attempts into a robust entanglement generation service, that can produce entanglement between controllable quantum nodes connected by an (chain of) automated quantum node. Requests can be made by higher layers to the link layer to produce entanglement, where robust means that the link layer endows the physical system with additional guarantees: a request for entanglement generation will (eventually) be fulfilled or result in a time-out. This can be achieved by instructing the physical layer to perform many attempts to produce entanglement until success.

Built on top of the link layer rests the network layer, which is responsible for producing long-distance entanglement between nodes that are neither connected directly, nor connected by a chain of automated quantum nodes at the physical

layer. This may be achieved by means of entanglement swapping, using the link layer to generate entanglement between neighboring controllable nodes. In addition, it contains an entanglement manager that keeps track of entanglement in the network, and which may choose to pre-generate entanglement to service later requests from higher layers. It is possible that the network layer and entanglement manager may eventually be separated.

To assist the SQ use case, a transport layer takes responsibility for transmitting qubits deterministically (e.g. using teleportation). One may question why this warrants a separate layer, rather than a library. Use of a dedicated layer allows two nodes to pre-share entanglement that is used as applications of the system demand it. Here, entanglement is not assigned to one specific application (purpose ID, Section 2.4.1). This potentially increases the throughput of qubit transmission via teleportation, as teleportation requires no additional connection negotiation, but only forward communication from a sender to the receiver. Implementing such functionality in a library would incur delays in application behavior as entanglement would need to be generated on-demand rather than supplying it from pre-allocated resources.

2.4. Design Considerations for quantum link layer

2.4.1. Desired Service

The link layer offers a robust entanglement creation service between a pair of controllable quantum nodes A and B that are connected by a quantum link, which may include automated nodes along the way. This service allows higher layers to operate independently of the underlying hardware platform, depending only on high-level parameters capturing the hardware capabilities.

Requesting entanglement

Our use cases bring specific requirements for such a service. Entanglement creation can be initiated at either A or B by a CREATE request from the higher layer with parameters:

- *Remote node* with whom entanglement generation is desired if the node is connected directly to multiple others.
- *Type of request* - create and keep (K), create and measure (M), and remote state preparation (R). The first type of request (K) stores entanglement, addressing the use cases CK and NL (see Section 2.3.3). The second (M) leads to immediate measurement, supporting the use case MD. The reason for distinguishing these two scenarios is twofold: first, we will show later (Section 2.4.4) that a higher throughput can for some implementations be achieved for M than for K on the same system. Second, simple photonic quantum hardware without a quantum memory and sophisticated processing capabilities [70] only supports the M mode of operation. In R, a measurement is performed only at one node. Since it behaves like K, we will only expand upon R in the appendix A.

- *Number of entangled pairs to be created.* Allowing the higher layer to request several pairs at once can increase throughput by avoiding additional processing delays due to increased inter-layer communication (as compared to classical networks [71, Table 2]). It also helps the CK use case where an application actually needs several pairs concurrently.
- *Atomic* is a flag that indicates that the request should be satisfied as a whole without interruption by other requests.
- *Consecutive* is a flag indicating an OK is returned for each pair made for a request (typical for NL use case). Otherwise, an OK is sent only when the entire request is completed (more common in application use cases).
- *Waiting time, t_{\max} (and time units)* can be used to indicate the maximum time that the higher layer is willing to wait for completion of the request. This allows a general timeout to be set, and enables the NL and CK use case to specify strict requirements since the requested pairs may no longer be desirable if they are delivered too late.
- A *purpose ID* can be specified which allows the higher layer to tag the entanglement for a specific purpose. For an application use case, this purpose ID may be considered analogous to a port number found in the TCP/IP stack. Including it in the CREATE request allows both nodes to immediately provide the entanglement to the right application and proceed processing without incurring further communication delays. Reducing any additional communication overhead is necessary due to the noisy nature of quantum devices. A purpose ID is also useful to identify entanglement created by the NL use case for a specific long-distance path. We envision that an entanglement manager who may decide to pre-generate entanglement would use a special tag to indicate "ownership" of the requested pairs. For the NL use case for example, if the entanglement request does not correspond to a pre-agreed path, then the remote node may refuse to engage in entanglement generation. Finally, because quantum resources are scarce, a purpose ID enables rejection of requests from remote nodes based on scheduling or security policies.
- A *priority* that may be used by a scheduler. Here we use only three priorities in our simulations (use cases NL, MD and CK), but we remark that in the future more fine grained priorities may find use. For now, we merely provision space for such information for traffic engineering purposes.
- *Random basis choice* to be used for measurements in MD requests. May be used to specify measurement bases that are sampled from uniformly by the local and remote nodes from a set of bases commonly used in QKD (see section 1.4).
- *Measurement basis* for the local and remote nodes should one desire all measurements be performed in a fixed basis. Default is a measurement in the standard basis. Other bases may be specified in terms of rotations around the Bloch sphere axes of a qubit (see section 1.4).

- Finally, we allow a specification of a purely quantum parameter (see section 1.4), the *desired minimum fidelity*, F_{\min} , of the entanglement [46]. Here, it is sufficient to note that the fidelity $0 \leq F \leq 1$ measures the quality of entanglement, where a higher value of F means higher entanglement quality. The ideal target state has $F = 1$, while $F \geq 1/2$ is often desirable [72]. Higher fidelity implies lower quantum bit error rate (QBER), which captures the probability that measurements on the entangled state deviate from the ideal outcomes (see section 1.4).

The reason for allowing different F_{\min} instead of fixing one for each hardware platform is that the same platform can be used to produce higher or lower fidelity pairs, where a higher fidelity pair costs more time to prepare. An example of this is the use of entanglement distillation [73, 74] where two lower quality pairs are combined into one higher quality one. Another is the choice of bright state population α (see Section 2.4.4), which can be chosen to trade-off fidelity and throughput. In practice, the necessary minimum fidelity required to execute either long distance entanglement generation or application protocols may be obtained by the requirements for the successful operation of said protocols, and differs significantly across protocols. Such minimum fidelity requirements are typically concluded from an analytical or numerical analysis of such protocols, and are not yet known for many proposed application protocols.

Response to entanglement requests

If entanglement has been produced successfully, an OK message should be returned. In addition, the use cases specified in Section 2.3.3 desire several other pieces of information, which may also be tracked at higher layer:

- An entanglement identifier Ent_{ID} unique in the network during the lifetime of the entanglement. This allows both nodes to immediately process the entanglement without requiring an additional round of communication degrading the entanglement due to limited memory lifetimes.
- A qubit ID for K -type (create and keep) requests which identifies where the local qubit is in the quantum memory device.
- The "Goodness" G , which for K requests is an estimate (see section 1.4) of the fidelity — where $G \geq F_{\min}$ should hold — and for M an estimate of the QBER (see section 1.4).
- The measurement outcome for M type requests.
- The time of entanglement creation.
- The time the goodness parameter was established. The goodness may later be updated given fixed information about the underlying hardware platform.

Explicit OK messages from the link layer are desired for several reasons which derive from the task of the link layer to turn low probability generation at the physical layer

into a robust service: First, before an entanglement swapping or other operation may be performed by the network layer we need to know entanglement has been produced. Second, applications demand knowledge of entanglement identifiers or measurement outcomes to proceed successfully.

Evidently, there are many possibilities of failure resulting in the return of error messages. This includes:

- Timeout when a request could not be fulfilled in a specific time frame (TIME-OUT).
- An immediate rejection of the request because the requested fidelity is not achievable in the given time frame (UNSUPP).
- The quantum storage is permanently (MEMEXCEEDED) or temporarily (OUT-OFMEM) too small to simultaneously store all pairs of an atomic request.
- Refusal by the remote node to participate (DENIED).

Finally, we allow an EXPIRE message to be sent, indicating that the entanglement is no longer available. This in principle can be indicated by a quantum memory manager (see Section 2.5.2) instead of the protocol, but we will show that this allows for recovery from unlikely failures.

Fixed hardware parameters

Not included in these request or response messages are parameters that are fixed for the specific hardware platform, or change only very infrequently. As such, these may be obtained by the higher-level software by querying the low level system periodically, similarly to some classical network architectures (e.g. [75]). Such parameters include:

- The number of available qubits.
- The qubit memory lifetimes.
- Possible quantum operations.
- Attainable fidelities and generation time.
- The class of states that are produced.

The latter refers to the fact that more information about that state than just the fidelity allows optimization at layers above the link layer.

2.4.2. Performance Metrics

Before designing any protocols that adhere to these requirements, we consider the performance metrics that such protocols may wish to optimize. Standard metrics from networking also apply here, such as *throughput* (entangled pairs/s), and the *latency*. We distinguish between:

- Latency per request (time between submission of a CREATE request and its successful completion at a requesting node).
- Latency per pair (time between CREATE and OK at requesting node).
- Latency per request divided by the number of requested pairs (which we denote as the *scaled latency*).

Given that requests may originate at both A and B , we also demand *fairness*, i.e., the metrics should be roughly independent of the origin of the request. Here, we also care about genuinely quantum quality metrics, specifically the fidelity F (at least F_{\min}).

The non-quantum reader may wonder about the significance of F , and why we do not simply maximize throughput (e.g. [42, 76]) or minimize latency (e.g. [44, 77]). For instance, QKD (a MD use case as listed in Section 2.3.3), requires a minimum quantum bit error rate (QBER) between measurement outcomes at A and B (related to F , see section 1.4). A lower F results in a larger QBER, allowing less key to be produced per pair. We may thus achieve a higher throughput, but a lower number of key bits per second, or key generation may become impossible.

2.4.3. Error Detection

Link layer protocols for classical communication typically aim to correct or detect errors, e.g. using a CRC. In principle, there exists an exact analogy at the quantum level: We could use a checksum provided by a quantum error correcting code (QECC) [46, 78] to detect errors. This is technologically challenging and experimental implementations of QECC are in very early stages [79–81]: to use a QECC for information traveling 5km, we would need to create highly entangled quantum states of many qubits, combined with quantum operations of extremely high precision [82]. Yet, apart from technological limitations, future quantum link layer protocols may not use quantum checksums due to different use case requirements. We typically only demand some minimum fidelity F_{\min} with high confidence that may also fluctuate slightly for pairs produced within a time window. That is, the applications do not expect all errors to be corrected for them.

As we thus allow imperfect pairs to be delivered to an application, we instead use a different mechanism: we intersperse test rounds during entanglement generation (for details, see appendix A.1) to verify the quality of the link, by estimating the fidelity of the generated entanglement. Such test rounds are easy to produce without the need for complex gates or extra qubits. Evidently, there exists an exact analogy in the classical networking world, where we would transmit test bits to measure the current quality of transmission, e.g. a direct analogy to network profiling [75] to gain confidence that the non-test bits are also likely to be transmitted with roughly the same amount of error. Yet, there we typically care about correctness of a specific data item, rather than an enabling resource like entanglement.

2.4.4. Physical Entanglement Generation

Let us now explain how heralded entanglement generation is actually performed between two controllable nodes A and B (see appendix A for details). As an example, we focus on the hardware platform available to us (NV in diamond, Figure 2.2), but analogous implementations have been performed using remote Ion Traps [25] and Neutral Atoms [26].

In all cases (NV, Ion Trap, Neutral Atom, and others), processing nodes A and B are few-qubit quantum computers, capable of storing and manipulating qubits. They are connected to an intermediate station called the *heralding station* H over optical fibers. This station is a much simpler automated node, built only from linear optical elements. Each node can have two types of qubits: *memory qubits* as a local memory, and *communication qubits* with an optical interface, that can be entangled with a photon. To produce entanglement, a time synchronized trigger is used at both A and B to create entanglement between each communication qubit, and a corresponding traveling qubit (photon). These photons are sent from A and B to H over fiber. When both arrive at H , H performs an automatic entanglement swapping operation which succeeds with some probability. Since H has no quantum memory, both photons must arrive at H at the same time to succeed. Success or failure is then transmitted back from H to the nodes A and B over a standard classical channel (e.g. 100Base-T). In the case of success, one of several entangled states may be produced, which can however be converted to one other using local quantum gates at A or B . The heralding signal is used to indicate which state was produced. After a generation attempt, the communication qubit may be moved to a memory qubit, in order to free the communication qubit to produce the next entangled pair. Many parameters influence the success and quality of this process, such as the quality of the qubits themselves, the probability of emission of a photon given a trigger signal, losses in fiber, and quality of the optical elements such as detectors used at H (Figure 2.2).

To understand this process in more detail, consider the NV platform (Figure 2.2) (see e.g. [23] for details on this process, and [83] for an overview of the NV platform in general). Two different schemes for producing entanglement have been implemented, that differ in how the qubits are encoded into photons (time-bin [54], or presence/absence of a photon [55]). While physically different, both of these schemes fit into the framework of our physical and link layer protocols.

To evaluate the performance of the protocol (Section 2.6) and provide intuition of timings, we compare to data from the setup [23] which uses presence/absence of a photon as encoding. A microwave pulse prepares the communication qubit depending on a parameter α , followed by a laser pulse to trigger photon emission (total duration $5.5\mu\text{s}$). A pair ($|\Psi^+\rangle$ or $|\Psi^-\rangle$) is successfully produced with fidelity $F \approx 1 - \alpha$ with probability $p_{\text{succ}} \approx 2\alpha p_{\text{det}}$, where $p_{\text{det}} \ll 1$ is, given that a photon was emitted, the probability of heralding success. The parameter α thus allows a trade-off between the rate of generation (p_{succ}), and the quality metric F . Other factors that impact the fidelity are memory decoherence, detector dark-counts, phase instability, losses, imperfect operations and more (see appendix A.3.2). For K type requests, we may store the pair in the communication qubit, or move to a memory

qubit (gate duration $1040\mu\text{s}$ for the qubit considered). The quality of this qubit degrades as we wait for H to reply. For M type requests, we may choose to measure immediately before receiving a reply (here readout takes $3.7\mu\text{s}$). Important is the time of an attempt t_{attempt} (time preparing the communication qubit until receiving a reply from H , and completion of any post-processing such as moving to memory), and the maximum attempt rate r_{attempt} (maximum number of attempts that can be performed per second not including waiting for a reply from H or post-processing). The rate r_{attempt} can be larger than $1/t_{\text{attempt}}$: (1) for M the communication qubit is measured before receiving the reply from H and thus allows for multiple attempts to overlap and (2) for K, if the reply from H is failure, then no move to memory is done.

For performance evaluation we consider two physical setups as an example (see appendix A.2.2) with additional parameters hereafter referred to as the Lab scenario and the QL2020 scenario. The Lab scenario already realized [23] with 1 m distance to the station from both A and B (communication delay to H negligible), $p_{\text{succ}} \approx \alpha \cdot 10^{-3}$ (F vs. α , Figure 2.8). For M requests, we act the same for Lab and QL2020 and always measure immediately before parsing the response from H to ease comparison (thus $t_{\text{attempt}} = 1/r_{\text{attempt}} = 10.12 \mu\text{s}$ which includes electron readout $3.7 \mu\text{s}$, photon emission $5.5 \mu\text{s}$ and a 10 % extra delay to avoid race conditions). For K requests in Lab, $t_{\text{attempt}} = 1045 \mu\text{s}$ but $1/r_{\text{attempt}} \approx 11 \mu\text{s}$ as memory qubits need to be periodically initialized ($330 \mu\text{s}$ every $3500 \mu\text{s}$). The QL2020 scenario has not been realized and is based on a targeted implementation connecting two Dutch cities by the end of 2020 ($\approx 10\text{km}$ from A to H with a communication delay of $48.4\mu\text{s}$ in fiber, and $\approx 15\text{km}$ from B to H with a $72.6\mu\text{s}$ delay). Frequency conversion of 637nm to 1588nm needs to be performed on the photons emitted in our modeled NV center, where fiber losses at 1588nm are taken to be 0.5dB/km (values for deployed QL2020 are $0.43\text{-}0.47\text{ dB/km}$). We assume the use of optical cavities to enhance photon emission [84, 85] giving a probability of success $p_{\text{succ}} \approx \alpha \cdot 10^{-3}$. F is worse due to increased communication times from H . For QL2020, $t_{\text{attempt}} = 145 \mu\text{s}$ for M (trigger, wait for reply from H) and $t_{\text{attempt}} = 1185 \mu\text{s}$ for K (trigger, wait for reply from H , swap to carbon). Maximum attempt rates are $1/r_{\text{attempt}} = 10.120 \mu\text{s}$ (M) and $1/r_{\text{attempt}} \approx 165 \mu\text{s}$ (K).

2.4.5. Hardware Considerations

Quantum hardware imposes design considerations for any link layer protocol based on top of such experiments for generating entanglement.

Trigger generation: Entanglement can only be produced if both photons arrive at the heralding station at the same time. This means that the low level system requires tight timing control; such control (ns scale) is also required to keep the local qubits stable. This imposes hard real time constraints at the lowest level, with dedicated timing control (AWG) and software running on a dedicated microcontroller (Adwin ProII). We expect that a physical layer protocol built on heralded entanglement without the use of additional quantum memories would operate over distances up to 100km . As such, providing timing synchronization at the required level may be done using existing techniques such as White Rabbit [86]. Timing constraints

to perform entanglement swapping over larger distances at higher layers, or using automated nodes with memories are less stringent. When considering a functional allocation between the physical and link layer in the quantum network stack, this motivates taking all timing synchronization to happen at the physical layer. At this layer, we may then also timestamp classical messages traveling to and from H , to form an association between classical control information and entangled pairs.

Scheduling and flow control: Consequently, we make the link layer responsible for all higher level logic, including scheduling, while keeping the physical layer as simple as possible. An example of scheduling other than priorities, is flow control which controls the speed of generation, depending on the availability of memory on the remote node to store such entanglement.

Note that depending on the number of communication qubits, and parallelism of quantum operations that the platforms allows, a node also needs a global scheduler for the entire system and not only the actions of the link layer.

Noise due to generation: One may wonder why one does not continuously trigger entanglement generation locally whenever the node wants a pair, or why one does not continuously produce pairs and then this entanglement is either discarded or otherwise made directly available. In the NV system, triggering entanglement generation causes the memory qubits to degrade faster [87, 88]. As such we would like to achieve agreement between nodes to avoid triggering unless entanglement it is indeed desired.

This consideration also yields a security risk: if an attacker could trick a node into triggering entanglement generation, without a matching request on the other side, this would allow a rapid destruction of contents of the nodes' local quantum memory. For this reason, we want classical communication to be authenticated which can be achieved using standard methods.

Memory allocation: Decisions on which qubits to use for what purpose lies in the domain of higher level logic, where more information is available. We let such decisions be taken by a global quantum memory manager (QMM), which can assist the link layer to make a decision on which qubits to employ. It can also translate logical qubit IDs into physical qubit IDs in case multiple qubits are used to redundantly form one logical storage qubit.

2.5. Protocols

We now present our protocols satisfying the requirements and considerations set forth in Sections 2.3 and 2.4. The entanglement generation protocol (QEGP) at the link layer, uses the midpoint heralding protocol (MHP) at the physical layer. Classical communication is authenticated, and made reliable using standard methods (e.g. 802.1AE [89], authentication only).

2.5.1. Physical Layer MHP

Our MHP is a lightweight protocol built directly on top of physical implementations of the form of Section 2.4.4, supplementing them with some additional control information. With minor modifications this MHP can be adapted to other forms of

heralded entanglement generation between controllable nodes, even using multiple automated middle nodes [90].

The MHP is meant to be implemented directly at the lowest level subject to tight timing constraints. Protocol execution is divided into time slots, which are synchronized between the two neighboring nodes (Section 2.4.4). In each time slot, the MHP polls the higher layer (Figure 2.3, the link layer QEGP) to determine whether entanglement generation is required in this slot. A batched operation is possible, should the delay incurred by the polling exceed the minimum time to make one entanglement generation attempt - *the MHP cycle* - and hence dominate the throughput. MHP keeps no other state. Upon polling, the higher layer may respond "no" in which case no attempt to produce entanglement will be made or with "yes", additionally providing parameters to use in the attempt. These parameters include the type of request (M, measure) or (K, store) passed on from the higher layer, for which the MHP takes the following actions.

Protocol for Create and Keep (K)

The parameters given to the MHP with a "yes" response contain the following:

- An ID for the attempt that is forwarded to H ,
- Generation parameters (α , Section 2.4.4),
- The device qubits for storing the entanglement,
- A sequence of operations to perform on the device memory ¹.

The higher layer may instruct the MHP to perform a gate on the communication qubit depending on the heralding signal from H allowing the conversion from the $|\Psi^-\rangle$ state to the $|\Psi^+\rangle$ state, before returning completion to the higher layer. Entanglement generation is then triggered at the start of the next time interval, using the generation parameter α , and a GEN message is sent to H which includes a timestamp, and the given ID. The motivation for including the ID is to protect against errors in the classical control, for example losses.

The station H uses the timestamp to link the message to a detection window in which the accompanying photons arrived. Should messages from both nodes arrive, the midpoint verifies that the IDs transmitted with the GEN messages match, and checks the detection counts (Figure 2.2) from the corresponding detection window. The midpoint will then send a REPLY message indicating success or failure, and in the case of success, which of the two states, $|\Psi^+\rangle$ and $|\Psi^-\rangle$, was produced. The REPLY additionally contains a sequence number uniquely identifying the generated pair of entangled qubits chosen by H , which later enables the QEGP to assign unique entanglement identifiers. This REPLY and the ID is forwarded to the link layer for post-processing. Note that the REPLY may be received many MHP cycles later, allowing the potential for emission multiplexing (Section 2.5.2).

¹Less abstractly, by specifying microwave and laser pulse sequences controlling the chip (see appendix A).

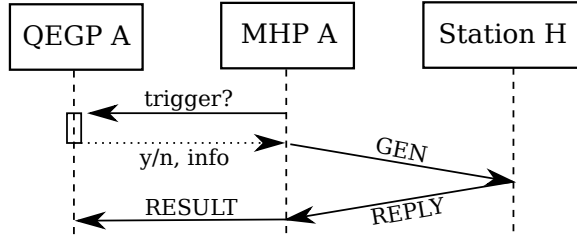


Figure 2.3: Timeline of the MHP polling higher layers to see if entanglement should be produced.

Protocol for Create and Measure (M)

Handling M type requests is very similar, differing only in two ways: Instead of performing a gate on the communication qubit, the “yes” message requests the MHP to perform a measurement on the communication qubit in a specified basis once the photon has been emitted, even before receiving the response from H . The outcome of the measurement and the REPLY are passed back to the QEGP. In practice, the communication time from transmitting a GEN message to receiving a REPLY may currently exceed the duration of such a local measurement ($3.7 \mu\text{s}$ vs. communication delay Lab 9.7 ns, and QL2020 145 μs). The MHP may thus choose to perform the measurement immediately (communication delay exceeds measurement delay) such as in Figure 2.4, or only after receiving the response (measurement delay exceeds communication delay).

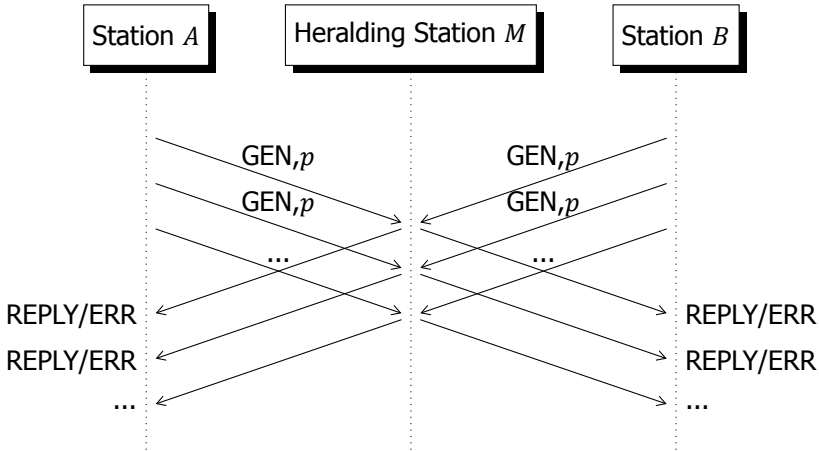


Figure 2.4: Timeline of multiplexing photon emission in the MHP: multiple GEN_{p} messages are sent one by one. For details about the GEN and REPLY/ERR messages, see figs. A.18 and A.19

2.5.2. Link Layer QEGP

Here we present an implementation of a link layer protocol, dubbed QEGP (quantum entanglement generation protocol), satisfying the service requirements put forth in

Section 2.4 (see appendix A.4 for details and message formats). We build up this protocol from different components:

Distributed Queue

Both nodes that wish to establish entangled link(s) must trigger their MHP devices in a coordinated fashion (Section 2.4.4). To achieve such agreement, the QEGP employs a distributed queue comprised of synchronized local queues at the controllable nodes. These local queues can be used to separate requests based on priority, where here we employ 3 queues for the different use cases (CK, NL, MD). Due to low errors in classical communication (estimated $< 4 \times 10^{-8}$ on QL2020, see appendix A.3.6), we let one node hold the master copy of the queue, and use a simple two-way handshake for enqueuing items, and a windowing mechanism to ensure fairness. Queue items include a *min_time* that specifies the earliest possible time a request is deemed ready for processing by both nodes (depending on their distance). Specifying *min_time* prevents either node from beginning entanglement generation in different timesteps. We note that while the distributed queue requires timing synchronization for such functionality, the timing constraints are looser than those found at the physical layer. Hence, sufficient synchronization may be obtained by piggy-backing on the mechanisms used at the physical layer, or by using PTP [91].

One may wonder why we employ a distributed queue to coordinate entanglement rather than utilizing classical discussion after entanglement has been generated. Recall from Section 2.4.5 that the memory lifetimes of qubits are very short. By agreeing on coordination in advance, we reduce the amount of noise introduced into the qubits before they are used by applications. An alternative design choice worthwhile exploring would be to employ the heralding midpoint as the master of the distributed queue. Such a construction may allow coordination of entanglement generation between several endnodes connected to a common midpoint station.

Quantum Memory Management (QMM)

The QEGP uses the node's QMM (Section 2.4.5) to determine which physical qubits to use for generating or storing entanglement.

Fidelity Estimation Unit (FEU)

In order to provide information about the quality of entanglement, the QEGP employs a fidelity estimation unit. This unit is given a desired quality parameter F_{\min} , and returns generation parameters (such as α) along with an estimated minimal completion time. Such a fidelity estimate can be calculated based on known hardware capabilities such as the quality of the memory and operations. To further improve this base estimate the QEGP intersperses test rounds.

Physical Translation Unit (PTU)

The link layer protocol processes CREATE requests in a hardware-independent manner. To resolve physical gate instructions that must be provided to the MHP and underlying platform, a physical translation unit that converts hardware-independent

instruction descriptions into hardware-dependent instructions is used. For example, the PTU may convert the Euler decomposition of a single-qubit gate or a pair of physical qubit ids for a two-qubit gate (such as moving the state of one to the other) into a sequence of physical instructions that should be issued to the hardware below. This unit also converts entanglement generation parameters like α supplied by the FEU into the corresponding physical instruction (here, a specific microwave pulse).

Scheduler

The QEGP scheduler decides which request in the queue should be served next. In principle, any scheduling strategy is suitable as long as it is deterministic, ensuring that both nodes select the same request locally. This limits two-way communication, which adversely affects entanglement quality due to limited memory lifetimes.

Protocol

Figure 2.5 presents an architecture diagram visualizing the operation. The protocol begins when a higher layer at a controllable node issues a CREATE operation to the QEGP specifying a desired number of entangled pairs along with F_{min} and t_{max} (Section 2.4.1). Upon receipt of a request the QEGP will query the FEU to obtain hardware parameters (α), and a minimum completion time (depending on α). If this time is larger than t_{max} , the QEGP immediately rejects the request (UNSUPP). Should the request pass this evaluation, the local node will compute a fitting *min_time* specifying the earliest MHP polling cycle the request may begin processing. The node then adds the request into the distributed queue shared by the nodes. This request may be rejected by the peer should the remote node have queue rules that do not accept the specified purpose ID. Then, the QEGP locally rejects the request (DENIED).

The local scheduler selects the next request to be processed, given that there exists a ready one (as indicated by *min_time*). The QMM is then used to allocate qubits needed to fulfill the specified request type (create and keep K or create and measure M). The QEGP will then again ask the FEU to obtain a current parameter α due to possible fluctuations in hardware parameters during the time spent in the queue. The scheduler then constructs a “yes” response to the MHP containing α from the FEU, along with an ID containing the unique queue ID of the request in the distributed queue, and number of pairs already produced for the request. This response is then forwarded to the local MHP upon its next poll to the QEGP. If no request is ready for processing, a “no” response is returned to the MHP. At this point the MHP behaves as described in the previous section and an attempt at generating entanglement is made.

Whenever a REPLY and ID is received from the MHP, the QEGP uses the ID to match the REPLY to an outstanding request, and evaluates the REPLY for correctness. Should the attempt be successful, the number of outstanding pairs in the request is decremented, and an OK message is propagated to higher layers containing the information specified in Section 2.4.1, where the Goodness is obtained from the FEU.

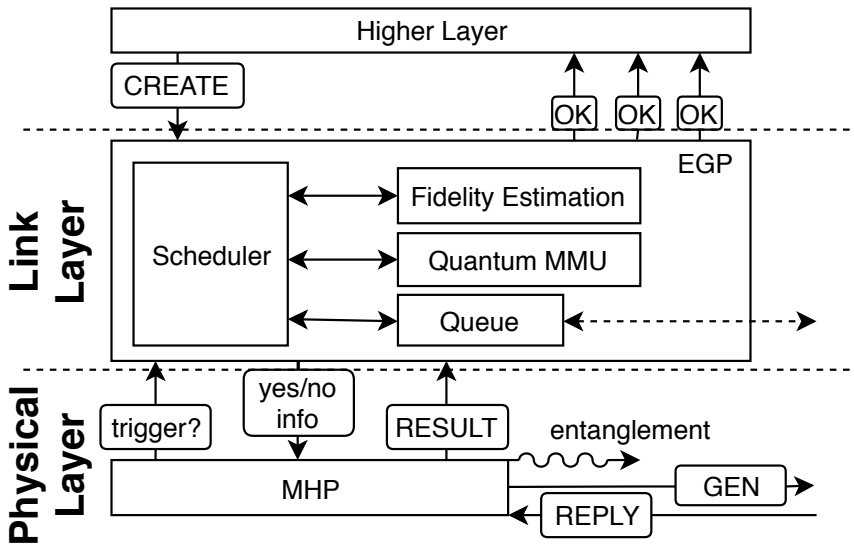


Figure 2.5: Flow diagram of the MHP and QEGP operation. The QEGP handles CREATE requests and schedules entanglement generation attempts are issued to the MHP. Replies from the midpoint are parsed and forwarded to the QEGP from request management.

In the appendix A, we consider a number of examples to illustrate decisions and possible pitfalls in the QEGP. One such example is the possibility of *emission multiplexing* [92]: The QEGP can be polled by the MHP before receiving a response from the MHP for the previous cycle. This allows the choice to attempt entanglement generation multiple times in succession before receiving a reply from the midpoint, e.g., in order to increase the throughput for the MD use case.

Errors such as losses on the classical control link can lead to an inconsistency of state (of the distributed queue) at A and B from which we need to recover. Inconsistencies can also affect the higher layer, e.g. with node A issuing an OK to higher layer, but not node B . Since the probability of e.g. control message losses is extremely low, we choose not to perform additional two-way discussion to further curb all inconsistencies at the link layer. Instead, the QEGP can issue an EXPIRE message for an OK already issued if inconsistency is detected later, e.g. when the remote node never received an OK for this pair.

2.6. Evaluation

We investigate the performance of our link layer protocol using a purpose built discrete event simulator for quantum networks (NetSquid [93], Python/C++) based on DynAA [94] (see appendix A.2 for details and more simulation results). Both the MHP and QEGP are implemented in full in Python, running on simulated nodes that have simulated versions of the quantum hardware components, fiber connections, etc. All simulations were performed on the supercomputer *Cartesius* at SURF-

sara [95], in a total of 2578 separate runs using a total of 94244 core hours, and 707 hours time in the simulation (~ 250 billion MHP cycles). One simulated second currently takes about two core minutes on average, since in each entanglement generation attempt (every $10.12 \mu\text{s}$ for type MD) multiple events are scheduled and handled and the 16×16 -matrix representing the state of the two photons and electrons is updated based on multiple sources of noise and gate operations. The code used for the simulation can be found at [96] and complete data at [97].

We conduct the following simulation runs:

- Long runs: To study robustness of our protocol, we simulate the 169 scenarios described below for an extended period of time. Each scenario was simulated twice for 120 wall time hours, simulating 502 – 13437 seconds. We present and analyze the data from these runs in sections 2.6.1, 2.6.2 and appendix A.2.2.
- Short runs: We perform the following simulations for a shorter period of time (24 wall time hours, reaching 67 – 2356 simulated seconds):
 - Performance trade-offs: To study the trade-off between latency, throughput and fidelity we sweep the incoming request frequency and the requested minimum fidelity, see Figure 2.6.
 - Metric fluctuations: To be able to study the impact of different scheduling strategies on the performance metrics, we run 4 scenarios, 102 times each. The outcomes of these simulation runs are discussed in section 2.6.3.

To explore the performance at both short and long distances, the underlying hardware model is based on the Lab and QL2020 scenarios, where we validate the physical modeling of the simulation against data collected from the quantum hardware system of the Lab scenario already realized (Figure 2.8). For the quantum reader we note that while our simulations can also be used to predict behavior of physical implementations (such as QL2020), the focus here is on the performance and behavior of the link layer protocol. This includes a first investigation of how different scheduling strategies can affect traditional performance metrics (such as throughput) in relation of genuinely quantum ones (the fidelity) for different use cases.

We structure the evaluation along the three different use cases (NL, CK, MD), leading to a total of 169 different simulation scenarios. First, we use different kinds of requests:

- *NL* (network layer): K type request, consecutive flag, priority 1 (highest), store qubit in memory qubit.
- *CK* (create and keep) an application asking for one or more long-lived pairs: K type request, immediate return flag, priority 2 (high), store qubit in memory qubit.
- *MD* (measure directly): M type request, consecutive flag, priority 3 (lowest).

For an application such as QKD, one would not set the immediate return flag in practice for efficiency, but we do so here to ease comparison to the other two scenarios. Measurements in *MD* are performed in randomly chosen bases X , Z and Y (see section 1.4).

In each MHP cycle, we randomly issue a new CREATE request for a random number of pairs k (max k_{\max}), and random use case $P \in \{NL, CK, MD\}$ with probability $f_P \cdot p_{\text{succ}}/(E \cdot k)$, where p_{succ} is the probability of one attempt succeeding (Section 2.4.4), f_P is a fraction determining the load in our system of kind P , and E is the expected number of MHP cycles to make one attempt ($E = 1$ for MD and $E \approx 1.1$ for NL/CK in Lab due to memory re-initialization and post-processing; $E \approx 16$ for NL/CK in QL2020 due to classical communication delays with H ($145\mu\text{s}$)). This probability is chosen in a way such that the queue (without a max size) storing these requests is stable (finite expected length) for $\sum_{P \in \{NL, CK, MD\}} f_P < 1$ and otherwise unstable (infinite expected length). In the long runs, we first study single kinds of requests (only one of MD/CK/NL), with $f_P = 0.7$ (Low), 0.99 (High) or 1.5 (Ultra). For these long runs, we fix one target fidelity $F_{\min} = 0.64$ to ease comparison. For each of the 3 kinds (MD/CK/NL), we examine:

- $k_{\max} = 1$, one pair per request.
- $k_{\max} = 3$, one to three pairs per request.
- $k_{\max} = 255$, one to 255 pairs per request (only for MD).

For Ultra the number of outstanding requests intentionally grows until the queue is full (max 256), to study an overload of our system.

To study fairness, we take 3 cases of who issues the CREATE for each single kind (MD/CK/NL) scenario:

- all from A (master of the distributed queue).
- all from B .
- A or B are randomly chosen with equal probability.

To examine scheduling, we additionally consider long runs with mixed kinds of requests (appendix A.2.2, e.g. Figure 2.7).

2.6.1. Robustness

To study robustness, we artificially increase the probability of losing classical control messages (100 Base T on QL2020 fiber $< 4 \times 10^{-8}$ (see appendix A.3.6)), which can lead to an inconsistency of state of the QEGP, but also higher layers (Section 2.5.2). These classical control messages are part of both the MHP protocol and QEGP protocol including the distributed queue. We ramp up loss probabilities up to 10^{-4} (see appendix A.2.2) and observe our recovery mechanisms work to ensure stable execution in all cases (35 runs, 281 - 3973 s simulated time), with only small impact to the performance parameters (maximum relative differences ² to the case of no

²Relative difference between m_1 and m_2 is $|m_1 - m_2|/\max(|m_1|, |m_2|)$

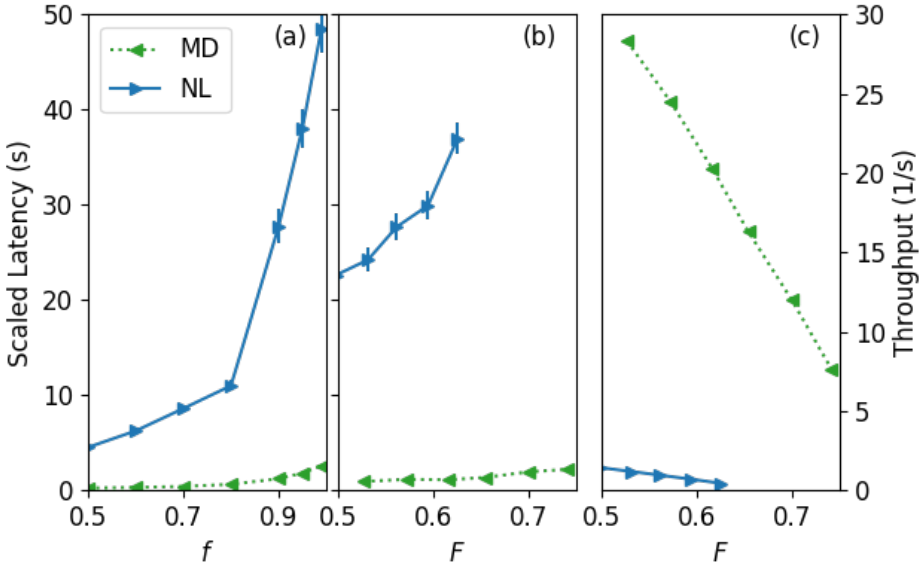


Figure 2.6: Performance trade-offs. (a) Scaled latency vs. f_p determining fraction of throughput (b) Scaled latency vs. fidelity F_{min} . Demanding a higher F_{min} lowers the probability of success (Section 2.4.4), meaning (c) throughput directly scales with F_{min} (each point averaged over 40 short runs each 24 h, 93 – 2355 s simulated time, QL2020, $k_{max} = 3$, for (b,c) $f_p = 0.99$). Higher F_{min} not possible for NL in (b).

losses, fidelity (0.005), throughput (0.027), latency (0.629), number of OKs (0.026) with no EXPIRE messages). We see a relatively large difference for latency, which may however be due to latency not reaching steady state during the simulation (70×70 core hours), see the next section.

2.6.2. Performance Metrics

We first consider runs including only a single kind of request (MD/CK/NL). In addition to the long runs, we conduct specific short runs examining the trade-off between latency and throughput for fixed target fidelity F_{min} (Figure 2.6(a)), and the trade-off between latency (throughput) and the target fidelity in Figure 2.6(b) (Figure 2.6(c)). As described in section 2.4.4, the probability of successful entanglement generation, and therefore throughput, is directly proportional to one minus fidelity of the generated pair.

Below we present the metrics extracted from the long runs with single kinds of requests:

Fidelity: As a benchmark, we began by recording the average fidelity F_{avg} in all 169 scenarios with fixed minimum fidelity. We observe that F_{avg} is independent of the other metrics but does depend on the distance, and whether we store or measure: $0.745 < F_{\text{avg}} < 0.757$ (NL/CK Lab), $0.626 < F_{\text{avg}} < 0.653$ (NL/CK QL2020), $0.709 < F_{\text{avg}} < 0.779$ (MD Lab), $0.723 < F_{\text{avg}} < 0.767$ (MD QL2020) (Fidelity MD extracted from QBER measurements, see section 1.4). This is to be expected since (1) we fix one F_{min} and (2) we consider an NV platform with only 1 available memory qubit so no change in quality is observed by using different memory qubits (Lab).

Throughput: All scenarios High and Ultra in Lab reach an average throughput th_{avg} (1/s) of $6.05 < th_{\text{avg}} < 6.47$ NL/CK and $6.51 < th_{\text{avg}} < 7.09$ for MD. It is expected that MD has higher throughput, since no memory qubit needs to be initialized. The time to move to memory ($1040\mu\text{s}$) is less significant since many MHP cycles are needed to produce one pair, but we only move once. As expected for Low the throughput is slightly lower in all cases, $4.44 < th_{\text{avg}} < 4.72$ NL/CK, and $4.86 < th_{\text{avg}} < 5.22$ MD. For QL2020, the throughput for NL/CK is about 14 times lower, since we need to wait ($145\mu\text{s}$) for a reply from H before MHP can make a new attempt.

Latency: The scaled latency highly depends on the incoming request frequency as the longer queue causes higher latency. However, from running the same scenarios many (102) times for a shorter period (24 wall time hours) (see Section 2.6.3), we see that the average scaled latency fluctuates a lot, with a standard deviation of up to 6.6 s in some cases. For QL2020 with NL requests specifying 1-3 pairs from both nodes, we observe an average scaled latency of 10.97 s Low, 142.9 s High and 521.5 s Ultra. For MD requests, 0.544 s Low, 3.318 s High and 32.34 s Ultra. The longer scaled latency for NL is largely due to longer time needed to create a pair, and not that the queues are longer (average queue length for NL: 3.83 Low, 56.3 High, 214 Ultra), and for MD: 3.23 Low, 22.4 High and 219 Ultra).

Fairness: For 103 scenarios of the long runs (single kinds of requests (MD/CK-/NL) randomly from A and B), we see only slight differences in fidelity, throughput or latency between requests from A and B . Maximum relative differences do not exceed: fidelity 0.033, throughput 0.100, latency 0.073, number of OKs 0.100 (for Ultra).

2.6.3. Scheduling

We take a first step studying the effect of scheduling strategies on the performance when using mixed kinds of requests. Part of simulating the performance of a scheduling strategies can certainly be done without implementing all details of the physical entanglement generation. However, since we do simulate these details we can first confirm that different scheduling strategies below do not affect the average fidelity in these scenarios. Here, we examine two simple scheduling strategies: (1) first-come-first-serve (FCFS) and (2) a strategy where NL (priority 1) has a strict highest priority, and use a weighted fair queue (WFQ) for CK (priority 2) and MD (priority 3), where CK has 10 times the weight of MD. With these scheduling strategies, we simulate two different request patterns ((i) uniform and (ii) no NL more MD), 102 times over 24 wall time hours each and extract the performance

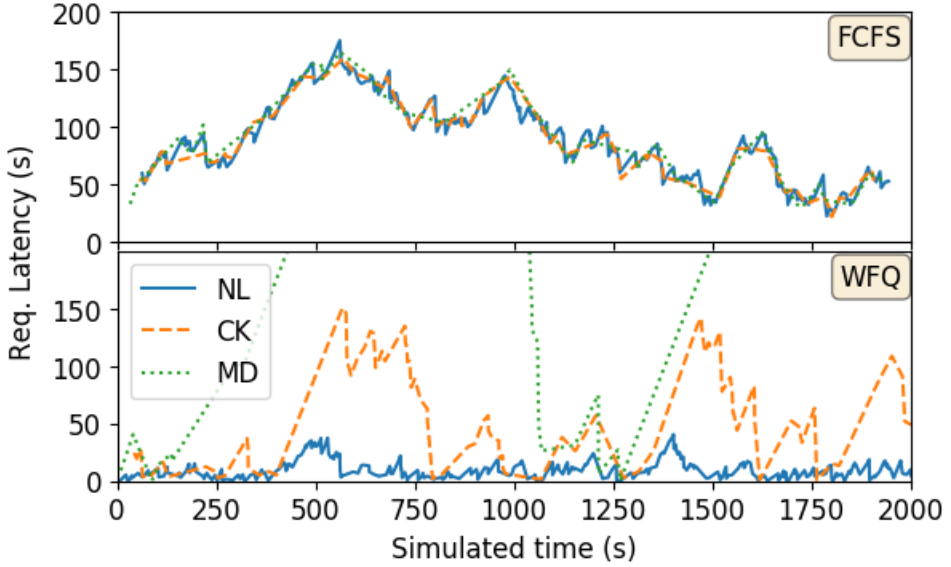


Figure 2.7: Request latency vs. time for two scheduling scenarios (long runs simulated 120 h wall time). As expected the max. latency for *NL* is decreased due to strict priority. In this scenario, there are more incoming *NL* requests ($f_{NL} = 0.99 \cdot 4/5$, $f_{CK} = 0.99 \cdot 1/5$ and $f_{MD} = 0.99 \cdot 1/5$).

metrics of throughput and scaled request latency (Table 2.1).

As expected we see a drastic decrease of the average scaled latency for *NL* when giving it strict priority: 10.3 s with FCFS and 3.5 s with WFQ. For *CK* there is similarly a decrease in average scaled latency, however smaller than for *NL*, of 10.1 s (FCFS) and 6.5 s (WFQ). For *MD* the average scaled latency goes up in both cases when using WFQ instead of FCFS, by factors of 2.49 (uniform) and 1.28 (no *NL* more *MD*).

We observe that the throughput gets less affected by the scheduling strategy than the latency for these scenarios. The maximal difference between the throughput for FCFS and WFQ is by a factor of 1.16 (for *MD* in the scenario of no *NL* and more *MD*). Furthermore, we see that the total throughput for all requests goes down from 2.75 (5.99) 1/s for FCFS to 2.44 (5.92) 1/s for WFQ in the case of uniform (no *NL* more *MD*).

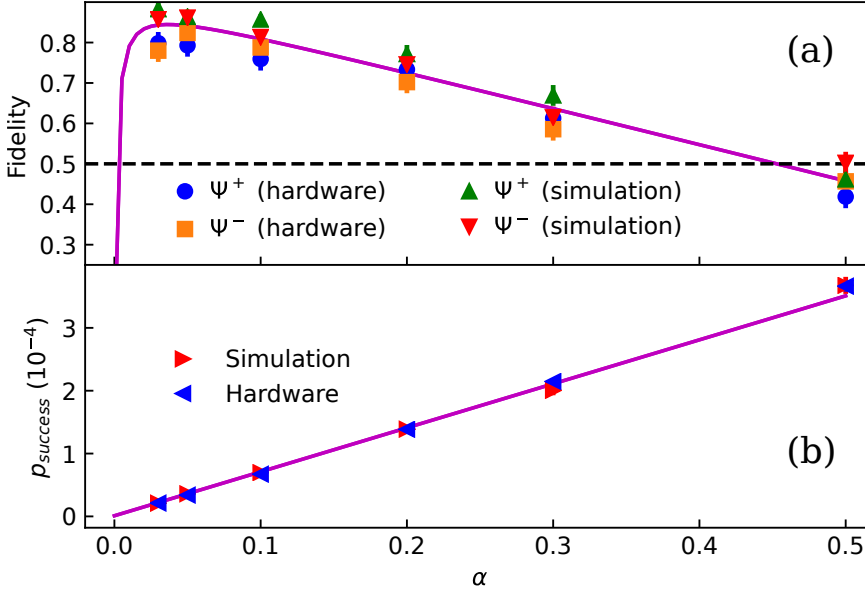


Figure 2.8: Validation against data from NV hardware (Lab scenario). Fidelity (a) and probability an attempt succeeds (b) in terms of α (Section 2.4.4) shows good agreement between hardware and simulation points (each at least 300 pairs averaged, 5s–117s simulated time, 500k–10.000k attempts, 122 hours wall time). Theoretical model [23] as visual guide (solid line).

Table 2.1: Throughput (T) and scaled latency (SL) using scheduling strategies FCFS and WFQ for two request patterns: (i) with $f_{NL} = f_{CK} = f_{MD} = 0.99 \cdot 1/3$, i.e. a uniform load of the different priorities and (ii) with $f_{NL} = 0$, $f_{CK} = 0.99 \cdot 1/3$ and $f_{MD} = 0.99 \cdot 4/5$, i.e. no *NL* and more *MD*. The physical setup: QL2020 and number of pairs per request: 2 (*NL*), 2 (*CK*), and 10 (*MD*). Each value average over 102 short runs each 24 h, with standard error in parentheses.

| T (1/s) | <i>NL</i> | <i>CK</i> | <i>MD</i> |
|-----------|---------------|---------------|---------------|
| (i) FCFS | 0.146 (0.003) | 0.144 (0.003) | 2.464 (0.056) |
| (i) WFQ | 0.154 (0.003) | 0.156 (0.003) | 2.130 (0.063) |
| (ii) FCFS | - | 0.086 (0.003) | 5.912 (0.033) |
| (ii) WFQ | - | 0.096 (0.003) | 5.829 (0.049) |

| SL (s) | <i>NL</i> | <i>CK</i> | <i>MD</i> |
|-----------|----------------|----------------|---------------|
| (i) FCFS | 10.272 (0.654) | 10.063 (0.631) | 1.740 (0.120) |
| (i) WFQ | 3.520 (0.085) | 6.548 (0.361) | 4.331 (0.336) |
| (ii) FCFS | - | 5.659 (0.313) | 0.935 (0.062) |
| (ii) WFQ | - | 2.503 (0.100) | 1.194 (0.093) |

2.7. Conclusion

Our top down inventory of design requirements, combined with a bottom up approach based on actual quantum hardware allowed us to take quantum networks a step further on the long path towards their large-scale realization. Our work readies QL2020, and paves the way towards the next step, a robust network layer control protocol. The link layer may now be used as a robust service without detailed knowledge of the physics of the devices. Due to the relatively small size of initial quantum networks, close attention was paid to application use cases even at the link layer. We expect that in the future, the network layer will have a similar interface to higher layers as the link layer itself, and nodes internal to the network will not run applications themselves. Scheduling strategies catering to different use cases may at this stage be applied primarily at the network layer at the level of long-distance links, which are then directly passed to applications running at the end nodes requesting long-distance entanglement. We expect that at the network layer, and when considering larger quantum memories, smart scheduling strategies will be important not only to combat memory lifetimes but also to coordinate actions of different nodes in time, calling for significant effort in computer science and engineering.

Acknowledgment

We thank Kenneth Goodenough for comments on earlier drafts. This work was supported by ERC Starting Grant (SW), ERC Consolidator Grant (RH), EU Flagship on Quantum Technologies, Quantum Internet Alliance (No. 820445), NWO VIDI (SW), and Marie Skłodowska-Curie action Spin-NANO (No. 676108). The research in this chapter poses no ethical issues.

References

- [1] A. Dahlberg, M. Skrzypczyk, T. Coopmans, L. Wubben, F. Rozpędek, M. Pompili, A. Stolk, P. Pawełczak, R. Knegjens, J. de Oliveria Filho, R. Hanson, and S. Wehner, *A link layer protocol for quantum networks*, in [ACM SIGCOMM 2019 Conference](#), SIGCOMM '19 (ACM, New York, NY, USA, 2019) p. 15.
- [2] C. H. Bennett, G. Brassard, C. Crépeau, R. Jozsa, A. Peres, and W. K. Wootters, *Teleporting an unknown quantum state via dual classical and einstein-podolsky-rosen channels*, *Physical Review Letters* **70**, 1895 (1993).
- [3] M. Zukowski, A. Zeilinger, M. A. Horne, and A. K. Ekert, "event-ready-detectors" bell experiment via entanglement swapping, *Physical Review Letters* **71**, 4287 (1993).
- [4] S. Wehner, D. Elkouss, and R. Hanson, *Quantum internet: A vision for the road ahead*, *Science* **362**, eaam9288 (2018).
- [5] H. J. Kimble, *The quantum internet*, *Nature* **453**, 1023 (2008).

- [6] C. H. Bennett and G. Brassard, *Quantum Cryptography: Public Key Distribution, and Coin-Tossing*, in *Proc. 1984 IEEE International Conference on Computers, Systems, and Signal Processing* (1984) pp. 175–179.
- [7] A. K. Ekert, *Quantum cryptography based on bell's theorem*, *Physical Review Letters* **67**, 661 (1991).
- [8] D. Gottesman, T. Jennewein, and S. Croke, *Longer-baseline telescopes using quantum repeaters*, *Physical Review Letters* **109**, 070503 (2012).
- [9] P. Komar, E. M. Kessler, M. Bishof, L. Jiang, A. S. Sørensen, J. Ye, and M. D. Lukin, *A quantum network of clocks*, *Nature Physics* **10**, 582 (2014).
- [10] M. Ben-Or and A. Hassidim, *Fast quantum byzantine agreement*, in *Proceedings of the Thirty-seventh Annual ACM Symposium on Theory of Computing*, STOC '05 (ACM, New York, NY, USA, 2005) pp. 481–485.
- [11] V. S. Denchev and G. Pandurangan, *Distributed quantum computing: A new frontier in distributed systems or science fiction?* *ACM SIGACT News* **39**, 77 (2008).
- [12] A. Broadbent, J. Fitzsimons, and E. Kashefi, *Universal blind quantum computation*, in *Proceedings of the 2009 50th Annual IEEE Symposium on Foundations of Computer Science*, FOCS '09 (IEEE, Washington, DC, USA, 2009) pp. 517–526, [arXiv:0807.4154](https://arxiv.org/abs/0807.4154) .
- [13] A. M. Childs, *Secure assisted quantum computation*, *Quantum Info. Comput.* **5**, 456 (2005), [arXiv:0111046 \[quant-ph\]](https://arxiv.org/abs/0111046) .
- [14] J. F. Dynes, H. Takesue, Z. L. Yuan, A. W. Sharpe, K.-I. Harada, T. Honjo, H. Kamada, O. Tadanaga, Y. Nishida, M. Asobe, and A. J. Shields, *Efficient entanglement distribution over 200 kilometers*, *Optics express* **17**, 11440 (2009).
- [15] T. Inagaki, N. Matsuda, O. Tadanaga, M. Asobe, and H. Takesue, *Entanglement distribution over 300 km of fiber*, *Optics express* **21**, 23241 (2013).
- [16] J. Yin, Y. Cao, Y.-H. Li, S.-K. Liao, L. Zhang, J.-G. Ren, W.-Q. Cai, W.-Y. Liu, B. Li, H. Dai, G.-B. Li, Q.-M. Lu, Y.-H. Gong, Y. Xu, S.-L. Li, F.-Z. Li, Y.-Y. Yin, Z.-Q. Jiang, M. Li, J.-J. Jia, G. Ren, D. He, Y.-L. Zhou, X.-X. Zhang, N. Wang, X. Chang, Z.-C. Zhu, N.-L. Liu, Y.-A. Chen, C.-Y. Lu, R. Shu, C.-Z. Peng, J.-Y. Wang, and J.-W. Pan, *Satellite-based entanglement distribution over 1200 kilometers*, *Science* **356**, 1140 (2017), [arXiv:1707.01339\[quant-ph\]](https://arxiv.org/abs/1707.01339) .
- [17] R. Alléaume, C. Branciard, J. Bouda, T. Debuisschert, M. Dianati, N. Gisin, M. Godfrey, P. Grangier, T. Langer, N. Lutkenhaus, C. Monyk, P. Painchault, M. Peev, A. Poppe, T. Pornin, J. Rarity, R. Renner, G. Ribordy, M. Riguidel, L. Salvail, A. Shields, H. Weinfurter, and A. Zeilinger, *Using quantum key distribution for cryptographic purposes: a survey*, *Theoretical Computer Science* **560**, 62 (2014).

- [18] C. Elliott, D. Pearson, and G. Troxel, *Quantum cryptography in practice*, in *Proceedings of the 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, SIGCOMM '03 (ACM, New York, NY, USA, 2003) pp. 227–238.
- [19] Q. Xchange, *Quantum Xchange*, <https://quantumxc.com> (2019).
- [20] B. Hensen, H. Bernien, A. E. Dréau, A. Reiserer, N. Kalb, M. S. Blok, J. Ruitenbergh, R. F. L. Vermeulen, R. N. Schouten, C. Abellán, W. Amaya, V. Pruneri, M. W. Mitchell, M. Markham, D. J. Twitchen, D. Elkouss, S. Wehner, T. H. Taminiau, and R. Hanson, *Loophole-free Bell inequality violation using electron spins separated by 1.3 kilometres*, *Nature* **526**, 682 (2015), [arXiv:1508.05949](https://arxiv.org/abs/1508.05949).
- [21] C. E. Bradley, J. Randall, M. H. Abobeih, R. Berrevoets, M. Degen, M. A. Bakker, R. F. L. Vermeulen, M. Markham, D. J. Twitchen, and T. H. Taminiau, *A 10-qubit solid-state spin register with quantum memory up to one minute*, [arXiv preprint \(2019\)](https://arxiv.org/abs/1905.02094), [arXiv:1905.02094 \[quant-ph\]](https://arxiv.org/abs/1905.02094).
- [22] M. H. Abobeih, J. Cramer, M. A. Bakker, N. Kalb, M. Markham, D. J. Twitchen, and T. H. Taminiau, *One-second coherence for a single electron spin coupled to a multi-qubit nuclear-spin environment*, *Nature Communications* **9**, 2552 (2018), [arXiv:1801.01196](https://arxiv.org/abs/1801.01196).
- [23] P. C. Humphreys, N. Kalb, J. P. Morits, R. N. Schouten, R. F. Vermeulen, D. J. Twitchen, M. Markham, and R. Hanson, *Deterministic delivery of remote entanglement on a quantum network*, *Nature* **558**, 268 (2018).
- [24] W. Pfaff, B. J. Hensen, H. Bernien, S. B. van Dam, M. S. Blok, T. H. Taminiau, M. J. Tiggelman, R. N. Schouten, M. Markham, D. J. Twitchen, and R. Hanson, *Unconditional quantum teleportation between distant solid-state quantum bits*, *Science* **345**, 532 (2014), [arXiv:1404.4369](https://arxiv.org/abs/1404.4369).
- [25] D. L. Moehring, P. Maunz, S. Olmschenk, K. C. Younge, D. N. Matsukevich, L. Duan, and C. Monroe, *Entanglement of single-atom quantum bits at a distance*, *Nature* **449**, 68 (2007).
- [26] J. Hofmann, M. Krug, N. Ortegel, L. Gérard, M. Weber, W. Rosenfeld, and H. Weinfurter, *Heralded entanglement between widely separated atoms*, *Science* **337**, 72 (2012).
- [27] A. Delteil, Z. Sun, W.-b. Gao, E. Togan, S. Faelt, and A. Imamoglu, *Generation of heralded entanglement between distant hole spins*, *Nature Physics* **12**, 218 (2016), [arXiv:1507.00465](https://arxiv.org/abs/1507.00465).
- [28] R. Valivarthi, M. G. Puigibert, Q. Zhou, G. H. Aguilar, V. B. Verma, F. Marsili, M. D. Shaw, S. W. Nam, D. Oblak, and W. Tittel, *Quantum teleportation across a metropolitan fibre network*, *Nature Photonics* **10**, 676 (2016), [arXiv:1605.08814](https://arxiv.org/abs/1605.08814).

- [29] B. Julsgaard, A. Kozhekin, and E. S. Polzik, *Experimental long-lived entanglement of two macroscopic objects*, *Nature* **413**, 400 (2001).
- [30] C.-W. Chou, H. de Riedmatten, D. Felinto, S. V. Polyakov, S. J. Van Enk, and H. J. Kimble, *Measurement-induced entanglement for excitation stored in remote atomic ensembles*, *Nature* **438**, 828 (2005).
- [31] A. Narla, S. Shankar, M. Hatridge, Z. Leghtas, K. M. Sliwa, E. Zalys-Geller, S. O. Mundhada, W. Pfaff, L. Frunzio, R. J. Schoelkopf, and M. H. Devoret, *Robust concurrent remote entanglement between two superconducting qubits*, *Physical Review X* **6**, 031036 (2016).
- [32] N. Sangouard, C. Simon, H. De Riedmatten, and N. Gisin, *Quantum repeaters based on atomic ensembles and linear optics*, *Reviews of Modern Physics* **83**, 33 (2011).
- [33] R. Van Meter and J. Touch, *Designing quantum repeater networks*, *IEEE Communications Magazine* **51**, 64 (2013).
- [34] R. Van Meter, *Quantum networking and internetworking*, *IEEE Network* **26**, 59 (2012).
- [35] R. Van Meter, T. D. Ladd, W. J. Munro, and K. Nemoto, *System design for a long-line quantum repeater*, *IEEE/ACM Transactions on Networking* **17**, 1002 (2009), [arXiv:0705.4128](https://arxiv.org/abs/0705.4128).
- [36] L. Aparicio, R. Van Meter, and H. Esaki, *Protocol design for quantum repeater networks*, in *Proceedings of the 7th Asian Internet Engineering Conference*, AINTEC '11 (ACM, New York, NY, USA, 2011) pp. 73–80.
- [37] A. Pirker and W. Dür, *A quantum network stack and protocols for reliable entanglement-based networks*, *New Journal of Physics* **21**, 033003 (2019).
- [38] S. Lloyd, J. H. Shapiro, F. N. C. Wong, P. Kumar, S. M. Shahriar, and H. P. Yuen, *Infrastructure for the quantum internet*, *SIGCOMM Comput. Commun. Rev.* **34**, 9 (2004).
- [39] B. Liu, B. Zhao, Z. Wei, C. Wu, J. Su, W. Yu, F. Wang, and S. Sun, *Qphone: A quantum security voip phone*, in *Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM*, SIGCOMM '13 (ACM, New York, NY, USA, 2013) pp. 477–478.
- [40] W. Yu, B. Zhao, and Z. Yan, *Software defined quantum key distribution network*, *2017 3rd IEEE International Conference on Computer and Communications, ICC 2017 2018-Janua*, 1293 (2018).
- [41] A. Aguado, E. Hugues-Salas, P. A. Haigh, J. Marhuenda, A. B. Price, P. Sibson, J. E. Kennard, C. Erven, J. G. Rarity, M. G. Thompson, A. Lord, R. Nejabati, and D. Simeonidou, *Secure nfv orchestration over an sdn-controlled optical network with time-shared quantum key distribution resources*, *J. Lightwave Technol.* **35**, 1357 (2017).

- [42] A. Bremler-Barr, Y. Harchol, and D. Hay, *Openbox: A software-defined framework for developing, deploying, and managing network functions*, in *Proceedings of the 2016 ACM SIGCOMM Conference*, SIGCOMM '16 (ACM, New York, NY, USA, 2016) pp. 511–524.
- [43] M. Handley, C. Raiciu, A. Agache, A. Voinescu, A. W. Moore, G. Antichi, and M. Wójcik, *Re-architecting datacenter networks and stacks for low latency and high performance*, in *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, SIGCOMM '17 (ACM, New York, NY, USA, 2017) pp. 29–42.
- [44] F. Chen, R. K. Sitaraman, and M. Torres, *End-user mapping: Next generation request routing for content delivery*, in *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, SIGCOMM '15 (ACM, New York, NY, USA, 2015) pp. 167–181.
- [45] L. Zheng, C. Joe-Wong, C. W. Tan, M. Chiang, and X. Wang, *How to bid the cloud*, in *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, SIGCOMM '15 (ACM, New York, NY, USA, 2015) pp. 71–84.
- [46] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*, 10th ed. (Cambridge University Press, Cambridge, 2010).
- [47] R. Van Meter, *Quantum Networking*, 1st ed. (John Wiley & Sons, Ltd, Hoboken, NJ, USA, 2014).
- [48] J. Strand, A. L. Chiu, and R. Tkach, *Issues for routing in the optical layer*, *IEEE Comm. Mag.* **39**, 81 (2001).
- [49] H. J. Briegel, W. Dür, J. I. Cirac, and P. Zoller, *Quantum repeaters: the role of imperfect local operations in quantum communication*, *Physical Review Letters* **81**, 5932 (1998).
- [50] W. J. Munro, K. Azuma, K. Tamaki, and K. Nemoto, *Inside quantum repeaters*, *IEEE Journal of Selected Topics in Quantum Electronics* **21**, 78 (2015).
- [51] W. J. Munro, A. M. Stephens, S. J. Devitt, K. A. Harrison, and K. Nemoto, *Quantum communication without the necessity of quantum memories*, *Nature Photonics* **6**, 777 (2012).
- [52] W.-B. Gao, C.-Y. Lu, X.-C. Yao, P. Xu, O. Gühne, A. Goebel, Y.-A. Chen, C.-Z. Peng, Z.-B. Chen, and J.-W. Pan, *Experimental demonstration of a hyper-entangled ten-qubit schrödinger cat state*, *Nature physics* **6**, 331 (2010).
- [53] S. Muralidharan, J. Kim, N. Lütkenhaus, M. D. Lukin, and L. Jiang, *Ultrafast and fault-tolerant quantum communication across long distances*, *Physical Review Letters* **112**, 250501 (2014).

- [54] S. D. Barrett and P. Kok, *Efficient high-fidelity quantum computation using matter qubits and linear optics*, *Physical Review A* **71**, 060310 (2005).
- [55] C. Cibrillo, J. I. Cirac, P. Garcia-Fernandez, and P. Zoller, *Creation of entangled states of distant atoms by interference*, *Physical Review A* **59**, 1025 (1999).
- [56] D. D. Awschalom, R. Hanson, J. Wrachtrup, and B. B. Zhou, *Quantum technologies with optically interfaced solid-state spins*, *Nature Photonics* **12**, 516 (2018).
- [57] K. Mattle, H. Weinfurter, P. G. Kwiat, and A. Zeilinger, *Dense coding in experimental quantum communication*, *Physical Review Letters* **76**, 4656 (1996).
- [58] J. Brendel, N. Gisin, W. Tittel, and H. Zbinden, *Pulsed energy-time entangled twin-photon source for quantum communication*, *Physical Review Letters* **82**, 2594 (1999).
- [59] H. Bernien, B. Hensen, W. Pfaff, G. Koolstra, M. S. Blok, L. Robledo, T. Tamniiau, M. Markham, D. J. Twitchen, L. Childress, and R. Hanson, *Heralded entanglement between solid-state qubits separated by three metres*, *Nature* **497**, 86 (2013).
- [60] B. B. Blinov, D. L. Moehring, L. Duan, and C. Monroe, *Observation of entanglement between a single trapped atom and a single photon*, *Nature* **428**, 153 (2004).
- [61] S. Ritter, C. Nölleke, C. Hahn, A. Reiserer, A. Neuzner, M. Uphoff, M. Mücke, E. Figueroa, J. Bochmann, and G. Rempe, *An elementary quantum network of single atoms in optical cavities*, *Nature* **484**, 195 (2012).
- [62] K. Nemoto, M. Trupke, S. J. Devitt, B. Scharfenberger, K. Buczak, J. Schmiedmayer, and W. J. Munro, *Photonic quantum networks formed from *nv*-centers*, *Scientific reports* **6**, 26284 (2016).
- [63] N. Kalb, A. Reiserer, S. Ritter, and G. Rempe, *Heralded storage of a photonic quantum bit in a single atom*, *Physical Review Letters* **114**, 220501 (2015).
- [64] I. Damgård, S. Fehr, L. Salvail, and C. Schaffner, *Secure identification and qkd in the bounded-quantum-storage model*, *Theoretical Computer Science* **560**, 12 (2014), [arXiv:0708.2557](https://arxiv.org/abs/0708.2557) .
- [65] A. Chailloux and I. Kerenidis, *Optimal bounds for quantum bit commitment*, in *Proceedings of the 2011 IEEE 52Nd Annual Symposium on Foundations of Computer Science*, FOCS '11 (IEEE Computer Society, Washington, DC, USA, 2011) pp. 354–362.
- [66] D. Aharonov, A. Ta-Shma, U. V. Vazirani, and A. C. Yao, *Quantum bit escrow*, in *Proceedings of the Thirty-second Annual ACM Symposium on Theory of Computing*, STOC '00 (ACM, New York, NY, USA, 2000) pp. 705–714.

- [67] I. B. Damgård, S. Fehr, L. Salvail, and C. Schaffner, *Cryptography in the bounded-quantum-storage model*, *SIAM Journal on Computing* **37**, 1865 (2008).
- [68] S. Wehner, C. Schaffner, and B. M. Terhal, *Cryptography from noisy storage*, *Physical Review Letters* **100**, 220502 (2008).
- [69] J. Ribeiro and F. Grosshans, *A tight lower bound for the bb84-states quantum-position-verification protocol*, *arXiv preprint (2015)*, [arXiv:1504.07171 \[quant-ph\]](https://arxiv.org/abs/1504.07171) .
- [70] V. Scarani, H. Bechmann-Pasquinucci, N. J. Cerf, M. Dušek, N. Lütkenhaus, and M. Peev, *The security of practical quantum key distribution*, *Reviews of modern physics* **81**, 1301 (2009).
- [71] Y. J. Liu, P. X. Gao, B. Wong, and S. Keshav, *Quartz: A new design element for low-latency dcns*, in *Proceedings of the 2014 ACM Conference on SIGCOMM*, SIGCOMM '14 (ACM, New York, NY, USA, 2014) pp. 283–294.
- [72] M. Horodecki and P. Horodecki, *Reduction criterion of separability and limits for a class of distillation protocols*, *Physical Review A* **59**, 4206 (1999).
- [73] W. Dür and H. J. Briegel, *Entanglement purification and quantum error correction*, *Reports on Progress in Physics* **70**, 1381 (2007).
- [74] N. Kalb, A. A. Reiserer, P. C. Humphreys, J. J. W. Bakermans, S. J. Kamerling, N. H. Nickerson, S. C. Benjamin, D. J. Twitchen, M. Markham, and R. Hanson, *Entanglement distillation between solid-state quantum network nodes*, *Science* **356**, 928 (2017), [arXiv:1703.03244](https://arxiv.org/abs/1703.03244) .
- [75] I. Marinos, R. N. M. Watson, and M. Handley, *Network stack specialization for performance*, in *Proceedings of the Twelfth ACM Workshop on Hot Topics in Networks*, HotNets-XII (ACM, New York, NY, USA, 2013) pp. 9:1–9:7.
- [76] R. Singh, M. Ghobadi, K.-T. Foerster, M. Filer, and P. Gill, *Radwan: Rate adaptive wide area network*, in *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*, SIGCOMM '18 (ACM, New York, NY, USA, 2018) pp. 547–560.
- [77] F. R. Dogar, T. Karagiannis, H. Ballani, and A. Rowstron, *Decentralized task-aware scheduling for data center networks*, in *Proceedings of the 2014 ACM Conference on SIGCOMM*, SIGCOMM '14 (ACM, New York, NY, USA, 2014) pp. 431–442.
- [78] B. M. Terhal, *Quantum error correction for quantum memories*, *Reviews of Modern Physics* **87**, 307 (2015).
- [79] J. Cramer, N. Kalb, M. A. Rol, B. Hensen, M. S. Blok, M. Markham, D. J. Twitchen, R. Hanson, and T. H. Taminiau, *Repeated quantum error correction on a continuously encoded qubit by real-time feedback*, *Nature communications* **7**, 11526 (2016).

- [80] D. Riste, S. Poletto, M. Huang, A. Bruno, V. Vesterinen, O.-P. Saira, and L. DiCarlo, *Detecting bit-flip errors in a logical qubit using stabilizer measurements*, *Nature communications* **6**, 6983 (2015).
- [81] A. D. Córcoles, E. Magesan, S. J. Srinivasan, A. W. Cross, M. Steffen, J. M. Gambetta, and J. M. Chow, *Demonstration of a quantum error detection code using a square lattice of four superconducting qubits*, *Nature communications* **6**, 6979 (2015).
- [82] K. Azuma, K. Tamaki, and H.-K. Lo, *All-photonic quantum repeaters*, *Nature Communications* **6**, 6787 (2015).
- [83] L. Childress and R. Hanson, *Diamond *nv* centers for quantum computing and quantum networks*, *MRS Bulletin* **38**, 134 (2013).
- [84] D. Riedel, I. Söllner, B. J. Shields, S. Starsielec, P. Appel, E. Neu, P. Maletinsky, and R. J. Warburton, *Deterministic enhancement of coherent photon generation from a nitrogen-vacancy center in ultrapure diamond*, *Physical Review X* **7**, 031040 (2017).
- [85] S. Bogdanović, S. B. van Dam, C. Bonato, L. C. Coenen, A.-M. J. Zwerver, B. Hensen, M. S. Liddy, T. Fink, A. Reiserer, M. Lončar, and R. Hanson, *Design and low-temperature characterization of a tunable microcavity for diamond-based quantum networks*, *Applied Physics Letters* **110**, 171103 (2017).
- [86] J. L. Serrano, P. F. B. Álvarez, M. Cattin, E. G. Cota, J. F. Lewis, P. Moreira, T. Włostowski, G. Gaderer, P. Loschmidt, J. Dědič, R. C. Bär, T. Fleck, M. C. Kreider, C. Prados, and S. Rauch, *The white rabbit project*, in *Proceedings of ICALEPCS* (TUC004, Kobe, Japan, 2009) p. 3.
- [87] A. Reiserer, N. Kalb, M. S. Blok, K. J. van Bemmelen, T. H. Taminiau, R. Hanson, D. J. Twitchen, and M. Markham, *Robust Quantum-Network Memory Using Decoherence-Protected Subspaces of Nuclear Spins*, *Physical Review X* **6**, 021040 (2016), [arXiv:1603.01602](https://arxiv.org/abs/1603.01602) .
- [88] N. Kalb, P. C. Humphreys, J. J. Slim, and R. Hanson, *Dephasing mechanisms of diamond-based nuclear-spin memories for quantum networks*, *Physical Review A* **97**, 1 (2018).
- [89] IEEE 802.1 working group, *802.1ae - media access control (mac) security*, (2015).
- [90] S. Guha, H. Krovi, C. A. Fuchs, Z. Dutton, J. A. Slater, C. Simon, and W. Tittel, *Rate-loss analysis of an efficient quantum repeater architecture*, *Physical Review A* **92**, 022357 (2015).
- [91] V. Shankarkumar, L. Montini, T. Frost, and G. Dowd, *Precision Time Protocol Version 2 (PTPv2) Management Information Base*, RFC 8173 (RFC Editor, 2017).

- [92] S. B. van Dam, P. C. Humphreys, F. Rozpędek, S. Wehner, and R. Hanson, *Multiplexed entanglement generation over quantum networks using multi-qubit nodes*, *Quantum Science and Technology* **2**, 034002 (2017).
- [93] QuTech, *NetSQUID*, <https://netsquid.org/> (2018).
- [94] J. Filho, Z. Papp, R. Djapic, and J. Oostveen, *Model-based design of self-adapting networked signal processing systems*, in *Proc. SASO* (IEEE, Philadelphia, PA, USA, 2013) pp. 41–50.
- [95] SURFsara, *Cartesius*, <https://userinfo.surfsara.nl/systems/cartesius> (2018).
- [96] A. Dahlberg, M. Skrzypczyk, T. Coopmans, L. Wubben, F. Rozpędek, M. Pompili, A. Stolk, P. Pawłczak, R. Knegjens, J. de Oliveira Filho, R. Hanson, and S. Wehner, *Code used in simulations*, <https://github.com/SoftwareQuTech/QLinkLayerSimulations> (2019).
- [97] A. Dahlberg, M. Skrzypczyk, T. Coopmans, L. Wubben, F. Rozpędek, M. Pompili, A. Stolk, P. Pawłczak, R. Knegjens, J. de Oliveira Filho, R. Hanson, and S. Wehner, *Data from simulations*, <https://dataverse.nl/dataverse/QLinkLayer> (2019).

3

SimulaQron - A simulator for developing quantum internet software

Axel Dahlberg, Stephanie Wehner

We introduce a simulator of a quantum internet with the specific goal to support software development. A quantum internet consists of local quantum processors, which are interconnected by quantum communication channels that enable the transmission of qubits between the different processors. While many simulators exist for local quantum processors, there is presently no simulator for a quantum internet tailored towards software development. Quantum internet protocols require both classical as well as quantum information to be exchanged between the network nodes, next to the execution of gates and measurements on a local quantum processor. This requires quantum internet software to integrate classical communication programming practises with novel quantum ones.

SimulaQron is built to enable application development and explore software engineering practises for a quantum internet. SimulaQron can be run on one or more classical computers to simulate local quantum processors, which are transparently connected in the background to enable the transmission of qubits or the generation of entanglement between remote processors. Application software can access the simulated local quantum processors to execute local quantum instructions and measurements, but also to transmit

Parts of this chapter have been published in Quantum Science and Technology [1].

qubits to remote nodes in the network. SimulaQron features a modular design that performs a distributed simulation based on any existing simulation of a quantum computer capable of integrating with Python. Programming libraries for Python and C are provided to facilitate application development.

3.1. Introduction

A quantum internet enables quantum communication between remote quantum processors in order to solve problems that are infeasible classically. Many applications of a quantum internet are already known, the most famous of which is quantum key distribution (QKD) [2, 3] which allows two network nodes to establish an encryption key. Examples of other applications are secure identification [4] and other two-party cryptographic tasks [5], clock synchronization [6], secure delegated quantum computation [7], and even extending the baseline of telescopes [8]. For many of these applications, only relatively simple quantum processors capable of operating on a handful of qubits are required, as they draw their power from quantum entanglement. Entanglement can be realized using already one qubit at each end point, and its capabilities cannot be replicated using classical communication. In the case of QKD, for example, quantum processors capable only of preparing and measuring single qubits can already be sufficient [2].

The first quantum networks that connect remote quantum processors capable of operating on several qubits each are expected to be deployed within the coming years. End-nodes, holding such quantum processors and on which applications run, will be able to send qubits and generate entanglement between each other using the network. Depending on the distance between the end-nodes, different ways to realize the quantum communication can be used, including the use of quantum repeaters [9–13]. In order to execute arbitrary quantum internet applications on these networks, it is essential to create a development framework in which software for these applications, running on the end-nodes, can be written and debugged. Writing software for a quantum internet shares some similarities with programming a quantum computer, but in addition poses new challenges. Similar to programming a quantum computer, we wish to execute quantum gates and measurements on each local quantum processor. Techniques for optimizing such gates and mapping them to the underlying hardware can be borrowed from quantum computing efforts, and are hence not the subject of this work. What differentiates programming a quantum network is the need for a close integration between classical and quantum messages exchanged during the course of the protocol. The need for such integration arises at several layers of abstraction, and poses significant design challenges. On a lower level, a quantum network stack is needed to create and track entanglement in the network, which requires classical control messages to be exchanged. On a higher level, applications which desire to create and use such entanglement do themselves exchange classical messages during the course of a protocol, requiring standard classical network programming techniques to integrate with quantum ones. A functional allocation of a network stack was introduced in chapter 2. We note that the CQC-interface as presented here have later been replaced by an improved version which we now call NetQASM [14].

3.1.1. What SimulaQron does

SimulaQron is a simulator providing a tool for software development for a quantum internet, freely available online [17]. Specifically, SimulaQron simulates several quantum processors held by the end-nodes of the network, connected by a

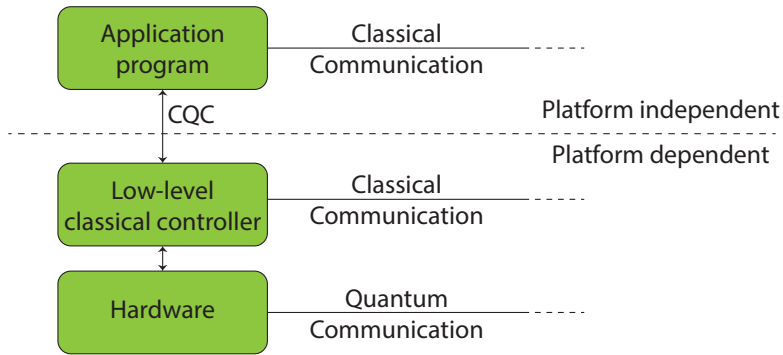


Figure 3.1: High-level schematic of the quantum network system of one node in the network. At the lowest level lies the quantum hardware, which in the most general case consists of a quantum processor capable of storing and manipulating qubits. The quantum processor has an optical interface over which it can generate entanglement with neighbouring network nodes, and send/receive qubits. Sending and receiving qubits may - depending on the implementation - be realized by first generating entanglement, followed by quantum teleportation. At present, quantum network nodes that support local quantum processing as well as the generation of long-lived entanglement are NV centers in diamond [15] and Ion Traps [16]. The quantum hardware is controlled by a necessarily platform dependent control system that may possess a classical communication interface to neighbouring nodes, for example to communicate the successful generation of heralded entanglement [15]. Together the quantum hardware and low-level control system form the platform dependent quantum processing system. SimulaQron provides a stand-in for such platform dependent quantum processing systems, and the quantum communication between them. The platform dependent system provides a universal interface, which we call CQC (Classical Quantum Combiner). CQC can be understood as an extended instruction set which - next to supporting “standard” quantum instructions such as performing gates or measurements - features special features tailored to a quantum network. This includes, for example, commands to produce entanglement or transmit qubits. Applications can be realized in the platform independent part of the quantum internet system by sending the appropriate instructions using CQC to the underlying quantum processing system. For simplicity, SimulaQron allows for direct communication between any two nodes in the network. However, a different topologically can be realized by using SimulaQron in a restricted fashion. Quantum network applications generally require the exchange of classical communication, and the integration between such classical communication and the use of the local quantum processing system is an integral aspect of application software development. In practice, this classical communication may be solved transparently using the same physical medium as used for the low level control, but this is not a requirement.

simulated quantum communication channels. This allows the simulation of single quantum networks, as well as inter-connected quantum networks forming a quantum internet. The quantum communication channels between the end-nodes in a real implementation of the network, can be realized in different ways, for example using quantum repeaters. The simulation of quantum repeaters and their performance is an important aspect of developing a quantum network. Numerous simulations of quantum repeaters have been conducted such as for example [18–20], the objective of SimulaQron however is a rather different one in that it aims provide a platform for application development to software engineers. SimulaQron can be used to develop the software running at the end-nodes together with classical communication between these. Figure 3.1 provides a high level schematic of such a quantum internet system, where SimulaQron should be understood as

a stand-in for the quantum processing system (platform dependent) in order to enable platform independent software development to proceed without access to quantum hardware.

SimulaQron precisely mimicks a real network, and allows each simulated processor to run on a different classical computer. Each local processing system supports the execution of local quantum gates and measurements, but also network specific commands, for example to generate entanglement. SimulaQron transparently simulates the exchange of qubits and the creation of entanglement between remote processors in the background, making this functionality available to applications. This is achieved by classical communication between the computers hosting the simulated quantum processors, as depicted in the schematic of Figure 3.2.

To perform the simulation of local quantum processors itself, SimulaQron uses existing simulators of quantum processors. What's more, SimulaQron's modular structure does in principle allow any simulation of a quantum computer accessible via Python to serve as a backend. The key novelty in SimulaQron is to leverage such backends into a distributed simulation that maps locally simulated entangled qubits to remote network nodes, in order to simulate the availability of entanglement between distant quantum computers. We emphasize that applications using SimulaQron's simulated entanglement, evidently do not provide the security guarantees afforded by real entanglement. The objective is instead to use SimulaQron as a development platform to write software realizing these applications which can later run on real quantum hardware and use real entanglement in order to achieve these guarantees. SimulaQron can be used as a tool for software development in all areas ranging from the implementation of the actual applications, the development of application level abstractions and programming libraries, to exploring and implementing a quantum network stack [21].

3.1.2. What SimulaQron does not

We emphasize that SimulaQron is written as a tool for software engineers, with the objective of allowing software engineering efforts for a quantum internet to proceed. Its goal is thus to allow developers to write software that can later run with no or minimal modification on real quantum internet hardware.

For quantum experts, we remark that SimulaQron does not aim to achieve an efficient simulation of a large scale quantum internet in order to test quantum repeater schemes, error correcting codes, or study the effects of noise on distant qubits. Evidently, given that SimulaQron can use any local simulation of a quantum processor in Python as a backend, it is straightforward to let it use a backend simulating noisy qubits. Noise can also be added manually by the application, for example, by probabilistically applying Pauli gates to the qubits during the protocol. We remark however that such adhoc additions of noise do not allow us to make general and accurate statements about the performance of quantum network applications in the presence of noise. Noise in quantum devices is highly time dependent, and hence the amount of noise an application experiences is highly dependent on time delays - for example, how long it takes classical and quantum data to travel from one node to another. If SimulaQron would simply be programmed to apply

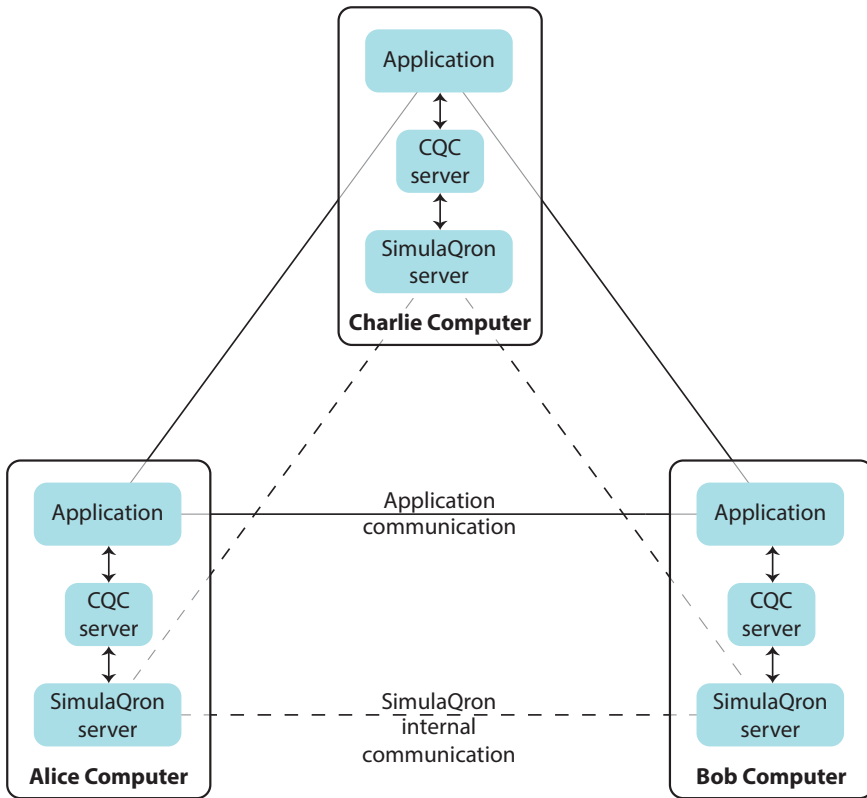


Figure 3.2: A schematic overview of the communication in a quantum network simulated by SimulaQron. The simulation of the quantum hardware at each node is handled by the SimulaQron backend server. Communication between the SimulaQron servers is needed to simulate the network, for example to simulate entanglement. Opting for this method enables a distributed simulation, i.e. the computers in the figure can be physically different computers. The CQC servers provide the CQC interface (see Figure 3.1) to the applications running on the network and the simulated platform dependent quantum processing system (see also Figure 3.3). Internally the CQC servers establish a connection to the backend. The choice to use two distinct servers is motivated by modularity, in that the same CQC servers can in principle give access to multiple backends (see also Section 3.1.2). Finally the applications can communicate classically, as they would in a real implementation of a quantum network.

a time dependent noise based on the message delays during simulation, then this would only be meaningful if these delays mirrored the exact time delays in the real network. What's more, even for executing local gates alone, the execution time of the simulation on the classical computer does not provide reliable timings.

In order to explore the precise effect of noise in a quantum network, it is essential to be able to model time very precisely. This can be achieved using a technique called discrete event simulation that is well known from classical networking. This is subject of a separate simulation platform (NetSquid [22]), which performs a discrete event simulation of a large scale quantum internet capable of precisely modelling timing delays and hence study the effects of delays in quantum communication,

as well as classical control communication on the performance of quantum protocols and a quantum network stack. NetSquid was only recently made publicly available and differs from SimulaQron in several key aspects apart from its ability to model time: first, it is a monolithic simulation and does not provide a real simulation network on different computers as SimulaQron does. As such, it does not mirror software engineering practices in networked programming. Second, it does not provide a real time interactive experience of working on a quantum network as provided by SimulaQron. We remark that we have recently extended NetSquid to support the new version of CQC, which we call NetQASM [14]. This way the performance of applications written for SimulaQron can be tested under controlled timing conditions, and realistic time dependent noise models of different hardware platforms, using NetSquid. This work is however not yet public.

Also for software engineers, we remark that there are some things which are purposefully not handled by SimulaQron. This includes, for example, the management and tracking of entanglement required by a quantum network stack, which may on the other hand be explored and implemented using SimulaQron. The reason being that a quantum network stack, together with protocols for managing and tracking entanglement was only developed after SimulaQron [21, 23]. Importantly, we also note that for the same reason no efforts have been made to secure or authenticate access to the simulated quantum processors. A SimulaQron backend server will by default happily accept requests from any client connecting to it, and in particular allow access to any qubit given its correct identifiers. It is clear that mechanisms for implementing access control to quantum nodes and qubits are important to realize in software for a quantum internet, highlighting the need for a tool like SimulaQron as a stand-in for quantum hardware in order to develop them.

3.1.3. Related Work

There is to our knowledge no analogue of SimulaQron available for application development on a quantum internet. There are of course numerous simulators of a quantum computers available, some of which could in principle be used as a local simulation backend in conjunction with SimulaQron, with the distributed simulation handled by SimulaQron. Freely available ones include ProjectQ [24] which is written in Python and contains an optimizing compiler. Other simulators include Liquid [25] (available as a binary), Forest by Rigetti [26], QX [27], the simulation backend of the IBM Quantum Experience [28] (can be accessed in Python using QISKit), and the Microsoft Quantum Development Kit [29] (using the programming language Q#). Presently, the quantum backend of SimulaQron is simply realized using QuTip [30], which is not designed to be an efficient simulator, but more than sufficient for the purpose of simulating a small network. Using QuTip also has the advantage of providing a very easy way to extend SimulaQron to operate on noisy instead of perfect qubits, should one wish to gain insights into whether, for example, software performing error correction for a quantum protocol has been implemented correctly. Again, we emphasize that while this allows verification of the correct implementation of such error correction schemes, SimulaQron is not meant to accurately model time dependent noise in the network.

3.1.4. Overview

SimulaQron itself is written in Python [31] due to the popularity of this language in the quantum information community, making it easier to extend if desired. Internally, SimulaQron also makes use of the Twisted framework for Python [32]. Twisted is a library providing functions that facilitate the development of network applications in Python. SimulaQron's simulated quantum internet can be programmed in two ways as outlined in Figure 3.3. In Section 3.2 we will provide an overview of the design and inner workings of SimulaQron. In Section 3.3 we discuss the integration with the CQC interface. We provide two simple examples on how to program SimulaQron using the Python CQC library in Section 3.4, together with performance analysis of SimulaQron for some test-cases. Further examples, programming templates as well as an API documentation can be found online [17]. Finally, we discuss future developments and extensions of SimulaQron.

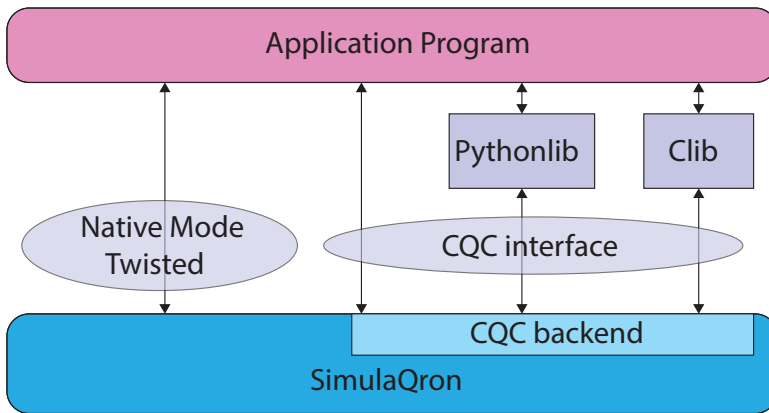


Figure 3.3: SimulaQron's simulated quantum internet can be programmed in two ways. The first proceeds by directly accessing the SimulaQron backend using Twisted in Python. This way of programming gives full access to the backend, but is highly specific to Twisted and unlikely to be available on any real quantum network devices of the future. The second way of programming SimulaQron is by using the Classical-Quantum-Combiner (CQC) interface. CQC specifies a packet format for issuing commands to a quantum network node and we intend to make a (possibly refined) version of CQC available on the planned 2020 quantum internet demonstrator in the Netherlands using actual quantum network hardware. For ease of programming, we provide two libraries that encapsulate the CQC interface. More specifically, we provide libraries in both Python and C for programming quantum internet applications, which internally connect to SimulaQron over the CQC interface. Libraries for other languages are easy to add and can access SimulaQron using the CQC interface.

3.2. Backend

Let us now first describe the SimulaQron backend, which can be accessed in Python using Twisted's Perspective Broker [32]. We note that for programming applications there is no need to ever access the backend directly, as the Python library using CQC provides a much easier way to program applications. CQC is an interface (close to) what we intend to make available on the quantum internet demonstrator in the Netherlands that can be accessed from any programming language. As mentioned,

QC has after this work been reworked and is now called NetQASM [14].

3.2.1. Challenges

The main challenge in providing a simulator suitable for programming quantum internet applications is to simulate quantum entanglement. Mathematically, any quantum state can be written as a density matrix $\rho \in \mathbb{C}^{d \times d}$, where d is the dimension of the quantum system (see [33], or [34, Weeks 0 and 1] for an introduction to quantum information). Crucially, for two entangled qubits A and B , we have $\rho \neq \sum_j p_j \rho_j^A \otimes \rho_j^B$, with $\rho_j^A \in \mathbb{C}^{2 \times 2}$, $\rho_j^B \in \mathbb{C}^{2 \times 2}$, and \otimes denoting the tensor product. This means that we cannot factorize ρ into different components ρ_j^A and ρ_j^B that could be simulated individually on two different network nodes. Instead, we need to simulate the entire matrix ρ as one, while making qubits A and B virtually available at two different nodes in the network.

One way of achieving this, is to let one simulating node hold all qubits in the network in the same register, consisting of a matrix of dimension $2^n \times 2^n$ where n is the number of qubits. This places a prohibitively large load on the simulating node, preventing large networks to be simulated. One step towards making the simulation more efficient is to still use only one central simulating node, but keep - as much as possible - different registers. For example, by initially placing each qubit in its own register requiring the simulation of only a 2×2 matrix per initial qubit, and only merging registers as qubits become entangled with each other. This improvement is indeed implemented in SimulaQron whenever qubits are created at one node. Nevertheless if this was the only optimization, a significant load would remain at one node in the network.

Consequently, we here go one step further and choose a distributed approach to simulation, described in detail below. In summary, each node keeps a quantum register of its own, in which a number of qubits are simulated. Qubits which are virtually available at other nodes are mapped back to registers simulated at in principle any other node, typically holding the other qubit of any entanglement. This allows the network to grow dynamically, and distributes simulation efforts amongst many computers. It is especially suited to situations where large amounts of fragmented entanglement exists in the network. This is naturally the case if the simulated network is very large but different subsets of nodes are executing protocols between them at any one time. This is well motivated from the usage of the classical internet, where many subsets of a few nodes each communicate with each other, but we do not see all nodes on the internet engaging in a joint application protocol. Another natural example, which we will also consider in a performance analysis below, is given by a situation in which a large number of nodes in a ring keeps entanglement with its neighbours - for example to transmit a qubit by forward teleportation along a chain of nodes. Our analysis shows no significant problems in simulating 60 nodes using 120 qubits on a single desktop machine, while if we had placed all 120 qubits in a single register we would need to keep track of a 2^{120} dimensional matrix.

While a distributed simulation is more efficient and allows a dynamic growth of the network, it also brings new challenges. Quantum gates performed on two qubits can cause two qubits that were previously unentangled to become entangled.

Consequently, if the two qubits were previously simulated in two different registers in the network, then a register merge is required. This is made challenging by the fact, that other nodes may simultaneously try and manipulate qubits simulated in said registers, or even perform operations that require a different register merge. It is in general challenging to manage access to distributed data in the presence of many concurrent requests to use it. SimulaQron solves such concurrency issues using a relatively simple but carefully crafted internal locking mechanism.

3

3.2.2. Design overview

Together, these considerations motivate the design of the SimulaQron backend depicted in Figure 3.4. The SimulaQron backend consists of running a client and server program on each classical computer - a virtualNode - that wants to participate in the simulated quantum network. All such programs connect to each other, forming the simulated quantum internet backend.

SimulaQron itself does not provide a new quantum simulator, but rather builds a distributed simulation on top of an existing one using a modular design. In the initial release, we have simply used QuTip for the underlying simulation, but essentially any existing simulator (or even real quantum computing - but not yet networked - hardware) that can be addressed via Python can easily be used in conjunction with SimulaQron. An existing simulator can be made available to SimulaQron as a backend by providing an interface in the form of a quantumEngine to function as a quantumRegister. This register supports addressing individual qubits by their position in the register, that is, for a register with n qubits. For example, if a Hadamard gate H is requested on qubit j , quantumRegister is responsible for applying this operation on the underlying quantum simulation. In the simple case of QuTip, this means simply applying the unitary $\text{id}^{\otimes j-1} \otimes H \otimes \text{id}^{\otimes n-j}$ to the matrix representing ρ (see crudeSimulator.py).

Building on top of an underlying quantumRegister, SimulaQron uses a simulatedQubit object to represent each qubit simulated in the underlying quantumRegister. Manipulation of qubits then follows exclusively by manipulating such simulatedQubits without interacting directly with the quantumRegister. In particular, each simulatedQubit keeps track of its own position in the register, allowing easy manipulation and update of qubits that are physically simulated at a particular node. For example, if a qubit is measured it can be removed from the underlying quantumRegister, effectively shrinking the size of the matrices. This allows the simulation to proceed without having the underlying register grow arbitrarily while new qubits are being created and discarded. Removal from the quantumRegister can be done by updating only the simulated qubits (and not the corresponding virtual ones that may be held by other nodes in the simulation), and hence the rest of the simulation can proceed to access the simulated qubits without being aware that the underlying register has been shrunk. Figure 3.4 illustrates the relationship between the quantumRegister and the simulatedQubits. We note that simulatedQubit objects are local to the node performing their actual simulation in the quantumRegister.

3.2.3. Virtual Qubits

Each `simulatedQubit` object is then associated with a `virtualQubit` object. Importantly, a `virtualQubit` does not need to reside at the same network node as the corresponding `simulatedQubit`, but can be transferred to other nodes than the one performing the backend simulation. Specifically, each `virtualNode` can hold a number of `virtualQubits` which can either be simulated locally (i.e., the `simulated qubit` and the `virtual qubit` are located at the same node), or at a remote node. Twisted's Perspective Broker is used to marshal the mapping of virtual qubits back to `simulated qubit` objects at remote nodes on the simulated network.

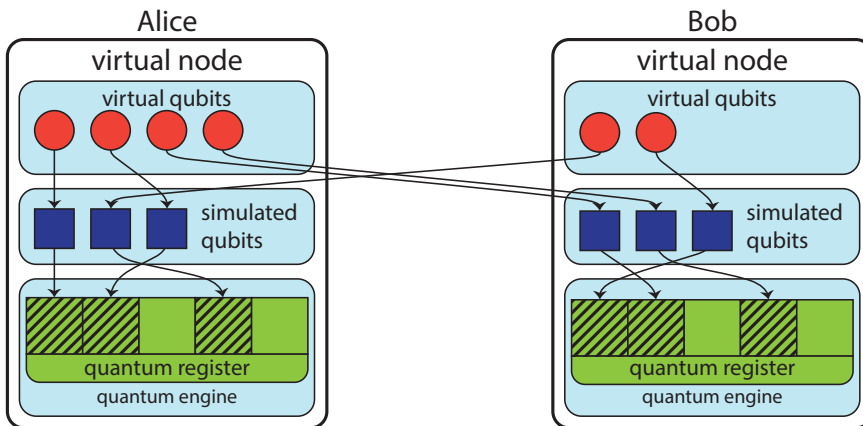


Figure 3.4: A visualization of the interplay between different internal components of SimulaQron. The `simulatedQubits` (blue squares) are objects handled locally in a `virtualNode`. These `simulatedQubits` point to a part of the `quantumRegister`, which stores the quantum state simulated by the `virtualNode`. Operations on the simulated state in the `quantumRegister` are handled by the `quantumEngine`. Additionally, a `virtualNode` also has `virtualQubits` (red circles). These `virtualQubits` point to `simulatedQubits`, possibly in a different `virtualNode`. The `virtualQubits` correspond to the actual qubits a node would have in a physical implementation of the quantum network.

This way it is possible for SimulaQron to simulate entanglement between network nodes: If simulated nodes A and B hold entanglement, then each `virtualNode` has a `virtualQubit` that can be processed at nodes A and B as if they were truly entangled. In the background, however, the two virtual qubits are mapped back to two `simulatedQubits` within the same `quantumRegister` that may be located at either A or B , or even at some other node.

Any application wishing to use SimulaQron consists of a client program at each computer that connects to the SimulaQron server backend. SimulaQron will make virtual qubits available to any client connecting to the `virtualNode` server, as well as allow the creation of new qubits and registers. Using Twisted's Perspective Broker, the client has full access to the specific `virtualQubit`, allowing it to perform gates, measurements, or send the qubit to other nodes in the simulated quantum network.

3.2.4. Register merges

One simple way to deal with virtual qubits is to have them simulated at one single classical computer in the network. Evidently, this puts a large strain on that specific computer, which then has to simulate every single qubit in the network. Here, we have instead chosen a distributed simulation, which is more challenging to realize, but does allow many nodes to take part in the simulation, as long as the global entanglement in each quantum state is not too large. This is typically the case, when investigating how to program applications that may each run on only a subset of the nodes in the network.

We hence allow any `virtualNode` to hold a `quantumRegister` containing simulated qubits that may eventually be sent as virtual qubits to other nodes in the network. This enables a distributed simulation in which each classical computer participating in the simulated quantum internet contributes its share to realizing the overall simulation. Yet, it is clear that this approach brings additional challenges. Specifically, consider two virtual qubits A and B at one network node, which are mapped back to two simulated qubits in *different* `quantumRegisters`. An example of such a situation is given in Figure 3.5. If the protocol now requests the `virtualNode` to perform an entangling gate between A and B , qubits A and B may now become entangled and can thus no longer be simulated in different registers.

Entangling gates may thus require a register merge in order for the simulation to proceed. More precisely, entangling A and B then requires that the two distinct quantum registers which hold the corresponding simulated qubits to be merged into one single quantum register, followed by an update of the corresponding `simulatedQubits` that the `virtualQubits` are mapped to.

SimulaQron solves this problem by transparently merging the registers in the background. For the application dealing with only the `virtualQubits` such a merge is invisible, but the simulated qubits representing the virtual qubits in question are updated.

It is clear that locking is required to ensure consistency in performing such register merges in case multiple requests to merge a register arise in the network at the same time. At present, SimulaQron implements a very simple and relatively inefficient locking mechanism tailored to acquiring the minimal set of locks required to perform and update. As two separate locks for registers and simulated qubits are used, a deadlock can arise which is dealt with using a simple randomized backoff to marshal competition for locks. Future versions of SimulaQron may be enhanced by a more sophisticated locking mechanism.

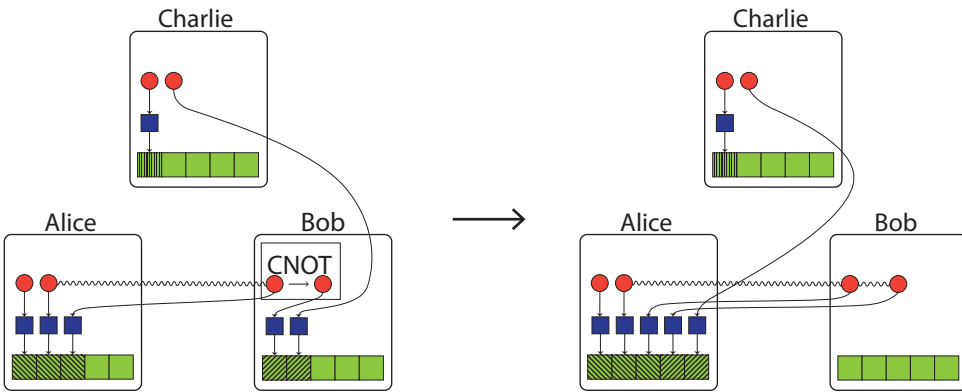


Figure 3.5: A visualization of a register merge, initialized by a CNOT performed in the virtualNode Bob. Alice and Bob initially share a EPR-pair, represented by a curvy line between the two virtualQubits. These entangled qubits are simulated in the virtualNode Alice. Bob then performs a CNOT between his part of the entangled state (control) and another qubit (target) simulated at Bob. This operations requires a register merge between Alice and Bob, since the three relevant qubits will now be in a GHZ-state, which needs to be simulated locally. The registers of Alice and Bob are therefore merged and all qubits simulated by either Alice or Bob are now simulated by Alice. Note that this also includes the virtualQubit of Charlie, which is initially simulated by Bob.

3.3. CQC

The CQC¹ instruction format provides a low level language programmable at the level of quantum gates and measurements, tailored specifically to include certain commands and behaviour that is useful in the quantum internet domain. The CQC interface provides a way for applications to be developed independently of the underlying platform and for these to be executed on any platform which provide the CQC interface. On top of the backend, SimulaQron realizes a CQC interface (classical-quantum combiner) as implemented by the so-called CQC backend. SimulaQron can therefore be programmed using any language capable of connecting to the CQC server backend over a TCP connection, and sending packets of the required form, specified by the CQC message format. In this section we describe the CQC interface and its messages in more detail.

Two libraries are included to program SimulaQron via the CQC interface in Python and C. Programming SimulaQron via this Python library is the easiest way to make use of SimulaQron and the recommended way to get started using SimulaQron. Below, we provide two simple examples; many more examples can be found in the online documentation [17].

We expect that the present CQC interface will undergo further evolution until its use in the Dutch demonstration network. As such, we describe the main ideas and what functionality exists. The current version of the CQC interface including the precise message format is available online [17] and will be updated when a newer version exist. CQC specifies a particular instruction format for requests and replies

¹CQC has now been reworked and is instead called NetQASM [14].

from the CQC backend. Requests may be given as a single message to the CQC interface, or an entire batch of messages at once specifying a complex operation consisting of many actions in succession.

Messages in the CQC interface can be of different *types*, as described in section 3.3.1. One of these types is COMMAND which instructs SimulaQron (or the physical hardware) to perform a certain command. This command could for example be a gate or to send a qubit to another node. More details on the message type COMMAND is given in section 3.3.2.

In addition to the type of message and what command to perform, CQC messages will also contain information regarding for example the CQC version, an application identifier, the length of the message (including additional commands if any) and - if applicable - the qubit identifiers a command concerns, the IP and port-number the node a qubit should be sent to. Other options that can be specified is whether notification should be returned when the command is finished and if the node should be blocked during the execution of the command, see table 3.1.

| Option | Effect |
|------------|--|
| OPT_NOTIFY | Send a notify when command is done |
| OPT_ACTION | Execute further commands when done |
| OPT_IFTHEN | Execute further commands based on result |
| OPT_BLOCK | Block until command done |

Table 3.1: Options that can be specified when sending a COMMAND message.

There is also the possibility to attach a list of commands that should be executed directly after the command specified in the CQC message is completed, or if, for example, a measurement outcome takes on a certain value. As such, it is assumed that the CQC backend provides a rudimentary form of classical logic next to the quantum specific instructions that allow certain simple processing to be executed in the CQC backend, or indeed the actual hardware. That is, these commands can be executed without having to send messages back and fourth through the CQC interface between each command. Avoiding these messages between each command allows the sequence of commands to be executed faster, which is important for a physically implemented quantum network, subject to decoherence.

3.3.1. Message types

We describe in this section message types currently implemented in the CQC interface. As mentioned above, in future version of the interface, changes may be made and the latest version can be found online. The types of messages sent from the application to the hardware (either simulated by SimulaQron or physically implemented) are specified in table 3.2.

A HELLO message can be used to check that the connection to the hardware/SimulaQron is up and also to get some specifications about the hardware. The commands that can be specified using the type COMMAND is discussed in the following section. FACTORY is a type similar to COMMAND, but here a certain command should be executed a specified number of times. An example of the use of FACTORY

| Type | Effect |
|-------------|--------------------------------------|
| TP_HELLO | Alive check (get hardware spec.) |
| TP_COMMAND | Execute (list of) command |
| TP_FACTORY | Execute (list of) command repeatedly |
| TP_GET_TIME | Get creation time of qubit |

Table 3.2: Types of messages from the application to the hardware/SimulaQron.

is to instruct the hardware to continuously produce a specified number of entangled EPR-pairs. An EPR-pair could then ready to be used whenever it is needed by the protocol running on the network.

There are also message types returned by the hardware to the application, as specified in table 3.3.

| Type | Effect |
|-----------------|---------------------------|
| TP_NEW_OK | Qubit was allocated |
| TP_EXPIRE | Qubit has expired |
| TP_DONE | Done with command |
| TP_RECV | Received qubit |
| TP_EPR_OK | Created EPR pair |
| TP_MEASOUT | Measurement outcome |
| TP_INF_TIME | Return timing information |
| ERR_GENERAL | General purpose error |
| ERR_NOQUBIT | No more qubit |
| ERR_UNKNOWN | Unknown qubit ID |
| ERR_UNAVAILABLE | Cannot allocate qubit |
| ERR_DENIED | No access to qubit |
| ERR_VERSION | CQC version not supported |
| ERR_UNSUPP | Sequence not supported |
| ERR_TIMEOUT | Timeout |

Table 3.3: Types of messages from the hardware/SimulaQron to the application.

3.3.2. Possible commands

The different commands currently supported in the CQC interface when using the message type COMMAND are specified in table 3.4. The full CQC packet format includes qubit and entanglement identifiers required by the commands below [17]. We remark that we include a command to create pairwise entanglement, since this reflects the way two nodes are entangled by a heralded entanglement generation scheme (see Figure 3.1 for a high level implementation schematic and information). When a qubit is received or entanglement has been generated, a message, TP_RECV or TP_EPR_OK respectively, is returned to the application. This message notifies the application that the command was successful and that possible corrections required by the entanglement generation scheme has been applied.

| Command | Effect |
|---------------------|---|
| CMD_NEW | Ask to allocate a new qubit |
| CMD_ALLOCATE | Ask to allocate multiple qubits |
| CMD_RELEASE | Release qubit to be used by other app. |
| CMD_RESET | Reset qubit to $ 0\rangle$ |
| CMD_MEASURE | Measure qubit (demolition) |
| CMD_MEASURE_INPLACE | Measure qubit (non-demolition) |
| CMD_SEND | Send qubit to another node |
| CMD_RECV | Ask to receive qubit |
| CMD_EPR | Create EPR pair with another node |
| CMD_RECV_EPR | Ask to receive half of EPR pair |
| CMD_SWAP | Entanglement swapping |
| CMD_I | Identity |
| CMD_X | Pauli X |
| CMD_Y | Pauli Y |
| CMD_Z | Pauli Z |
| CMD_H | Hadamard |
| CMD_K | K gate (Z to Y) |
| CMD_T | T gate |
| CMD_ROT_X | Rotation around X (multiple of $\frac{2\pi}{256}$) |
| CMD_ROT_Y | Rotation around Y (multiple of $\frac{2\pi}{256}$) |
| CMD_ROT_Z | Rotation around Z (multiple of $\frac{2\pi}{256}$) |
| CMD_CNOT | CNOT (this qubit as control) |
| CMD_CPHASE | CPHASE (this qubit as control) |

Table 3.4: Commands that can be specified when using the type COMMAND.

The angle of rotation for the single-qubit rotations are currently specified by one byte and is therefore discretized to 256 possible angles. When the creation of an EPR pair is requested, a message containing an entanglement identifier will be returned to the application. At present, there is no protocol to track and manage entanglement in a network available. We are currently developing and testing such a protocol, which will be referenced on the SimulaQron website once released [17].

3.4. Examples

We provide here a simple example of how to program SimulaQron using the Python CQC library (see Figure 3.3). More examples, also using the C CQC library, can be found in the full online documentation [17]. Before running the example presented here, the simulated network we consider needs to be configured and the servers that does the communication in the backend of SimulaQron has to be setup. Information on how to configure the simulated network and setup the servers can be found in the online documentation [17]. In what follows, we will hence assume that the SimulaQron and CQC backends have been setup already, simulating the hardware

of two quantum internet nodes labelled Alice and Bob. We remark that the names Alice and Bob are translated by the Python CQC library into IP addresses according to the configuration file specified on [17].

3.4.1. Sending BB84 States

The first example we consider has no classical communication on the application level between Alice and Bob. This implies that we do not need to set up a separate client/server interaction to exchange information at the level of applications. This means that Alice and Bob only connect locally to the CQC backend of SimulaQron, which functions as the quantum hardware at their network node, to issue quantum instructions. In this example, Alice will send a single qubit to Bob.

Code for Alice

The first thing we need to do is to initialize an object called a CQCConnection, which does the communication with SimulaQron using the CQC interface. Once this connection is set up, Alice has access to her own locally simulated quantum hardware.

```
1 from SimulaQron.cqc.pythonLib.cqc import *
2
3 # Establish connection to SimulaQron
4 Alice=CQCConnection("Alice")
```

The argument that CQCConnection takes should be the name specified in the configuration file for the CQC-network. Once the connection is set up we can then create our first qubit. The qubit-object takes a CQCConnection as argument when initialized. When operations are applied to the qubit, the CQCConnection is used to communicate with SimulaQron, such that the corresponding simulatedQubit is updated.

```
1 # Create new qubit
2 q=qubit(Alice)
```

Alice will then send Bob one out of the four states

$$|0\rangle, |1\rangle, |+\rangle, |-\rangle \quad (3.1)$$

where $|\pm\rangle = \frac{1}{\sqrt{2}}(|0\rangle \pm |1\rangle)$, depending on the choice of the bits h_A and x set in the program. The states in equation (3.1) are usually called BB84-states and can be described by two binary variables as follows

$$H^{h_A} X^x |0\rangle \quad h_A, x \in \{0, 1\}, \quad (3.2)$$

where H is the Hadamard operation and X is the Pauli-X operation, defined as

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}, \quad X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}. \quad (3.3)$$

3

The second part of the code on Alice's side will then apply the operation $H^{h_A} X^x$ and send the qubit to Bob. In our simple example, Alice is done using the quantum internet and hence we close the CQCConnection.

```

1 # Determine which BB84 state to use
2 h_A=1;x=0
3
4 # if x=1, flip |0> to |1>
5 if x == 1: q.X()
6
7 # if h_A==1, convert to Hadamard basis
8 if h_A==1: q.H()
9
10 # Send qubit to Bob
11 Alice.sendQubit(q, "Bob")
12
13 # Close connection to SimulaQron
14 Alice.close()

```

Code for Bob

We will now describe the code on Bobs side. In our example, Bob does nothing but wait for a qubit to arrive. Once he receives one, he will measure it in the standard- $(\{|0\rangle, |1\rangle\})$ (for $h_B = 0$) or the Hadamard-basis $(\{|+\rangle, |-\rangle\})$ (for $h_B = 1$) and print the measurement outcome. The code on Bobs side can be seen below, where h_B determines that basis Bob measures in.


```

1 from SimulaQron.cqc.pythonLib.cqc import *
2
3 # Establish a connection to SimulaQron
4 Bob=CQCConnection("Bob")
5
6 # Choose which basis we measure in
7 h_B=1
8
9 # Wait to receive a qubit
10 q=Bob.recvQubit()
11
12 # If we chose the Hadamard basis
13 # to measure in, apply H
14 if h_B==1: q.H()
15
16 # Measure the qubit in the standard
17 # basis and store the outcome in m
18 m=q.measure()
19
20 # Print measurement outcome
21 print("Bobs meas. outcome: {}".format(m))
22
23 # Close connection to SimulaQron
24 Bob.close()

```

Note that if $h_A = h_B$ the measurement outcome for Bob will be Alice's choice of x with probability one.

3.4.2. Teleporting a qubit

In our second example, Alice teleports a qubit to Bob. This example demonstrates how to create shared EPR pairs, and also how to perform additional classical communication from Alice and Bob on the application level. This classical communication can be realized using standard client/servers programming. An inefficient, but convenient testing tool, to perform classical communication tool is provided by the Python library that does not require knowledge of classical client/server programming.

Code for Alice

As in the previous example, the first thing that happens is that a CQCConnection is initialized to handle the communication to SimulaQron via the CQC interface.

```

1 # Initialize the connection
2 Alice=CQCConnection("Alice")

```

By calling the method createEPR, Alice makes a request to generate an EPR-pair with Bob, i.e. to generate the state $\frac{1}{\sqrt{2}} (|00\rangle_{AB} + |11\rangle_{AB})$.

```

1 # Make an EPR pair with Bob
2 qA=Alice.createEPR("Bob")

```

When the EPR-pair has been generated, Alice prepares another qubit q in the state $|+\rangle = H|0\rangle$, which she wants to teleport to Bob.

3

```

1 # Create a qubit to teleport
2 q=qubit(Alice)
3
4 # Prepare the qubit to teleport in |+>
5 q.H()

```

Alice makes a Bell-measurement between the qubit qA in the EPR-pair shared with Bob and the qubit q . The Bell-measurement is done by applying a CNOT gate with q as control and qA as target, followed by a Hadamard gate on q and measuring both qubits in the standard basis.

```

1 # Apply the local teleportation operations
2 q.cnot(qA)
3 q.H()
4
5 # Measure the qubits
6 a=q.measure()
7 b=qA.measure()
8 print("Alice meas. out.: a={}, b={}".format(a,b))

```

At this point Alice needs to communicate to Bob what her measurement outcome was, such that Bob can recover the state which is teleported. This classical communication can be realized by setting up a client/server interaction between Alice and Bob. There is a built-in feature in the Python library that realize this functionality, which have been developed for ease of use for someone not familiar with a client/server setup. This communication is also handled by the object `CQCCConnection`. For Alice to send a message to Bob, the method `Alice.sendClassical("Bob",msg)` is simply called, where `msg` is the message she wish to send to Bob. The method opens a socket connection to Bob, sends the message and closes the connection again. Note that if this method is never called, a socket connection is never opened.

We emphasise that to have classical communication between the applications, one is not forced to use the built-in functionality realized by the `CQCCConnection`. A standard client/server setup can also be used.

```

1 # Send corrections to Bob
2 Alice.sendClassical("Bob", [a,b])
3
4 # Stop the connections
5 Alice.close()

```

Code for Bob

As mentioned, Bob will need to know the measurement outcomes from Alice and will therefore setup a server to be able to receive these. Bob will then receive the qubit q_B which is part of the EPR-pair generated with Alice. By calling the method `recvClassical`, Bob receives the measurement outcomes that Alice sent. Corrections are then performed, depending on these measurement outcomes. The qubit q_B will then be in the state Alice prepared, i.e. the state $|+\rangle$. Finally, Bob measures the qubit q_B which gives 0 or 1 with equal probability.

```

1 # Initialize the connection
2 Bob=CQCCConnection("Bob")
3
4 # Make an EPR pair with Alice
5 qB=Bob.recvEPR()
6
7 # Receive info about corrections
8 data=Bob.recvClassical()
9 message=list(data)
10 a=message[0]
11 b=message[1]
12
13 # Apply corrections
14 if b==1: qB.X()
15 if a==1: qB.Z()
16
17 # Measure qubit
18 m=qB.measure()
19 print("Bob meas. out.: {}".format(m))
20
21 # Stop the connection
22 Bob.close()

```

3.4.3. Performance

In this section we present a practical performance analysis of SimulaQron, which has been performed by running the following four tests:

- (a) A network with n nodes in a ring is used to teleport a qubit n times from a sender, through all the nodes and back to the sender again. The sender records the time it took for the qubit to traverse the network. We test two different cases, depending on when the nodes create the EPR-pairs for teleportation: (1) Each node creates an EPR-pair with the next node on the ring

only once the qubit to be teleported is received. (2) The creation of all EPR-pairs starts in advance. The qubit is teleported forward as soon as the next EPR-pair becomes available. Case (1) is denoted "EPR on the fly" in Figure 3.6 and case (2) as "EPR first". This test is executed on the same computer.

- (b) A network consisting of two nodes is used to teleport a qubit back and forth between the nodes n times. This means teleporting a qubit $2n$ times. In this test each node is simulated on its own physical computer, which both are on the same Ethernet network. The time it takes to perform all the teleportations of the qubit is again recorded.
- (c) Here we test how much time it takes for one node to initialize n qubits and later measure them. No two-qubit gate, that can entangle the qubits, is used in this test.
- (d) One node initializes a GHZ state on n qubits, i.e. $|\text{GHZ}_n\rangle = \frac{1}{\sqrt{2}}(|0\rangle^{\otimes n} + |1\rangle^{\otimes n})$, and measures all the qubits. The time this takes is recorded.

The runtime for these tests can be found in Figure 3.6. Note that for these simulations there are three processes running for each node: one performing the actual simulation, one listening to incoming CQC messages and the application program which sends the CQC messages, see Figure 3.2. Thus, in the case of 60 nodes in test (a), there are in fact 180 processes running on a single computer. These processes communicate over TCP, which creates some delay in the runtime of the simulation.

Since quantumRegisters are only merged when needed, i.e when a two-qubit gate is performed between qubits in different quantumRegisters, the runtime highly depends on how large multi-partite entangled states are produced in the simulation. For example, in test (c) non-entangled qubits are generated in one node and the runtime then scales linearly with the number of qubits. On the other hand in test (d), all qubits are in a single entangled state which requires all qubits to be in the same quantumRegister and the runtime then scales super-exponentially with the number of qubits. We emphasize that this runtime is a direct consequence of using QuTip as a backend for the simulation. Creating a GHZ-state directly in QuTip gives the same runtime-scaling as in Figure 3.6(d). Thus, by using a different backend for the simulation, the runtime to create larger entangled states can certainly be improved.

As mentioned, new qubits are always put in different registers and these registers are only merged if a two-qubit gate is performed between qubits in different registers. Currently fixed decisions are used to decide which node will store the register after the merge, depending on which qubit is the target and which is control in the two-qubit gate. There is clearly a room for improvement of the performance, if the direction of the register merge is chosen in a more dynamic way, depending on the protocol being simulated. At this point we do not have enough usage data to know what good decisions for the direction of the register merge are. We also point out that we make no effort in trying to split registers if qubits later become

unentangled by some other gate, since this is unlikely to be feasible in general and would require a lot of extra computations.

We emphasize that SimulaQron is not intended to substitute simulators dedicated to handle large multipartite states, as in the case for quantum computation. Despite this, developing software realizing for example distributed quantum computation using SimulaQron, is in principle possible since universal quantum computation is supported by the provided operations. However, SimulaQron's use-cases are for larger (or smaller) networks where far from all nodes are entangled in a single state. Nonetheless, this does not exclude cases where each node share some entanglement with another node, for example as in test (a) where all the nodes share two EPR-pairs with two other nodes, since these EPR-pairs can be simulated in different quantumRegisters and does not require a single quantumRegister for the whole network.

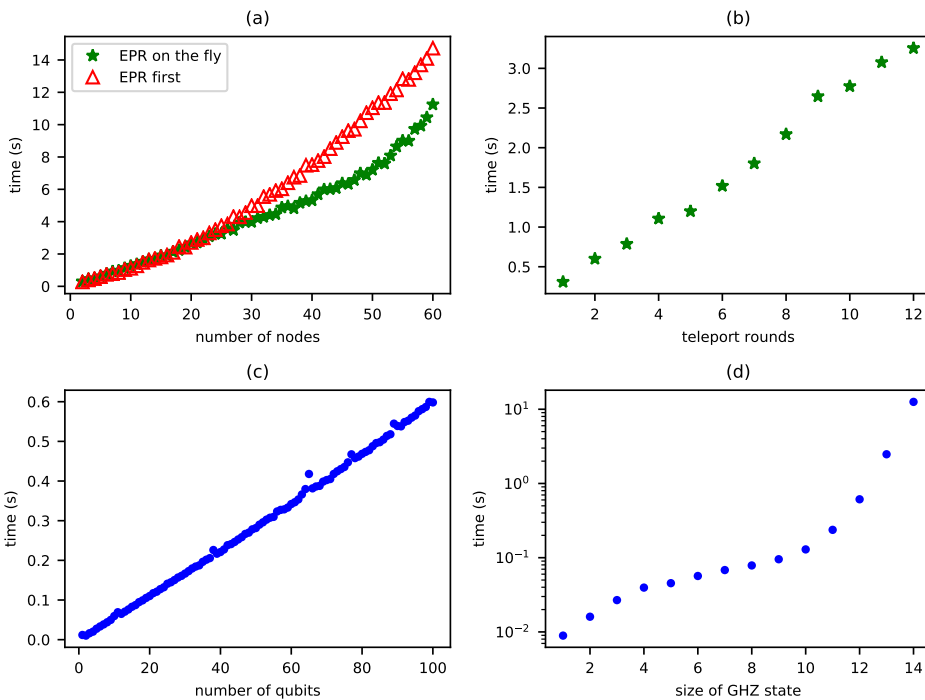


Figure 3.6: Runtimes for the four tests (see Section 3.4.3). All the tests were executed on one (two for (b)) iMac with a 3.2 GHz Intel Core i5 processor with 8 GB of 1600 MHz RAM running Python 3.6.0 with Twisted 17.9.0.

3.5. Conclusion

SimulaQron provides access to simulated quantum internet hardware, enabling application development. SimulaQron is undergoing continuous development and im-

improvements and new features may be added over time.

CQC itself can be understood as a very low level language by itself, supported directly by SimulaQron, and tailored specifically to the needs and behaviour of quantum internet application programs. At present, CQC is hidden away from view by the Python and C libraries that enable application development at a much higher level. Indeed, programming using the Python CQC library is the fastest way to get started writing a quantum internet application using SimulaQron. It is conceivable that CQC will be integrated into higher level languages directly by a means of a compiler, which produces CQC commands instead of using a dedicated library. As mentioned, this interface has after this work been improved and reworked and instead goes under the name NetQASM [14].

We might also imagine that more convenient higher level programming environments and libraries become available in the future. While the Python CQC library already offers many conveniences, such as dealing with qubits as Python objects, performing specific gates to - for example - generate BB84 states, it is still a relatively low level approach to programming the quantum internet. Indeed, many protocols depend on exchanging BB84 states, and their creation and processing could in the future be handled by a set of quantum internet software libraries.

Acknowledgment

We thank Ronald Hanson, Tracy Northup, Peter Humphreys, Norbert Kalb and Arie van Deursen for interesting discussions and feedback. AD and SW were supported by STW Netherlands, and NWO VIDI grant, and an ERC Starting grant.

References

- [1] A. Dahlberg and S. Wehner, *SimulaQron—a simulator for developing quantum internet software*, [Quantum Science and Technology](#) **4**, 015001 (2018).
- [2] C. H. Bennett and G. Brassard, *Quantum Cryptography: Public Key Distribution, and Coin-Tossing*, in *Proc. 1984 IEEE International Conference on Computers, Systems, and Signal Processing* (1984) pp. 175–179.
- [3] A. K. Ekert, *Quantum cryptography based on bell's theorem*, *Physical Review Letters* **67**, 661 (1991).
- [4] I. Damgård, S. Fehr, L. Salvail, and C. Schaffner, *Secure identification and qkd in the bounded-quantum-storage model*, [Theoretical Computer Science](#) **560**, 12 (2014), [arXiv:0708.2557](#) .
- [5] J. Kaniewski and S. Wehner, *Device-independent two-party cryptography secure against sequential attacks*, [New Journal of Physics](#) **18** (2016), [10.1088/1367-2630/18/5/055004](#), [arXiv:1601.06752](#) .
- [6] R. Jozsa, D. S. Abrams, J. P. Dowling, and C. P. Williams, *Quantum Clock Synchronization Based on Shared Prior Entanglement*, [Physical Review Letters](#) **85**, 2010 (2000), [arXiv:0004105 \[quant-ph\]](#) .

- [7] A. M. Childs, *Secure assisted quantum computation*, *Quantum Info. Comput.* **5**, 456 (2005), [arXiv:0111046 \[quant-ph\]](https://arxiv.org/abs/0111046) .
- [8] A. Kellerer, *Quantum telescopes*, *Astronomy and Geophysics* **55**, 1 (2014), [arXiv:1403.6681](https://arxiv.org/abs/1403.6681) .
- [9] S. Muralidharan, L. Li, J. Kim, N. Lütkenhaus, M. D. Lukin, and L. Jiang, *Optimal architectures for long distance quantum communication*, *Scientific Reports* **6**, 1 (2016), [arXiv:1509.08435](https://arxiv.org/abs/1509.08435) .
- [10] R. Van Meter, *Quantum Networking*, 1st ed. (John Wiley & Sons, Ltd, Hoboken, NJ, USA, 2014).
- [11] N. Sangouard, C. Simon, H. De Riedmatten, and N. Gisin, *Quantum repeaters based on atomic ensembles and linear optics*, *Reviews of Modern Physics* **83**, 33 (2011).
- [12] K. Azuma, K. Tamaki, and H.-K. Lo, *All-photonic quantum repeaters*, *Nature Communications* **6**, 6787 (2015).
- [13] W. J. Munro, K. Azuma, K. Tamaki, and K. Nemoto, *Inside quantum repeaters*, *IEEE Journal of Selected Topics in Quantum Electronics* **21**, 78 (2015).
- [14] A. Dahlberg, B. van der Vecht, C. Delle Donne, M. Skrzypczyk, W. Kozłowski, and S. Wehner, *Netqasm - a low-level instruction set architecture for hybrid quantum-classical programs in a quantum internet*, (2020), in preparation.
- [15] P. C. Humphreys, N. Kalb, J. P. Morits, R. N. Schouten, R. F. Vermeulen, D. J. Twitchen, M. Markham, and R. Hanson, *Deterministic delivery of remote entanglement on a quantum network*, *Nature* **558**, 268 (2018).
- [16] I. V. Inlek, C. Crocker, M. Lichtman, K. Sosnova, and C. Monroe, *Multispecies Trapped-Ion Node for Quantum Networking*, *Physical Review Letters* **118**, 1 (2017), [arXiv:1702.01062](https://arxiv.org/abs/1702.01062) .
- [17] *SimulaQron*, <http://www.simulaqron.org> (2020).
- [18] F. Rozpędek, K. Goodenough, J. Ribeiro, N. Kalb, V. C. Vivoli, A. Reiserer, R. Hanson, S. Wehner, and D. Elkouss, *Parameter regimes for a single sequential quantum repeater*, *Quantum Science and Technology* **3**, 34002 (2018).
- [19] H. Krovi, S. Guha, Z. Dutton, J. A. Slater, C. Simon, and W. Tittel, *Practical quantum repeaters with parametric down-conversion sources*, *Applied Physics B: Lasers and Optics* **122**, 1 (2016), [arXiv:1505.03470](https://arxiv.org/abs/1505.03470) .
- [20] S. Muralidharan, J. Kim, N. Lütkenhaus, M. D. Lukin, and L. Jiang, *Ultrafast and fault-tolerant quantum communication across long distances*, *Physical Review Letters* **112**, 250501 (2014).

- [21] A. Dahlberg, M. Skrzypczyk, T. Coopmans, L. Wubben, F. Rozpędek, M. Pompili, A. Stolk, P. Pawełczak, R. Knegjens, J. de Oliveria Filho, R. Hanson, and S. Wehner, *A link layer protocol for quantum networks*, in *ACM SIGCOMM 2019 Conference*, SIGCOMM '19 (ACM, New York, NY, USA, 2019) p. 15.
- [22] QuTech, *NetSQUID*, <https://netsquid.org/> (2018).
- [23] W. Kozłowski, A. Dahlberg, and S. Wehner, *Designing a quantum network protocol*, arXiv preprint arXiv:2010.02575 (2020).
- [24] D. S. Steiger, T. Häner, and M. Troyer, *ProjectQ: An Open Source Software Framework for Quantum Computing*, arXiv preprint (2016), arXiv:1612.08091 .
- [25] D. Wecker and K. M. Svore, *LIQUi|>: A Software Design Architecture and Domain-Specific Language for Quantum Computing*, arXiv preprint (2014), arXiv:1402.4467 .
- [26] *Forest by Rigetti*, <https://www.rigetti.com/forest> (2020).
- [27] N. Khammassi, I. Ashraf, X. Fu, C. Almudever, and K. Bertels, *QX: A high-performance quantum computer simulation platform*, *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017* , 464 (2017).
- [28] *IBM Q Experience*, <https://quantumexperience.ng.bluemix.net/qx/experience> (2020).
- [29] *Microsoft Quantum Development Kit*, <https://www.microsoft.com/en-us/quantum/development-kit> (2020).
- [30] J. R. Johansson, P. D. Nation, and F. Nori, *QuTiP: An open-source Python framework for the dynamics of open quantum systems*, *Computer Physics Communications* **183**, 1760 (2012), arXiv:1110.0573 .
- [31] *Python*, <https://www.python.org/> (2020).
- [32] *Twisted*, <https://twistedmatrix.com/trac/> (2020).
- [33] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*, 10th ed. (Cambridge University Press, Cambridge, 2010).
- [34] *MOOC: Quantum Cryptography on*, <https://www.edx.org> (2020).

4

Graph states and single-qubit operations

Axel Dahlberg, Jonas Helsen, Stephanie Wehner

In this chapter we introduce graph states and how certain single-qubit operations, that preserve this class, act on them. Furthermore, we describe how questions on transforming graph states in a quantum network relate to questions in graph theory.

Parts of this chapter have been published in Phil. Trans. R. Soc. [1].

4.1. Introduction

A key concept in realizing quantum technologies is the preparation of specific resource states, which then enable further quantum processing. For example, many quantum network protocols first ask to prepare a specific resource state that is shared amongst the network nodes, followed by measurements and exchange of classical communication. The simplest instance of this concept is indeed quantum key distribution [2, 3], in which we first produce a maximally entangled state, followed by random measurements. Similarly, measurement-based quantum computing [4] proceeds by first preparing the quantum device in a large resource state, followed by measurements on the qubits.

An important class of such resource states are graph states. These states can be described by a simple undirected and unweighted graph where the vertices correspond to the qubits of the state [5]. The graph state of a given graph is formed by initializing each qubit $v \in V(G)$ in the state $|+\rangle_v = \frac{1}{\sqrt{2}}(|0\rangle_v + |1\rangle_v)$ and for each edge $(u, v) \in E(G)$ applying a controlled phase gate between qubits u and v . Apart from their broad range of applications, an appealing feature of graph states is that they can be efficiently described classically. Specifically, to describe a graph state on n qubits, only $\frac{n(n-1)}{2}$ bits are needed to specify the edges of the graph. This is in sharp contrast to the 2^n complex numbers required to describe a general quantum state [6]. It turns out that for graph states, and indeed the more general class of *stabilizer states*, their evolution under *Clifford* operations and *Pauli* measurement can be simulated efficiently on a classical computer [7].

Well-known applications of graph states include cluster states [8] used in measurement based quantum computing where, together with arbitrary single-qubit measurements, these states form a universal resource for measurement-based quantum computation [4]. Graph states also arise as logical codewords of many error-correcting codes [9]. In the domain of quantum networking, a specific class of graph states is of particular interest. Specifically, these are states which are GHZ-like, i.e., they are equivalent to the GHZ-state up to single-qubit Clifford operations. GHZ-states have been shown to be useful for applications such as quantum secret sharing [10], anonymous transfer [11], conference key agreement [12] and clock synchronization [13]. It turns out that graph states described by either a star graph or a complete graph are precisely those GHZ-like states [5].

Given the desire for graph states, we may thus ask how they can effectively be prepared, and transformed. We consider the situation in which we already have a specific starting state (the *source* state), and we wish to transform it to a desired *target* state, using an available set of operations. Motivated by the fact that on a quantum network or distributed quantum processor, local operations are typically much faster and easier to implement, we consider the set of operations consisting of single-qubit Clifford operations, single-qubit Pauli measurements, and classical communication (LC + LPM + CC). Applications of an efficient algorithm that finds a series of operations to transform a source to a target state includes the ability to make effective routing decisions for state preparation on a distributed quantum processor or network. Here, fast decisions are essential since quantum memories

are inherently noisy and the source state will therefore become useless if too much time is spent on making a decision. Such algorithms could also be used as a design tool in the study of quantum repeater schemes [14], and the discovery of effective code switching procedures in quantum error correction [15, 16].

In this chapter we set our notation and recall various concepts which will be used throughout the rest of the thesis. We start by providing the definitions of graph states, qubit-minors and the relation to vertex-minors. We then recall the definitions of local complementation and vertex deletion as operations on graphs. These operations are useful in the context of graph states since they completely capture the action of LC+LPM+CC on graph states. Furthermore, we discuss circle graphs and their various characterizations and discuss how local complementation behaves on these graphs. We also introduce the concept of semi-ordered Eulerian tours, which is a key technical concept for the results later in this thesis. Finally we discuss distance-hereditary graphs, which form a subclass of circle graphs. We discuss how these graphs can be built up out of elementary pieces and prove some technical results which will be used in later chapters.

4.1.1. Previous work

It turns out that single-qubit Clifford operations on graph states correspond to an operation called *local complementation* [17] on the corresponding graph [18]. Furthermore, single-qubit Pauli measurements and classical communication correspond to local complementations and vertex-deletions [5]. The graphs reachable from G by performing local complementations and vertex-deletions are called *vertex-minors* of G . Vertex-minors are well-studied objects in graph theory [19]. To understand which graph states are related under LC + LPM + CC operations we will introduce the notion of a *qubit-minor* in section 4.4. A qubit-minor of a graph state $|G\rangle$ is another graph state $|G'\rangle$ such that $|G\rangle$ can be transformed to $|G'\rangle$ using only LC + LPM + CC operations. We show, theorem 4.4.2, that the notion of qubit-minors is equivalent to the notion of vertex-minors, in the sense that the graph state $|G'\rangle$ is a qubit-minor of $|G\rangle$ if and only if the graph G' is a vertex-minor of G .

Vertex-minors play an important role in algorithmic graph theory, together with the notion of *rank-width*, which is a complexity measure on graphs. Specifically, one can efficiently decide membership of a graph in some set of graphs, if this set is closed under taking vertex-minors and of fixed (bounded) rank-width [19]. An example of such a set is the set of *distance-hereditary graphs*, which are exactly the graphs with rank-width one [19]. Another example of a set of graphs which is closed under taking vertex-minors are *circle graphs*, which are however of unbounded rank-width ([20, Proposition 6.3] and [21]). An appealing connection between the rank-width of graphs, and the entanglement in the corresponding graph states was identified in [22], where it is shown that the rank-width of a graph equals the Schmidt-rank width of the corresponding graph state. The Schmidt-rank width of a quantum state is an entanglement measure. Specifically, the higher rank-width a graph has, the more entanglement there is in the corresponding graph state, in terms of this measure. Another interpretation of the *Schmidt-rank width* is that it captures how complex the quantum state is. One reason for this interpretation is

that quantum states can be described using a technique called *tree-tensor networks* and it was shown in [22] that the minimum dimension of the tensors needed to describe a state is in fact given by the *Schmidt-rank width*.

In the domain of complexity theory, the rank-width and related measures such as the tree- and clique-width [23, 24] also form a measure of the inherent complexity of instances to graph problems, and feature prominently in the study of fixed-parameter tractable (FPT) algorithms [25]. Specifically, a problem is called fixed-parameter tractable in terms of a parameter r , if any instance I of the problem of fixed r , is solvable in time $f(r) \cdot |I|^{O(1)}$, where $|I|$ is the size of the instance and f is a computable function of r [25]. In this work, r is the rank-width, and for graphs of constant rank-width the techniques of Courcelle [26] and its generalizations [27], can be used to obtain polynomial time algorithms for problems such as Graph Coloring [28], or Hamiltonian Path [29]. While very appealing from a complexity theory point of view, a direct application of these techniques does not usually lead to polynomial time algorithms that are also efficient in practice, since $f(r)$ is often prohibitively large.

Since the problem of deciding whether a graph state $|G'\rangle$ is a qubit-minor of $|G\rangle$ (`QubitMinor`) is equivalent to deciding if G' is a vertex-minor of G (`VertexMinor`), an efficient algorithm for `VertexMinor` directly provides an efficient algorithm for `QubitMinor`. This in turn can be used for fast decisions on how to transform graph states in a quantum network or distributed quantum processor. However, not much was previously known about the computational complexity of `VertexMinor` and therefore whether efficient algorithms exists. For a related but slightly more restrictive minor-relation, namely *pivot-minors* it has been shown in [30] that checking whether a graph G has a pivot-minor isomorphic to another graph G' is NP-Complete . We show in the upcoming chapters that also `VertexMinor` is NP-Complete .

However, for fixed rank-width, we show in chapter 5 how one can apply the techniques of Courcelle [26], to obtain an FPT algorithm for our problem that is efficient if both the size of G' , as well as the rank-width of G are bounded. Indeed, a powerful method for deciding if a graph problem is fixed-parameter tractable is by Courcelle's theorem and its generalizations [27]. It turns out that also for our case, a direct implementation of Courcelle's theorem does not give an algorithm that can be used in practice. In fact, in the case of `VertexMinor`, this constant factor obtained by applying the techniques of Courcelle in chapter 5 can be shown to be a tower of twos

$$f(r) = 2^{2^{\dots^{2^r}}} \quad (4.1)$$

where r is the rank-width of the input graph G and the height of the tower is 10. In chapter 6 on the other hand we provide efficient algorithms for restricted versions of the problem which can be used in practice.

4.2. Notation and definitions

Here we introduce some notation and vocabulary that will be used throughout this thesis. We assume familiarity of the general notation of quantum information the-

ory, see [6] for more details.

Quantum operations

The Pauli matrices will be denoted as

$$\mathbb{I} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \quad Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \quad (4.2)$$

and for the single-qubit Pauli group \mathcal{P} or \mathcal{P}_1 . The single-qubit Clifford group \mathcal{C} consists of operations which leave the Pauli group $\mathcal{P} = \langle \mathbb{I}, X, Z \rangle$ invariant. More formally, \mathcal{C} is the normalizer of the Pauli group, i.e.

$$\mathcal{C} = \{C \in \mathcal{U} : (\forall P \in \mathcal{P} : CPC^\dagger \in \mathcal{P})\}, \quad (4.3)$$

where \mathcal{U} is the single-qubit unitary operations. The n -qubit Pauli group \mathcal{P}_n is the n -fold single-qubit Pauli group whose elements are the tensor-products of elements of \mathcal{P}_1 .

Assume that $\mathbf{v} = (v_1, \dots, v_i, \dots, v_n)$ are the labels of a qubit which is part of some multi-qubit state $|\psi\rangle_{v_1 \dots v_i \dots v_n}$. We will then denote P_{v_i} as the operation

$$P_{v_i}^{(v_1 \dots v_n)} = (\mathbb{I})_{v_1} \otimes \dots \otimes (P)_{v_i} \otimes \dots \otimes (\mathbb{I})_{v_n}, \quad (4.4)$$

where $P \in \{\mathbb{I}, X, Y, Z\}$. We will never write an explicit ordering of the qubits in a multi-qubit state $|\psi\rangle_{v_1 \dots v_i \dots v_n}$ and rather write $|\psi\rangle_V$, where V is the set $\{v_1, \dots, v_n\}$. For explicit calculations one just needs to use a consistent ordering. Similarly for the operation $P_{v_i}^{(v_1 \dots v_n)}$ we will write $P_{v_i}^V$ or even P_{v_i} when it is clear which set V is considered.

Sequences and words

A *sequence* $\mathbf{X} = x_1 x_2 \dots x_k$ is an ordered, possibly empty, tuple of elements in some set X . We also call a sequence a *word* and its elements *letters*. We write $\mathbf{X} \subseteq X$, when all letters of \mathbf{X} are in the set X . A *sub-word* \mathbf{X}' of \mathbf{X} , is a word which can be obtained from \mathbf{X} by iteratively deleting the first or last element of \mathbf{X} . We denote the concatenation of two words $\mathbf{X}_1 = x_1 \dots x_{k_1}$ and $\mathbf{X}_2 = y_1 \dots y_{k_2}$ as $\mathbf{X}_1 \|\mathbf{X}_2 = x_1 \dots x_{k_1} y_1 \dots y_{k_2}$. We also denote the 'mirror image' by an overset tilde, e.g. if $\mathbf{X} = ab$ then $\tilde{\mathbf{X}} = ba$.

Sets

The set containing the natural numbers from 1 to n is denoted $[n]$. The symmetric difference $X\Delta Y$ between two sets X and Y is the set of elements of X and Y that occur in X or Y exclusively, i.e. $X\Delta Y = (X \cup Y) \setminus (X \cap Y)$.

We use the following notation for sets of consecutive natural numbers

$$[k, n] \equiv \{i \in \mathbb{N} : k \leq i < n\} \quad (4.5)$$

$$[n] \equiv \{i \in \mathbb{N} : 0 \leq i < n\} \quad (4.6)$$

Graphs

A simple undirected graph $G = (V, E)$ is a set of vertices V and a set of edges E . Edges are 2-element subsets of V for simple undirected graphs. Importantly,

we only consider labeled graphs, i.e. we consider a complete graph with vertices $\{1, 2, 3\}$ to be different from a complete graph with vertices $\{2, 3, 4\}$, even though these graphs are isomorphic. The reason for considering labeled graphs is that these will be used to represent graph states on specific qubits, possibly at different physical locations in the case of a quantum network. In a simple undirected graph, there are no multiple edges or self-loops, in contrast with a multi-graph: An undirected multi-graph $H = (V, E)$ is a set of vertices V and a multi-set of edges E . For undirected multi-graphs, edges are unordered pairs of elements in V . We will often write $V(G) = V$ and $E(G) = E$ to mean the vertex- and edge-set of the (multi-)graph $G = (V, E)$.

Next we list some glossary about (multi-)graphs:

4

- If a vertex $v \in V$ is an element of an edge $e \in E$, i.e. $v \in e$, then v and e are said to be *incident* to one another.
- Two vertices which are incident to a common edge are called *adjacent*.
- The set of all vertices adjacent to a given vertex v in a (multi-)graph G is called the *neighborhood* $N_v^{(G)}$ of v . We will sometimes just write N_v if it is clear which (multi-)graph is considered.
- The *degree* of a vertex v is the number of neighbors of v , i.e. $|N_v|$.
- A *k-regular* (multi-)graph is a (multi-)graph such that every vertex in the (multi-)graph has degree k .
- A *walk* $W = v_1 e_1 v_2 \dots e_k v_{k+1}$ is an alternating sequence of vertices and edges such that e_i is incident to v_i and v_{i+1} for $i \in [k]$.
- The vertices v_1 and v_{k+1} are called the *ends* of W .
- If the ends of a walk are the same vertex, it is called *closed*.
- A *trail* is a walk which does not include any edge twice.
- A closed trail is called a *tour*.
- A *path* is a walk which does not include any vertex twice, apart from possibly the ends.
- A closed path is called a *cycle*.
- Two vertices u and v are called *connected* if there exists a path with u and v as ends.
- A (multi-)graph is called *connected* if any two vertices are connected in the (multi-)graph.
- $G' = (V', E')$ is a *subgraph* of $G = (V, E)$ if $V' \subseteq V$ and $E' \subseteq E$.

- An *induced subgraph* $G[V']$ of $G = (V, E)$ is a subgraph on a subset $V' \subseteq V$ and with the edge-set

$$E' = \{(u, v) \in E : u, v \in V'\}. \quad (4.7)$$

- A *connected component* of a (multi-)graph $G = (V, E)$ is a connected induced subgraph $G[V']$ such that no vertex in V' is adjacent to a vertex in $V \setminus V'$ in the (multi-)graph G .
- A *cut-vertex* v of a (multi-)graph $G = (V, E)$ is a vertex such that $G[V \setminus \{v\}]$ has strictly more connected components than G
- The *distance* $d_G(v, w)$ between two vertices v, w in a (multi-)graph G is equal to the number of edges in the shortest path that connects v and w .
- The *complement* G^c of a graph $G = (V, E)$ is a graph with vertex-set $V^c = V$ and edge-set

$$E^c = \{(u, v) \in V \times V : (u, v) \notin E \wedge u \neq v\}. \quad (4.8)$$

A graph G is assumed to be simple and undirected, unless specified and will be denoted as G, G_i, G', \tilde{G} or similar. A multi-graph is assumed to be undirected, unless specified and will be denoted as H, H_i, H', \tilde{H} or similar. Furthermore, 3-regular simple graphs will be denoted as R, R_i, R' or similar and 4-regular multi-graphs as F, F_i, F' or similar. We will denote the *complete graph* on a set of vertices V as K_V and the *star graph* on a set of vertices V with center c by $S_{V,c}$. We will often not care about the choice of center, writing S_V to mean any choice of star graph on the vertex set V .

4.3. Stabilizer states

A stabilizer state $|\mathcal{S}\rangle$ on n qubits is defined by its stabilizer group \mathcal{S} , which is a subgroup of the Pauli group \mathcal{P}_n [15]. The stabilizer state is defined to be a state such that it is an eigenstate of all elements of \mathcal{S} with an eigenvalue of $+1$, i.e. $s|\mathcal{S}\rangle = |\mathcal{S}\rangle$ for $s \in \mathcal{S}$. To avoid $|\mathcal{S}\rangle$ being a trivial zero state there are two requirements of \mathcal{S} , (1) $-I \notin \mathcal{S}$ and (2) all elements of \mathcal{S} should commute¹. Furthermore, for $|\mathcal{S}\rangle$ to be a unique state (up to a global phase), \mathcal{S} needs to be of size 2^n and can therefore be described by n independent generators. As an example consider the stabilizer group \mathcal{S}_0 generated by $X \otimes X$ and $Z \otimes Z$. One can check that \mathcal{S}_0 describes the state

$$|\mathcal{S}_0\rangle = \frac{1}{\sqrt{2}} (|0\rangle \otimes |0\rangle + |1\rangle \otimes |1\rangle) \quad (4.9)$$

4.4. Graph states

A graph state is a multi-partite quantum state $|G\rangle$ which is described by a graph G , where the vertices of G correspond to the qubits of $|G\rangle$. The graph state is

¹Elements of the Pauli group either commute or anti-commute.

formed by initializing each qubit $v \in V(G)$ in the state $|+\rangle_v = \frac{1}{\sqrt{2}}(|0\rangle_v + |1\rangle_v)$ and for each edge $(u, v) \in E(G)$ applying a controlled phase gate between qubits u and v . Importantly, all the controlled phase gates commute and are invariant under changing the control- and target-qubits of the gate. This allows the edges describing these gates to be unordered and undirected. Formally, a graph state $|G\rangle$ is given as

$$|G\rangle = \prod_{(u,v) \in E(G)} C_Z^{(u,v)} \left(\bigotimes_{v \in V(G)} |+\rangle_v \right), \quad (4.10)$$

where $C_Z^{(u,v)}$ is a controlled phase gate between qubit u and v , i.e.

$$C_Z^{(u,v)} = |0\rangle\langle 0|_u \otimes \mathbb{I}_v + |1\rangle\langle 1|_u \otimes Z_v \quad (4.11)$$

and Z_v is the Pauli-Z matrix acting on qubit v .

A graph state is also a stabilizer state [5] with a stabilizer group generated by

$$g_v = X_v \prod_{u \in N_v} Z_u \quad \text{for } v \in V(G) \quad (4.12)$$

In fact, any stabilizer state can be transformed to some graph state using only single-qubit Clifford operations [18]. Furthermore, given a stabilizer state, a graph state equivalent under single-qubit Clifford operations can be found efficiently in time $O(n^3)$.

The GHZ states are an important class of stabilizer states given as

$$|\text{GHZ}\rangle_k = \frac{1}{\sqrt{2}} \left(|0\rangle^{\otimes k} + |1\rangle^{\otimes k} \right). \quad (4.13)$$

It is easy to verify that any graph state given by a star or complete graph, i.e. $|S_{V,c}\rangle$ or $|K_V\rangle$, can be turned into a GHZ state on the qubits V using only single-qubit Clifford operations. Furthermore, it is easy to see² that no other graph states are single-qubit Clifford equivalent to the GHZ-states.

In the section 4.4.1 we will discuss local complementations and vertex-deletions on graph states. It turns out that single-qubit Clifford operations (LC), single-qubit Pauli measurements (LPM) and classical communication (CC): LC + LPM + CC, which take graph states to graph states, can be completely characterized by local complementations and vertex-deletions on the corresponding graphs, see sections 4.4.1 and 4.4.2. More concretely, any sequence of single-qubit Clifford operations, mapping graph states to graph states, can be described as some sequence of local complementations on the corresponding graph. Moreover, measuring qubit v of a graph state $|G\rangle$ in the Pauli- X , Pauli- Y or Pauli- Z basis, gives a stabilizer state that is single-qubit Clifford equivalent to $|X_v(G)\rangle$, $|Y_v(G)\rangle$, $|Z_v(G)\rangle$ respectively. The operations X_v , Y_v and Z_v are graph operations consisting of sequences

²This follows from the fact that no other graph is LC-equivalent to the star or complete graph and that graph states are single-qubit Clifford if and only if their corresponding graphs are LC-equivalent, see section 4.4.1

of local complementations together with the deletion of vertex v , which we define in definition 4.4.10. As mentioned the post-measurement state of for example a Pauli- X measurement on qubit v is only single-qubit Clifford equivalent to the graph state $|X_v(G)\rangle$. The single-qubit Clifford operations that take the post-measurement state to $|X_v(G)\rangle$ depend on the outcome of the measurement of the qubit v and act on qubits adjacent to v [5]. This means classical communication is required to announce the measurement result at the vertex v to its neighboring vertices.

We now introduce the notion of a *qubit-minor* which captures exactly which graph states can be reached from some initial graph state under LC + LPM + CC. Formally we define a qubit-minor as:

Definition 4.4.1 (qubit-minor). Assume $|G\rangle$ and $|G'\rangle$ are graph states on the sets of qubits V and U respectively. $|G'\rangle$ is called a qubit-minor of $|G\rangle$ if there exists a sequence of single-qubit Clifford operations (LC), single-qubit Pauli measurements (LPM) and classical communication (CC) that takes $|G\rangle$ to $|G'\rangle$, i.e.

$$|G\rangle \xrightarrow{\text{LC, LPM+CC}} |G'\rangle \otimes |\text{junk}\rangle_{V \setminus U}. \quad (4.14)$$

If $|G'\rangle$ is a qubit-minor of $|G\rangle$, we denote this as

$$|G'\rangle < |G\rangle. \quad (4.15)$$

◊

In chapter 4.5 we will show that the notion of qubit-minors for graph states is equivalent to the notion of *vertex-minors* for graphs. We will define and discuss vertex-minors in section 4.5, however we formally state the relation between vertex-minors here. For a proof see section 4.5.

Theorem 4.4.2. Let $|G\rangle$ and $|G'\rangle$ be two graph states such that no vertex in G' has degree zero. $|G'\rangle$ is then a qubit-minor of $|G\rangle$ if and only if G' is a vertex-minor of G , i.e.

$$|G'\rangle < |G\rangle \iff G' < G. \quad (4.16)$$

◊

Theorem 4.4.2 is very powerful since it allows us to consider graph states under LC+LPM+CC, purely in terms of vertex-minors of graphs. We will in the upcoming chapters use the formalism of vertex-minors to study the computational complexity of `QubitMinor` and provide efficient algorithms for solving this problem. Next, we will in more detail show the equivalence between single-qubit operations on graph states and the mentioned graph operations.

4.4.1. Local Clifford operations

Let's consider the following sequence of single-qubit Clifford operations

$$U_v^{(G)} = \exp\left(-i\frac{\pi}{4}X_v\right) \prod_{u \in N_v} \exp\left(i\frac{\pi}{4}Z_u\right). \quad (4.17)$$

As shown in [5], the operation $U_v^{(G)}$ on the state $|G\rangle$ can be seen as an operation on the graph G since

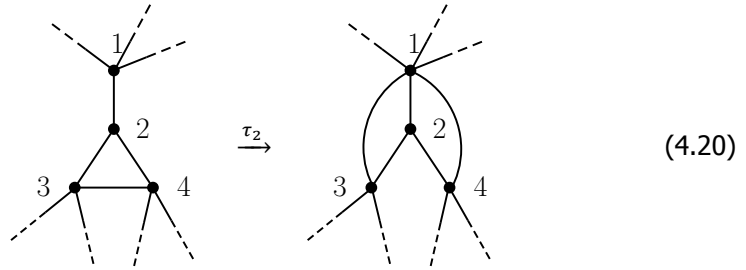
$$U_v^{(G)} |G\rangle = |\tau_v(G)\rangle \tag{4.18}$$

where τ_v is a local complementation on the vertex v , as defined in definition 4.4.3 and illustrated in equation (4.20).

Definition 4.4.3 (local complementation). A local complementation τ_v acts on a vertex v of a graph G by complementing the induced subgraph on the neighborhood of v . The neighborhoods of the graph $\tau_v(G)$ are therefore given by

$$N_u^{\tau_v(G)} = \begin{cases} N_u \Delta (N_v \setminus \{u\}) & \text{if } (u, v) \in E(G) \\ N_u & \text{else} \end{cases}, \tag{4.19}$$

where Δ denotes the symmetric difference between two sets. ◊



Surprisingly, any single-qubit Clifford operation which takes some graph state to another graph state can be seen as some sequence of local complementations on the corresponding graphs. This was proven in [18] and we restate this theorem here.

Theorem 4.4.4 (Van den Nest [18]). *Two graph states $|G\rangle$ and $|G'\rangle$ are equivalent under single-qubit Clifford operations if and only if their corresponding graphs G and G' are related by some sequence of local complementations.* ◊

Let $\mathbf{m} = v_1 v_2 \dots v_l$ be a sequence of vertices of G , then we denote the sequence of local complementations on the vertices in \mathbf{m} as

$$\tau_{\mathbf{m}}(G) = \tau_{v_l} \circ \dots \circ \tau_{v_2} \circ \tau_{v_1}(G). \tag{4.21}$$

If there exists a sequence \mathbf{m} such that $\tau_{\mathbf{m}}(G) = G'$ then we write this as $G \sim_{\text{LC}} G'$. Theorem 4.4.4 can therefore be stated as

$$|G\rangle \sim_{\text{LC}} |G'\rangle \iff G \sim_{\text{LC}} G'. \tag{4.22}$$

Testing whether two graphs are LC-equivalent can be done in time $\mathcal{O}(n^4)$, where n is the size of the graphs, as shown in [31].

Notable about local complementation is its action on star and complete graphs. For a vertex set V and $c \in V$ we have that $\tau_c(S_{V,c}) = K_V$ and for any $v \in V$ we have

$\tau_v(K_V) = S_{V,v}$. This means all star graphs on a vertex set V are equivalent to each other under local complementation and also to the complete graph on V . Moreover, no other graph is equivalent to the star or complete graph.

Another operation which we will make use of is the pivot.

Definition 4.4.5 (Pivot). A pivot ρ_e is a graph operation specified by an edge $e = (u, v)$, taking a graph G to $\rho_e(G)$ such that

$$\rho_e(G) = \tau_v \circ \tau_u \circ \tau_v(G). \quad (4.23)$$

◇

The pivot can simply be specified by an undirected edge since

$$\tau_v \circ \tau_u \circ \tau_v(G) = \tau_u \circ \tau_v \circ \tau_u(G) \quad \text{if } (u, v) \in E(G) \quad (4.24)$$

as shown in [32].

It will be useful to be able to specify a pivot simply by a vertex v . We therefore also introduce the following definition:

Definition 4.4.6. The graph operation ρ_v is specified by a vertex, taking a graph G to $\rho_v(G)$ such that

$$\rho_v(G) = \begin{cases} \rho_{e_v}(G) & \text{if } |N_v| > 0 \\ G & \text{if } |N_v| = 0 \end{cases} \quad (4.25)$$

where e_v is an edge incident on v chosen in some consistent way. For example we could assume that the vertices of G are ordered and that $e_v = (v, \min(N_v))$. The specific choice will not matter but importantly e_v only depends on G and v , and the same therefore holds for $\rho_v(G)$. ◇

Another fundamental operation on a graph is that of vertex-deletion, which relates to measuring a qubit of a graph state in the standard basis [5]. We denote the deletion of vertex v from the graph G as $G \setminus v = G[V(G) \setminus \{v\}]$. It turns out that given a sequence of local complementations and vertex-deletions, acting on some graph, one can always perform the vertex-deletions at the end of the sequence and arrive at the same graph. This fact follows inductively from the following lemma.

Lemma 4.4.7. Let $G = (V, E)$ be a graph and $v, u \in V$ be vertices such that $v \neq u$, then

$$\tau_v(G \setminus u) = \tau_v(G) \setminus u. \quad (4.26)$$

◇

Proof. Note first that it is important that $v \neq u$ since the operation $\tau_v(G \setminus u)$ is otherwise undefined. To prove that the graphs $G_1 = \tau_v(G \setminus u)$ and $G_2 = \tau_v(G) \setminus u$ are equal, we show that the neighborhoods of any vertex in the graphs are the same, i.e. $N_w^{(G_1)} = N_w^{(G_2)}$ for all $w \in V(G) \setminus u$. The local complementation only changes the neighborhoods for vertices which are adjacent to v , so for any vertex $w \neq u$ which is not adjacent to v , we have that

$$N_w^{(G_1)} = N_w^{(G_2)} = N_w^{(G)} \setminus \{u\}. \quad (4.27)$$

On the other hand, for a vertex w which is adjacent to v , its neighborhood becomes

$$N_w^{(G_1)} = (N_w^{(G)} \setminus \{u\}) \Delta \left((N_v^{(G)} \setminus \{u\}) \setminus \{w\} \right) = \left(N_w^{(G)} \Delta (N_v^{(G)} \setminus \{w\}) \right) \setminus \{u\} = N_w^{(G_2)} \quad (4.28)$$

by the definition of a local complementation. \square

4.4.2. Local Pauli measurements

How should the corresponding graph of a graph state be updated when a measurement is performed? In [33] it is shown how the Pauli projectors $P_v^{(X,\pm)}$, $P_v^{(Y,\pm)}$, $P_v^{(Z,\pm)}$ act on graph states³

$$P_v^{(Z,\pm)} |G\rangle = \frac{1}{2} |Z, \pm\rangle_v \otimes U_v^{(Z,\pm)} |G \setminus v\rangle \quad (4.29)$$

$$P_v^{(Y,\pm)} |G\rangle = \frac{1}{2} |Y, \pm\rangle_v \otimes U_v^{(Y,\pm)} |\tau_v(G) \setminus v\rangle \quad (4.30)$$

$$P_v^{(X,\pm)} |G\rangle = \frac{1}{2} |X, \pm\rangle_v \otimes U_e^{(X,\pm)} |\rho_e(G) \setminus v\rangle \quad \text{if } |N_v| > 0 \quad (4.31)$$

where e is an edge of G incident on the vertex v . Choosing a different edge e' incident on v gives a single-qubit Clifford-equivalent graph state, i.e. $|\rho_{e'}(G) \setminus v\rangle \sim_{\text{LC}} |\rho_e(G) \setminus v\rangle$. The operators $U_v^{(P,\pm)}$ are sequences of single-qubit Clifford operations and take the post-measurement state to a graph state. The exact form of these correction operators can be found in [33] but we will only need the ones for measurements in the standard (Z -) basis, which are given by

$$U_v^{(Z,+)} = \mathbb{I}_v, \quad U_v^{(Z,-)} = \prod_{u \in N_v} Z_u. \quad (4.32)$$

Since these correction operators of eqs. (4.29) to (4.31) are sequences of single-qubit Clifford operations it is therefore the case that

$$|G \setminus v\rangle, \quad |\tau_v(G) \setminus v\rangle, \quad |\rho_e(G) \setminus v\rangle \quad (4.33)$$

are all qubit-minors of $|G\rangle$. Furthermore, Bouchet proved the following.

Lemma 4.4.8 (Bouchet, (9.2) in [32]). *If $G \sim_{\text{LC}} G'$ then $G' \setminus v$ is LC-equivalent to $G \setminus v$, $\tau_v(G) \setminus v$ or $\rho_e(G) \setminus v$, where e is some fixed edge incident on v in G . \diamond*

We therefore have the following lemma.

Lemma 4.4.9. *Let $|G\rangle$ be a graph state, $v \in V(G)$ be a vertex and $e \in E(G)$ be an edge incident on v . Furthermore, assume that $|G'\rangle$ is a qubit-minor of $|G\rangle$, where $V(G') = V(G) \setminus v$ and that G' has no vertices of degree zero. Then $|G'\rangle$ is single-qubit Clifford-equivalent to at least one of the three states in equation (4.33). \diamond*

³For the special case when $|N_v| = 0$, a measurement in the X -basis does not change the graph state since this is then $|G\rangle = |+\rangle_v \otimes |G \setminus v\rangle$.

Proof. Since $|G'\rangle$ is a qubit-minor of $|G\rangle$ we know that there exists a sequence \mathcal{W} of single-qubit Clifford operations, single-qubit Pauli measurements and classical communication that takes $|G\rangle$ to $|G'\rangle$, by definition. Any single-qubit Pauli measurement on a qubit u gives a product states between qubit u and the rest of the qubits, see equations (4.29)-(4.31). The sequence \mathcal{W} cannot therefore contain a single-qubit Pauli measurement on a qubit u , different from v , since u has non-zero degree in the graph G' by assumption. Without loss of generality we can in fact assume that \mathcal{W} is a sequence of single-qubit Clifford operations followed by a measurement of v in the standard basis and then by another sequence of single-qubit Clifford operations. The reason why it is sufficient to consider a measurement in the standard basis is that any other Pauli measurement can be simulated by performing some single-qubit Clifford operation followed by a measurement in the standard basis. Assume that the sequence of single-qubit Clifford operations before (after) the measurement is described by the sequences of local complementations \mathbf{m} (\mathbf{m}'), which exists due to theorem 4.4.4. We therefore have that

$$\tau_{\mathbf{m}'}(\tau_{\mathbf{m}}(G) \setminus v) = G'. \quad (4.34)$$

Using lemma 4.4.8 we have that $\tau_{\mathbf{m}}(G) \setminus v$, and therefore G' , is LC-equivalent to either $G \setminus v$, $\tau_v(G) \setminus v$ or $\rho_e(G) \setminus v$. Finally, by theorem 4.4.4 the lemma follows. \square

To simplify notation we also introduce the following three graph operations, which exactly captures how Pauli measurements act on graph states.

Definition 4.4.10. The graph operations X_v , Y_v and Z_v , specified with a vertex v , act on a graph G by transforming it to

$$X_v(G) = \rho_v(G) \setminus v, \quad Y_v(G) = \tau_v(G) \setminus v, \quad Z_v(G) = G \setminus v \quad (4.35)$$

When we need to specify which edge incident on v the pivot of X_v acts on, we write $X_v^{(u)}(G) = \rho_{(u,v)}(G) \setminus v$. We call the operations X_v , Y_v and Z_v for *measurement operations*. \diamond

Equations (4.36) to (4.38) show examples of how these operations can act on

graphs.

$$Z_6 \left(\begin{array}{c} 3 \\ 2 \bullet \quad \bullet 4 \\ 0 \\ 1 \bullet \quad \bullet 5 \\ 6 \end{array} \right) = \begin{array}{c} 3 \\ 2 \bullet \quad \bullet 4 \\ 0 \\ 1 \bullet \quad \bullet 5 \end{array} \quad (4.36)$$

$$Y_5 \left(\begin{array}{c} 1 \\ 2 \bullet \quad \bullet 4 \\ 3 \bullet \quad \bullet 5 \end{array} \right) = \begin{array}{c} 1 \\ 2 \bullet \quad \bullet 4 \\ 3 \bullet \quad \bullet 5 \end{array} \quad (4.37)$$

$$X_1^{(2)} \left(\begin{array}{c} 3 \\ 1 \bullet \quad \bullet 2 \\ 4 \bullet \quad \bullet 5 \quad \bullet 6 \end{array} \right) = \begin{array}{c} 3 \\ 2 \bullet \quad \bullet 4 \\ 5 \bullet \quad \bullet 6 \end{array} \quad (4.38)$$

The operation $X_v^{(u)}$ is the most complicated one, so we will here quickly describe what happens to a graph when $X_v^{(u)}$ is applied. One can check that after the operation $X_v^{(u)}$, the vertex u will have the neighbors that v previously had, except v itself. Furthermore, some edges between vertices in $(N_v \cup N_u) \setminus \{u, v\}$ will be complemented, i.e. removed if present or added if not. To know which of these edges gets complemented, let's introduce the following three sets

$$V_{vu} = N_v \cap N_u, \quad V_v = N_v \setminus (N_u \cup \{u\}), \quad V_u = N_u \setminus (N_v \cup \{v\}) \quad (4.39)$$

which form a partition of $(N_v \cup N_u) \setminus \{u, v\}$. In eq. (4.38), these sets are $V_{12} = \{3\}$, $V_1 = \{4\}$ and $V_2 = \{5, 6\}$. An edge (w_1, w_2) between vertices in $(N_v \cup N_u) \setminus \{u, v\}$ gets complemented if and only if w_1 and w_2 belong to different sets of the partition (V_{vu}, V_v, V_u) . All other edges in the graph, i.e. edges containing a vertex not in $N_v \cup N_u$, will be unchanged.

4.5. Vertex-minors

Using the two operations local complementation and vertex-deletion, we can formulate the notion of a vertex-minor of a graph.

Definition 4.5.1 (Vertex-minor). A graph G' is called a vertex-minor of G if and only if there exist a sequence of local complementations and vertex-deletions that takes G to G' . If G' is a vertex-minor of G we write this as

$$G' < G \quad (4.40)$$

and if G' is not a vertex-minor of G then

$$G' \not\prec G. \quad (4.41)$$

◇

Equivalently, due to lemma 4.4.7, G' is called a vertex-minor of G if there exists a sequence of vertices \mathbf{m} such that

$$\tau_{\mathbf{m}}(G)[V(G')] = G' \quad (4.42)$$

Vertex-minors were first studied in [32] but by the name of l -reductions. Note that if G_1 and G_2 are two LC-equivalent graphs, then $G' < G_1$ if and only if $G' < G_2$.

In the previous sections we have seen that single-qubit Clifford operations that take graph states to graph states can be seen as local complementations on the corresponding graph and similarly for single-qubit Pauli measurements and vertex-deletions. The relation between qubit-minors and vertex-minors is captured by theorem 4.4.2 which we now prove.

Proof of theorem 4.4.2. Assume first that $|G'\rangle$ is a qubit-minor of $|G\rangle$. By the same arguments as in the proof of lemma 4.4.9 we then have that there exists a sequence of vertices m such that

$$\tau_{\mathbf{m}}(G)[V(G')] = G' \quad (4.43)$$

and by definition that G' is a vertex-minor of G . Assume now on the other hand that G' is a vertex-minor of G , i.e. that $\tau_{\mathbf{m}}(G)[V(G')] = G'$ for some \mathbf{m} . We can then go from $|G\rangle$ to $|G'\rangle$ by simply performing the single-qubit Clifford operations corresponding to the sequence of local complementation specified by \mathbf{m} and then measure the qubits $V(G) \setminus V(G')$ in the standard basis. If the correct corrections, i.e. $U_v^{(Z,\pm)}$, are applied after the measurements, the state $|G'\rangle$ is reached. □

So to check whether a graph state has a certain qubit-minor we can check if the corresponding graph has a certain vertex-minor. Note that one can also include the case where G' has vertices of degree zero. Let's denote the vertices of G' which have degree zero as U . We then have that

$$|G'\rangle < |G\rangle \Leftrightarrow G'[V(G) \setminus U] < G. \quad (4.44)$$

It is interesting to consider under which conditions a graph G' is a vertex-minor of another graph G . As theorem 4.5.3 below states, to decide whether $G' < G$ it is sufficient to check whether G' is LC-equivalent to at least one out of $3^{|V(G)|-|V(G')|}$ graphs.

It turns out that the three operations $\{X_v, Y_v, Z_v\}$ from definition 4.4.10 are sufficient to check whether some graph is a vertex-minor of another graph. This is formalized in theorem 4.5.3 below. We first introduce some definitions needed to prove the theorem.

It is sometimes convenient to denote a sequence of the operations X, Y, Z on a sequence of vertices:

Definition 4.5.2. Let $\mathbf{u} = (u_1, \dots, u_n)$ be a sequence of vertices. Let $P_{\mathbf{u}} = P_{u_n} \circ \dots \circ P_{u_1}$ denote a sequence of operations, where $P_{u_i} \in \{X_{u_i}, Y_{u_i}, Z_{u_i}\}$ for each i . We call $P_{\mathbf{u}}$ a *measurement operation sequence*. Let $\mathcal{P}_{\mathbf{u}}$ denote all such $P_{\mathbf{u}}$, i.e.

$$\mathcal{P}_{\mathbf{u}} = \{P_{u_n} \circ \dots \circ P_{u_1} : P_{u_i} \in \{X_{u_i}, Y_{u_i}, Z_{u_i}\}\} \quad (4.45)$$

Furthermore, let \mathcal{P}_U denote the union of all $\mathcal{P}_{\pi(\mathbf{u})}$ for all permutations π , where U is the set $\{u_1, \dots, u_n\}$. \diamond

We are now ready to prove a generalization of lemma 4.4.9, which algorithm 5.1 is built on.

Theorem 4.5.3. Let G and G' be two graphs and U be the set $V(G) \setminus V(G') = \{v_1, \dots, v_{n-k}\}$. Then we have that

$$G' < G \iff \exists P \in \mathcal{P}_U : G' \sim_{\text{LC}} P(G). \quad (4.46)$$

\diamond

Proof. If there exists a P in \mathcal{P}_U such that $G' \sim_{\text{LC}} P(G)$ then we clearly have that $G' < G$, since any such P is some sequence of local complementations and vertex-deletions. Assume now that $G' < G$. We will prove by induction on $n - k$ that there exists a P in \mathcal{P}_U such that $G' \sim_{\text{LC}} P(G)$. For $n - k = 1$ this follows directly from lemma 4.4.8. Assume therefore that it is true for $n - k = l$. We now show that this implies that it is also true for $n - k = l + 1$. Since $G' < G$ we know that $\tau_m(G)[V(G')] = G'$ for some m . Let v be a vertex in $V(G')$ and consider the graph $\tilde{G} = \tau_m(G)[V(G') \cup \{v\}]$. Note that $G' = \tilde{G} \setminus v$. Clearly we have that $G' < \tilde{G} < G$ and by the induction assumption we know that

$$\exists P \in \mathcal{P}_{U \setminus \{v\}} : \tilde{G} \sim_{\text{LC}} P(G). \quad (4.47)$$

Then from lemma 4.4.8 we know that $G' = \tilde{G} \setminus v$ is LC-equivalent to at least one of the following graphs

$$X_v(P(G)), \quad Y_v(P(G)), \quad Z_v(P(G)) \quad (4.48)$$

and the theorem follows. \square

Note that in 4.5.3 $\mathcal{P}_{\mathbf{u}}$ is indexed simply with the set associated to the word \mathbf{u} since the statement is independent of the ordering of the elements of \mathbf{u} . A direct corollary of theorem 4.5.3 is therefore:

Corollary 4.5.3.1. Let G and G' be two graphs. Furthermore, let \mathbf{u} and \mathbf{u}' be two ordered tuples such that each element of $V(G) \setminus V(G')$ occurs exactly once in both \mathbf{u} and \mathbf{u}' . Then we have that

$$\exists P \in \mathcal{P}_{\mathbf{u}} : G' \sim_{\text{LC}} P(G) \iff \exists P \in \mathcal{P}_{\mathbf{u}'} : G' \sim_{\text{LC}} P(G). \quad (4.49)$$

\diamond

Proof. This follows directly from theorem 4.5.3 since both sides in eq. (4.49) are true if and only if $G' < G$. \square

Note that theorem 4.5.3 does not give an efficient method to check if G' is a vertex-minor of G , since the set $\mathcal{P}_{\mathbf{u}}$ is of exponential size for all \mathbf{u} . However, a non-efficient algorithm can be constructed, see algorithm 5.1. To study this problem further we formally define the vertex-minor problem.

Problem 4.5.4 (*VertexMinor*). Given a graph G and a graph G' defined on a subset of $V(G)$, decide whether G' is a vertex-minor of G . \diamond

We will often consider the special case where G' is a star graph $S_{V'}$ defined on a subset V' of $V(G)$. Remember that a graph state described by a star graph is single-qubit Clifford equivalent to a GHZ-state. Thus checking if $S_{V'}$ is a vertex-minor of G is equivalent to checking if $|G\rangle$ can be transformed to GHZ-state on the qubits V' by only using LC + LPM + CC. We will give this problem a separate name.

Problem 4.5.5 (*StarVertexMinor*). Given a graph G and a vertex subset V' of $V(G)$, decide whether $S_{V'}$ is a vertex-minor of G . \diamond

Note that we have not specified which star graph on V' we use. This is not ambiguous since all star graphs on V' are equivalent under local complementation. In the rest of the text we will often leave the choice of star graph open.

4.6. Rank-width

In this section we introduce the notion of rank-width, which is a complexity measure of a graph. It is in some ways similar to the tree-width, introduced in [34]. The tree-width captures essentially how tree-like the graph is. This is useful for finding algorithms for problems on graphs of bounded tree-width, motivated by the fact that many graph problems are easy on trees. More on algorithms for problems on graphs of bounded tree-width can be found in [25]. Rank-width, compared to tree-width, captures a larger class of graphs with similar complexity. For example, the complete graph has very low complexity, due to its highly symmetric nature, but the tree-width is in this case maximal. On the other hand the rank-width is one for both trees and complete graphs. In fact, it turns out that the graphs of rank-width one are exactly the distance-hereditary graphs, see [19].

We start by defining the cut-rank of a graph. To do this we will use the following notation for a graph G with vertices V and adjacency matrix Γ and two subsets of the vertices $A, B \subseteq V$; $\Gamma[A, B]$ is the $|A| \times |B|$ -matrix describing the connections between the sets A and B . So, for $a \in A$ and $b \in B$, the element $(\Gamma[A, B])_{ab}$ is 1 if (a, b) is an edge in G and 0 otherwise.

Definition 4.6.1 (cut-rank). Let's assume that A is a subset of the vertices V of some graph G with adjacency matrix Γ . The cut-rank $\text{cutrk}_A(G)$ of G with respect to A , is then defined as

$$\text{cutrk}_A(G) \equiv \text{rank}_{\mathbb{F}_2}(\Gamma[A, V \setminus A]), \quad (4.50)$$

where $\text{rank}_{\mathbb{F}_2}$ is the rank over the finite field of order two. \diamond

Note that the cut-rank is symmetric in the sense that

$$\text{cutrk}_A(G) = \text{rank}_{\mathbb{F}_2}(\Gamma[A, V \setminus A]) = \text{rank}_{\mathbb{F}_2}(\Gamma[V \setminus A, A]^\top) = \text{cutrk}_{V \setminus A}(G). \quad (4.51)$$

Interestingly the cut-rank with respect to A of a graph G is in fact equal to the Schmidt-rank of the state $|G\rangle$ with respect to the bipartition $(A, V \setminus A)$.⁴

Next we define what is called a rank-decomposition of a graph.

Definition 4.6.2 (rank-decomposition). A rank-decomposition of a graph G is a pair $\mathcal{R} = (\mathcal{T}, \mu)$, where \mathcal{T} is a subcubic tree and μ is a bijection $\mu : V(G) \rightarrow \{l : l \text{ is a leaf of } \mathcal{T}\}$. A subcubic tree is a tree with at least two vertices and each vertex has degree less or equal to 3. Any edge e in \mathcal{T} splits the tree into two connected components upon deletion and therefore induces a partition (A_e, B_e) of the leaves. The width of an edge e of the subcubic tree is defined as the cut-rank of the corresponding partition. Furthermore the width of the rank-decomposition is defined as the maximum width over all edges, i.e.

$$\text{width}_{\mathcal{R}}(G) \equiv \max_{e \in E(\mathcal{T})} \text{cutrk}_{\mu^{-1}(A_e)}(G). \quad (4.52)$$

To simplify notation we write the cut-rank induced by a rank-decomposition (\mathcal{T}, μ) and an edge e as

$$\text{cutrk}_{\mu^{-1}(\mathcal{T}, e)}(G) \equiv \text{cutrk}_{\mu^{-1}(A_e)}(G). \quad (4.53)$$

◇

This allows us to define the rank-width of a graph.

Definition 4.6.3 (rank-width). The rank-width $\text{rwd}(G)$ of a graph G is the minimum width over all rank-decompositions, i.e.

$$\text{rwd}(G) \equiv \min_{\mathcal{R}} \text{width}_{\mathcal{R}}(G) = \min_{(\mathcal{T}, \mu)} \max_{e \in E(\mathcal{T})} \text{cutrk}_{\mu^{-1}(\mathcal{T}, e)}(G). \quad (4.54)$$

◇

The rank-width of the graph G is related to the entanglement of the state $|G\rangle$, although as a relatively unknown entanglement monotone. In [22] the corresponding entanglement monotone is called the *Schmidt-rank width* and is defined for general quantum states. For graph states, the Schmidt-rank width of the state and the rank-width of the corresponding graph coincide. There they also give an interpretation of the Schmidt-rank width as a quantifier for the optimal description of the state using a tree tensor network.

4.7. Circle graphs

Here we introduce circle graphs and representations of these under the action of local complementations. Circle graphs are graphs with edges represented as intersections of chords on a circle. These graphs are also sometimes called alternance

⁴This is proven in [5], see proposition 10.

graphs since they can be described by a double occurrence word such that the edges of the graph are then given by the alternances induced by this word. We will make use of the latter description here, which was introduced by Bouchet in [35] and also described in [36]. This description is also related to yet another way to represent circle graphs, as Eulerian tours of 4-regular multi-graphs, introduced by Kotzig in [37]. For an overview and the history of circle graphs see for example the book by Golombic [38].

4.7.1. Double occurrence words

Let us first define double occurrence words and equivalence classes of these. This will allow us to define circle graphs.

Definition 4.7.1 (Double occurrence word). A double occurrence word \mathbf{X} is a word with letters in some set V , such that each element in V occurs exactly twice in \mathbf{X} . Given a double occurrence word \mathbf{X} we will write $V(\mathbf{X}) = V$ for its set of letters. \diamond

Definition 4.7.2 (Equivalence class of double occurrence words). We say that a double occurrence word \mathbf{Y} is equivalent to another \mathbf{X} , i.e. $\mathbf{Y} \sim \mathbf{X}$, if \mathbf{Y} is equal to \mathbf{X} , the mirror $\tilde{\mathbf{X}}$ or any cyclic permutation of \mathbf{X} or $\tilde{\mathbf{X}}$. We denote by $\mathbf{d}_{\mathbf{X}} = \{\mathbf{Y} : \mathbf{Y} \sim \mathbf{X}\}$ the equivalence class of \mathbf{X} , i.e. the set of words equivalent to \mathbf{X} . \diamond

Next we define alternances of these equivalence classes, which will represent the edges of an alternance graph.

Definition 4.7.3 (Alternance). An *alternance* (u, v) of the equivalence class $\mathbf{d}_{\mathbf{X}}$ is a pair of distinct elements $u, v \in V$ such that a double occurrence word of the form $\dots u \dots v \dots u \dots v \dots$ is in $\mathbf{d}_{\mathbf{X}}$. \diamond

Note that if (u, v) is an alternance of $\mathbf{d}_{\mathbf{X}}$ then so is (v, u) , since the mirror of any word in $\mathbf{d}_{\mathbf{X}}$ is also in $\mathbf{d}_{\mathbf{X}}$.

Definition 4.7.4 (Alternance graph). The alternance graph $\mathcal{A}(\mathbf{X})$ of a double occurrence word \mathbf{X} is a graph with vertices $V(\mathbf{X})$ and edges given exactly by the alternances of $\mathbf{d}_{\mathbf{X}}$, i.e.

$$E(\mathcal{A}(\mathbf{X})) = \{(u, v) \in V(\mathbf{X}) \times V(\mathbf{X}) : (u, v) \text{ is an alternance of } \mathbf{d}_{\mathbf{X}}\} \quad (4.55)$$

\diamond

Note that since $\mathcal{A}(\mathbf{X})$ only depends on the equivalence class of \mathbf{X} , the alternance graphs $\mathcal{A}(\mathbf{X})$ and $\mathcal{A}(\mathbf{Y})$ are equal if $\mathbf{X} \sim \mathbf{Y}$. Now we can formally define circle graphs.

Definition 4.7.5 (Circle graph). A graph G which is the alternance graph of some double occurrence word \mathbf{X} is called a circle graph. \diamond

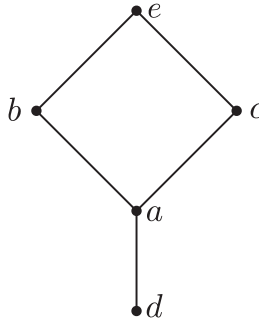


Figure 4.1: An example of a circle graph induced by the double-occurrence word $adcb aebced$.

4.7.2. Eulerian tours on 4-regular multi-graphs

There is yet another way to represent circle graphs, closely related to double occurrence words, as Eulerian tours of 4-regular multi-graphs.

Definition 4.7.6 (Eulerian tour). Let F be a connected 4-regular multi-graph. An Eulerian tour U on F is a tour that visits each edge in F exactly once. \diamond

Any 4-regular multi-graph is Eulerian, i.e. has a Eulerian tour, since each vertex has even degree [39].

Furthermore, any Eulerian tour on a 4-regular multi-graph F traverses each vertex exactly twice, except for the vertex which is both the start and the end of the tour. Such a Eulerian tour induces therefore a double occurrence word, the letters of which are the vertices of F , and consequently a circle graph as described in the following definition.

Definition 4.7.7 (Induced double occurrence word). Let F be a connected 4-regular multi-graph on k vertices $V(F)$. Let U be a Eulerian tour on F of the form

$$U = x_1 e_1 x_2 \dots x_{2k-1} e_{2k-1} x_{2k} e_{2k} x_1. \quad (4.56)$$

with $x_i \in V$. Note that every element of V occurs exactly twice in U . From a Eulerian tour U as in eq. (4.56) we define an induced double occurrence word as

$$m(U) = x_1 x_2 \dots x_{2k-1} x_{2k}. \quad (4.57)$$

To denote the alternance graph given by the double occurrence word induced by a Eulerian tour, we will write $\mathcal{A}(U) \equiv \mathcal{A}(m(U))$. \diamond

Similarly to double occurrence words, we also introduce equivalence classes of Eulerian tours under cyclic permutation or reversal of the tour.

Definition 4.7.8 (Equivalence class of Eulerian tours). Let F be a connected 4-regular multi-graph and U be an Eulerian tour on F . We say that an Eulerian tour U' on F is equivalent to U , i.e. $U \sim U'$, if U' is equal to U , the reversal \tilde{U} or any cyclic permutation of U or \tilde{U} . We denote by \mathbf{t}_U the equivalence class of U , i.e. the set of Eulerian tours on F which are equivalent to U . \diamond

It is clear that if the Eulerian tours U and U' on a 4-regular multi-graph F are equivalent, then so are the double occurrence words $m(U)$ and $m(U')$. Furthermore, as for double occurrence words, two equivalent Eulerian tours on a connected 4-regular multi-graph induce the same alternance graph.

4.7.3. Local complementations on circle graphs

We will now introduce an operation $\tilde{\tau}_v$ on double occurrence words that will be the equivalent of performing a local complementation on the corresponding alternance graph.

Definition 4.7.9 ($\tilde{\tau}_v$). Let \mathbf{X} be a double occurrence word and v be an element in $V(\mathbf{X})$. We can then always find sub-words \mathbf{A} , \mathbf{B} and \mathbf{C} not containing v , such that $\mathbf{X} = \mathbf{A}v\mathbf{B}v\mathbf{C}$. Note that some of the sub-words \mathbf{A} , \mathbf{B} and \mathbf{C} are possibly empty. The operation $\tilde{\tau}_v$ acting on a double occurrence word is then defined as

$$\tilde{\tau}(\mathbf{A}v\mathbf{B}v\mathbf{C}) = \mathbf{A}v\tilde{\mathbf{B}}v\mathbf{C}. \quad (4.58)$$

If $\mathbf{v} = (v_1, \dots, v_l)$ is a sequence of elements of $V(\mathbf{X})$ we use the notation $\tilde{\tau}_{\mathbf{v}}(\mathbf{X}) = \tilde{\tau}_{v_l} \circ \dots \circ \tilde{\tau}_{v_1}(\mathbf{X})$. \diamond

The operation $\tilde{\tau}_v$ in the above definition maps equivalence classes to equivalence classes, as defined in definition 4.7.2. That is, if $\mathbf{X} \sim \mathbf{Y}$ and $v \in V(\mathbf{X})$, then $\tilde{\tau}_v(\mathbf{X}) \sim \tilde{\tau}_v(\mathbf{Y})$. For example, assume that \mathbf{Y} is the mirror of \mathbf{X} , i.e. $\mathbf{Y} = \tilde{\mathbf{X}}$. Then we see that

$$\tilde{\tau}_v(\mathbf{X}) = \mathbf{A}v\tilde{\mathbf{B}}v\mathbf{C} \sim \widetilde{\mathbf{A}v\mathbf{B}v\mathbf{C}} = \tilde{\mathbf{C}}v\mathbf{B}v\tilde{\mathbf{A}} = \tilde{\tau}_v(\mathbf{Y}). \quad (4.59)$$

The case when \mathbf{Y} is obtained by a cyclic permutation of \mathbf{X} can be checked similarly.

In [36] it was shown that the alternance graph of $\mathcal{A}(\tilde{\tau}_v(\mathbf{X}))$, where \mathbf{X} is a double occurrence word and $v \in V(\mathbf{X})$, is the same as the graph obtained by performing a local complementation at v , i.e.

$$\tau_v(\mathcal{A}(\mathbf{X})) = \mathcal{A}(\tilde{\tau}_v(\mathbf{X})). \quad (4.60)$$

Similar to the above we can also define an operation $\bar{\tau}_v$ on Eulerian tours U on 4-regular multi-graphs which also has the effect of a local complementation on the graph $\mathcal{A}(U)$.

Definition 4.7.10 ($\bar{\tau}_v$). Let F be a connected 4-regular multi-graph. Let U be a Eulerian tour on F and v be a vertex in V . Let P_v be the first subtrail of U that starts and ends at v , i.e. $U = U_1P_vU_2$, from some U_1 and U_2 . We define $\bar{\tau}_v(U)$ to be the Eulerian tour obtained by traversing U_1 , the reversal of P_v and then U_2 , i.e. $\bar{\tau}_v(U) = U_1\tilde{P}_vU_2$. When $\mathbf{v} = v_1 \dots v_l$ is a sequence of vertices in V we write $\bar{\tau}_{\mathbf{v}}(U) \equiv \bar{\tau}_{v_l} \circ \dots \circ \bar{\tau}_{v_1}(U)$. \diamond

Note in particular that $\bar{\tau}_v(U)$, where U is an Eulerian tour on F , is also a Eulerian tour on F .

We have now defined τ -operations on circle graphs, $\tilde{\tau}$ -operations on double occurrence words and $\bar{\tau}$ -operations on Eulerian tours of 4-regular multi-graphs.

They are given similar names since they are in some sense the same operation but in different representations of circle graphs. To see this note that

$$m(\bar{\tau}_v(U)) = m(U_1 \tilde{P}_v U_2) = \tilde{\tau}_v(m(U)) \quad (4.61)$$

where $U = U_1 P_v U_2$ as in definition 4.7.10. From eq. (4.60) and the shorthand $\mathcal{A}(U) = \mathcal{A}(m(U))$ we also have that

$$\mathcal{A}(\bar{\tau}_v(U)) = \mathcal{A}(\tilde{\tau}_v(m(U))) = \tau_v(\mathcal{A}(U)). \quad (4.62)$$

The operation $\bar{\tau}_v$ on Eulerian tours of 4-regular multi-graphs was introduced by Kotzig in [40], where he called it a κ -transformation.

As stated by Bouchet in [36], Kotzig [40] proved that any two Eulerian tours of a 4-regular multi-graph are related by a sequence of κ -transformations.

4

Theorem 4.7.11 (Proposition 4.1 in [36], [40]). *Let U and U' be Eulerian tours on the same connected 4-regular multi-graph. Then there exists a sequence \mathbf{v} such that $\tau_{\mathbf{v}}(U) \sim U'$.* \diamond

4.7.4. Vertex-deletion on circle graphs

When we are considering vertex-minors of circle graphs, it is useful to have an operation on the double occurrence word that corresponds to the deletion of a vertex in the corresponding alternance graph. Let $\mathbf{X} = \mathbf{A}v\mathbf{B}v\mathbf{C}$ be a double occurrence word and v be an element in $V(\mathbf{X})$. We will denote by $\mathbf{X} \setminus v$ the deletion of the element v , i.e.

$$\mathbf{X} \setminus v \equiv (\mathbf{A}v\mathbf{B}v\mathbf{C}) \setminus v = \mathbf{ABC}. \quad (4.63)$$

The resulting word \mathbf{ABC} is also a double occurrence word and furthermore we have that

$$\mathcal{A}(\mathbf{X}) \setminus v = \mathcal{A}(\mathbf{X} \setminus v). \quad (4.64)$$

If $W = \{w_1, w_2, \dots, w_l\}$ is a subset of V , we will write $\mathbf{X} \setminus W$ as the deletion of all elements in W , i.e.

$$\mathbf{X} \setminus W = (\dots((\mathbf{X} \setminus w_1) \setminus w_2) \dots) \setminus w_l. \quad (4.65)$$

Connected to this we can also define an induced double occurrence sub-word $\mathbf{X}[W] = \mathbf{X} \setminus (V \setminus W)$. The reason for calling this an induced double occurrence sub-word stems from its relation to induced subgraphs of the alternance graph as

$$\mathcal{A}(\mathbf{X})[W] = \mathcal{A}(\mathbf{X}[W]). \quad (4.66)$$

4.7.5. Vertex-minors of circle graphs

Since we now have expressions for local complementation and vertex deletion on circle graphs in terms of double occurrence words, we can consider vertex-minors of circle graphs completely in terms of double occurrence words. More precisely we have the following theorem.

Theorem 4.7.12. *Let G be a circle graph such that $G = \mathcal{A}(\mathbf{X})$ for some double occurrence word \mathbf{X} . Then G' is a vertex-minor of G if and only if there exist a sequence \mathbf{v} of elements in $V(G) = V(\mathbf{X})$ such that*

$$G' = \mathcal{A}(\tilde{\tau}_{\mathbf{v}}(\mathbf{X})[V(G')]). \quad (4.67)$$

◊

Proof. By using eq. (4.66) and eq. (4.62) on the right hand side of eq. (4.67) we have that

$$\mathcal{A}(\tilde{\tau}_{\mathbf{v}}(\mathbf{X})[V(G')]) = \mathcal{A}(\tilde{\tau}_{\mathbf{v}}(\mathbf{X})) [V(G')] = \tau_{\mathbf{v}}(\mathcal{A}(\mathbf{X})) [V(G')] \quad (4.68)$$

Since G' is a vertex-minor of $G = \mathcal{A}(\mathbf{X})$ if and only if there exist a sequence \mathbf{v} of elements in $V(G)$ such that

$$G' = \tau_{\mathbf{v}}(G) [V(G')] \quad (4.69)$$

the theorem follows. \square

We can also consider vertex minors of circle graphs in terms of their representations as Eulerian tours on connected 4-regular multi-graphs, which we will use in section 6.2 to prove that `VertexMinor` is NP-Complete . Theorem 4.7.11, together with eq. (4.62), implies that connected 4-regular multi-graphs describe equivalence classes of circle graphs under local complementations. Bouchet pointed out this fact in [36]. We formalize this here as a theorem together with a formal proof:

Theorem 4.7.13. *Let U be an Eulerian tour of a connected 4-regular multi-graph F with vertices V . Then (1) any graph LC-equivalent to $\mathcal{A}(U)$ is an alternance graph of some Eulerian tour of F and (2) any alternance graph of a Eulerian tour of F is a graph LC-equivalent to $\mathcal{A}(U)$. \diamond*

Proof. We start by proving (1), so let us therefore assume that G is a graph LC-equivalent to $\mathcal{A}(U)$. This means, by definition, that there exist a sequence \mathbf{v} of vertices in V such that $G = \tau_{\mathbf{v}}(\mathcal{A}(U))$. By using eq. (4.62) we have that

$$G = \mathcal{A}(\tilde{\tau}_{\mathbf{v}}(U)). \quad (4.70)$$

which shows that G is an alternance graph induced by a Eulerian tour of F , since $\tilde{\tau}_{\mathbf{v}}(U)$ is a Eulerian tour on F . To prove (2), assume that U' is a Eulerian tour of F . We will now prove that the alternance graph of U' , $\mathcal{A}(U')$, is LC-equivalent to $\mathcal{A}(U)$. By theorem 4.7.11, we know that there exists a sequence of $\tilde{\tau}_v$ -transformations that relates U and U' , i.e. there exist a sequence \mathbf{v} such that

$$\tilde{\tau}_{\mathbf{v}}(U) \sim U'. \quad (4.71)$$

Since these Eulerian tours are equivalent, their induced alternance graphs are equal, i.e.

$$\mathcal{A}(\tilde{\tau}_{\mathbf{v}}(U)) = \mathcal{A}(U'). \quad (4.72)$$

Finally, using eq. (4.62) on the above equation gives

$$\tau_{\mathbf{v}}(\mathcal{A}(U)) = \mathcal{A}(U') \quad (4.73)$$

which shows that $\mathcal{A}(U)$ and $\mathcal{A}(U')$ are indeed LC-equivalent. \square

Similarly to theorem 4.7.12 we can decide if a circle graph has a certain vertex-minor by considering Eulerian tours of a 4-regular graph, which is captured in the following theorem.

Theorem 4.7.14. *Let F be a connected 4-regular multi-graph and let G be a circle graph such that $\mathcal{A}(U)$ for some Eulerian tour U on F . Then G' is a vertex-minor of G if and only if there exist a Eulerian tour U' on F such that*

$$G' = \mathcal{A}(m(U'))[V(G')]. \quad (4.74)$$

◊

Proof. Lets first assume that G' is a vertex-minor of G . This means that there exists a sequence \mathbf{v} such that $G' = \tau_{\mathbf{v}}(G)[V(G')]$. Since $G = \mathcal{A}(U)$ we have that

$$G' = \tau_{\mathbf{v}}(\mathcal{A}(U))[V(G')] \quad (4.75)$$

$$= \mathcal{A}(\bar{\tau}_{\mathbf{v}}(U))[V(G')] \quad (4.76)$$

$$= \mathcal{A}(m(\bar{\tau}_{\mathbf{v}}(U)))[V(G')] \quad (4.77)$$

where we used eq. (4.62) in the first step and eq. (4.66) in the second. We therefore see that $U' = \bar{\tau}_{\mathbf{v}}(U)$ is a Eulerian tour on F satisfying eq. (4.74).

To prove the converse let us assume that there exist a Eulerian tour U' on F satisfying eq. (4.74). From theorem 4.7.11 we know that there exist a sequence \mathbf{v} such that $U' = \bar{\tau}_{\mathbf{v}}(U)$. We can then replace U' by $\bar{\tau}_{\mathbf{v}}(U)$ in eq. (4.74) such that

$$G' = \mathcal{A}(m(\bar{\tau}_{\mathbf{v}}(U)))[V(G')] \quad (4.78)$$

$$= \mathcal{A}(m(\bar{\tau}_{\mathbf{v}}(U)))[V(G')] \quad (4.79)$$

$$= \tau_{\mathbf{v}}(\mathcal{A}(U))[V(G')] \quad (4.80)$$

where we again made use of eq. (4.66) and eq. (4.62). From eq. (4.80) we see that G' is indeed a vertex-minor of G , see definition 4.5.1, since $G = \mathcal{A}(U)$. \square

4.7.6. Semi-Ordered Eulerian tours

As discussed in section 4.4, the question of whether a graph state $|G\rangle$ can be transformed into a GHZ-state on the qubits V' corresponds to whether the graph G has vertex-minors on V' in the form of star or complete graphs. From the previous sections we have seen that circle graphs and their vertex-minors can be described by Eulerian tours on connected 4-regular multi-graphs. A natural question is therefore: Given a set of vertices V' , what property should a connected 4-regular multi-graph F satisfy, such that $S_{V'}$ is a vertex-minor of $\mathcal{A}(U)$, for some Eulerian tour U on F . As we will see in this section, a necessary and sufficient condition is that F allows for what we call a *semi-ordered Eulerian tour* (SOET) with respect to V' .

The existence of a SOET on a 4-regular graph F with respect to some vertex set V' will therefore be a key technical tool when considering `StarVertexMinor` on circle graphs, as described in section 6.2. We formally define this new concept of a SOET as follows.

Definition 4.7.15 (SOET). Let F be a 4-regular multi-graph and let $V' \subseteq V(F)$ be a subset of its vertices. Furthermore, let $\mathbf{s} = s_1 s_2 \dots s_k$ be a word with letters in V' such that each element of V' occurs exactly once in \mathbf{s} and where $k = |V'|$. A semi-ordered Eulerian tour U with respect to V' is a Eulerian tour such that $m(U) = \mathbf{X}_0 s_1 \mathbf{X}_1 s_2 \dots s_k \mathbf{X}_k s_1 \mathbf{Y}_1 s_2 \dots s_k \mathbf{Y}_k$ for some \mathbf{s} and where $\mathbf{X}_0, \mathbf{X}_1, \dots, \mathbf{X}_k, \mathbf{Y}_1, \dots, \mathbf{Y}_k$ are words (possibly empty) with letters in $V \setminus V'$. This can also be stated as $m(U)[V'] = \mathbf{ss}$, for some \mathbf{s} . \diamond

Note that the multi-graph F is not assumed to be simple, so multi-edges and self-loops are allowed. A SOET is a Eulerian tour on F that traverses the elements of V' in some order once and then again in the same order. The particular order in which the SOET traverses V' will not be important here, only that it traverses V' in the same order twice. An example of a graph that allows for a SOET with respect to the set $V' = \{a, b, c, d\}$ can be seen in fig. 4.2a. A SOET for this graph is for example $m(U) = abcdaebced$. The graph in fig. 4.2b on the other hand does not allow for any SOET with respect to the set $V' = \{a, b, c, d\}$.

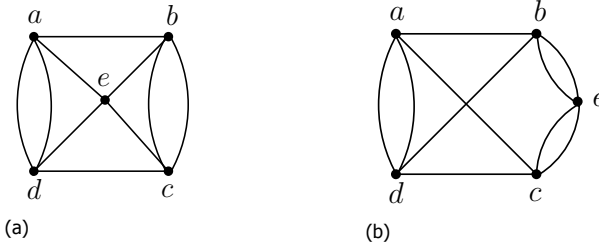


Figure 4.2: Examples of two 4-regular multi-graphs. The graph in fig. 4.2a allows for a SOET with respect to the set $\{a, b, c, d\}$ but the graph in fig. 4.2b does not.

We also formally define the SOET -decision problem, which takes a 4-regular multi-graph F and a subset V' of the vertices as input and asks to decide whether or not the graph F allows for a Semi-Ordered Eulerian Tour with respect to the vertex set V' .

Problem 4.7.16 (SOET). Let F be a 4-regular multi-graph and let V' be a subset $V(F)$. Decide whether there exists a SOET U on F with respect to the set V' . \diamond

As mentioned, the reason for introducing the notion of a SOET is that a 4-regular multi-graph F allows for a SOET with respect to a subset $V' \subseteq V(F)$ if and only if a star graph on V' is a vertex-minor of an alternance graph $\mathcal{A}(U)$ induced by a Eulerian tour U on F . This is captured in the following theorem, formulated as a corollary of theorem 4.7.14.

Corollary 4.7.16.1. Let F be a connected 4-regular multi-graph and let G be a circle graph given by the alternance graph of a Eulerian tour U on F , i.e. $G = \mathcal{A}(U)$. Then $S_{V'}$ is a vertex-minor of G if and only if F allows for a SOET with respect to V' . \diamond

Proof. Note first that $S_{V'} \leq G$ if and only if $K_{V'} \leq G$, since $S_{V'}$ and $K_{V'}$ are LC equivalent. From theorem 4.7.14 we know that $K_{V'}$ is a vertex-minor of G if and only if there exist an Eulerian tour U' on F such that

$$K_{V'} = \mathcal{A}(m(U')[V']). \quad (4.81)$$

It is easy to verify that $\mathcal{A}(\mathbf{X})$ is a complete graph on V' if and only if $\mathbf{X} = s_1 s_2 \dots s_k s_1 s_2 \dots s_k$ where $\mathbf{s} = s_1 s_2 \dots s_k$ is a word with letters in V' such that each element of V' occur exactly once in \mathbf{s} . The result then follows, since $m(U')[V']$ is of this form if and only if U' is a SOET with respect to V' . \square

One can see that the existence of a SOET on a 4-regular multi-graph F with respect to V' , imparts an ordering on the subset of vertices V' . We will in particular be interested in vertices in V' that are 'consecutive' with respect to the SOET. Consecutiveness is defined as follows.

Definition 4.7.17 (Consecutive vertices). Let F be a 4-regular graph and U a SOET on F with respect to a subset $V' \subseteq V(F)$. Two vertices $u, v \in V'$ are called consecutive in U if there exist a sub-word $u\mathbf{X}v$ or $v\mathbf{X}u$ of $m(U)$ such that no letter of \mathbf{X} is in V' . \diamond

We also define the notion of a "maximal sub-word" associated with two consecutive vertices.

Definition 4.7.18 (Maximal sub-words). Let F be a 4-regular multi-graph and U a SOET on F with respect to a subset $V' \subseteq V(F)$. The double occurrence word induced by U is then of the form $m(U) = \mathbf{X}_0 s_1 \mathbf{X}_1 s_2 \dots s_k \mathbf{X}_k s_1 \mathbf{Y}_1 s_2 \dots s_k \mathbf{Y}_k$, where $k = |V'|$, $s_1, \dots, s_k \in V'$ and $X_0, \dots, X_k, Y_1, \dots, Y_k$ are words (possibly empty) with letters in $V(F) \setminus V'$. For $i \in [k - 1]$, we call \mathbf{X}_i and \mathbf{Y}_i the two maximal sub-words associated with the consecutive vertices s_i and s_{i+1} . Furthermore, we call \mathbf{X}_k and $\mathbf{Y}_k \mathbf{X}_0$ the two maximal sub-words associated with the consecutive vertices s_k and s_1 . Given two consecutive vertices u and v , we will denote their two maximal sub-words as \mathbf{X} and \mathbf{X}' , \mathbf{Y} and \mathbf{Y}' or similar. \diamond

4.8. Leaves, twins and axils

In this section we will consider certain vertices called leaves, twins and axils. First we will prove that such vertices can in many cases be removed when considering the vertex-minor problem, which can simplify the problem significantly. We capture this in theorem 4.8.4. This motivates us to consider distance-hereditary graphs, since it turns out that these are exactly the graphs that can be reached from a single-vertex graph by adding leaves or performing twin-splittings. These graphs therefore always have at least one leaf or twin. We will leverage these properties in section 6.3.1 to find an efficient algorithm for `StarVertexMinor` when the input graph is distance hereditary. We define and consider distance-hereditary graphs in section 4.8.1.

Let us first formally define leaves, twins and axils.

Definition 4.8.1 (Leaves and axils). A *leaf* is vertex with degree one. An *axil* is the unique neighbor of a leaf. \diamond

Definition 4.8.2 (Twin). A *twin* is a vertex v such that there exist a different vertex u with the same neighborhood, i.e. v is a twin if and only if

$$\exists u \in V \setminus \{v\} : (N_v \setminus \{u\} = N_u \setminus \{v\}). \quad (4.82)$$

A vertex u as in eq. (4.82) is called a *twin-partner* of v and v, u form a *twin-pair*. If v and u are adjacent, they form a *true twin-pair* and otherwise a *false twin-pair*. \diamond

Definition 4.8.3 (Foliage). The *foliage* of a graph G is the set of leaves, axils and twins in a graph G and is denoted

$$T(G) = \{v \in V(G) : v \text{ is a leaf, axil or twin}\} \quad (4.83)$$

\diamond

We are now ready to prove the following theorem which can be used to simplify some instances of `VertexMinor`, in particular when considering distance-hereditary graphs, see section 4.8.1.

Theorem 4.8.4. *Let G be a graph, G' be a connected graph and v be a vertex in G but not in G' . Then the following is true:*

- *If v is a leaf or a twin, then G' is a vertex-minor of G if and only if G' is a vertex-minor of $G \setminus v$, i.e.*

$$G' < G \iff G' < (G \setminus v). \quad (4.84)$$

- *If v is an axil, then G' is a vertex-minor of G if and only if G' is a vertex-minor of $\tau_w \circ \tau_v(G) \setminus v$, where w is the leaf associated to v , i.e.*

$$G' < G \iff G' < (\tau_w \circ \tau_v(G) \setminus v). \quad (4.85)$$

\diamond

Proof. Firstly, if G' is a vertex-minor of $G \setminus v$, then clearly G' is also a vertex-minor of G .

This means we only need to prove the other direction. Assume therefore that G' is a vertex-minor of G . We start by proving the case where v is a leaf in G . The cases where v is an axil or a twin in G then follow by a short argument.

Hence assume that v is a leaf in $V \setminus V'$, where $V = V(G)$ and $V' = V(G')$. Furthermore, let \mathbf{u} be a sequence of vertices such that each element of $V \setminus V'$ occurs exactly once in \mathbf{u} . Since G' is a vertex-minor of G , we know by theorem 4.5.3 that there exists some sequence of operations $P \in \mathcal{P}_{\mathbf{u}}$, such that $P(G) \sim_{\text{LC}} G'$. Let us denote the i -th operation in P as $P^{(i)}$, such that $P = P^{(n-k)} \circ \dots \circ P^{(1)}$, where $n = |G|$

and $k = |G'|$. Remember that each operation $P^{(i)}$ deletes the i -th vertex of \mathbf{u} from the graph. Furthermore, let's denote the sequence of operations from i through j in P as

$$P_i^j = P^{(j)} \circ \dots \circ P^{(i+1)} \circ P^{(i)}. \quad (4.86)$$

By corollary 4.5.3.1 we know that such a P exist for all orderings \mathbf{u} of the vertices in $V \setminus V'$. Without loss of generality we can assume that v is the first element in \mathbf{u} . This means that $P^{(1)}$ is either Z_v , Y_v or X_v . We will now treat all three these cases separately.

If $P^{(1)}$ is Z_v or Y_v , then since v is a leaf we have that

$$P^{(1)}(G) = G \setminus v. \quad (4.87)$$

Then it is easy to see that G' is also a vertex-minor of $G \setminus v$, since

$$G' \sim_{\text{LC}} P(G) = P_2^{n-k} \circ P^{(1)}(G) = P_2^{n-k}(G \setminus v) \quad (4.88)$$

If $P^{(1)}$ is X_v then the axil of v cannot be in $V \setminus V'$, since the operation X_v on a leaf disconnects the axil from its neighbors. Lets denote the axil of v by w and assume again w.l.o.g. that the ordering of $V \setminus V'$ is such that w is the second element of \mathbf{u} . Since w is a disconnected vertex after $P^{(1)}$, any of the three operations $\{X_w, Y_w, Z_w\}$ act the same, i.e. deleting w . So the action of X_v followed by $P^{(2)} \in \{X_w, Y_w, Z_w\}$ is the same as deleting both v and w or in other words

$$P_1^2(G) = Z_w(G \setminus v) \quad (4.89)$$

It is again clear that G' is then a vertex-minor of $G \setminus v$, since

$$G \sim_{\text{LC}} P(G) = P_3^{n-k} \circ P_1^2(G) = P_3^{n-k} \circ Z_w(G \setminus v) \quad (4.90)$$

with a satisfying sequence taking $G \setminus v$ to an LC-equivalent graph of G' being $(Z_w, P^{(3)}, \dots, P^{(n-k)})$. This proves the theorem when v is a leaf.

Now assume that v is a twin in G . To prove that the theorem also hold for twins, we first show that a twin can always be transformed into a leaf by local complementations. Assume that v and w are false twins, and denote one of their common neighbors as n .⁵ Then the graph $\tilde{G} = \tau_w \circ \tau_n(G)$ is a graph where v is a leaf and w is an axil. Since LC-equivalent graphs have the same vertex-minors, G' is also a vertex-minor of \tilde{G} . From what we showed above and that v is a leaf, G' is also a vertex-minor of $\tilde{G} \setminus v$. Finally, G' is then also a vertex-minor of

$$\tau_n \circ \tau_w(\tilde{G} \setminus v) = \tau_n \circ \tau_w \circ \tau_w \circ \tau_n(G) \setminus v = G \setminus v \quad (4.91)$$

where we used lemma 4.4.7. An almost identical argument can be made for the case where v and w are true twins by considering the graph $\tilde{G} = \tau_w(G)$.

Now assume v is an axil in G . If v is an axil in G and $v \notin G'$, then v is a leaf in the graph $\tilde{G} = \tau_w \circ \tau_v(G)$, where w is the leaf of v in G . Since by assumption $G' < G$, we know that $G' < \tilde{G}$ and from the cases of leaves we have that also $G' < \tilde{G} \setminus v$, since v is a leaf in \tilde{G} . This completes the proof. \square

⁵Note that twins always have at least one common neighbor, except for the graph K_2 where the twins are anyway also leaves.

4.8.1. Distance-hereditary graphs

In this section we introduce distance-hereditary graphs. As shown by Bouchet in [41], distance-hereditary graphs are exactly the graphs with rank-width one. These graphs have nice properties which we make use of in section 6.3.1.

Definition 4.8.5 (Distance-hereditary). A graph G is distance-hereditary if and only if, for each connected induced subgraph $G[A]$ and for any two vertices $u, v \in A$ the distance between u and v is the same in G and in $G[A]$, i.e.

$$d_G(u, v) = d_{G[A]}(u, v). \quad (4.92)$$

◇

Trees, i.e. graphs without loops, is clearly a direct subset of distance-hereditary graphs. The simplest example of a graph which is not distance-hereditary is the five-cycle C_5 . To see this pick two vertices which have distance two in C_5 and denote their unique common neighbor by v . The distance between the same vertices in the connected induced subgraph $C_5[V \setminus v]$ is three and thus not the same as in C_5 . It turns out that distance-hereditary graphs are exactly the graphs which do not contain a vertex-minor isomorphic to C_5 [36]. We also note that distance-hereditary graphs form a strict subclass of circle graphs [36]. This has been shown before [36], but we provide a simple proof here for completeness.

Theorem 4.8.6. Any distance-hereditary graph is also a circle graph. ◇

Proof. We will prove this by instead proving the contrapositive statement, i.e. if a graph is not a circle graph then it is not a distance-hereditary graph. In [36] Bouchet proved that a graph is not a circle graph if and only if it has a vertex-minor isomorphic to one of the graphs in fig. 4.3. Similarly, a graph is not distance-hereditary if and only if it has a vertex-minor isomorphic to the 5-cycle C_5 [36, 42]. Since each graph in fig. 4.3 has a vertex-minor isomorphic to C_5 , this shows that if a graph is not a circle graph then it is also not distance-hereditary. □

Note that distance-hereditary graphs are a proper subset of circle graphs, i.e. there are graphs which are circle graphs but not distance-hereditary. For example C_5 is such a graph.

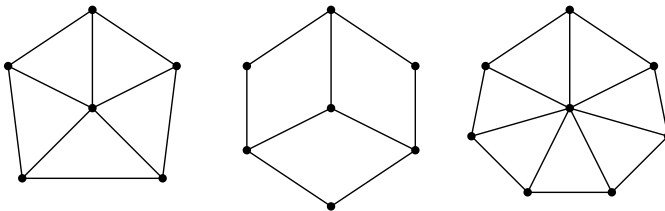


Figure 4.3: A graph is not a circle graph if and only if it has one of these three graphs as a vertex-minor. In [36] this is called a complete set of circle graph obstructions.

In [42] an equivalent property of distance-hereditary is shown: A graph is distance-hereditary if and only if it can be obtained from a single-vertex graph using the following three operations:

- *Add a leaf*: Let u be a vertex in a graph G . Add the vertex v and the edge (u, v) to G .
- *False twin-split*: Let u be a vertex in a graph G . Add the vertex v and the edges $\{(v, x) : x \in N_u\}$.
- *True twin-split*: Let u be a vertex in a graph G . Add the vertex v and the edges $\{(v, x) : x \in \{u\} \cup N_u\}$.

Note that this implies that a distance-hereditary graph always has at least one leaf or twin, i.e. the foliage is non-empty. This fact will be a critical element of the algorithm presented in section 6.3.1.

We will now show that we can in fact rephrase this using only the operations of local complementation and adding leaves. Formally we have the following theorem.

Theorem 4.8.7. *A graph G is distance-hereditary if and only if it can be built using only the operations of (1) local complementations and (2) adding leaves, starting from a graph with a single vertex.* \diamond

Proof. The property of being distance-hereditary is invariant under local complementations and clearly also from adding leaves. Thus we directly see that the property that a graph can be built by the operations above is a sufficient condition for a graph being distance-hereditary. We will now prove that it is also a necessary condition.

We know (from [42]) that a necessary (and sufficient) condition for a graph being distance-hereditary is that it can be built using only the three operations above. We now show that any of these operations can be performed using local complementations and adding leaves, from which the theorem follows:

1. *Add a leaf*: Trivial.
2. *True twin-split* Assume that the original graph is G and that vertex u is split into a true twin pair with v . The same graph can be reached by the following sequence of operations
 - (a) Local complementation on u .
 - (b) Add leaf v with neighbor u .
 - (c) Local complementation on u .
3. *False twin-split* Assume that the original graph is G and that vertex u is split into a false twin pair with v and that w is a common neighbor of u and v . Note that w must exist if $|G| > 1$ and the case of $|G| = 1$ is trivial. The same graph can be reached by the following sequence of operations
 - (a) Local complementation on w .
 - (b) Split u into a *true* twin pair with v .
 - (c) Local complementation on w .

Note that we already proved above that performing true twin splits can be performed with local complementations and adding leaves.

□

In the rest of this section we prove some properties of the foliage for distance-hereditary graphs, which we make use of in section 6.3.1 to find an efficient algorithm for `VertexMinor` on distance-hereditary graphs. First we show that the twin relation is in fact transitive. This is a technical lemma we will use in later theorems.

Lemma 4.8.8. *Let G be a graph and let u be a vertex of G that is a twin and has twin-partners $\{t_1, t_2, \dots, t_k\}$. Then all vertices in $u \cup \{t_1, t_2, \dots, t_k\}$ are pairwise twins.*

◊

Proof. Since u and t_i form a twin-pair, for $i \in \{1, 2, \dots, k\}$, we have that

$$N_u \setminus \{t_i\} = N_{t_i} \setminus \{u\} \quad (4.93)$$

which implies that

$$N_{t_i} = (N_u \setminus \{t_i\}) \cup \{u\}. \quad (4.94)$$

Thus, we have that

$$N_{t_i} \setminus \{t_j\} = ((N_u \setminus \{t_i\}) \cup \{u\}) \setminus \{t_j\} \quad (4.95)$$

$$= ((N_u \setminus \{t_j\}) \cup \{u\}) \setminus \{t_i\} \quad (4.96)$$

$$= \underbrace{(N_u \setminus \{t_i\}) \cup \{u\}}_{N_{t_j} \setminus \{u\}} \setminus \{t_i\} = N_{t_j} \setminus \{t_i\}. \quad (4.97)$$

This shows that, for $i \neq j$, t_i and t_j form a twin-pair. □

Next we prove that adding leaves to a graph G or performing (true or false) twin-splits never decreases the size of the foliage $T(G)$.

Lemma 4.8.9. *Assume G is a connected, distance-hereditary graph. Let G' be a graph formed by doing a twin-split on G or adding a leaf to G . Then*

$$|T(G')| \geq |T(G)|, \quad (4.98)$$

where $T(G)$ is the foliage of G . ◊

Proof. To prove this, let us first consider the case when $|G| \leq 2$. Since G is connected it is necessary the case that $G = K_1$ or $G = K_2$:

- If $G = K_1$, then $G' = K_2$ and $|T(G')| = 2 \geq 0 = |T(G)|$.
- If $G = K_2$, then $G' = K_3$ or $G' = P_3$ and $|T(G')| = 3 \geq 2 = |T(G)|$.

Let's now consider the case when $|G| > 2$. We consider the two cases when G' is formed by adding a leaf and performing a twin-split separately:

- Assume G' is formed by adding a leaf v to G , making u an axil of G' . Note first that if $u \notin T(G)$, then $|T(G)|$ can only increase since no vertex in $T(G)$ was affected. Let's therefore assume that $u \in T(G)$. There are then three possibilities: (1) u is a leaf, (2) u is an axil but not a twin and (3) u is a twin. We consider these three cases separately:
 - (1) Assume u is a leaf in G . Then the axil of u in G , is not in $T(G')$, but both u and v are. Therefore $|T(G)| = |T(G')|$.
 - (2) Assume u is an axil but not a twin in G . Then u is also an axil in G' and we have that $|T(G')| = |T(G)| + 1$.
 - (3) Assume u is a twin in G .
 - ◊ Assume there is only one twin-pair containing u in G . Then the twin-partner of u in G , is not in $T(G')$, but both u and v are. Therefore $|T(G')| = |T(G)|$.
 - ◊ Assume there is more than one twin-pair containing u in G . Then the twin-partners of u are all pairwise twins, by lemma 4.8.8, and will still be in G' . Therefore $|T(G')| = |T(G)| + 1$.
- Assume G' is formed by twin-splitting u in G , creating v and making v and u a twin-pair. Note first that if $u \notin T(G)$, then $|T(G)|$ can only increase since no vertex in $T(G)$ was affected. Let's therefore assume that $u \in T(G)$. There are then three possibilities: (1) u is a leaf, (2) u is an axil but not a twin and (3) u is a twin. We consider these three cases separately:
 - (1) Assume u is a leaf in G . Then the axil of u in G is either still an axil in G' or not, depending on if u and v are true or false twins. In either case, $|T(G')| \geq |T(G)|$ since $v \in T(G')$.
 - (2) Assume u is an axil but not a twin in G . Note that all leaves with u as an axil in G are also twins. These vertices are also twins in G' since they are all now also adjacent to v . Thus, $|T(G')| = |T(G)| + 1$.
 - (3) Assume u is a twin in G .
 - ◊ Assume there is only one twin-pair in G containing u . Then the twin-partner of u in G , may or may not still be a twin-partner of u in G' depending on whether the considered twin-pairs are true or false. Therefore the size of $T(G)$ either remains the same or increases by one since again $v \in T(G')$.
 - ◊ Assume there are more than one twin-pair in G containing u . Then the twin-partners of u are all pairwise twins, by lemma 4.8.8, and will still be in G' . Therefore $|T(G')| = |T(G)| + 1$.

□

We now make use of the above theorem to prove that the foliage has a certain minimum size.

Theorem 4.8.10. Assume G is a connected, distance-hereditary graph and $2 \leq k \leq 4$, then

$$|G| \geq k \quad \Rightarrow \quad |T(G)| \geq k. \quad (4.99)$$

◇

Proof. First we explicitly check that the graphs on 2, 3 and 4 vertices has $T(G) = 2$, $T(G) = 3$ and $T(G) = 4$, respectively.⁶ Then by lemma 4.8.9 and the fact that all distance-hereditary graphs can be built up by twin-splits and adding leaves [41], the result follows. □

We point out that the theorem does not hold for $k > 4$. Consider for example a path graph P_k on more than four vertices. It is easy to see that size of the foliage in this case is $|P_k| = 4$.

Finally we show that an interesting property regarding the foliage, in relation to cut-vertices.⁷

Corollary 4.8.10.1. Assume that G is a connected distance-hereditary graph and that $v \in G$ is a cut-vertex. Denote the connected components of $G \setminus v$ by G_1, G_2, \dots, G_k , where k is the number of connected components of $G \setminus v$. Then for any $1 \leq i \leq k$, there exist a vertex $u \in G_i$ such that $u \in T(G)$. ◇

Proof. Pick an arbitrary connected component G_i with vertices V_i . If G_i is just a single vertex, then this vertex is necessarily a leaf in G and is therefore in $T(G)$. Now assume that $|G_i| > 1$, then by using theorem 4.8.10, we have that there exist at least one twin-pair not containing v or a leaf which is not v in $G[\{v\} \cup V_i]$. This proves the corollary. □

References

- [1] A. Dahlberg and S. Wehner, *Transforming graph states using single-qubit operations*, *Phil. Trans. R. Soc. A* 376, One contribution of 15 to a discussion meeting issue 'Foundations of quantum mechanics and their impact on contemporary society' (2018), 10.1098/rsta.2017.0325.
- [2] A. K. Ekert, *Quantum cryptography based on bell's theorem*, *Physical Review Letters* **67**, 661 (1991).
- [3] C. H. Bennett and G. Brassard, *Quantum Cryptography: Public Key Distribution, and Coin-Tossing*, in *Proc. 1984 IEEE International Conference on Computers, Systems, and Signal Processing* (1984) pp. 175–179.
- [4] R. Raussendorf and H. J. Briegel, *A one-way quantum computer*, *Physical Review Letters* **86**, 5188 (2001), arXiv:0510135v1 [quant-ph] .

⁶The number of non-isomorphic graphs on 2, 3 and 4 vertices are 1, 2 and 6, respectively.

⁷A cut-vertex is a vertex such that when it is deleted, the number of connected components increases.

- [5] M. Hein, W. Dür, J. Eisert, R. Raussendorf, M. V. den Nest, H. J. Briegel, M. V. den Nest, and H. J. Briegel, *Entanglement in Graph States and its Applications, Quantum Computers, Algorithms and Chaos* **162**, 1 (2006), [arXiv:0602096 \[quant-ph\]](#) .
- [6] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*, 10th ed. (Cambridge University Press, Cambridge, 2010).
- [7] D. Gottesman, *The Heisenberg Representation of Quantum Computers, Proceedings of the XXII International Colloquium on Group Theoretical Methods in Physics* **1**, 32 (1998), [arXiv:9807006 \[quant-ph\]](#) .
- [8] M. A. Nielsen, *Cluster-state quantum computation*, *Reports on Mathematical Physics* **57**, 147 (2006).
- [9] D. Schlingemann and R. F. Werner, *Quantum error-correcting codes associated with graphs*, *Physical Review A* **65**, 8 (2002), [arXiv:0012111 \[quant-ph\]](#) .
- [10] D. Markham and B. C. Sanders, *Graph states for quantum secret sharing*, *Physical Review A - Atomic, Molecular, and Optical Physics* **78**, 042309 (2008), [arXiv:0808.1532](#) .
- [11] M. Christandl and S. Wehner, *Quantum anonymous transmissions*, in *Advances in Cryptology - ASIACRYPT 2005*, edited by B. Roy (Springer Berlin Heidelberg, Berlin, Heidelberg, 2005) pp. 217–235.
- [12] J. Ribeiro, G. Murta, and S. Wehner, *Fully device-independent conference key agreement*, *Physical Review A* **97**, 022307 (2018), [arXiv:1708.00798](#) .
- [13] P. Komar, E. M. Kessler, M. Bishof, L. Jiang, A. S. Sørensen, J. Ye, and M. D. Lukin, *A quantum network of clocks*, *Nature Physics* **10**, 582 (2014).
- [14] K. Azuma, K. Tamaki, and H.-K. Lo, *All-photonic quantum repeaters*, *Nature Communications* **6**, 6787 (2015).
- [15] D. Gottesman, *PhD Theses, Ph.D. thesis*, California Institute of Technology (2004), [arXiv:9705052 \[quant-ph\]](#) .
- [16] H. P. Nautrup, N. Friis, and H. J. Briegel, *Fault-tolerant interface between quantum memories and quantum processors*, *Nature Communications* **8**, 1321 (2017).
- [17] A. Bouchet, *κ -Transformations, Local Complementations and Switching*, in *Cycles and Rays*, edited by G. Hahn, G. Sabidussi, and R. E. Woodrow (Springer Netherlands, Dordrecht, 1990) pp. 41–50.
- [18] M. Van den Nest, J. Dehaene, and B. De Moor, *Graphical description of the action of local clifford transformations on graph states*, *Physical Review A* **69**, 022316 (2004).

- [19] S. I. Oum, *Rank-width and vertex-minors*, *Journal of Combinatorial Theory. Series B* **95**, 79 (2005).
- [20] S. Oum and P. Seymour, *Approximating clique-width and branch-width*, *Journal of Combinatorial Theory, Series B* **96**, 514 (2006).
- [21] B. Courcelle, *Circle graphs and monadic second-order logic*, *Journal of Applied Logic* **6**, 416 (2008).
- [22] M. Van den Nest, W. Dür, G. Vidal, and H. J. Briegel, *Classical simulation versus universality in measurement-based quantum computation*, *Physical Review A* **75**, 012337 (2007), [arXiv:0608060 \[quant-ph\]](https://arxiv.org/abs/0608060) .
- [23] B. Courcelle, J. Engelfriet, and G. Rozenberg, *Handle-rewriting hypergraph grammars*, *Journal of computer and system sciences* **46**, 218 (1993).
- [24] U. Bertele and F. Brioschi, *Nonserial dynamic programming* (Academic Press, 1972).
- [25] R. G. Downey and M. R. Fellows, *Parameterized Complexity*, *Proc 6th Annu Conf on Comput Learning Theory* **5**, 51 (1999).
- [26] B. Courcelle and S. il Oum, *Vertex-minors, monadic second-order logic, and a conjecture by Seese*, *Journal of Combinatorial Theory. Series B* **97**, 91 (2007).
- [27] B. Courcelle and J. Engelfriet, *Graph Structure and Monadic Second-Order Logic: A Language Theoretic Approach*, 1st ed. (Cambridge University Press, New York, NY, USA, 2011).
- [28] R. Galian and P. Hliněný, *On parse trees and Myhill-Nerode-type tools for handling graphs of bounded rank-width*, *Discrete Applied Mathematics* **158**, 851 (2010).
- [29] A. Langer, F. Reidl, P. Rossmanith, and S. Sikdar, *Practical algorithms for MSO model-checking on tree-decomposable graphs*, *Computer Science Review* **13-14**, 39 (2014).
- [30] K. K. Dabrowski, F. Dross, J. Jeong, M. M. Kanté, O. Kwon, S. Oum, and D. Paulusma, *Computing small pivot-minors*, in *Graph-Theoretic Concepts in Computer Science*, edited by A. Brandstädt, E. Köhler, and K. Meer (Springer International Publishing, Cham, 2018) pp. 125–138.
- [31] A. Bouchet, *An efficient algorithm to recognize locally equivalent graphs*, *Combinatorica* **11**, 315 (1991), [arXiv:0702057v2 \[cs\]](https://arxiv.org/abs/0702057v2) .
- [32] A. Bouchet, *Graphic presentations of isotropic systems*, *Journal of Combinatorial Theory, Series B* **45**, 58 (1988).
- [33] M. Hein, J. Eisert, and H. J. Briegel, *Multiparty entanglement in graph states*, *Physical Review A - Atomic, Molecular, and Optical Physics* **69**, 062311 (2004), [arXiv:0307130 \[quant-ph\]](https://arxiv.org/abs/0307130) .

- [34] N. Robertson and P. Seymour, *Graph minors: Algorithmic aspects of tree-width*, *J Algorithms* **7**, 309 (1986).
- [35] A. Bouchet, *Caractérisation des symboles croisés de genre nul*, *Comptes Rendus de l'Académie des Sciences* **274**, 724 (1972).
- [36] A. Bouchet, *Circle Graph Obstructions*, *Journal of Combinatorial Theory, Series B* **60**, 107 (1994).
- [37] A. Kotzig, *Quelques remarques sur les transformations κ* , in *seminaire Paris* (1977).
- [38] M. C. Golumbic, *Algorithmic graph theory and perfect graphs*, 2nd ed. (North-Holland Publishing Co., 2004).
- [39] N. Biggs, E. K. Lloyd, and R. J. Wilson, *Graph Theory, 1736-1936* (Oxford University Press, 1976).
- [40] A. Kotzig, *Eulerian lines in finite 4-valent graphs and their transformations*, in *Colloquium on Theory of Graphs (1966 : Tihany, Hungary)* (1968) pp. 219–230.
- [41] A. Bouchet, *Transforming trees by successive local complementations*, *Journal of Graph Theory* **12**, 195 (1988).
- [42] H.-J. Bandelt and H. M. Mulder, *Distance-hereditary graphs*, *Journal of Combinatorial Theory, Series B* **41**, 182 (1986).

5

Transforming graph states using single-qubit operations

Axel Dahlberg, Stephanie Wehner

Stabilizer states form an important class of states in quantum information, and are of central importance in quantum error correction. Here, we provide an algorithm for deciding whether one stabilizer (target) state can be obtained from another stabilizer (source) state by single-qubit Clifford operations (LC), single-qubit Pauli measurements (LPM), and classical communication (CC) between sites holding the individual qubits. What's more, we provide a recipe to obtain the sequence of LC + LPM + CC operations which prepare the desired target state from the source state, and show how these operations can be applied in parallel to reach the target state in constant time. Our algorithm has applications in quantum networks, quantum computing, and can also serve as a design tool - for example, to find transformations between quantum error correcting codes. We provide a software implementation of our algorithm that makes this tool easier to apply.

A key insight leading to our algorithm is to show that the problem is equivalent to one in graph theory, which is to decide whether some graph G' is a vertex-minor of another graph G . The vertex-minor problem is in general NP-Complete, but can be solved efficiently on graphs which are not too complex. A measure of the complexity of a graph is the rank-width which equals the Schmidt-rank width of a subclass of stabilizer states called graph states, and thus intuitively is a measure of entanglement. Here we show that the

Parts of this chapter have been published in Phil. Trans. R. Soc. [1].

vertex-minor problem can be solved in time $O(|G|^3)$ where $|G|$ is the size of the graph G , whenever the rank-width of G and the size of G' are bounded. Our algorithm is based on techniques by Courcelle for solving fixed parameter tractable problems, where here the relevant fixed parameter is the rank width. The second half of this chapter serves as an accessible but far from exhausting introduction to these concepts, that could be useful for many other problems in quantum information.

5.1. Introduction

Motivated by the previous chapter we now proceed to study the computational complexity of `VertexMinor` and therefore of `QubitMinor`. The computational complexity of these problems was, to the authors' knowledge, previously unknown. In chapter 6 we show that this decision problem is in fact \mathbb{NP} -Complete. There is therefore no efficient algorithm that solves this question in general, unless $\mathbb{P} = \mathbb{NP}$. In this chapter on the other hand we show that it is fixed-parameter tractable by using techniques by Courcelle which we will review.

5.1.1. Results

Here, we show that `QubitMinor` can be solved in cubic time in the number of qubits of $|G\rangle$ on instances where the *Schmidt-rank width* of $|G\rangle$ and the number of qubits of $|G'\rangle$ are bounded¹. This is our first main result which we formally state in theorem 5.1.1 and prove in section 5.4.3.

Theorem 5.1.1. *There exists an algorithm that decides if $|G'\rangle$ is a qubit-minor of $|G\rangle$, and therefore if G' is a vertex-minor of G , and has running time*

$$\mathcal{O}(f(|G'|, r) \cdot |G|^3), \quad (5.1)$$

where r is the rank-width of G which is equal to the Schmidt-rank width of $|G\rangle$, $|G|$ denotes the number of vertices in the graph G and f is some computable function. If conjecture 5.4.2 is true then there exists an algorithm to the same problem but with running time

$$\mathcal{O}(f(r) \cdot |G|^3). \quad (5.2)$$

◊

Our second main result concerns GHZ-states, which are useful for many applications on a quantum network and is therefore an important target state. One can easily check that the state $|\text{GHZ}\rangle_U$ is single-qubit Clifford equivalent to the graph state $|K_U\rangle$, where K_U is the complete graph with vertex-set U . It is therefore the case that $|\text{GHZ}\rangle_U$ can be mapped from $|G\rangle$ by LC + LPM + CC if and only if $|K_U\rangle$ is a qubit-minor of $|G\rangle$. We show that this question can be solved efficiently if the Schmidt-rank width of $|G\rangle$ is bounded, as captured in theorem 5.1.2 and proven in section 5.4.3.

Theorem 5.1.2. *There exists an algorithm that decides if $|K_U\rangle$ is a qubit-minor of $|G\rangle$, and therefore if K_U is a vertex-minor of G and has running time*

$$\mathcal{O}(f(r) \cdot |G|^3), \quad (5.3)$$

where r is the rank-width of G which is equal to the Schmidt-rank width of $|G\rangle$, $|G|$ denotes the number of vertices in the graph G and f is some computable function.

◊

¹Note that the time-dependence on the size of G' can be removed if conjecture 5.4.2 is true.

Note in particular that the running time in theorem 5.1.2 does not depend on U , even if conjecture 5.4.2 is false. Similarly to theorem 5.1.2 one can also decide if a graph state has a qubit-minor on a subset U with a given property², efficiently on graph states with bounded Schmidt-rank width, see theorem 5.4.5.

Both of the two main results, theorem 5.1.1 and theorem 5.1.2, rely on a variant of Courcelle's theorem, which we describe more in detail in section 5.4. Courcelle's theorem states that a large class of graph problems are fixed-parameter tractable. This means that there exist algorithms for these problems which are efficient in the size of the input graphs, provided a certain parameter of these graphs is bounded. This is a very powerful theorem, but a direct implementation of the algorithm given by Courcelle's theorem is not useful in practice. The reason being that even though the algorithm is efficient, the hidden constant factor of the algorithm's asymptotic runtime is huge. This huge constant factor is unavoidable since the theorem is so general and captures many $\mathbb{N}\mathbb{P}$ -Complete problems. On the other hand, by knowing that a problem can be efficiently solved, one can usually find a more tailored efficient algorithm for the problem at hand, that does not have a huge hidden constant in the runtime. In chapter 6, we provide an efficient algorithm without a huge hidden constant for the problem of deciding whether $|K_U\rangle$ is a qubit-minor of some graph $|G\rangle$, if $|G\rangle$ has Schmidt-rank width one. There are also many other approaches to find practical algorithms for problems captured by Courcelle's theorem, see for example [2] or [3].

We have implemented many of the concepts and algorithms mentioned in this chapter in SAGE [4] and MONA [5]. Both the code in SAGE and MONA can be freely accessed from the git-repository at [6]. The functionalities provided by this repository include:

- A function taking two graphs, G and G' , as input and returns True if the graph states $|G\rangle$ and $|G'\rangle$ are equivalent under single-qubit Clifford operations and otherwise returns False. The function has a runtime of $\mathcal{O}(|G|^4)$ and is an implementation of the algorithm described in [7, 8].
- A function taking two graphs, G and G' , as input and returns a sequence of operations that takes $|G\rangle$ to a graph state which is single-qubit Clifford equivalent to $|G'\rangle$, if $|G'\rangle < |G\rangle$ and otherwise returns False. This function uses a more sophisticated version of the non-efficient algorithm described in section 5.2.
- A function taking a graph G and a set U as input and either returns a sequence of operations that takes $|G\rangle$ to a graph state which is single-qubit Clifford equivalent to $|K_U\rangle$ and therefore $|\text{GHZ}\rangle_U$ or returns False. If the function returns False and G has rank-width one, then $|K_U\rangle \not\prec |G\rangle$ as we show in chapter 6. The runtime of this function is $\mathcal{O}(|U||G|^3)$.
- As described in section 5.4 one can express whether $|G'\rangle < |G\rangle$ in a logic called monadic second-order logic. We have implemented the expression for

²Expressible in C_2MS .

$|G'| < |G|$ in MONA which is a software to translate such logic expressions to finite-state automata. This can then be used to construct efficient algorithms for graphs of bounded rank-width.

5.1.2. Overview

We start by providing a non-efficient but correct algorithm for deciding the vertex-minor problem, taking any graph as input, in section 5.2. Note that such an algorithm is necessarily non-efficient, unless $\mathbb{P} = \mathbb{NP}$, since the problem it solves is in general \mathbb{NP} -Complete. Furthermore, we describe how the corresponding operations on the graph states can be applied in constant time in section 5.3. In section 5.4 we provide an efficient algorithm for graphs with bounded rank-width by making use of monadic second-order logic and Courcelle's theorem. It is our intention that this section can also be used as a short introduction for those not familiar with these concepts.

5.2. A non-efficient but general algorithm

Here we describe an algorithm that decides if G' is a vertex-minor of G and returns a sequence of local complementations m such that $\tau_m(G)[V(G')] \sim_{\text{LC}} G'$ if such a sequence exists. We will use same notation and terminology as introduced in chapter 4. This algorithm works for any G and G' and has a running time of $\mathcal{O}(3^{n-k}(k^4 + (n-k)n^2))$, where $k = |G'|$ and $n = |G|$. Obviously this algorithm is not efficient, due to the exponential scaling in the size-difference of the graphs, but is still useful for smaller graphs or when $n-k$ is bounded. For finite n , the algorithm described in this section is also a useful benchmark for the efficient algorithm described in section 5.4 for graphs of bounded rank-width. In appendix B.1 we show how the runtime of this algorithm can be slightly improved.

From theorem 4.5.3 we see that to check if G' is a vertex-minor of G it is sufficient to check if one of the graphs in

$$\{P(G) : P \in \mathcal{P}_{V(G) \setminus V(G')}\} \quad (5.4)$$

are LC-equivalent to G' . Note that there are possibly $3^{|V(G) \setminus V(G')|}$ graphs in the set in equation (5.4) to check. As mentioned earlier it is possible to check whether two graphs are LC-equivalent in time $\mathcal{O}(k^4)$, where k is the size of the graphs. The explicit algorithm for checking if G' is a vertex-minor of G is stated in algorithm 5.1 and theorem 5.2.1 captures the proof that it is correct and what its running time is.

Theorem 5.2.1. *Algorithm 5.1 returns a sequence \mathbf{m} such that $\tau_{\mathbf{m}}(G)[V(G')] \sim_{\text{LC}} G'$ if $G' < G$ and returns \perp if $G' \not< G$. Furthermore the runtime is $\mathcal{O}(3^{n-k}(k^4 + (n-k)n^2))$, where $k = |G'|$ and $n = |G|$. \diamond*

Proof. We first prove that the algorithm is correct. Since the algorithm calls itself recursively with the three graphs as in in line 13, 17 and 21 it will generate all the graphs in equation (5.4). That is, the graphs tested for LC-equivalence against G' in line 6 are exactly the graphs in equation (5.4). Furthermore, if at least one of the base-cases, i.e. line 5-10, return an empty sequence then the top-level call to

Algorithm 5.1 Non-efficient algorithm that decides if $G' < G$.

Input: (G, G') .

Output: A sequence \mathbf{m} such that $\tau_{\mathbf{m}}(G)[V(G')] \sim_{\text{LC}} G'$ if $G' < G$.
 \perp if $G' \not< G$.

```

1: function is_vm( $G, G'$ )
2:   if  $V(G') \not\subseteq V(G)$  then
3:     return  $\perp$ 
4:   end if
5:   if  $V(G) = V(G')$  then
6:     if  $G \sim_{\text{LC}} G'$  then
7:       return [] # Return an empty sequence
8:     else
9:       return  $\perp$ 
10:    end if
11:  else
12:    Let  $v$  be a vertex in  $V(G) \setminus V(G')$ 
13:    Let  $\mathbf{m} = \text{is\_vm}(Z_v(G))$ 
14:    if  $\mathbf{m} \neq \perp$  then
15:      return  $\mathbf{m}$ 
16:    else
17:      Let  $\mathbf{m} = \text{is\_vm}(Y_v(G))$ 
18:      if  $\mathbf{m} \neq \perp$  then
19:        return  $\mathbf{m} \parallel [v]$  # Concat.  $[v]$  to  $\mathbf{m}$ , return.
20:      else
21:        Let  $u$  be a vertex incident to  $v$  in  $G$ 
22:        Let  $\mathbf{m} = \text{is\_vm}(X_v(G))$ 
23:        if  $\mathbf{m} \neq \perp$  then
24:          return  $\mathbf{m} \parallel [u, v, u]$  # Concat.  $[u, v, u]$  to  $\mathbf{m}$ , return.
25:        else
26:          return  $\perp$ 
27:        end if
28:      end if
29:    end if
30:  end if
31: end function

```

the algorithm will return a sequence \mathbf{m} such that $\tau_{\mathbf{m}}(G)[V(G')] \sim_{\text{LC}} G'$. On the other hand, if all of the base-cases returns \perp then the top-level call returns \perp . By theorem 4.5.3 it follows that the algorithm is correct.

Let's now consider the runtime of the algorithm. We assume that the graphs are given as their adjacency matrices. Let's denote the running time by $T(n, k)$. Picking a vertex v from the set $V(G) \setminus V(G')$, as in line 12, can be done in time $\mathcal{O}(n)$. There are three recursive calls on line 13, 17 and 21, where also four local complementa-

tions are performed³. Each local complementation can be done in quadratic time in the size of the graph. From the definition of local complementation, definition 4.4.3, we see that this can be done by adding the row of the adjacency matrix corresponding to the vertex where the local complementation is performed to the rows of its neighbors, where adding means vector-addition modulo 2. We therefore have that the running time of the full algorithm has the following recursive expression

$$T(n, k) = 3T(n - 1, k) + \mathcal{O}(n^2) \quad (5.5)$$

where $T(k, k) = \mathcal{O}(k^4)$ from testing LC-equivalence of the graphs as the base-case. By induction we see that the running time can be expressed as

$$T(k + l, k) = 3^l \mathcal{O}(k^4) + \sum_{i=0}^{l-1} 3^i \mathcal{O}((k + l - i)^2) \quad (5.6)$$

Evaluating the above expression for $l = n - k$ we get

$$T(n, k) = 3^{n-k} (\mathcal{O}(k^4) + \sum_{i=0}^{n-k-1} \mathcal{O}((n - i)^2)) \quad (5.7)$$

$$= \mathcal{O}(3^{n-k}(k^4 + (n - k)n^2)) \quad (5.8)$$

□

5

5.3. Constant time transformation

So far we have considered the task of finding the operations that take some graph state to its qubit-minor, but what is the best way to apply these operations to the state when they are found? Assume that we have, by some classical (or quantum) algorithm, found a sequence of operations that takes us from the current state $|G\rangle$ to the target state $|G'\rangle$, i.e. $|G'\rangle$ is a qubit-minor of $|G\rangle$. There are different ways to express these operations, for example as a sequence of single-qubit Clifford operations and single-qubit Pauli measurements or as a sequence of local complementations and vertex-deletions on the corresponding graph. From the previous section we have also seen how these different representations can be mapped to each other. Let's therefore assume that we have expressed the sequence of operations as local complementations \mathbf{m} followed by vertex-deletions of the vertices in $V(G) \setminus V(G')$. The reason for doing this is that we can now perform the single-qubit Clifford operations corresponding to \mathbf{m} in parallel and then simultaneously measure all the qubits in $V(G) \setminus V(G')$ in the standard basis. The simultaneous measurements in the standard basis are possible since the corrections $U_v^{(Z, \pm)}$ are either the identity or Z on the neighbors of v and do therefore not change the measurement basis of neighboring vertices, in contrast to Pauli X and Y measurement.

³Three local complementations for the pivot.

We still need to know what corrections that are needed, depending on the measurement outcomes of the qubits $V(G) \setminus V(G')$. In appendix B.2 we show that vertex $v \in V(G')$ only need to apply $(Z_v)^{y_v}$, where

$$y_v = \sum_{u \in N_v^{(G)} \setminus V(G')} x_u \pmod{2} \quad (5.9)$$

and $x_u \in \{0, 1\}$ is the measurement outcome of node u . In other words, a Z operations is applied to qubit v if the parity of the measurement outcomes of the neighbors of v (in G) is odd. Otherwise, no correction is applied to qubit v . We emphasize that the corrections of the qubit v only depend on the measurement outcomes of the neighborhood of that qubit in the graph G .

Another advantage of only performing measurements in the standard basis is that in some cases it is possible to extract $|G'\rangle$ without destroying all the rest of the entanglement in the original state. More specifically, consider the vertices that are adjacent to at least one vertex in $V(G')$ but which are not in $V(G')$ themselves. These vertices are exactly the ones in the set

$$N_{V(G')} = \left(\bigcup_{v \in V(G')} N_v^{(\tau_m(G))} \right) \setminus V(G'). \quad (5.10)$$

Assume that $N_{V(G')} \neq V(G) \setminus V(G')$. Then the deletion of all the vertices in $N_{V(G')}$ from the graph $\tau_m(G)$ gives a graph with at least two connected components $G' = \tau_m(G)[V(G')]$ and $\tau_m(G)[V(G) \setminus (N_{V(G')} \cup V(G'))]$. Let's denote the second part as G_{rest} . Note that G_{rest} could consist of more than one connected component. We can then see that if, after performing the single-qubit Clifford operations corresponding to \mathbf{m} , we only measure the qubits in $N_{V(G')}$ in the standard basis, followed by corrections, we arrive at the following state

$$|G'\rangle \otimes |G_{\text{rest}}\rangle \otimes \bigotimes_{v \in N_{V(G')}} |+\rangle_v. \quad (5.11)$$

The entanglement in $|G_{\text{rest}}\rangle$ is then not wasted. Since there are in fact multiple sequences of vertices m such that $\tau_m(G)[V(G')] = G'$, one can also try to minimize the neighborhood $N_{V(G')}$ and therefore maximize the size of $|G_{\text{rest}}\rangle$.

5.4. Efficient algorithm based on theorem by Courcelle

As mentioned, we show in chapter 6 that the problem of deciding if G' is a vertex-minor of G is NP -Complete in general. Fortunately the problem is fixed-parameter tractable in the rank-width of G and in general the size of G' , which follows from results by Oum and Courcelle in [9] as we show below. The statement that a problem is fixed parameter-tractable in some parameter r means that there exists

an algorithm that solves the problem and has running time

$$\mathcal{O}(f(r) \cdot p(n)) \tag{5.12}$$

where p is some polynomial and n is the size of the input to the problem. Many \mathbb{NP} -Complete problems are fixed-parameter tractable, which means that their time complexity is not necessarily super-polynomial in the input size but rather in the parameter r . For \mathbb{NP} -Complete fixed-parameter tractable problems, the factor $f(r)$ must scale super-polynomially with n in the worst case, unless $\mathbb{P} = \mathbb{NP}$.

In 1990, Courcelle proved that a large class of graph problems are fixed-parameter tractable in the tree-width of the graph [10]. Courcelle's theorem states that any graph problem specified by a monadic second-order logic (MS) formula can be solved in linear time on graphs of bounded tree-width. The tree-width is a notion which essentially describes how tree-like a graph is [11]. Many problems that are hard in general become tractable on trees, as for example the subgraph isomorphism problem [12]. The same holds for graphs which are not too different from trees, i.e. have a low tree-width, which is exactly what Courcelle's theorem states. Since the original theorem by Courcelle, there has also been many generalizations including the same statement but using rank-width. Bounded rank-width captures a larger class of graph than tree-width, for example complete graphs have minimal rank-width. Importantly here is that rank-width is invariant under local complementations and non-increasing under vertex-deletions [13]. More details on rank-width can be found in section 4.6.

MS logic is an extension of first-order logic, which allows for quantification over sets [14]. Courcelle's theorem actually holds for a strictly more expressive logic called counting monadic second-order logic (CMS) where one can also express whether the size of a set is zero modulo p [14]. A sublanguage of CMS is C_2 MS where p is restricted to be 2 and one can therefore express whether the size of a set is even or odd.

Any graph problem specified by a C_2 MS formula can be solved in cubic time on graphs of bounded rank-width, which is due to theorem 6.55 in [14]. We state this formally in theorem 5.4.1. This is the result we make use of in this section to find an efficient algorithm for the vertex-minor problem.

It turns out that the vertex-minor problem is expressible in C_2 MS, which we formally state in theorem 5.4.4, by collecting results proven by Courcelle and Oum in [9]. By theorem 5.4.1 and theorem 5.4.4 we see that the problem of deciding whether $|G'|$ is a qubit-minor $|G\rangle$ is fixed-parameter tractable in the rank-width of G and in the size of G' in general, as we captured in theorem 5.1.1. The reason the qubit-minor problem is fixed-parameter tractable in both $\text{rwd}(G)$ and $|G'|$ is because the formula $\text{VM}_{G'}$ in equation (5.31) depends on G' . Note that if conjecture 5.4.2 is true this dependence of $|G'|$ in the running time can be removed. If G' is restricted to be a certain type of graph or if we ask the question whether G has a vertex-minor on the subset $U \subseteq V(G)$ with a given property instead, then the running time does not need to depend G' or U respectively, see theorems 5.1.2 and 5.4.5, even if conjecture 5.4.2 is false.

Algorithm 5.2 below gives a high-level description of how to solve the vertex-

minor problem efficiently on graphs of bounded rank-width. The overall runtime of the algorithm is $\mathcal{O}(f(|G'|, r) \cdot |G|^3)$ and is dominated by line 6. In line 5, the C_2 MS formula VM' defined in equation (5.51) can be constructed in time $\mathcal{O}(|G'|)$. An assignment α_y as in line 6 can be found in time $\mathcal{O}(f(|G'|, r) \cdot |G|^3)$ due to Courcelle's theorem for CMS selection problems [14] together with a proof similar to theorem 5.4.1. Finally a sequence of switchings m taking the Eulerian vector $(\emptyset, V(G), \emptyset)$ to the $\alpha_y(X_e, Y_e, Z_e)$ can be done in time $\mathcal{O}(|G|)$ as we show in appendix B.4.

Algorithm 5.2 Algorithm that decides if $G' < G$. Runtime: $\mathcal{O}(f(|G'|, r) \cdot |G|^3)$

Input: (G, G') where $\text{rwd}(G) = r$.

Output: A sequence m such that $\tau_m(G)[V(G')] = G'$ if $G' < G$.

\perp if $G' \not< G$.

```

1: function is_vm( $G, G'$ )
2:   if  $V(G') \not\subseteq V(G)$  then
3:     return  $\perp$ 
4:   end if
5:   Construct the  $C_2$ MS formula  $VM'_{G'}(X, X_e, Y_e, Z_e)$  # See eq. 5.51.
6:   Find an assignment  $\alpha_y$  such that  $G \models VM'_{G'}(X \mapsto V(G'), \alpha_y(X_e, Y_e, Z_e))$ 
7:   if There is no such  $\alpha_y$  then
8:     return  $\perp$ 
9:   else
10:    # See section 5.4.4
11:    Find a sequence of switchings  $m$  taking  $(\emptyset, V(G), \emptyset)$  to  $\alpha_y(X_e, Y_e, Z_e)$ 
12:    return  $m$ 
13:   end if
14: end function

```

5.4.1. Monadic second-order logic

Monadic second-order logic is a sublanguage of second-order logic which in turn is an extension of first-order logic. In MS one can quantify over sets⁴ compared to first-order logic which is restricted to only quantification over elements. For a detailed reference on MS and its extensions, see the book by Courcelle and Engelfriet [14].

MS formulas on graphs uses variables which are either vertex variables x, y, \dots or set variables X, Y, \dots , which are sets of vertices. A MS formula is a finite string built up by the atomic formulas $x = y$, $x \in X$ and $\text{adj}(x, y)$ ⁵ together with the following recursive rules [2]:

1. If ϕ is a formula, then so is $\neg\phi$.
2. If $\phi_1 \dots \phi_l$ are formulas, then so are $(\phi_1 \vee \dots \vee \phi_l)$ and $(\phi_1 \wedge \dots \wedge \phi_l)$.

⁴In second-order logic one can more generally quantify over predicates. MS is restricted to quantification over predicates with one argument (monadic), which is equivalent to quantification over sets.

⁵Expressing whether (x, y) is an edge in the considered graph.

3. If ϕ is a formula, then so are $\exists x : \phi$, $\forall x : \phi$, $\exists X : \phi$ and $\forall X : \phi$.

The variables of a formula which are not part of a quantifier, as in rule (iii) above, are called free variables. A formula with no free variables is called a sentence. We write $\phi(X_1, \dots, X_l, x_1, \dots, x_m)$ for a formula with free variables $X_1 \dots X_l x_1 \dots x_m$. To simplify formulas we will sometimes make use of the following abbreviations

$$(\phi \Rightarrow \psi) \equiv (\neg\phi \vee \psi) \quad (5.13)$$

$$(\phi \Leftrightarrow \psi) \equiv ((\phi \Rightarrow \psi) \wedge (\psi \Rightarrow \phi)) \quad (5.14)$$

$$(X \subseteq Y) \equiv (\forall x : (x \in X \Rightarrow x \in Y)). \quad (5.15)$$

A MS formula is related to a graph G by the atomic formula $\text{adj}(x, y)$, which is true if and only if (x, y) is an edge in G . If a MS sentence ϕ is true on a graph G , we say that G models ϕ and write this as $G \models \phi$. For a formula with free variables, an assignment α is a mapping from the free variables to vertices and subsets of vertices of a graph G . If a MS formula $\phi(X_1, \dots, X_l, x_1, \dots, x_m)$ is true on a graph G with the assignment α , we say that α satisfies ϕ on G and write this as

$$G \models \phi(\alpha(X_1), \dots, \alpha(X_l), \alpha(x_1), \dots, \alpha(x_m)). \quad (5.16)$$

If α assigns v to the free variable x , we write this as $x \mapsto v$. Furthermore, if a formula has free variables $\mathcal{X} = X_1 \dots X_l x_1 \dots x_m$ then we sometimes write $\phi(\mathcal{X})$ and $\phi(\alpha(\mathcal{X}))$ for an assignment of these variables. When considering a formula on a graph we implicitly assume that the quantifiers are over the vertex-set of the graph, i.e.

$$G \models \forall x : \phi(x) \quad \text{iff} \quad \bigwedge_{v \in V(G)} G \models \phi(x \mapsto v). \quad (5.17)$$

As an example the complete graph satisfies the following formula

$$K_n \models \forall x, y : (\neg(x = y) \Rightarrow \text{adj}(x, y)). \quad (5.18)$$

There are many NP-Complete problems that can be defined in MS, including for example 3-colorability [3]. In extensions of MS, one can also consider optimization problems such as minimum vertex cover and traveling sales person [2].

5.4.2. MS problems and complexity

Given a MS formula, there are multiple problems one can consider. We will here be interested in model-checking, property-checking and selection problems, but will also include listing, counting and optimizing for completeness. Below we define these different problems and further details can be found in [14].

- Model-checking: Given a sentence ϕ and a graph, decide if $G \models \phi$.
- Property-checking: Given a formula $\phi(\mathcal{X})$, a graph G and an assignment α , decide if $G \models \phi(\alpha(\mathcal{X}))$.
- Selection: Given a formula $\phi(\mathcal{X})$ and a graph G , find an assignment α such that $G \models \phi(\alpha(\mathcal{X}))$ or if there is no such assignment output \perp .

- Listing: Given a formula $\phi(X)$ and a graph G , find the set of assignments $\{\alpha : G \models \phi(\alpha(X))\}$.
- Counting: Given a formula $\phi(X)$ and a graph G , find the size of the set of assignments $\{\alpha : G \models \phi(\alpha(X))\}$.
- Optimizing: Given a formula $\phi(X)$ and a graph G , find the maximum cardinality of a set assigned to X , i.e. $\max(|\alpha(X)| : G \models \phi(\alpha(X)))$.

In the above definitions we have only considered the problem of finding assignments to all the free variables of a formula for simplicity. One can also consider similar problems as above, where one is given a formula $\phi(X, Y)$, a graph G , an assignment α_y to the free variables Y and where the task is to find an assignment to the rest of the free variables, i.e. α_x such that $G \models \phi(\alpha_x(X), \alpha_y(Y))$. It turns out that all the above problems are fixed-parameter tractable⁶ in the formula and the clique-width of the graph, as shown in [14]. The same is therefore true for rank-width, since rank-width is bounded if and only if clique-width is bounded [15].

5

Theorem 5.4.1. *There exists an algorithm which checks whether an assignment α satisfies a C_2MS formula $\phi(X)$ on G , i.e. whether $G \models \phi(\alpha(X))$ or not, and has a running time*

$$\mathcal{O}(f(|\phi|, \text{rwd}(G)) \cdot |G|^3). \quad (5.19)$$

where $\text{rwd}(G)$ is the rank-width of G and f is an computable function. \diamond

Proof. Theorem 6.55 in [14] states that CMS model-checking problem can be solved in time

$$\mathcal{O}(f(\phi, \text{cwd}(G)) \cdot |G|^3), \quad (5.20)$$

where $\text{cwd}(G)$ is the clique-width of G . In section 6.4.1 of [14] it is also shown that the CMS property-checking problem can be reduced to the CMS model-checking problem and can therefore equivalently be solved in time as in equation (5.20). Furthermore, in definition 6.1 of [14] they show that the CMS model-checking problem can be solved in time

$$\mathcal{O}(f(|\phi|, \text{cwd}(G)) \cdot |G|^3), \quad (5.21)$$

since there are only finitely many sentences of size bounded by a given integer. The theorem then follows since the rank-width is bounded if and only if the clique-width is bounded, which is because

$$\text{rwd}(G) \leq \text{cwd}(G) \leq 2^{\text{rwd}(G)+1} - 1, \quad (5.22)$$

as shown by Oum in [15]. \square

⁶Note that the output of the listing problem is possibly super-polynomial in the size of the graph. Thus for the listing problem, fixed-parameter tractable means polynomial scaling in the size of the input plus the size of the input.

In many cases it is in fact the quantifier rank, see definition 5.4.3 below, which dominates the runtime to solve MS problems. For example in [16] it is shown by Langer, Rossmanith and Sikdar that the MS model-checking problem, i.e. if $G \models \phi$, can be solved in time $\mathcal{O}(f(\text{qr}(\phi), \text{c wd}(G)) \cdot |G|^3)$. As discussed with Langer, it is probably possible to extend this statement to also the CMS model-checking problem and therefore CMS property-checking problem. We therefore make the following conjecture,

Conjecture 5.4.2. *There exists an algorithm which checks whether an assignment α satisfies a C_2 MS formula $\phi(\mathcal{X})$ on G , i.e. whether $G \models \phi(\alpha(\mathcal{X}))$ or not, and has a running time*

$$\mathcal{O}(f(\text{qr}(\phi), \text{rwd}(G)) \cdot |G|^3). \quad (5.23)$$

where $\text{qr}(\phi)$ is the quantifier rank of ϕ , $\text{rwd}(G)$ is the rank-width of G and f is a computable function. \diamond

Definition 5.4.3. The quantifier rank $\text{qr}(\phi)$ of a formula ϕ is the maximum number of nested quantifiers as defined in [2] and can be found by the following recursive relations.

$$\text{qr}(\phi) = 0, \text{ if } \phi \text{ is atomic} \quad \text{qr}(\exists x : \phi(x)) = \text{qr}(\phi) + 1 \quad (5.24)$$

$$\text{qr}(\neg\phi) = \text{qr}(\phi) \quad \text{qr}(\exists X : \phi(X)) = \text{qr}(\phi) + 1 \quad (5.25)$$

$$\text{qr}(\phi \vee \psi) = \max(\{\text{qr}(\phi), \text{qr}(\psi)\}) \quad \text{qr}(\forall x : \phi(x)) = \text{qr}(\phi) + 1 \quad (5.26)$$

$$\text{qr}(\phi \wedge \psi) = \max(\{\text{qr}(\phi), \text{qr}(\psi)\}) \quad \text{qr}(\forall X : \phi(X)) = \text{qr}(\phi) + 1 \quad (5.27)$$

\diamond

5.4.3. Vertex-minor as C_2 MS formula

In [9] Courcelle and Oum show how one can express whether a graph G' is a vertex-minor of G in counting monadic second-order logic C_2 MS. We restate this here and also provide the explicit C_2 MS formula and its quantifier rank. In appendix B.3 we explicitly provide all subformulas we make use of here, which can also be found in [9]. These formulas and the statement of this section build heavily on the concept of an *isotropic system* which was introduced by Bouchet in [17]. We will describe isotropic systems more in detail in chapter 9. The reason for the relation to isotropic systems is that an isotropic system describes an equivalence class of graphs under local complementation. Importantly, an isotropic system⁷ $S(G)$ given by a graph G has a number of *Eulerian vectors* and each of these Eulerian vectors describes a LC-equivalent graph to G . Furthermore, any LC-equivalent graph to G is described by some Eulerian vector of $S(G)$. We will here describe a Eulerian vector by three pairwise disjoint subsets of the vertices of G whose union is $V(G)$ and write this as a tuple (X_e, Y_e, Z_e) . The set of Eulerian vectors of $S(G)$ will be denoted as $\mathcal{E}(S(G))$.

⁷There are actually multiple isotropic system $S(G, A, B)$ related to a graph, depending on the choice of supplementary vectors A and B , see [17]. Here we consider a canonical isotropic system $S(G)$ and chose the supplementary vectors to be $A = (\omega, \dots, \omega)$ and $B = (1, \dots, 1)$, where ω is a primitive element of \mathbb{F}_4

The formula $\text{Eul}(X_e, Y_e, Z_e)$ in equation (B.16) describes whether (X_e, Y_e, Z_e) is a Eulerian vector of $S(G)$, i.e.

$$G \models \text{Eul}(X_e \mapsto U_e, Y_e \mapsto V_e, Z_e \mapsto W_e) \quad \text{iff} \quad (U_e, V_e, W_e) \in \mathcal{E}(S(G)). \quad (5.28)$$

As mentioned above, the set of graphs described by the Eulerian vectors of $S(G)$ are exactly the LC-equivalent graphs to G . Bouchet used this to develop an efficient algorithm to test LC-equivalence between graphs in [7], which has been used to efficiently test single-qubit Clifford equivalence of graph states in [8]. Let's denote the LC-equivalent graph to G described by the Eulerian vector (X_e, Y_e, Z_e) as $\mathcal{G}(X_e, Y_e, Z_e)$. We will now find a formula that captures whether a graph G' is a vertex-minor of G . From the above and equation (4.42) we have that $G' < G$ if and only if there exists a Eulerian vector (X_e, Y_e, Z_e) such that

$$\mathcal{G}(X_e, Y_e, Z_e)[V(G')] = G'. \quad (5.29)$$

How do we express this as a C_2 MS formula? The formula $\text{Adj}(u, v, X_e, Y_e, Z_e)$ in equation (B.18) describes whether the edge (u, v) is an edge of the graph $\mathcal{G}(X_e, Y_e, Z_e)$, i.e.

$$G \models \text{Adj}(x \mapsto u, y \mapsto v, X_e \mapsto U_e, Y_e \mapsto V_e, Z_e \mapsto W_e) \quad \text{iff} \quad (u, v) \in E(\mathcal{G}(U_e, V_e, W_e)). \quad (5.30)$$

Using equation (5.30) we can express whether G' is a vertex-minor of G , as described in the following theorem.

Theorem 5.4.4. *For any G' with vertex-set $V(G') = \{x_1, \dots, x_k\}$, there exists a C_2 MS formula $\text{VM}_{G'}(x_1, \dots, x_k)$ such that*

$$G \models \text{VM}_{G'}(\alpha(x_1), \dots, \alpha(x_k)) \quad \text{iff} \quad \alpha(G') < G, \quad (5.31)$$

where x_i are the free variables of the formula, α is a bijection from $V(G')$ to a subset of $V(G)$ of size k . In equation (5.31), α functions both as an assignment of the free variables of VM and as a relabeling of the vertices in G' by $\alpha(G)$. This dual-purpose of α is valid since we identify the free variables of VM with the vertices of G' . To be precise, by $\alpha(G)$ we mean the graph

$$\alpha(G) = (\{\alpha(x) : x \in V(G')\}, \{(\alpha(x), \alpha(y)) : (x, y) \in E(G')\}). \quad (5.32)$$

Furthermore the length and quantifier rank of VM has the following scaling

$$|\text{VM}_{G'}| = \mathcal{O}(|G'|^2), \quad \text{qr}(\text{VM}_{G'}) = 10 = \mathcal{O}(1). \quad (5.33)$$

◊

Proof. We prove this by explicitly providing the C_2 MS formula as follows

$$\begin{aligned} \text{VM}_{G'}(x_1, \dots, x_k) = \exists X_e, Y_e, Z_e : & \left(\text{Eul}(X_e, Y_e, Z_e) \wedge \bigwedge_{(x,y) \in E(G')} \text{Adj}(x, y, X_e, Y_e, Z_e) \right. \\ & \left. \wedge \bigwedge_{(x,y) \notin E(G')} \neg \text{Adj}(x, y, X_e, Y_e, Z_e) \right) \end{aligned} \quad (5.34)$$

It is then clear that equation (5.31) is true, since if $G \models \text{VM}(\alpha(x_1), \dots, \alpha(x_k))$ then we know that there exist an LC-equivalent graph to G which induced subgraph on $V(G')$ has the edge-set $\{(\alpha(x), \alpha(y)) : (x, y) \in E(G')\}$. This is precisely $\alpha(G')$ and therefore $\alpha(G') < G$. Furthermore, if $\alpha(G') < G$ then we know that there exist a Eulerian vector (U_e, V_e, W_e) such that

$$\bigwedge_{(x,y) \in E(G')} \text{Adj}(\alpha(x), \alpha(y), U_e, V_e, W_e) \wedge \bigwedge_{(x,y) \notin E(G')} \neg \text{Adj}(\alpha(x), \alpha(y), U_e, V_e, W_e) \quad (5.35)$$

is true.

Next we show how the length and quantifier rank of VM scale with G' . Firstly the length clearly scales as

$$|\text{VM}| = \mathcal{O}(|E(G')|) = \mathcal{O}(|G'|^2). \quad (5.36)$$

The quantifier ranks of the subformulas used here are given in appendix B.3 and in particular we have that $\text{qr}(\text{Eul}) = 7$ and $\text{qr}(\text{Adj}) = 7$. Thus, we have that the quantifier rank of VM is

$$\text{qr}(\text{VM}) = 3 + \max(\{\text{qr}(\text{Eul}), \text{qr}(\text{Adj})\}) = 3 + \max(\{7, 7\}) = 10 = \mathcal{O}(1). \quad (5.37)$$

□

It is easy to see that theorem 5.1.1 is a direct consequence of theorem 4.4.2, theorem 5.4.1 and theorem 5.4.4.

As mentioned earlier, it is possible to specify in $C_2\text{MS}$ whether a graph has a vertex-minor on a subset of the vertices with a given property. We capture this in the following theorem.

Theorem 5.4.5. *Given a $C_2\text{MS}$ sentence ϕ specifying some graph property P , then there exists a $C_2\text{MS}$ formula $\text{Prop_VM}_\phi(X)$ capturing whether a graph has a vertex-minor on X which satisfies P , i.e.*

$$G \models \text{Prop_VM}_\phi(X \mapsto U) \quad \text{iff} \quad \exists G' : (V(G') = U) \wedge (G' < G) \wedge (G' \models \phi) \quad (5.38)$$

◇

Proof. Let ϕ' be the formula made from ϕ by replacing all instances of the predicate $\text{adj}(x, y)$ by the formula $\text{Adj}(x, y, X_e, Y_e, Z_e)$. If (U_e, V_e, W_e) is a Eulerian vector of $S(G)$, then we have that

$$G \models \phi'(X_e \mapsto U_e, Y_e \mapsto V_e, Z_e \mapsto W_e) \quad \text{iff} \quad \mathcal{G}(U_e, V_e, W_e) \models \phi. \quad (5.39)$$

The expression on the right of the above equation states that the LC-equivalent graph $\mathcal{G}(U_e, V_e, W_e)$ models ϕ , but what we want is that the induced subgraph $\mathcal{G}(U_e, V_e, W_e)[X]$ models ϕ . This can be done by restricting all the quantifiers in

ϕ' to the set X . Thus, let ϕ'' be the formula made from ϕ' by making the following changes to all the quantifiers

$$\forall y : \psi(y) \rightarrow \forall y : (y \in X \Rightarrow \psi(y)) \quad (5.40)$$

$$\forall Y : \psi(Y) \rightarrow \forall Y : (Y \subseteq X \Rightarrow \psi(Y)) \quad (5.41)$$

$$\exists y : \psi(y) \rightarrow \exists y : (y \in X \wedge \psi(y)) \quad (5.42)$$

$$\exists Y : \psi(Y) \rightarrow \exists Y : (Y \subseteq X \wedge \psi(Y)). \quad (5.43)$$

We then see that if (U_e, V_e, W_e) is an Eulerian vector of $S(G)$, then

$$G \models \phi''(X \mapsto U, X_e \mapsto U_e, Y_e \mapsto V_e, Z_e \mapsto W_e) \text{ iff } \mathcal{G}(U_e, V_e, W_e)[U] \models \phi. \quad (5.44)$$

The formula Prop_VM can then be built by checking if there exists a Eulerian vector such that G models ϕ'' , i.e.

$$\text{Prop_VM}_\phi(X) = \exists X_e, Y_e, Z_e : [\text{Eul}(X_e, Y_e, Z_e) \wedge \phi''(X, X_e, Y_e, Z_e)]. \quad (5.45)$$

□

Since it is possible to specify whether a graph is a complete graph in C_2MS , see equation (5.18), we see that theorem 5.1.2 directly follows from theorem 4.4.2, theorem 5.4.1 and theorem 5.4.5. Let's use the method in the proof of theorem 5.4.5 to explicitly find the formula expressing whether a graph has the complete graph as a vertex-minor on the subset X . Let Complete be the formula in equation (5.18) which is modeled by G if G is a complete graph. We first find Complete' by replacing the predicate $\text{adj}(x, y)$

$$\text{Complete}'(X_e, Y_e, Z_e) = \forall x, y : (\neg(x = y) \Rightarrow \text{Adj}(x, y, X_e, Y_e, Z_e)). \quad (5.46)$$

Then replacing the quantifiers as in equations (5.40)-(5.43) we get the formula

$$\begin{aligned} \text{Complete}''(X, X_e, Y_e, Z_e) &= \forall x : \left[x \in X \Rightarrow \left(\forall y : y \in X \Rightarrow (\neg(x = y) \Rightarrow \right. \right. \\ &\quad \left. \left. \text{Adj}(x, y, X_e, Y_e, Z_e)) \right) \right] \\ &= \forall x, y : \left[((x \in X) \wedge (y \in X) \wedge \neg(x = y)) \Rightarrow \right. \\ &\quad \left. \text{Adj}(x, y, X_e, Y_e, Z_e) \right] \end{aligned} \quad (5.47)$$

Finally by using the equation (5.45) we arrive at the formula

$$\text{Complete_VM}(X) = \exists X_e, Y_e, Z_e : [\text{Eul}(X_e, Y_e, Z_e) \wedge \text{Complete}''(X, X_e, Y_e, Z_e)] \quad (5.48)$$

which has the following property

$$G \models \text{Complete_VM}(X \mapsto U) \text{ iff } K_U < G \quad (5.49)$$

where K_U is the complete graph with vertex-set U .

5.4.4. Finding the sequence of operations

In this section so far, we have looked at the problem of deciding whether G' is a vertex-minor of G , but if this is true then how does one find the sequence of operations that takes G to G' ? Similarly, if $|G'\rangle$ is a qubit-minor of $|G\rangle$, what sequence of operations takes $|G\rangle$ to $|G'\rangle$? We will here describe two ways to find the sequence of operations.

Method 1: The first way is slightly simpler but increases the runtime from the decision problem by a factor of $|G|$. The idea is to use an algorithm that solves the decision problem of whether G' is a vertex-minor of G to iteratively find the sequence of operations. Let's therefore assume that we know that $G' < G$. Furthermore, let v be a vertex in $V(G) \setminus V(G')$. From theorem 4.5.3 we know that G' is a vertex-minor of at least one of the three graphs $X_v(G)$, $Y_v(G)$ or $Z_v(G)$. By using an algorithm for the decision problem we can decide which of these three graphs has G' as a vertex-minor. Let's denote one of these graphs by G_1 and the operation that takes G to G_1 as P_1 , i.e. $G_1 = P_1(G)$. That is, P_1 is either X_v , Y_v or Z_v , such that $G' < G_1 < G$. Now perform the same step again to find an operation P_2 taking G_1 to a graph G_2 which has G' as a vertex-minor. Perform the step $n - k$ times, where $n = |G|$ and $k = |G'|$. It is then clear that the sequence $P = P_{n-k} \circ \dots \circ P_1$ takes G to a LC-equivalent graph of G' . From P it is easy to find the induced sequence of local complementations m such that $\tau_m(G)[V(G')] \sim_{\text{LC}} G'$. Finally we can use the algorithm in [7] to find a sequence of local complementations m' such that $\tau_{m'} \circ \tau_m(G)[V(G')] = G'$. Assume that $|G'|$ and $\text{rwd}(G)$ are bounded⁸. Then according to theorem 5.1.1 the decision problem can be solved in time $\mathcal{O}(n^3)$. To find P we need to run the algorithm for the decision problem $\mathcal{O}(n - k)$ times and compute $P_i(G_{i-1})$ the same number of times. Computing $P_i(G_{i-1})$ can be done in time $\mathcal{O}(n^2)$, as described in section 5.2. Lastly finding the sequence m' can be done in time $\mathcal{O}(k^4)$. Thus, the total runtime is

$$\mathcal{O}((n^3 + n^2)(n - k)) + \mathcal{O}(k^4) = \mathcal{O}(n^4) \quad (5.50)$$

Method 2: Another way to find the sequence of operations that takes G to G' , given that $G' < G$, is to formulate the problem as a C_2 MS selection problem as described in section 5.4.2. Recall that an algorithm that solves the C_2 MS selection problem takes as input a formula ϕ with free variables \mathcal{X}, \mathcal{Y} , an assignment α_x to the free variables \mathcal{X} and a graph G and returns an assignment α_y such that $G \models \phi(\alpha_x(\mathcal{X}), \alpha_y(\mathcal{Y}))$ or returns \perp if no such assignment exists. We will now formulate a selection problem by making X_e, Y_e and Z_e free variables in equation (5.31) instead of quantifier variables. Therefore, let H be an isomorphic graph to G' with vertex-set $V(H) = \{x_1, \dots, x_k\}$ and define the C_2 MS formula

$$\text{VM}'_H(x_1, \dots, x_k, X_e, Y_e, Z_e) = \text{Eul}(X_e, Y_e, Z_e) \wedge \bigwedge_{(x,y) \in E(H)} \text{Adj}(x, y, X_e, Y_e, Z_e) \quad (5.51)$$

Let α_x be a bijection from $V(H)$ to $V(G')$ such that $\alpha_x(H) = G'$. We then have the following property of the formula VM'

$$\exists \alpha_e : G \models \text{VM}'_H(\alpha_x(x_1, \dots, x_k), \alpha_e(X_e, Y_e, Z_e)) \quad \text{iff} \quad G' < G. \quad (5.52)$$

⁸If conjecture 5.4.2 is true, then $|G'|$ does not need to be bounded.

From theorem 6.55 and similar arguments as in the proof of theorem 5.4.1 we know that we can solve the selection problem in time $\mathcal{O}(n^3)$ if k and $\text{rwd}(G)$ are bounded, where $n = |G|$ and $k = |G'|$. Thus, given an assignment to the free variables (X_e, Y_e, Z_e) , i.e. given an Eulerian vector (U_e, V_e, W_e) such that $\mathcal{G}(U_e, V_e, W_e)[V(G')] = G'$, the question is then how to find a sequence of local complementations that takes G to $\mathcal{G}(U_e, V_e, W_e)$. In appendix B.4 we show how to do this in time $\mathcal{O}(n)$, which shows that the total runtime is

$$\mathcal{O}(n^3) + \mathcal{O}(n) = \mathcal{O}(n^3). \quad (5.53)$$

5.5. Discussion

The problem of deciding whether a stabilizer state $|S_t\rangle$ can be obtained from another $|S_s\rangle$ by single-qubit Clifford operations, single-qubit Pauli measurement and classical communication is equivalent to deciding if some graph G' is a vertex-minor of another graph G . We showed here that the vertex-minor problem can be solved in cubic time in the size of G on instances where G has bounded rank-width and G' has bounded size, by using the theory of monadic second-order logic and a version of Courcelle's theorem. Furthermore, if conjecture 5.4.2 is true then the vertex-minor problem can be solved in cubic time in the size of G on the strictly larger class of instances where G has bounded rank-width and G' is arbitrary. A direct implementation of Courcelle's theorem is however not practical, due to a huge constant factor in the runtime of the algorithm. Finding more tailored algorithms for the vertex-minor problem on graphs of bounded rank-width is therefore of value. In chapter 6 we provide an efficient algorithm for graphs of rank-width one, which does not have a huge constant factor in the runtime.

Given some graph property P expressible in C_2MS one can decide if a graph G has a vertex-minor on a subset $U \subseteq V(G)$ that satisfies P , in cubic time in the size of G for graphs with bounded rank-width, as we show in section 5.4.3. The graph property P can be for example that the graph is a complete graph or that it is k -colourable. Testing for for example 2-colourable qubit-minors could be interesting in the context of purification since it has been shown that the purification schemes in [18] purify all 2-colourable graph states.

In section 5.4.4 we also showed how to find the sequence of operations that take $|G\rangle$ to its qubit-minor $|G'\rangle$. Finally in section 5.3 we showed how these operations can be applied in constant time in the size of the graph states and how this can be done without destroying all the rest of the entanglement in the source state. An open question is how to find an optimal sequence of operations that destroys a minimum amount of entanglement in the rest of the state.

Acknowledgment

Axel Dahlberg and Stephanie Wehner were supported by STW Netherlands, and NWO VIDI grant, and an ERC Starting grant.

We thank Alexander Langer, Robert Galian, Kenneth Goodenough, Jonas Helsen, Tim Coopmans, Jériémy Ribiero and Ben Criger for interesting discussions and feed-

back.

References

- [1] A. Dahlberg and S. Wehner, *Transforming graph states using single-qubit operations*, *Phil. Trans. R. Soc. A* **376**, One contribution of 15 to a discussion meeting issue 'Foundations of quantum mechanics and their impact on contemporary society' (2018), 10.1098/rsta.2017.0325.
- [2] A. Langer, F. Reidl, P. Rossmanith, and S. Sikdar, *Practical algorithms for MSO model-checking on tree-decomposable graphs*, *Computer Science Review* **13-14**, 39 (2014).
- [3] R. Ganian and P. Hliněný, *On parse trees and Myhill-Nerode-type tools for handling graphs of bounded rank-width*, *Discrete Applied Mathematics* **158**, 851 (2010).
- [4] SAGE, <http://www.sagemath.org/> (2020).
- [5] MONA, <http://www.brics.dk/mona/index.html> (2020).
- [6] *Git-repository with implemented code for graph state calculations*, <https://github.com/AckslD/vertex-minors> (2020).
- [7] A. Bouchet, *An efficient algorithm to recognize locally equivalent graphs*, *Combinatorica* **11**, 315 (1991), arXiv:0702057v2 [cs] .
- [8] M. Van den Nest, J. Dehaene, and B. De Moor, *Efficient algorithm to recognize the local clifford equivalence of graph states*, *Physical Review A* **70**, 034302 (2004), arXiv:0405023 [quant-ph] .
- [9] B. Courcelle and S. il Oum, *Vertex-minors, monadic second-order logic, and a conjecture by Seese*, *Journal of Combinatorial Theory. Series B* **97**, 91 (2007).
- [10] B. Courcelle, *The Monadic Second-Order Theory of Graphs. I. Recognizable Sets of Finite graphs*, *Information and Computation* **85**, 12 (1990).
- [11] N. Robertson and P. Seymour, *Graph minors: Algorithmic aspects of tree-width*, *J Algorithms* **7**, 309 (1986).
- [12] A. Gupta and N. Nishimura, *The complexity of subgraph isomorphism for classes of partial k-trees*, *Theoretical Computer Science* **164**, 287 (1996).
- [13] S. I. Oum, *Rank-width and vertex-minors*, *Journal of Combinatorial Theory. Series B* **95**, 79 (2005).
- [14] B. Courcelle and J. Engelfriet, *Graph Structure and Monadic Second-Order Logic: A Language Theoretic Approach*, 1st ed. (Cambridge University Press, New York, NY, USA, 2011).

- [15] S. Oum and P. Seymour, *Approximating clique-width and branch-width*, *Journal of Combinatorial Theory, Series B* **96**, 514 (2006).
- [16] A. Langer, P. Rossmanith, and S. Sikdar, *Linear-time algorithms for graphs of bounded rankwidth: A fresh look using game theory (extended abstract)*, *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* **6648 LNCS**, 505 (2011).
- [17] A. Bouchet, *Isotropic systems*, *European Journal of Combinatorics* **8**, 231 (1987).
- [18] H. Aschauer, W. Dür, and H. J. Briegel, *Multipartite entanglement purification for two-colorable graph states*, *Physical Review A - Atomic, Molecular, and Optical Physics* **71**, 1 (2005), arXiv:0303087 [quant-ph] .

6

How to transform graph states using single-qubit operations: computational complexity and algorithms

Axel Dahlberg, Jonas Helsen, Stephanie Wehner

Graph states are ubiquitous in quantum information with diverse applications ranging from quantum network protocols to measurement based quantum computing. Here we consider the question whether one graph (source) state can be transformed into another graph (target) state, using a specific set of quantum operations (LC + LPM + CC): single-qubit Clifford operations (LC), single-qubit Pauli measurements (LPM) and classical communication (CC) between sites holding the individual qubits. This question is of interest for effective routing or state preparation decisions in a quantum network or distributed quantum processor and also in the design of quantum repeater schemes and quantum error-correction codes.

We first show that deciding whether a graph state $|G\rangle$ can be transformed into another graph state $|G'\rangle$ using LC + LPM + CC is NP-Complete , which was previously not known. We also show that the problem remains NP-Complete even if $|G'\rangle$ is restricted to be the GHZ-state. However, we also provide efficient algorithms for two situations of practical interest.

Parts of this chapter have been published in Quantum Science and Technology [1].

Our results make use of the insight that deciding whether a graph state $|G\rangle$ can be transformed to another graph state $|G'\rangle$ is equivalent to a known decision problem in graph theory, namely the problem of deciding whether a graph G' is a vertex-minor of a graph G . The computational complexity of the vertex-minor problem was prior to this chapter an open question in graph theory. We prove that the vertex-minor problem is NP -Complete by relating it to a new decision problem on 4-regular graphs which we call the semi-ordered Eulerian tour (SOET) problem.

6.1. Introduction

In this chapter we continue our study of the computational complexity of `QubitMinor` and our search for efficient algorithms by building on the previous two chapters. In particular we prove that it is in general NP -Complete to decide whether a graph G' is a vertex-minor of another graph G . We however also give efficient algorithms for this problem whenever the input graphs belong to particular graph classes. An overview of the complexity of the problem for different classes of graphs considered in this chapter can be seen in fig. 1.3.

We will use the same notation and terminology from the previous two chapters. Before diving in to the details we will here outline the results and technique used. The results described in this section will be phrased informally. Details and formal statements is in the later parts of the chapter.

6.1.1. Results and proof techniques

We start by pointing out that our results of NP -Completeness and the presented algorithms also apply to the more general class of stabilizer states of relevance in quantum error correction. This is because any stabilizer state can be transformed to some graph state using only single-qubit Clifford operations. Furthermore, given a stabilizer state on n qubits, a graph state equivalent under single-qubit Clifford operations can be found efficiently in time $\mathcal{O}(n^3)$ [2].

Below we list the main results and proof techniques of this chapter. Our first result is a proof that `VertexMinor` and `QubitMinor` are both NP -Complete.

Theorem 6.1.1 (informal). *The problem of deciding whether a graph G' is a vertex-minor of another graph G is NP -Complete. This implies that `QubitMinor` is also NP -Complete.* \diamond

Our study of `QubitMinor` and `VertexMinor` is motivated by the fact that efficient algorithms that solve these problems can be used to make for example routing decisions in a quantum network. Unfortunately theorem 6.1.1 tells us that no such algorithms exist, unless $\text{NP}=\text{P}$. However, along with the proof of NP -Completeness we also provide efficient algorithms for the following three restricted variants of `VertexMinor` and `QubitMinor`:

1. Decide if a star graph on vertices V' is a vertex-minor of a distance-hereditary graph G . This is equivalent to deciding if the GHZ-state on qubits V' is a qubit-minor of a graph state $|G\rangle$ with Schmidt-rank width one.
2. For a fixed k , decide if a star graph on vertices V' , where $|V'| \leq k$, is a vertex-minor of a circle graph G . This corresponds to deciding if a GHZ-state of bounded size on qubits V' is a qubit-minor of a circle graph state $|G\rangle$ with unbounded entanglement.
3. Decide if a graph G' on vertices V' , where $|V'| \leq 3$ is a vertex-minor of a graph G .

For a visual overview of these different graph classes see fig. 1.3 and for more details chapter 4. We will from now on denote the special case of `VertexMinor` where G' is restricted to be a star graph as `StarVertexMinor`.

Theorem 6.1.2 (informal). *The algorithm presented in section 6.3.1, consisting of algorithm 6.1 and algorithm 6.2, solves `StarVertexMinor` in time*

$$O(|V(G')| \cdot |V(G)|^3) \quad (6.1)$$

and is correct if G is distance-hereditary, or equivalently if G has rank-width one. \diamond

The algorithm mentioned in the above theorem can therefore be used to decide how to transform graph states, with Schmidt-rank width one, to GHZ-states using single-qubit Clifford operations, single-qubit Pauli measurements and classical communication. As mentioned above, a more general method to find efficient algorithms for certain graph problems on graphs with bounded rank-width is by using Courcelle's theorem [3]. Compared to the algorithm provided by a direct implementation of Courcelle's theorem, see chapter 5, our algorithm presented here does not suffer from a huge constant factor in the runtime, as in eq. (4.1). In fact, besides providing proof for correctness and runtime, we have also implemented the algorithm [4] and see that it typically takes for example 50 ms to run for the case when $|V(G)| = 50$ on a standard desktop computer, see fig. 6.1.

We call k -`StarVertexMinor` the restriction of `StarVertexMinor` where G' is restricted to a star graph having k vertices, corresponding to a GHZ-state (up to LC) on k qubits.

Theorem 6.1.3 (informal). *k -`StarVertexMinor` is in \mathbb{P} if G is a circle graph¹.* \diamond

The above theorem implies that `StarVertexMinor` is fixed-parameter tractable in the size of the target star graph on circle graphs. Interestingly the class of circle graphs has unbounded rank-width ([8, Proposition 6.3] and [9]) and the corresponding graph states therefore have unbounded entanglement according to the Schmidt-rank width. Thus, theorem 6.1.3 is not captured by the results from Courcelle [3] and implies that efficient algorithms can be found even on graphs with unbounded rank-width.

Theorem 6.1.4 (informal). *Any connected graph G' on three vertices or less is a vertex-minor of any connected graph G if and only if the vertices of G' are also in G .* \diamond

Along with the above theorem we also provide an efficient algorithm for finding the transformation that takes the former graph to the latter.

We show this result by first proving that G has a vertex-minor which is connected, on any subset of its vertices. Then from the fact that there is only a single equivalence class for connected graphs on one, two or three vertices, respectively, under the considered operations, the result follows.

¹Not to be confused with cycle graph.

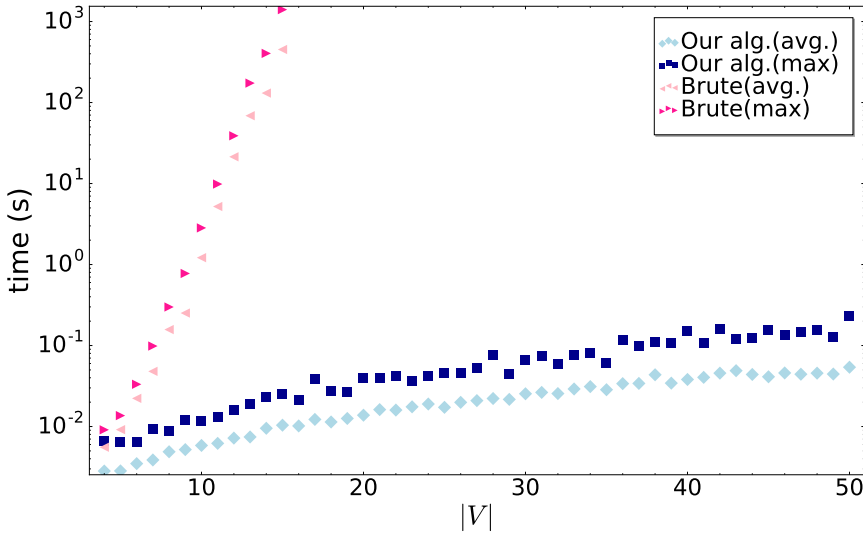


Figure 6.1: Average and maximal observed run-times for two algorithms that check if a GHZ-state on four qubits is a qubit-minor of a randomly generated connected graph state $|G\rangle$ on qubits V of Schmidt-rank width 1. Random connected graph states of Schmidt-rank width 1 are generated by starting from a single-qubit graph and randomly adding leaves or performing twin-splits, see section 4.8.1, which generates any connected graph state of Schmidt-rank width 1 [5]. “Our alg.” refers to the algorithm described in section 6.3.1 and “Brute” is the non-efficient algorithm 5.1. Algorithm 5.2 based on the techniques of Courcelle [6] is not depicted here since the pre-factor makes an application impractical in practice whenever $|V| < f(r)$ of eq. (4.1). For each size of V , 10 random graph states are generated for “Brute” and 100 random graph states for “Our alg.”, from which the average “(avg.)” and max “(max)” runtime is computed. Both algorithms are implemented in SAGE [7] and the tests were performed on an iMac with 3.2 GHz Intel Core i5 processor with 8 GB of 1600 MHz RAM.

Along with the mentioned theorems we also prove several theorems needed for the main results that may be interesting in their own right. For example we prove the following theorem which points out an interesting behaviour of bipartitions of vertices of a graph.

Theorem 6.1.5 (informal). *Assume G is a graph on the vertices $U \cup L$ such that $U \cap L = \emptyset$ and $U \neq \emptyset$. Furthermore, assume that for each l in L , there is at least one vertex in U not adjacent to l and for each u in U , there is at least one vertex in L adjacent to u . Then there exist two vertices u_1 and u_2 in U and two vertices l_1 and l_2 in L such that u_1 is adjacent to l_1 but not to l_2 and u_2 is adjacent to l_2 but not to l_1 .* \diamond

As mentioned, we prove that `StarVertexMinor` is NP-Complete on a strict subclass of circle graphs² and that `StarVertexMinor` is in \mathbb{P} on distance-hereditary graphs. These two graph classes are in fact disjoint, which we prove in the following theorem.

²These circle graphs are in fact circle graphs induced by Eulerian tours on triangular expansions of 3-regular graphs.

Theorem 6.1.6 (informal). *No circle graph induced by a Eulerian tour on a triangular expansion of some 3-regular graph is distance-hereditary.* \diamond

6.1.2. Overview

The chapter is structured as follows. Formal statement and proof of theorem 6.1.1 above is given in section 6.2 (as theorem 6.2.2), theorem 6.1.2 in section 6.3.1, theorem 6.1.3 in section 6.3.2 (as corollary 6.3.11.1) and theorem 6.1.4 in section 6.4 (as theorem 6.4.1). In section 6.2 we consider the computational complexity of the `VertexMinor` and `StarVertexMinor` problems. In particular we prove that both problems are NP-Complete (result 1.). In section 6.3 we provide an efficient algorithm for `StarVertexMinor` when the input graph is restricted to be distance-hereditary and prove that it is correct (result 2.). We also provide a fixed-parameter tractable algorithm for `StarVertexMinor` when the input graph is a circle graph and prove its correctness (result 3.). Finally we prove that any connected graph G' with three or less vertices is a vertex-minor of any connected graph G if $V(G') \subseteq V(G)$ and provide an efficient algorithm for finding the transformation that takes the former graph to the latter (result 4.).

6.2. Complexity

In this section we consider the computational time-complexity of deciding whether a graph state $|G\rangle$ can be transformed into another graph state $|G'\rangle$ using only single-qubit Clifford operations, single-qubit Pauli measurements and classical communication (`QubitMinor`). We will in fact prove that this problem is NP-Complete , even when $|G\rangle$ is in the restricted class of circle graph states and $|G'\rangle$ is a GHZ-state (up to LC). As we already shown in chapter 4, `QubitMinor` is equivalent to `VertexMinor`. We will here show that a highly restrictive version of the `VertexMinor` is NP-Hard , namely when G' is a star graph and G is in a strict subclass of circle graphs. Since we also prove that `VertexMinor` is in NP this then proves that `VertexMinor` is NP-Complete and our main result (corollary 6.2.2.1) follows.

6.2.1. `VertexMinor` is in NP

We begin by arguing that the vertex-minor problem is in NP . Given graphs G and G' such that $G' < G$, a witness to this relation would be a sequence of local complementations and vertex deletions that takes G to G' . It is not a priori clear that this sequence is polynomial in length w.r.t. to the number of vertices of G . However from theorem 4.5.3 one can argue that whenever there is such a sequence, there is also a sequence of polynomial length. This leads to the following lemma.

Lemma 6.2.1. *The decision problem `VertexMinor` is in NP .* \diamond

Proof. Let G and G' be graphs, on n and k vertices respectively. Furthermore, let \mathbf{u} be a sequence such that each element of $V(G) \setminus V(G')$ occur exactly once in \mathbf{u} . If $G' < G$ then, by theorem 4.5.3, there exists a sequence of operations $P \in \mathcal{P}_{\mathbf{u}}$,

as specified in definition 4.5.2, such that $P(G) \sim_{\text{LC}} G'$. Furthermore, the sequence of operations P consists of $\mathcal{O}(n - k)$ local complementations and vertex-deletions. A witness to the instance (G, G') will then be the sequence of operations P . On the other hand, if $G' \not\sim G$, then by theorem 4.5.3, there exist no $P \in \mathcal{P}_{\mathbf{u}}$ such that $P(G) \sim_{\text{LC}} G'$.

Given (G, G') and a sequence of operations P , a verifier can therefore perform the following protocol to check if (G, G') is a *yes*-instance of `StarVertexMinor`.

1. Compute $P(G)$.
2. Decide if $P(G) \sim_{\text{LC}} G'$ using Bouchet's algorithm for checking if two graphs are LC-equivalent [10].
3. Output *yes* if Bouchet's algorithm outputs `TRUE` and *no* otherwise.

The verifier will therefore output *yes* if P is such that $P(G) \sim_{\text{LC}} G'$ and *no* if $G' \not\sim G$, since then $P(G) \not\sim_{\text{LC}} G'$ for any P . Computing $P(G)$ can be done in time $\mathcal{O}(n^2(n-k))$, since each local complementation can be performed in time $\mathcal{O}(n^2)$ [10]. Furthermore, checking whether $P(G)$ and G' are LC-equivalent can be done in time $\mathcal{O}(k^4)$ using Bouchet's algorithm [10]. Thus the verifier will output *yes* or *no* in time $\mathcal{O}(n^2(n-k)) + \mathcal{O}(k^4)$. \square

6.2.2. `VertexMinor` is NP-Complete

Next we will argue that the problem `VertexMinor` is also NP-Hard and hence that it is NP-Complete. We will do this through a sequence of three reductions.

- Firstly we will reduce `StarVertexMinor` to `VertexMinor`. This is done in theorem 6.2.2.
- Secondly we will reduce the new problem `SOET`, which we introduce in section 4.7.6, to `StarVertexMinor`. This is done in section 6.2.2 using the results from section 4.7.6.
- Finally we will reduce the problem of deciding whether a 3-regular (or cubic) graph has a *Hamiltonian cycle* or not (`CubHam`), which is a known NP-complete problem [11], to the `SOET` problem. This is the most complicated part of the reduction and is done in several steps in section 6.2.2.

The reductions between these problems are summarized in fig. 6.2. Eventually we will have the following theorem, which can be considered the main theorem of this section.

Theorem 6.2.2. `VertexMinor` is NP-Complete \diamond

Proof. Note first that `StarVertexMinor` trivially reduces to `VertexMinor`. This is so since every *yes*(*no*)-instance of `StarVertexMinor` is also an *yes*(*no*)-instance

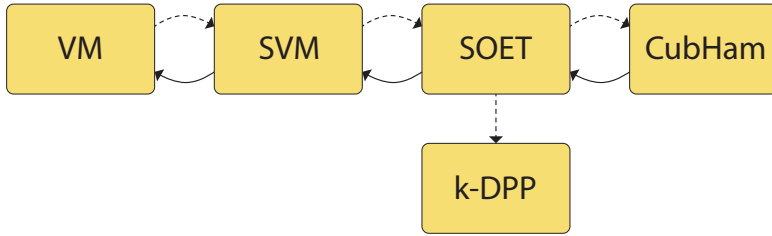


Figure 6.2: The different decision problems considered and how they can be reduced to each other. An solid arrow between problem A and B means that any instance of A can be reduced to B in polynomial time. A dashed line means that a subclass of instances of the A can be reduced to B in polynomial time. VertexMinor and StarVertexMinor are presented in section 4.5, SOET in section 4.7.6, CubHam in section 6.2.2 and $k\text{-DPP}$ (k -Disjoint Paths Problem) in section 6.3.2. The dashed line from StarVertexMinor to SOET is restricted to circle graphs. The dashed line from SOET to CubHam is restricted to triangle-expanded graphs $\Lambda(R)$ where the SOET is with respect to $V(R)$. Finally the dashed line from SOET to $k\text{-DPP}$ is for SOET problems where the SOET is with respect to a set of size k .

of VertexMinor . From theorem 6.2.3 we see that we can reduce the SOET problem to StarVertexMinor and finally from corollary 6.2.6.1 we see that we can reduce CubHam to the SOET problem. Since CubHam is a known NP-Complete problem this implies that VertexMinor is NP-Hard[11]. From lemma 6.2.1 we have that VertexMinor is in NP and hence it is NP-Complete. \square

As a direct corollary we have the following result.

Corollary 6.2.2.1. *Given two graph states $|G\rangle$ and $|G'\rangle$, deciding if $|G\rangle$ can be transformed into $|G'\rangle$ using only single-qubit Clifford operations, single-qubit Pauli measurement and classical communication is NP-Complete.* \diamond

Now we will detail every step in the above reduction. We begin with proving that the SOET decision problem reduces to StarVertexMinor .

Reducing the SOET problem to StarVertexMinor

In this section we show that the SOET problem reduces to StarVertexMinor . For this we will make use of the properties of circle graphs, discussed in section 4.7. In corollary 4.7.16.1 we showed that a 4-regular multi-graph F allows for a SOET with respect to a subset of its vertices $V' \subseteq V(F)$ if and only if an alternance graph $\mathcal{A}(U)$ (which is a circle graph), induced by some Eulerian tour on F , has $S_{V'}$ as a vertex-minor.

Since circle graphs are a subset of all simple graphs we can then decide whether a 4-regular graph F allows for a SOET with respect to some subset V' of it's vertices by constructing the circle graph induced by an Eulerian tour on F and checking whether it has a star-vertex minor on the vertex set V' . This leads to the following theorem.

Theorem 6.2.3. *The decision problem SOET reduces to StarVertexMinor .* \diamond

Proof. Let (F, V') be an instance of SOET , where F is a 4-regular multi-graph and V' a subset of the vertex set of F . Also let G be a circle graph induced by an Eulerian

tour U on F . From corollary 4.7.16.1 we see that G has $S_{V'}$ as a vertex-minor if and only if F allows for a SOET with respect to the vertex set V' . Since an Eulerian tour U can be found in polynomial time [12] and since G can be efficiently constructed given U , this concludes the reduction. \square

Reducing CubHam to the SOET problem

In this section we will prove that the SOET problem, as defined in problem 4.7.16, is NP-Complete by reducing the problem of deciding if a 3-regular graph is Hamiltonian (CubHam), a well-known NP-Complete problem [11], to the SOET problem (it is in NP by theorem 6.2.3 and lemma 6.2.1). For completeness we include the definition of a Hamiltonian graph.

Definition 6.2.4 (Hamiltonian). A graph is said to be Hamiltonian if it contains a Hamiltonian cycle. A Hamiltonian cycle is a cycle that visits each vertex in the graph exactly once. \diamond

We can use this to formally define the CubHam problem.

Problem 6.2.5 (CubHam). Let R be a 3-regular graph. Decide whether R is Hamiltonian. \diamond

The reduction of CubHam to the SOET problem is done by going through the following steps.

1. Introduce the notion of a (4-regular) triangular-expansion $\Lambda(R)$ of a 3-regular graph. This is done in definition 6.2.7.
2. Argue that given a 3-regular graph R , its triangular-expansion can be constructed efficiently. This is done in lemma 6.2.8.
3. Introduce the notions of *skip* and *true skip* that capture an essential behavior of SOETs on triangular-expansions of 3-regular graphs. This is done in section 6.2.2.
4. Prove that if a 3-regular graph R is Hamiltonian then the triangular-expansion $\Lambda(R)$ of R allows for a SOET with respect to the set $V(R)$. This is done in lemma 6.2.11.
5. Prove that if the triangular-expansion $\Lambda(R)$ of a 3-regular graph R allows for a special kind of SOET, called a HAMSOET with respect to R (HAMSOETs are defined in definition 6.2.12, but can be thought of as SOETs with no true skips), then the 3-regular graph R is Hamiltonian. This is done in lemma 6.2.13.
6. Prove that if the triangular-expansion $\Lambda(R)$ of a 3-regular graph R allows for a SOET with respect to $V(R)$ then it also allows for a HAMSOET with respect to R . This is done in lemma 6.2.14.

Performing all these steps will lead to the following theorems.

Theorem 6.2.6. *Let R be a 3-regular graph and $\Lambda(R)$ be its triangular-expansion as defined in definition 6.2.7. R is Hamiltonian if and only if $\Lambda(R)$ allows for a SOET with respect to $V(R)$.* \diamond

Proof. Let R be a 3-regular graph and let $\Lambda(R)$ be its triangular-expansion as defined in definition 6.2.7. If R is Hamiltonian then lemma 6.2.11 guarantees that $\Lambda(R)$ allows for a SOET with respect to the vertices $V(R)$. In the other direction, if $\Lambda(R)$ allows for a SOET with respect to the vertices $V(R)$ then we can see from lemma 6.2.13 that it also allows for a HAMSOET. The existence of a HAMSOET on $\Lambda(R)$ then implies, via lemma 6.2.13 that R has a Hamiltonian cycle and hence that it is Hamiltonian. This proves the theorem. \square

Corollary 6.2.6.1. *The SOET problem is NP-Complete.* \diamond

Proof. The Hamiltonian cycle problem (CubHam) is NP-Complete on 3-regular graphs [11]. We will reduce this problem to the SOET problem. Let R be an instance of CubHam , i.e. a 3-regular graph. From this 3-regular graph we can construct its triangular-expansion $\Lambda(R)$. In lemma 6.2.8 it is argued that this construction can be performed in $O(|V(R)|)$ time. We can then use theorem 6.2.6 to see that R is Hamiltonian if and only if $\Lambda(R)$ allows for a SOET with respect to the vertex set $V(R)$. Hence there exists an efficient reduction of CubHam to the SOET problem. This means that the SOET problem is NP-Hard. Furthermore, the SOET problem is in NP since it can be efficiently reduced to StarVertexMinor by theorem 6.2.3, which is in NP by lemma 6.2.1. Hence the SOET problem is NP-Complete. \square

Corollary 6.2.6.2. *The SOET problem is NP-Complete on graphs which are triangular-expansions of planar 3-regular triply-connected graphs, i.e. graphs in the set*

$$\{\Lambda(R) : R \text{ is planar, 3-regular and triply-connected}\}. \quad (6.2)$$

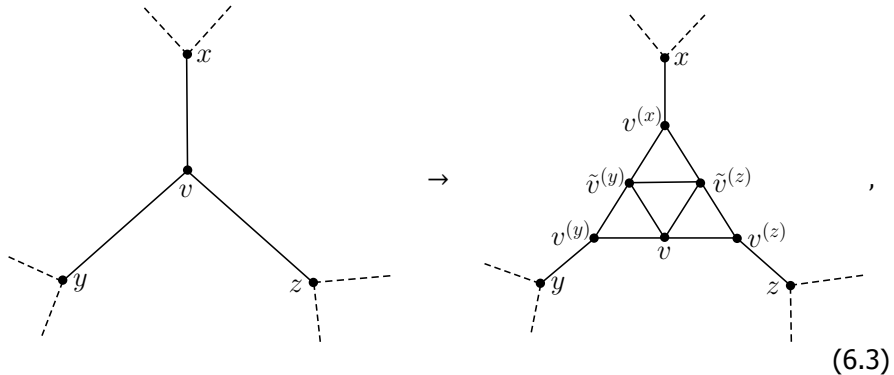
Proof. The proof is the same as the proof of corollary 6.2.6.1 but using the fact that CubHam is NP-Complete on planar triply-connected graphs. \square

Triangular-expansions

It now remains to prove lemmas 6.2.11, 6.2.13 and 6.2.14. These lemmas will relate Hamiltonian cycles on 3-regular graphs and SOETs on 4-regular multi-graphs by using a mapping from 3-regular graphs to 4-regular multi-graphs. We call this mapping 'triangular-expansion'. We have the following definition.

Definition 6.2.7 (Triangular-expansion). Let R be a 3-regular graph. A triangular-expansion $\Lambda(R)$ of a 3-regular graph R is constructed from R by performing the following two steps:

1. Replace each vertex v in R with the subgraph below



where x, y and z are the neighbors of v . We will denote this *triangle subgraph* associated to the vertex v with T_v , i.e. $T_v = G[\{v, v(x), v(y), v(z), \tilde{v}(y), \tilde{v}(z)\}]$.

2. Double every edge that is incident on two subgraphs $T_v, T_{v'}$.

◊

The graph $\Lambda(R)$ will be called a *triangular-expansion* of R . A multi-graph F that is the triangular-expansion of some 3-regular graph R will also be referred to as a triangular-expanded graph. Note that the triangular-expansion is not uniquely defined, since for each vertex $v \in R$ there is a choice how to orient the triangle with respect to the neighbors of v . Furthermore, the number of vertices in $\Lambda(R)$ is $6 \cdot |V(R)|$ and the number of edges is $2 \cdot |E(R)| + 9 \cdot |V(R)|$. In fig. 6.3 we show an example of a 3-regular graph and its triangular-expansion.

For a given triangle subgraph T_v in a triangular-expanded graph, we will refer to the vertices adjacent to other triangle subgraphs T_x, T_y, T_z as 'outer vertices' and label them according to the triangle subgraph they are adjacent to. Concretely we label the vertex in T_v that is adjacent to T_w as $v^{(w)}$, the index signifies which triangle subgraph it connects to.

In the following lemma we argue that this construction can be made efficiently in the size of R .

Lemma 6.2.8. *Let R be a 3-regular graph. We can construct its triangular-expansion in $O(|V(R)|^2)$ time.* ◊

Proof. Let R be a 3-regular graph. Without loss of generality assume some labeling on the vertices of R , i.e. $V(R) = \{v_1, \dots, v_k\}$ where $k = |V(R)|$. We begin by constructing the vertex set $V(\Lambda(R))$ off the triangular-expansion $\Lambda(R)$ of R .

For each $i \in [k]$ construct the set $V_i = \{v_i, \tilde{v}_i^{(v_j)}, \tilde{v}_i^{(v_{j'})}, v_i^{(v_j)}, v_i^{(v_{j'})}, v^{(v_j)}\}$ where $v_i \in V(R)$ and $v_j, v_{j'}, v_j$ are the three unique neighbors of v_i in R . Constructing this set takes $O(|V(R)|)$ as we must search the set of edges $E(R)$ of R to find the neighbors of v_i and we have that $|E(R)| = O(|V(R)|)$ since R is 3-regular. Thus

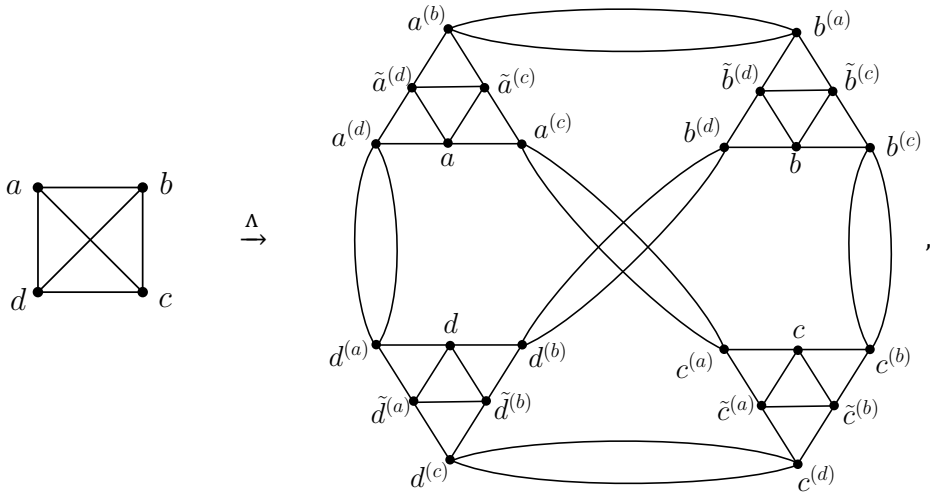


Figure 6.3: Figure showing the complete graph on vertices $V = \{a, b, c, d\}$ and its associated triangular-expansion $\Lambda(K_V)$.

6

constructing the set $V(\Lambda(R)) = \cup_{i \in [k]} V_i$ takes $O(|V(R)|^2)$ time.

Now we construct the edge multi-set $E(\Lambda(R))$ of the triangular-expansion of R . For each $i \in [k]$ we define the multi-set

$$\begin{aligned}
 E_i = \{ & (v_i, \tilde{v}_i^{(v_j)}), (v_i, \tilde{v}_i^{(v_{j'})}), (v_i, v_i^{(v_j)}), (v_i, v_i^{(v_{j'})}), (\tilde{v}_i^{(v_j)}, \tilde{v}_i^{(v_{j'})}), \\
 & (v_i^{(v_j)}, \tilde{v}_i^{(v_j)}), (v_i^{(v_{j'})}, \tilde{v}_i^{(v_{j'})}), (v_i^{(v_j)}, \tilde{v}_i^{(v_j)}), (v_i^{(v_{j'})}, \tilde{v}_i^{(v_{j'})}), \\
 & (v_i^{(v_j)}, v_j^{(v_i)}), (v_i^{(v_{j'})}, v_{j'}^{(v_i)}), (v_i^{(v_j)}, v_j^{(v_i)}) \}. \tag{6.4}
 \end{aligned}$$

This multi-set can be constructed in constant time. Hence the multi-set $E(\Lambda(R)) = \cup_{i \in [k]} E_i$ can be constructed in $O(|V(R)|)$ time. It is easy to check that the multi-graph defined by the vertex set $\Lambda(R)$ and edge multi-set $E(\Lambda(R))$ is indeed the triangular-expansion of R . This completes the lemma. □

Skips and true skips

A key insight in the behavior of the SOET problem on triangular-expanded graphs is the notion of *skips*. The word skip stems from the fact that since any SOET U on the triangular-expansion $\Lambda(R)$ of a 3-regular graph is a Eulerian tour, it must traverse any triangle subgraph T_v of $\Lambda(R)$ exactly three times. However in order for U to be a valid SOET with respect to $V(R)$ it must traverse the vertex v exactly two of those three times. This means it must *skip* the vertex v exactly once while traversing T_v .

We state a more formal definition of a (true) skip in terms of maximal sub-words (see definition 4.7.18).

Definition 6.2.9 (Skip). Let R be a 3-regular graph and let $\Lambda(R)$ be its triangular-expansion. Let U be a SOET on $\Lambda(R)$ with respect to $V(R)$. Let \mathbf{X} be a maximal sub-word of $m(U)$ (definition 4.7.18) associated to vertices $u, v \in V(R)$. We say the sub-trail described by \mathbf{X} makes a *skip* at a vertex $w \in V(R) \setminus \{u, v\}$ if $x_1^{(w)} w^{(x_1)}$ and $x_2^{(w)} w^{(x_2)}$ are sub-words of \mathbf{X} (up to reflection), where $x_1, x_2 \in V(R)$. Furthermore, if $x_1 \neq x_2$ then we say that the trail described by \mathbf{X} makes a true skip at w or sometimes that T_w contains a *true skip*. \diamond

Note that since \mathbf{X} is a maximal sub-word associated to u, v and $w \notin \{u, v\}$, w cannot be a letter of \mathbf{X} . As stated above, there is always exactly one maximal sub-word describing a sub-trail of a SOET that makes a skip at a certain triangle subgraph, as formalized in the following lemma. One can think of this lemma as giving necessary conditions for the existence of a SOET with respect to $V(R)$ on the triangular-expansion of a 3-regular graph R

Lemma 6.2.10. *Let R be a 3-regular graph and let $\Lambda(R)$ be its triangular-expansion. Let U be a Eulerian tour on $\Lambda(R)$. Let $w \in V(R)$ and let T_w be its triangle subgraph in $\Lambda(R)$. If U is a SOET on $\Lambda(R)$ with respect to $V(R)$ then there exists exactly one maximal sub-trail of U that makes a skip at w .* \diamond

Proof. We will prove this by showing that there are exactly three maximal sub-trails of U that traverse vertices in T_w and that exactly one of these makes a skip at w . Note first that the Eulerian tour U will enter and exit the triangle subgraph T_w exactly three times, since there are six edges incident to T_w . Hence there exists exactly three distinct edge-disjoint sub-trails, t_1, t_2 and t_3 of U that exit and enter T_w , i.e. the last vertices they traverse in T_w will be $x_i^{(w)}$, where x_i for $i \in [3]$ are the neighbors of w in R . Note that t_1, t_2 and t_3 each contain at least one vertex in T_w and they jointly traverse all edges in T_w (Since U is a Eulerian tour).

Now consider the vertex w . The Eulerian tour U traverses this vertex exactly twice. There are now two options for the trails t_1, t_2, t_3 . Either (1) one of the trails contains the vertex w exactly twice or (2) there are exactly two trails that contain the vertex w exactly once.

Now assume U is a SOET with respect to $V(R)$. If option (1) is true the tour U traverses the vertex w twice in succession before traversing any other vertex in the set V' . This is in contradiction with the assumption that U is a SOET. Hence if U is a SOET we must have that option (2) is true, that is, exactly two of the sub-trails t_1, t_2 and t_3 must contain w .

Lets assume without loss of generality that $w \in t_2$ and $w \in t_3$. Hence we have that $w \notin t_1$ and thus t_1 induces a sub-word $m(t_1)$ not containing w . The word

$m(t_1)$ can be extended to a maximal sub-word \mathbf{X} not associated to w but describing a sub-trail traversing vertices in T_w . Therefore by definition 6.2.9 \mathbf{X} describes a sub-trail making a skip at w . Furthermore, $m(t_2)$ has an overlap with two maximal sub-words associated with y_1, w and w, y_2 , respectively for some $y_1, y_2 \in V(R)$. Similarly for $m(t_3)$ and the vertices z_1, w and w, z_2 . Thus, the maximal sub-words that have an overlap with $m(t_2)$ or $m(t_3)$ do not make a skip at w .

Finally, there is no other maximal sub-word describing a sub-trail that traverses a vertex in T_w . The reason for this is that t_1, t_2 and t_3 jointly traverse all edges in T_w , so any maximal sub-trail traversing a vertex in T_w must have an overlap with t_1, t_2 or t_3 . The lemma then follows since we found exactly one maximal sub-word describing a sub-trail of U making a skip at w , i.e. the unique one that has an overlap with t_1 . \square

Equivalence between SOET's and Hamiltonian cycles

Now that we have defined the triangular-expansion $\Lambda(R)$ of a 3-regular graph R and discussed skips we can finally make the central argument of the reduction given in corollary 6.2.6.1. We begin by proving that if a 3-regular graph is Hamiltonian then its corresponding triangular-expansion $\Lambda(R)$ allows for a SOET w.r.t. the vertex set $V(R)$. We have the following lemma.

Lemma 6.2.11. *Let R be a 3-regular graph and $\Lambda(R)$ be a triangular-expansion as defined in definition 6.2.7. If R is Hamiltonian, then $\Lambda(R)$ allows for a SOET with respect to $V(R)$.* \diamond

Proof. Let R be Hamiltonian. This means there exists a Hamiltonian cycle M on R . We will prove that there exists a SOET U with respect to $V(R)$ on the triangular-expansion $\Lambda(R)$ of R by constructing, from the cycle M on R , a tour U that visits every vertex $v \in V(R)$ twice in the same order. We will then argue that this tour can always be lifted to a Eulerian tour and hence can be made into a SOET.

Note first that M induces an ordering on the vertices of R which without of loss of generality we will take to be v_1, \dots, v_k where $k = |V(R)|$. Note that for all $i \in [k-1]$ the vertices v_i and v_{i+1} are adjacent in R and so are v_k and v_1 . Now consider, for each $i \in [k-1]$ the triangle subgraphs $T_{v_i}, T_{v_{i+1}}$ and $T_{\hat{v}_i}$ in the triangular-expansion $\Lambda(R)$ of R where \hat{v}_i is the unique vertex adjacent to v_i in R that is not v_{i+1} or v_{i-1} . There are now three cases, depending on the orientation of the triangle subgraph T_{v_i} . Either (1) v_i is adjacent to $v_i^{(v_{i-1})}$ and $v_i^{(v_{i+1})}$ or (2) v_i is adjacent to $v_i^{(v_{i-1})}$ and $v_i^{(\hat{v}_i)}$ or (3) v_i is adjacent to $v_i^{(v_{i+1})}$ and $v_i^{(\hat{v}_i)}$.

For case (1), i.e. v_i is adjacent to $v_i^{(v_{i-1})}$ and $v_i^{(v_{i+1})}$ in T_{v_i} , we can define two edge-disjoint trails on the triangular-expansion $\Lambda(R)$ by their description as words $\mathbf{X}_i, \mathbf{X}'_i$ on the vertices of $\Lambda(R)$.

$$\mathbf{X}_i = v_i^{(v_{i-1})} \tilde{v}_i^{(v_{i-1})} v_i^{(\hat{v}_i)} \tilde{v}_i^{(v_{i+1})} v_i v_i^{(v_{i+1})} v_{i+1}^{(v_i)}, \quad (6.5)$$

$$\mathbf{X}'_i = v_i^{(v_{i-1})} v_i \tilde{v}_i^{(v_{i-1})} \tilde{v}_i^{(v_{i+1})} v_i^{(v_{i+1})} v_{i+1}^{(v_i)}. \quad (6.6)$$

An illustration of the trails described by these words is given in fig. 6.4a. We can similarly define two edge-disjoint trails for the cases (2) and (3). We will abuse notation and refer to the words as \mathbf{X}_i and \mathbf{X}'_i in all three cases. Importantly, for all three of these cases the trails described by $\mathbf{X}_i, \mathbf{X}'_i$ are edge-disjoint and both begin at $v_i^{(v_{i-1})}$ and end at $v_i^{(v_{i+1})}$. We can extend this definition to trails U_0, U'_0 described by the words $\mathbf{X}_0, \mathbf{X}'_0$, also for the case of the adjacent vertices v_k and v_1 . Note now that the walk U described by the word $\mathbf{W} = \mathbf{X}_0\mathbf{X}_1 \dots \mathbf{X}_{k-1}\mathbf{X}'_0\mathbf{X}'_1 \dots \mathbf{X}'_{k-1}$ is a tour and moreover that it traverses the vertices in $V(R)$ twice in the order v_1, \dots, v_k .

The tour U described by the word \mathbf{W} above visits every vertex in $V(R)$ such that $\mathbf{W}[V(R)] = v_1 \dots v_k v_1 \dots v_k$. However it is not yet a Eulerian tour, since it has not traversed all edges in $\Lambda(R)$. Note that the edges not traversed by U are precisely the triangular-expansions of the edges in R not traversed by the Hamiltonian cycle M . We can easily construct a Eulerian tour out of U by looping over all elements of the word \mathbf{X} and whenever we encounter a vertex $v_i^{(\hat{v}_i)}$ for all $i \in [k]$, where \hat{v}_i is defined as above, we check whether U already traverses the edges $(v_i^{(\hat{v}_i)}, \hat{v}_i^{(v_i)})$. If so we continue the loop and if not we insert the trail $(v_i^{(\hat{v}_i)}, \hat{v}_i^{(v_i)})\hat{v}_i^{(v_i)}(v_i^{(\hat{v}_i)}, \hat{v}_i^{(v_i)})v_i^{(\hat{v}_i)}$ into U at this position. This procedure is illustrated in fig. 6.4b. Now U is Eulerian and hence a SOET. This completes the proof. \square

Next we define a special type of SOET on triangular-expanded graphs, which we call HAMSOET s. These special SOET s on a triangular-expanded graph $\Lambda(R)$, are closely related to Hamiltonian cycles on R .

Definition 6.2.12 (HAMSOET). Let R be a 3-regular graph and $\Lambda(R)$ its triangular-expansion. Furthermore, let U be a SOET on $\Lambda(R)$ with respect to $V(R)$. U is called a HAMSOET with respect to R if, for all vertices $u, v \in V(R)$ we have that if u and v are consecutive with respect to the SOET U they are also adjacent in the graph R . \diamond

Next we prove that if the triangular-expansion of a 3-regular graph R allows for a HAMSOET with respect to R then the 3-regular graph R is Hamiltonian.

Lemma 6.2.13. Let R be a 3-regular graph and let $\Lambda(R)$ be its triangular-expansion. If $\Lambda(R)$ allows for a HAMSOET with respect to R , then R is Hamiltonian. \diamond

Proof. This follows by the definition of a HAMSOET. A HAMSOET U will induce a double occurrence word of the form

$$m(U) = \mathbf{X}_0 s_1 \mathbf{X}_1 s_2 \dots s_k \mathbf{X}_k s_1 \mathbf{X}'_1 s_2 \dots s_k \mathbf{X}'_k. \quad (6.7)$$

where $s_1, \dots, s_k \in V(R)$ with $k = |V(R)|$ and where \mathbf{X}_i and \mathbf{X}'_i are maximal sub-words associated to $s_i, s_{i+1} \in V(R)$. Now consider the induced double occurrence word $m(U)[V(R)]$. We have

$$m(U)[V(R)] = s_1 \dots s_k s_1 \dots s_k. \quad (6.8)$$

Consider now the sub-word $s_1 \dots s_k$ of $m(U)[V(R)]$. This sub-word describes a Hamiltonian cycle on R . To see this, note that each vertex in $V(R)$ occurs exactly

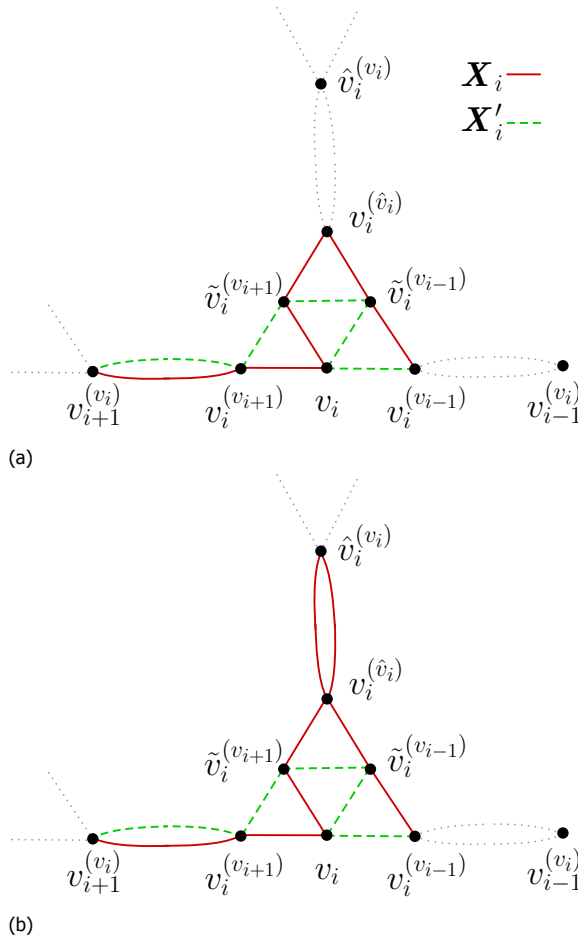


Figure 6.4: The trails used in the proof of lemma 6.2.11 to construct a SOET on a triangular-expanded graph $\Lambda(R)$ from a Hamiltonian cycle on R . Figure 6.4a shows the trails described by the words \mathbf{X}_i (solid red) and \mathbf{X}'_i (dashed green), defined in eqs. (6.5) and (6.6). Figure 6.4b shows how these trails can be extended to form a Eulerian tour and therefore a SOET with respect to $V(R)$. The dotted gray lines show edges of $\Lambda(R)$ which are not used by the trails.

once in $s_1 s_2 \dots s_k$. Furthermore, since s_i and s_{i+1} are consecutive in U , they are adjacent in R for all $i \in [k - 1]$, by definition of a HAMSOET. Finally, the same also holds for s_k and s_1 . Hence the tour on R described by $s_1 \dots s_k$ visits each vertex in R exactly once and is hence a Hamiltonian cycle. \square

We would like to prove that for any 3-regular graph R the existence of a SOET on its triangular-expansion $\Lambda(R)$ implies the existence of a Hamiltonian cycle on R . So far we have proven this only when $\Lambda(R)$ allows for a HAMSOET. However, a priori not all SOETs on triangular-expansions $\Lambda(R)$ have to be HAMSOETs. The reason for this is that consecutive vertices in a SOET are not necessarily adjacent

in R , since a SOET can contain true skips.

In the following lemma we will prove that consecutive vertices in a SOET are actually adjacent in R , except for two special cases. These special cases can be remedied since we show that for these cases we can always find a different SOET which is actually a HAMSOET.

The arguments proven below are understood the easiest when one reproduces the visual aids given in figs. 6.6 and 6.7 as one follows the arguments. We have the following lemma.

Lemma 6.2.14. *Let R be a 3-regular graph and $\Lambda(R)$ be its triangular-expansion. If $\Lambda(R)$ allows for a SOET with respect to $V(R)$, then $\Lambda(R)$ allows for a HAMSOET with respect to $V(R)$.* \diamond

Proof. To prove this lemma we will go through the following steps

Step 1 Note that if two vertices u, v in $V(R)$ (R being a 3-regular graph) are not adjacent in R but are consecutive in a SOET U with respect to $V(R)$ on $\Lambda(R)$, then the sub-trails of U described by the maximal sub-words associated to u and v must make a non-zero number of true skips in $\Lambda(R)$

Step 2 Argue by contradiction that this non-zero number of true skips can never be greater than one. This argument leverages lemma 6.2.15 which states that if a sub-trail of a SOET U with respect to $V(R)$ makes true skips at triangle subgraphs T_{w_1}, T_{w_2} for $w_1, w_2 \in V(R)$ and w_1, w_2 are adjacent in R then w_1, w_2 must actually be consecutive in the SOET U and lemma 6.2.16 which, in a slightly different initial situation than lemma 6.2.15, also concludes that two vertices must be consecutive in a SOET U with respect to $V(R)$.

Step 3 Argue by contradiction that there are only two possible ways for the sub-trails described by the maximal sub-words associated to u and v to make one true skip each. This argument also leverages lemmas 6.2.15 and 6.2.16.

Step 4 Argue that if a triangular-expansion of a 3-regular graph R allows for a SOET U as in **Step 3**, then it also allows for a SOET U' that is a HAMSOET.

Details of step 1

Let R be such that $\Lambda(R)$ allows for a SOET with respect to $V(R)$. Let U be any such SOET. If U is also a HAMSOET then we are directly done, therefore assume that U is not a HAMSOET. Since U is not a HAMSOET there must exist at least two vertices $u, v \in V(R)$ which are consecutive in U , but not adjacent in R . By definition, since u and v are consecutive and U is a SOET, there must exist exactly two different maximal sub-words \mathbf{X}, \mathbf{X}' of $m(U)$, associated to u and v . Since u and v are not adjacent in R , the trails described by \mathbf{X}, \mathbf{X}' must each traverse vertices in at least one (but possibly more) triangle subgraph that is not T_u or T_v . Since \mathbf{X} and \mathbf{X}' are maximal sub-words, they can not contain any vertex in $V(R)$ as a letter and hence the sub-trails described by \mathbf{X}, \mathbf{X}' must each make true skips (see

definition 6.2.9) at at least one triangle subgraph.

Assume w.l.o.g. that the sub-trail described by \mathbf{X} makes true skips at the triangle subgraphs $T_{s_1}, T_{s_2}, \dots, T_{s_r}$ in this order, and similarly for \mathbf{X}' and $T_{s'_1}, \dots, T_{s'_r}$. The true skips that the sub-trail described by \mathbf{X} makes, must be at different triangle subgraphs than the ones for \mathbf{X}' , since there can only be exactly one skip per triangle subgraph, due to lemma 6.2.10. The triangle subgraphs $T_{s_1}, T_{s_2}, \dots, T_{s_r}, T_{s'_1}, \dots, T_{s'_r}$ are therefore pairwise different. We will call this situation a rr' -skip, which is illustrated in fig. 6.5. Note that by assumption, r and r' are both greater than zero. We will first show that the case where either r or r' is greater than one can never occur if U is a SOET.

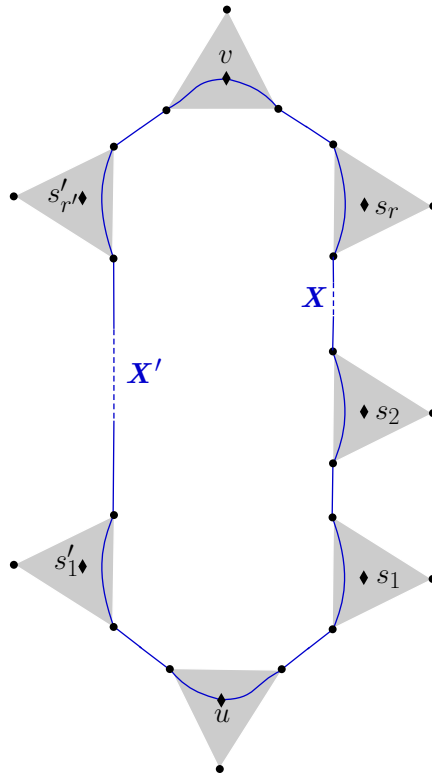


Figure 6.5: This figure is a visual aid for Step 1 of lemma 6.2.14. A simplified visualization of a triangular-expansion $\Lambda(R)$ of a 3-regular graph R is used. In the figure triangle subgraphs are shown in gray with only their outer vertices (circles) and vertices in $V(R)$ (diamonds) shown. Also shown are the sub-trails, of a SOET U , described by maximal sub-words \mathbf{X} and \mathbf{X}' associated to vertices u, v making true skips at triangle subgraphs T_{s_1}, \dots, T_{s_r} and $T_{s'_1}, \dots, T_{s'_r}$ respectively. These sub-trails always begin and end at a vertex in $V(R)$ but their path inside triangle subgraphs is not shown explicitly. Dashed lines are used to indicate that the sub-trails also traverse unspecified further parts of the graph $\Lambda(R)$.

Details of step 2

As a visual aid for the following argument, refer to fig. 6.6. We will make an argument by contradiction. Therefore let r be strictly greater than one. Consider the sub-trail described by the sub-word \mathbf{X} defined above. By assumption this sub-trail makes true skips at at least two adjacent triangle subgraphs. Take these to be T_{s_1} and T_{s_2} . As we will prove in lemma 6.2.15, if two adjacent triangle subgraphs T_{s_1}, T_{s_2} contain true skips, then s_1 and s_2 must be consecutive in the SOET U and that there is some maximal sub-word \mathbf{Y} of $m(U)$ associated to s_1, s_2 that contains the sub-word $s_1^{(s_2)} s_2^{(s_1)}$.

This implies that (by definition of SOET) there is some other maximal sub-word \mathbf{Y}' of $m(U)$ associated to s_1 and s_2 . This sub-word \mathbf{Y}' cannot contain the sub-word $s_1^{(s_2)} s_2^{(s_1)}$, as $s_1^{(s_2)} s_2^{(s_1)}$ already appears in both the maximal sub-word \mathbf{X} and the maximal sub-word \mathbf{Y} .

This implies the sub-trail described by \mathbf{Y}' must make a nonzero number of true skips at triangle subgraphs $T_{\hat{s}_1}, \dots, T_{\hat{s}_k}$ in that order. Now consider the triangle subgraphs T_{s_1} and $T_{\hat{s}_1}$. As we will prove below, lemma 6.2.16 applied to the triangle subgraphs $T_{s_1}, T_{\hat{s}_1}$ implies that the vertices s_1 and \hat{s}_1 must be consecutive in U . Call the maximal sub-word of $m(U)$ associated to these vertices \mathbf{W}_0 . Moreover, by lemma 6.2.16 this maximal sub-word contains the sub-word $s_1^{(\hat{s}_1)} \hat{s}_1^{(s_1)}$. Similarly we can see that the vertices s_2 and \hat{s}_k must be consecutive in U . Call the maximal sub-word of $m(U)$ associated to these vertices \mathbf{W}_k . By lemma 6.2.16 this maximal sub-word contains the sub-word $s_2^{(\hat{s}_{k+1})} \hat{s}_{k+1}^{(s_2)}$.

By applying lemma 6.2.16 again to all triangle subgraph pairs $T_{\hat{s}_i}, T_{\hat{s}_{i+1}}$ for $i \in [k-1]$ we come to the conclusion that \hat{s}_i, \hat{s}_{i+1} for $i \in [k-1]$ must be consecutive in U . Call the maximal sub-word of $m(U)$ associated to these vertices \mathbf{W}_i . By lemma 6.2.16 these maximal sub-words contain the sub-word $\hat{s}_i^{(\hat{s}_{i+1})} \hat{s}_{i+1}^{(\hat{s}_i)}$. This implies that U must traverse the vertices $s_1, \hat{s}_1, \dots, \hat{s}_k, s_2, s_1, \hat{s}_1, \dots, \hat{s}_k, s_2$ in order. Since by construction $\{s_1, s_2, \hat{s}_1, \dots, \hat{s}_k\} \neq V(R)$ we have that U is not a valid SOET on $V(R)$ (this can be seen by noting that e.g. T_{s_1} already contains a true skip, hence s_1 can not be part of the set). Hence by contradiction we must have $r \leq 1$. We can make the same argument for r' yielding $r' \leq 1$.

Details of step 3

As a visual aid for the following argument, refer to fig. 6.7. Now let $r = r' = 1$. This means the sub-trails described by the maximal sub-words \mathbf{X} and \mathbf{X}' associated to the vertices u and v make exactly one true skip each at triangle subgraphs triangle subgraphs $T_{s_1}, T_{s'_1}$ respectively. We will now argue that there are essentially only two ways that a SOET U with the above properties can exist on $\Lambda(R)$. This argument will again go in steps:

Step 3.1 Argue that the SOET U must have a maximal sub-word associated to the vertex s_1 and some other vertex x that describes a trail traversing the edge connecting the triangle subgraphs T_u and T_{s_1} .

Step 3.2 Argue that this sub-trail must make a true skip at the triangle subgraph T_u . Note that this is equivalent to arguing $x \neq u$.

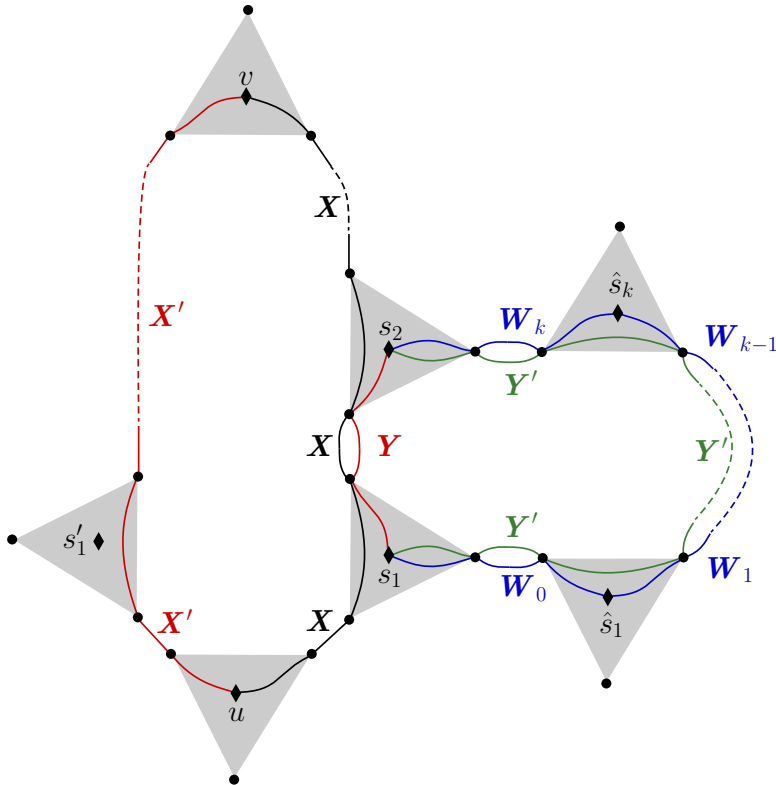


Figure 6.6: This figure is a visual aid for Step 2 of lemma 6.2.14. A simplified visualization of a triangular-expansion $\Lambda(R)$ of a 3-regular graph R is used. In the figure triangle subgraphs are shown in gray with only their outer vertices (circles) and the vertices in $V(R)$ (diamonds) shown. Also shown are sub-trail of a SOET U labeled by the maximal sub-words that describe them. These sub-trails always begin and end at a vertex in $V(R)$ but their path inside triangle subgraphs is not shown explicitly. Dashed lines are used to indicate that the sub-trails also traverse unspecified further parts of the graph $\Lambda(R)$. In this argument a contradiction is arrived at by first assuming the that the SOET U has a sub-trail described by the maximal sub-word \mathbf{X} associated to the vertices u, v makes true skips at triangle subgraphs T_{s_1}, T_{s_2} . Using lemma 6.2.15 it is shown that the maximal sub-word \mathbf{Y} must exist (associated to vertices s_1, s_2). This then means that the maximal sub-word \mathbf{Y}' must exist (also associated to s_1, s_2). The sub-trail described by this sub-word must make true skips at triangle subgraphs T_{w_1}, \dots, T_{w_k} . Using lemma 6.2.15 and lemma 6.2.16 it is then concluded that the SOET U must visit the vertices $s_1, w_1, \dots, w_k, s_2, s_1$ consecutively which means U is not a valid SOET (since $\{s_1, w_1, \dots, w_k, s_2\} \neq V(R)$). This is a contradiction.

Step 3.3 Apply the same sequence of arguments for the vertices s_1 and v and also for the vertices s'_1 and u and s'_1 and v .

Step 3.4 Conclude from the fact that the SOET U can only make a single true skip at the triangle subgraphs T_u and T_v that s_1 and s'_1 must be consecutive in U and moreover that the maximal sub-words \mathbf{Z}, \mathbf{Z}' associated to s_1 and s'_1 must make true skips at T_u and T_v respectively.

Details of step 3.1

Consider the second edge connecting the triangle subgraphs T_u and T_{s_1} . This is the edge $(u^{(s_1)}, s_1^{(u)})$. Since the SOET U is Eulerian it must traverse this edge. Note also that T_{s_1} already contains a true skip (made by the sub-trail described by \mathbf{X}). This means, by lemma 6.2.10 that any sub-trail of U crossing the edge $(u^{(s_1)}, s_1^{(u)})$ connecting T_u, T_{s_1} must traverse the vertex s_1 . Let \mathbf{Z} be the maximal sub-word describing the sub-trail starting at s_1 and containing the sub-word $u^{(s_1)}s_1^{(u)}$. This maximal sub-word is (by definition) associated to two vertices in $V(R)$. One of these vertices is s_1 and will label the other one x .

We will now argue that $x \neq u$ and hence that the sub-trail described by \mathbf{Z} makes a true skip at T_u . We do this by contradiction in the following argument.

Details of step 3.2

For this part of the argument assume that $x = u$. This means that u is consecutive to both s_1 and v . This means there must be some other maximal sub-word \mathbf{Z}' associated to s_1 and u . Note that this sub-word cannot contain the sub-word $u^{(s_1)}s_1^{(u)}$ as it is already contained in the maximal sub-words \mathbf{Z} and \mathbf{X}' .

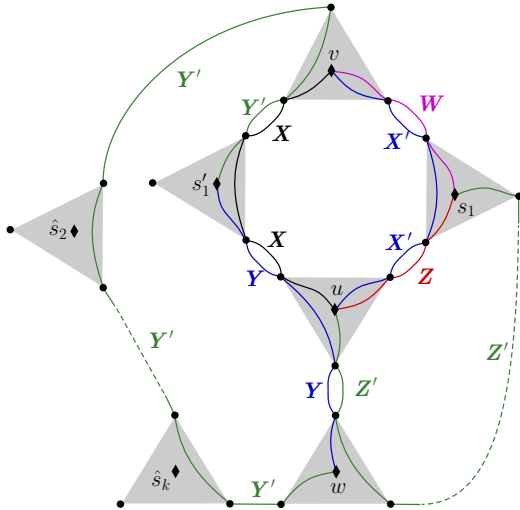
Now consider the unique third vertex that is adjacent to u in R (the vertex adjacent to u which is not s_1 or s'_1). Let us label this vertex w .

Since $T_{s'_1}$ already contains a true skip (which implies \mathbf{Z}' cannot connect to u by making a true skip at $T_{s'_1}$) the sub-trail described by the maximal sub-word \mathbf{Z}' must make a true skip at w . Now consider the maximal sub-word \mathbf{Y} of $m(U)$ that describes a sub-trail traversing the unused edge between $T_{s'_1}$ and T_u , i.e. \mathbf{Y} contains the sub-word $u^{(s'_1)}s'_1^{(u)}$. This sub-trail must be associated to s'_1 (since $T_{s'_1}$ already contains a true skip) and can not not be associated to u since u is already consecutive with two vertices in $V(R)$. This means the sub-trail described by the maximal sub-word \mathbf{Y} must make a true skip at the triangle subgraph T_u . Since the sub-word $u^{(s_1)}s_1^{(u)}$ is already contained in \mathbf{Z} and \mathbf{X}' the maximal sub-word \mathbf{Y} must contain the sub-word $u^{(w)}w^{(u)}$. Since T_w already contains a true skip this implies that w must be associated to the maximal sub-word \mathbf{Y} and hence that w and s'_1 are consecutive. This means there must be a second maximal sub-word \mathbf{Y}' associated to w and s'_1 .

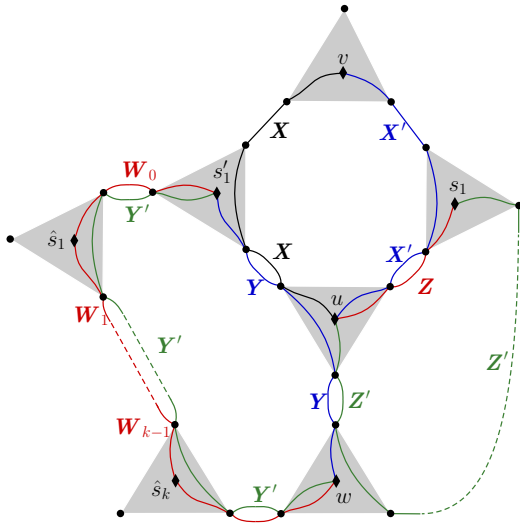
Since both edges connecting T_u and T_w have already been traversed by, the sub-trail described by the maximal sub-word \mathbf{Y}' must make true skips on some triangle subgraphs $T_{\hat{s}_1}, \dots, T_{\hat{s}_k}$.

There are now two possibilities. Either (1) we have that $\hat{s}_1 = v$ (see fig. 6.7a) or (2) that $\hat{s}_1 \neq v$ (see fig. 6.7b). We will now consider both of these cases:

1. If $\hat{s}_1 = v$ we can apply lemma 6.2.16 to the vertices v and s_1 to conclude that $\hat{s}_1 = v$ implies that v and s_1 are consecutive. Call the maximal sub-word connecting them \mathbf{W} . We now have that the vertices u, v and s_1 are pairwise consecutive to each other. Since $\{u, v, s_1\}$ is a strict subset of



(a)



(b)

Figure 6.7: This figure is a visual aid for Steps 3.1 and 3.2 of lemma 6.2.14. A simplified visualization of a triangular-expansion $\Lambda(R)$ of a 3-regular graph R is used. In the figure triangle subgraphs are shown in gray with only their outer vertices (circles) and vertices in $V(R)$ (diamonds) shown. Also shown are sub-trail of a SOET U labeled by the maximal sub-words that describe them. These sub-trails always begin and end at a vertex in $V(R)$ but their path inside triangle subgraphs is not shown explicitly. Dashed lines are used to indicate that the sub-trails also traverse unspecified further parts of the graph $\Lambda(R)$. In Step 3.2 of lemma 6.2.14 it is argued that if the sub-trail described by the maximal sub-word X' associated to vertices u, v makes a true skip at the triangle subgraph T_{s_1} then the sub-trail described by the maximal sub-word Z associated to s_1 and another vertex x must make a true skip at the triangle subgraph T_u (or equivalently that $x \neq u$). This is done by contradiction so it is assumed (as seen in both (a) and (b)) that $x = u$. This implies the existence of the maximal sub-word Z' which must make a true skip at T_w where w is the unique neighbor of u s.t. $w \neq s_1, s'_1$. This in turn implies that s'_1 and w must be consecutive and connected by a sub-trail described by the maximal sub-word labeled Y in both (a) and (b). This means another sub-trail connecting s'_1 and w must exist, which described by a maximal sub-word Y' . This sub-word makes true skips at triangle subgraphs $\hat{s}_1, \dots, \hat{s}_k$. There are now 2 options. Either, as shown in (a) we have that $v = \hat{s}_1$ or we have, as shown in (b) that $v \neq \hat{s}_1$. In the first case (a) lemma 6.2.16 is applied to conclude that v must be consecutive to s_1, s'_1 and u leading to a contradiction. In the second case (b) lemma 6.2.15 and lemma 6.2.16 are used to show that any SOET U must traverse the vertices $s'_1, w, \hat{s}_k, \dots, \hat{s}_1, s'_1$ in order leading to a contradiction.

$V(R), U$ cannot be a a SOET which is a contradiction.

- Now assume that $v \neq \hat{s}_1$. By lemma 6.2.16 we can now conclude that w and \hat{s}_k must be consecutive and that s'_1 and $\hat{s}_1 \neq v$ must be consecutive. We have to perform one last construction to prove the lemma. This construction is visualized in fig. 6.7b. Call the maximal sub-words

associated to these vertex pairs w and \hat{s}_k and s'_1 and \hat{s}_1 \mathbf{W}_k and \mathbf{W}_0 respectively. We can again use lemma 6.2.15 to conclude that \hat{s}_i is consecutive to \hat{s}_{i+1} for all $i \in [k - 1]$. Call the maximal sub-words associated to the vertices \hat{s}_i and \hat{s}_{i+1} \mathbf{W}_i . This implies that U must traverse the vertices $s'_1, \hat{s}_1, \dots, \hat{s}_k, w, s'_1, \hat{s}_1, \dots, \hat{s}_k, w$ in order. Since by construction $\{s'_1, s_w, \hat{s}_1, \dots, \hat{s}_k\} \neq V(R)$ we have that U is not a valid SOET on $V(R)$ (this can be seen by noting that e.g. T_u already contains a true skip, hence u can not be part of the set).

Hence in both cases we arrive at a contradiction. This means by contradiction that $x \neq u$ and thus that the sub-trail described by \mathbf{Z} makes a true skip at T_u .

Details of step 3.3

Now similarly to Step 3.1 consider the edge connecting T_{s_1} and T_v . We can again argue that there must exist a maximal sub-word \mathbf{Z}' associated to the vertex s_1 and some other vertex x' that describes a sub-trail that traverses this edge. By the same argument as Step 3.2 we can conclude that $x' \neq v$ and thus that \mathbf{Z}' describes a sub-trail making a true skip at T_v .

We can make the same argument for the vertex s'_1 establishing the existence of maximal sub-words $\hat{\mathbf{Z}}, \hat{\mathbf{Z}}'$ that describe sub-trails making true skips at T_u and T_v respectively.

Details of step 3.4

Note that the sub-trails described by \mathbf{Z} and $\bar{\mathbf{Z}}$ make true skips at the triangle subgraph T_u . Since T_u , by lemma 6.2.10 can only contain a single true skip and since $\mathbf{Z}, \bar{\mathbf{Z}}$ are maximal sub-words we must have that $\mathbf{Z} \sim \bar{\mathbf{Z}}$ and thus that the vertices s_1 and s'_1 are consecutive with respect to the SOET U . However since we must also conclude that $\mathbf{Z}' \sim \bar{\mathbf{Z}}'$ we have that s_1 and s'_1 are consecutive and have two maximal sub-words associated to them. Since there are no further constraints on U imposed by the the fact that the sub-words \mathbf{X}, \mathbf{X}' associated to the vertices v, u make true skips. Therefore SOETs with this type of behavior are in fact allowed. These SOETs can also be found explicitly, as can be seen in fig. 6.8. If a SOET U has this type of behavior (u and v are not adjacent in R but are consecutive in the SOET U on $\Lambda(R)$) we say that the SOET U has a *valid 11-skip*. Next we show that if a SOET U has a valid 11-skip, and thus is not a HAMSOET, it can always be turned into a HAMSOET by applying a fixed set of local complementation.

Details of step 4

We now show that a SOET with valid 11-skips can be turned into a HAMSOET. The two possibilities for a SOET with a valid 11-skip are shown in fig. 6.8. Note that these possibilities have the same 'local' structure, the only difference is how the rest of the SOET U is connected to the valid 11-skip. We first show the procedure that should be applied if the 11-skip is of the form in fig. 6.8a.

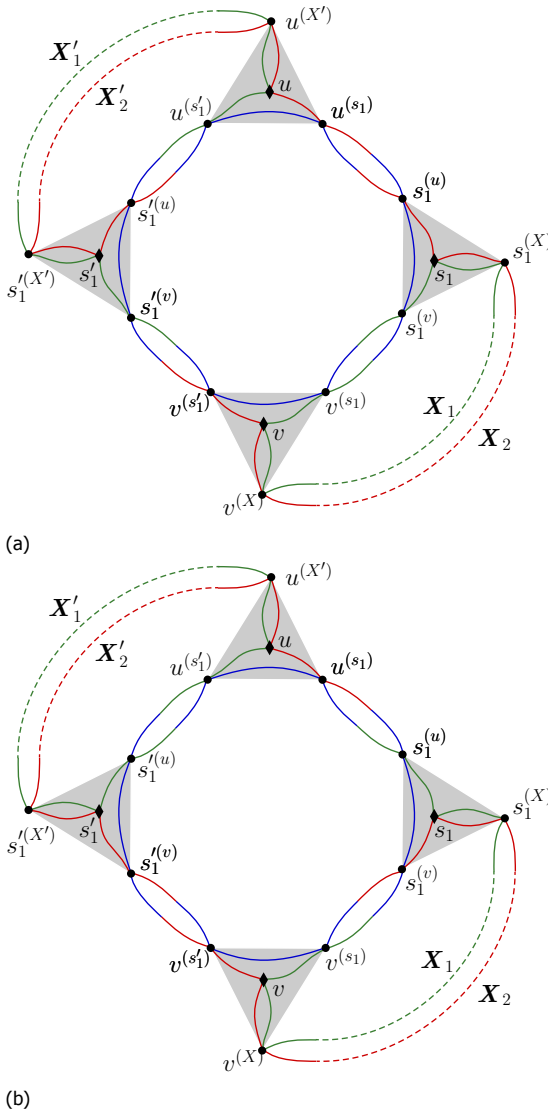


Figure 6.8: This figure is a visual aid for Step 4 of lemma 6.2.14. A simplified visualization of a triangular-expansion $\Lambda(R)$ of a 3-regular graph R is used. In the figure triangle subgraphs are shown in gray with only their outer vertices (circles) and the vertices in $V(R)$ (diamonds) shown. Also shown are sub-trail of a SOET U labeled by the maximal sub-words that describe them. These sub-trails always begin and end at a vertex in $V(R)$ but their path inside triangle subgraphs is not shown explicitly. Dashed lines are used to indicate that the sub-trails also traverse unspecified further parts of the graph $\Lambda(R)$. Figures 6.8a and 6.8b show the two valid 11-skips. The sub-word X_1 describes some sub-trail of U connecting T_v and T_{s_1} . With a slight abuse of notation we denote the endpoints of this trail by $v^{(X)}$ and $s_1^{(X)}$. Similarly for X_2 , X'_1 and X'_2 .

6

The SOET U in fig. 6.8a is of the form

$$m(U) = \mathbf{U}_1 v^{(s_1)} \mathbf{U}_2 v^{(s_1)} \mathbf{U}_3 s_1^{(u)} \mathbf{U}_4 s_1^{(u)} \mathbf{U}_5 \tag{6.9}$$

where

$$\mathbf{U}_1 = uu^{(s_1)}s_1^{(u)}s_1^{(v)} \quad (6.10)$$

$$\mathbf{U}_2 = vv^{(X)}\mathbf{X}_1s_1^{(X)}s_1s_1^{(v)} \quad (6.11)$$

$$\mathbf{U}_3 = v^{(s'_1)}s_1^{(v)}s_1^{(X')}s_1^{(X')}s_1^{(X')}s_1^{(X')}u^{(X')}uu^{(s'_1)} \quad (6.12)$$

$$\mathbf{U}_4 = s_1^{(v)}v^{(s'_1)}vv^{(X)}\mathbf{X}_2s_1^{(X)}s_1s_1^{(u)}u^{(s_1)}u^{(s'_1)} \quad (6.13)$$

$$\mathbf{U}_5 = s_1^{(X')}s_1^{(X')}s_1^{(X')}s_1^{(X')}u^{(X')} \quad (6.14)$$

where \mathbf{X}_1 is the word associated to the sub-trail connecting $v^{(X)}$ and $s_1^{(X)}$ as seen in fig. 6.8a and similarly for $\mathbf{X}'_1, \mathbf{X}_2, \mathbf{X}'_2$. Since our goal is to make this a HAMSOET we need to have that pairs of vertices in $V(R)$ are only consecutive w.r.t. U if they are adjacent in R . This can be done by applying $\bar{\tau}$ -operations to U at $v^{(s_1)}$ and $s_1^{(u)}$. The Eulerian tour U' after these operations will be described by

$$m(U') = m\left(\bar{\tau}_{(v^{(s_1)}, s_1^{(u)})}(U)\right) = \mathbf{U}_1v^{(s_1)}\widetilde{\mathbf{U}}_2v^{(s_1)}\mathbf{U}_3s_1^{(u)}\widetilde{\mathbf{U}}_4s_1^{(u)}\mathbf{U}_5 \quad (6.15)$$

where the over-set tilde indicates the mirror-inverting of a sub-word. Note that neither (u, v) or (s_1, s'_1) are consecutive anymore, but instead (u, s_1) and (v, s'_1) are now consecutive w.r.t. U' . To make sure that this procedure works we need to check two things: (1) U' is a SOET and (2) there are no additional consecutive pairs of vertices in U' that are not adjacent in R . To do this, let's look at the order U and U' traverse the vertices in $V(R)$, i.e. we will look at $m(U)[V(R)]$ and $m(U')[V(R)]$. Since U is a SOET we must have that $\mathbf{X}_1[V(R)] = \mathbf{X}_2[V(R)]$ and similarly $\mathbf{X}'_1[V(R)] = \mathbf{X}'_2[V(R)]$. Let's denote these words by $\mathbf{X}_V = \mathbf{X}_1[V(R)]$ and $\mathbf{X}'_V = \mathbf{X}'_1[V(R)]$. We then have that the double occurrence word of $m(U)$ induced by $V(R)$ is

$$m(U)[V(R)] = uv\mathbf{X}_Vs_1s_1'\mathbf{X}'_Vuv\mathbf{X}_Vs_1s_1'\mathbf{X}'_V \quad (6.16)$$

and similarly for U' we have

$$m(U')[V(R)] = us_1\widetilde{\mathbf{X}}_Vvs_1'\mathbf{X}'_Vus_1\widetilde{\mathbf{X}}_Vvs_1'\mathbf{X}'_V \quad (6.17)$$

It is therefore clear that the Eulerian tour U' is a SOET. Furthermore the only consecutive pairs of vertices in U' which were not consecutive in U are (u, s_1) and (v, s'_1) . Since (u, s_1) and (v, s'_1) are edges of R we see that we can iteratively apply this procedure to any valid 11-skip as in fig. 6.8a and turn the SOET into a HAMSOET. Similarly the SOET in fig. 6.8b can be turned into a HAMSOET by applying τ -operations to the vertices $s_1^{(u)}$ and $v^{(s'_1)}$. One can explicitly check this by applying the operations to U in fig. 6.8b which is given by

$$m(U) = \mathbf{U}_1s_1^{(u)}\mathbf{U}_2s_1^{(u)}\mathbf{U}_3v^{(s'_1)}\mathbf{U}_4v^{(s'_1)}\mathbf{U}_5 \quad (6.18)$$

where

$$\mathbf{U}_1 = uu^{(s_1)} \quad (6.19)$$

$$\mathbf{U}_2 = s_1^{(v)} v^{(s_1)} v v^{(X)} \mathbf{X}_1 s_1^{(X)} s_1 \quad (6.20)$$

$$\mathbf{U}_3 = u^{(s_1)} u^{(s'_1)} s_1^{(u)} s_1' s_1^{(X')} \mathbf{X}'_1 u^{(X')} uu^{(s'_1)} s_1^{(u)} s_1^{(v)} \quad (6.21)$$

$$\mathbf{U}_4 = v v^{(X)} \mathbf{X}_2 s_1^{(X)} s_1 s_1^{(v)} v^{(s_1)} \quad (6.22)$$

$$\mathbf{U}_5 = s_1^{(v)} s_1' s_1^{(X')} \mathbf{X}'_2 \quad (6.23)$$

with everything defined similarly to the case of fig. 6.8b. Going through a similar argument as above we can show that we can also turn the SOET U into a HAMSOET. This completes the lemma. \square

Lemma 6.2.15. *Let R be a 3-regular graph and $\Lambda(R)$ be its triangular-expansion. Also let u, v be adjacent vertices on R . Let U be a SOET on $\Lambda(R)$ with respect to $V(R)$. Let \mathbf{X} be a maximal sub-word of $m(U)$ not associated to u and/or v containing $u^{(v)}v^{(u)}$ and describing a sub-trail that makes true skips at T_u and T_v . Then u and v are consecutive in U and moreover $m(U)$ contains a sub-word of the form*

$$u\mathbf{Z}_1 u^{(v)} v^{(u)} \mathbf{Z}_2 v, \quad \mathbf{Z}_1 \subset V(T_u), \mathbf{Z}_2 \subset V(T_v), \quad (6.24)$$

\diamond

Proof. The situation described in the lemma is described graphically in fig. 6.9. Because U is a SOET the sub-word $v^{(u)}u^{(v)}$ must be contained exactly twice in $m(U)$. Note that at most one of these instances can be contained in the maximal sub-word \mathbf{X} . The other instance must be contained in a different maximal sub-word \mathbf{Z} . This maximal sub-word will be associated to two vertices w_1, w_2 . Note that since $v^{(u)}u^{(v)} \in \mathbf{Z}$ and $v^{(u)}u^{(v)} \in \mathbf{X}$ either we must have that $w_1 = u$ or that the sub-trail described by the maximal sub-word \mathbf{Z} makes a true skip at T_u . Since T_u already contains a true skip (made by the sub-trail described by \mathbf{X}) we must have that $w_1 = u$. We can make the same argument for the vertex v . This means the maximal sub-word \mathbf{Z} must be associated to u and v and hence that u, v must be consecutive in U and moreover we have that

$$\mathbf{Z} = \mathbf{Z}_1 u^{(v)} v^{(u)} \mathbf{Z}_2, \quad \mathbf{Z}_1 \subset V(T_u) \setminus \{u\}, \mathbf{Z}_2 \subset V(T_v) \setminus \{v\}. \quad (6.25)$$

\square

Lemma 6.2.16. *Let R be a 3-regular graph and $\Lambda(R)$ be its triangular-expansion. Also let u, v be adjacent vertices on R . Also take x_1, x_2 to be the vertices adjacent to u in R such that $x_1 \neq v, x_2 \neq v$. Let U be a SOET on $\Lambda(R)$ with respect to $V(R)$. Let \mathbf{Y} be a maximal sub-word of $m(U)$ not associated to u and/or v containing $u^{(x_1)}x_1^{(u)}$ and $u^{(x_2)}x_2^{(u)}$ and describing a sub-trail making a true skip at T_u . Also let \mathbf{X} be a maximal sub-word associated to u and a vertex $x_3 \neq v$ that describes a sub-trail*

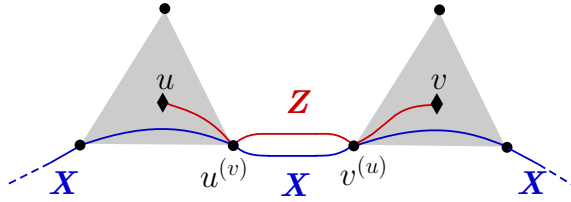


Figure 6.9: This figure is a graphical aid for lemma 6.2.15. Shown are two triangle subgraphs T_u, T_v (gray triangles) with outer vertices (circles) and the vertices u and v (diamonds) shown. The lemma starts from assuming the existence of the sub-trail described by the maximal sub-word \mathbf{X} , labeled as such in the figure. From this starting point the existence of the maximal sub-word \mathbf{Z} associated to the vertices u and v is derived.

making a true skip at v . Then u, v are consecutive and moreover $m(U)$ contains a sub-word of the form

$$u\mathbf{Z}_1u^{(v)}v^{(u)}\mathbf{Z}_2v, \quad \mathbf{Z}_1 \subset V(T_u), \mathbf{Z}_2 \subset V(T_v), \tag{6.26}$$

◇

Proof. The situation described in the lemma is described graphically in fig. 6.10. Because U is a SOET the sub-word $v^{(u)}u^{(v)}$ must be contained exactly twice in $m(U)$. Note that at most one of these instances can be contained in the maximal sub-word \mathbf{Y} and none can be contained in the maximal sub-word \mathbf{X} . This means there must be a maximal sub-word \mathbf{Z} of $m(U)$ (different from \mathbf{X} and \mathbf{Y}) containing $v^{(u)}u^{(v)}$. This maximal sub-word must again be associated with two vertices x, \hat{x} . If these vertices are not u, v then the sub-trail described by \mathbf{Z} must make true skips at T_u, T_v or both. Since both of these triangle subgraphs already contain true skips this is not possible and hence \mathbf{Z} must be associated to u and v which means they are consecutive. Moreover, by construction of \mathbf{Z} we have

$$\mathbf{Z} = \mathbf{Z}_1u^{(v)}v^{(u)}\mathbf{Z}_2, \quad \mathbf{Z}_1 \subset V(T_u) \setminus \{u\}, \mathbf{Z}_2 \subset V(T_v) \setminus \{v\}. \tag{6.27}$$

□

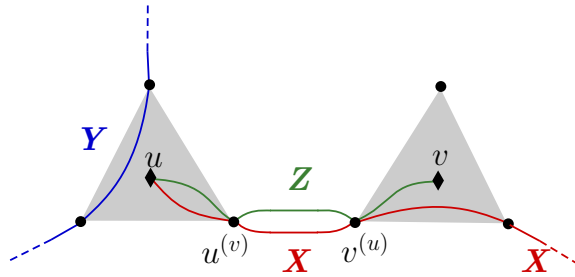


Figure 6.10: This figure is a graphical aid for lemma 6.2.16. Shown are two triangle subgraphs T_u, T_v (gray triangles) with outer vertices (circles) and the vertices u and v (diamonds) shown. The lemma starts from assuming the existence of the sub-trails described by the maximal sub-words \mathbf{Y} and \mathbf{X} , labeled as such in the figure. From this starting point the existence of the maximal sub-word \mathbf{Z} associated to the vertices u and v is derived.

6.3. Algorithms

In this section we provide algorithms for deciding if a graph state $|G\rangle$ can be transformed into another graph state $|G'\rangle$ using only LC + LPM + CC, (i.e. if $|G'\rangle$ is a qubit-minor of $|G\rangle$) when certain restrictions are put on the two graph states. We will again use the fact that $|G'\rangle$ is a qubit-minor of $|G\rangle$ if and only if G' is a vertex-minor of G , see theorem 4.4.2. Firstly, we describe an efficient algorithm to decide if $|G'\rangle$ is a qubit-minor of $|G\rangle$ whenever $|G'\rangle$ is a GHZ-state (up to LC) and $|G\rangle$ has Schmidt-rank width one. Phrased in graph theory, this is an algorithm for deciding if H is a vertex-minor of G , whenever H is a star graph and G is distance-hereditary. We prove that this algorithm always terminates and gives correct results. We also analyze its runtime and show that it is $\mathcal{O}(|n'| |n|^3)$, where n (n') is the number of qubits of $|G\rangle$ ($|G'\rangle$). Next we describe an algorithm for `QubitMinor`, whenever $|G'\rangle$ is a star graph and $|G\rangle$ is a circle graph state, i.e. that G is a circle graph. We prove that this algorithm is fixed-parameter tractable in the number of qubits of $|G'\rangle$.

6.3.1. Star graph as vertex-minor of a distance-hereditary graph

In this section we present an efficient algorithm for deciding whether a star graph on a given set of vertices V' is a vertex-minor of a given distance-hereditary graph G . This directly gives an efficient algorithm for deciding if a GHZ-state on a given set of qubits is a qubit-minor of a given graph state $|G\rangle$ with Schmidt-rank width one. Throughout this section we assume that the graph G is connected and distance-hereditary and that V' is a subset of its vertices. The algorithm presented in this section will return a sequence of vertices \mathbf{v} in $V(G)$, such that $\tau_{\mathbf{v}}(G)[V'] = S_{V'}$ if such a sequence exists and raise an error-flag otherwise, indicating that $S_{V'}$ is not a vertex-minor of G . We first present the algorithm in section 6.3.1, analyze its runtime in section 6.3.1 and prove that it is correct in section 6.3.1. The results of these sections imply the following theorem

Theorem 6.3.1. *Algorithm 6.1 takes a vertex-set V' and a graph G as input, rep-*

resented as an adjacency matrix, and has runtime $\mathcal{O}(|V'| |V(G)|^3)$. If the algorithm returns a sequence of vertices \blacktriangledown then the induced subgraph of $\tau_{\blacktriangledown}(G)$ on the vertices V' is a star graph. If the algorithm raises an error and G is distance-hereditary, then no star graph on vertices V' is a vertex-minor of G . \diamond

Proof. We provide a proof of the runtime of the algorithm in section 6.3.1 and proof of correctness in section 6.3.1. \square

An implementation in SAGE [7] of the algorithm can be found at [4].

The algorithm

We first give a rough sketch of the idea behind the algorithm. Remember that the task of the algorithm is to find a sequence of local complementations $\tau_{\blacktriangledown}$ such that the induced subgraph of $\tau_{\blacktriangledown}(G)$ on the vertices V' is a star graph.

The algorithm starts by choosing a one vertex c in V' which will become the center of the star graph on V' . It then proceeds by picking different vertices $v \in V'$ and making them adjacent to c by performing local complementations. After every vertex that is made adjacent to c the algorithm will check if the induced subgraph on the neighborhood of c is a star graph. If it is not it will attempt to turn it into a star graph by local complementations. If it fails at doing so it will raise an error and if it succeeds it will pick another vertex in V' and repeat the procedure until all vertices in V' are in the neighborhood of c . We will often call this process of making a vertex v adjacent to c ‘adding’ the vertex v to the star graph. To understand when the algorithm might fail we now zoom in on the situation where all but one vertex of V' has been added to the neighborhood of c . Let us call this vertex f . At this point in the algorithm the induced subgraph $G[V' \setminus \{f\}]$ is already a star graph (be previous successful iterations of this procedure).

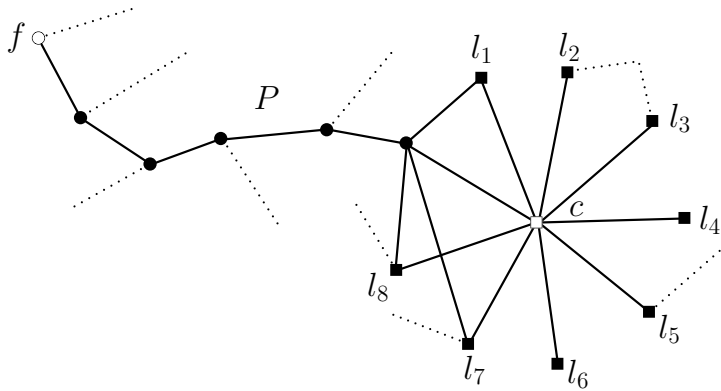
The task is now to turn $G[V']$ into a star graph by making f adjacent to the center c of $G[V' \setminus \{f\}]$ but to no other vertex of V' , and at the same time not change any edges in $G[V' \setminus \{f\}]$. This will be done in two steps, which are explained further below:

1. Make f and c adjacent, without changing any edges in $G[V' \setminus \{f\}]$. The star graph $S_{V'}$ is then a subgraph of the graph, but not necessarily an induced subgraph, since f could be also be adjacent to other vertices in V' than c . We will call these edges between f and vertices in $V' \setminus \{f\}$ *bad edges*. This first step is the task of algorithm 6.2 below. Interestingly, this step always succeeds if the graph is connected, even if the graph is not distance-hereditary.
2. Remove the bad edges, without changing any other edges between vertices in V' . The removal of the bad edges is the task of algorithm 6.1 below. Algorithm 6.1 tries to remove the bad edges by checking a few cases. Thus, one of the main results of this section is to prove that these cases provide a necessary condition for $S_{V'}$ being a vertex-minor of G .

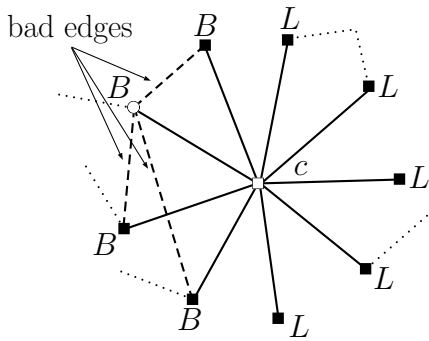
We will now describe the two main steps above of the algorithm in more detail. Let's denote the vertices as above and furthermore the current leaves in $G[V' \setminus \{f\}]$ as $V' \setminus \{c, f\} = \{l_1, \dots, l_k\}$.

Details of step 1:

The vertices c and f are made adjacent by performing local complementations along the shortest path P from c and f , see fig. 6.11a. The operations along the path P will in fact be either pivots, i.e. $\rho_{(u,v)} = \tau_v \circ \tau_u \circ \tau_v$, or single local complementations depending on the situation. The reason for this is to not remove edges between c and the l_i 's. The details of the operations along the path P are given in algorithm 6.2 together with the proof in section section 6.3.1.



(a)



(b)

Figure 6.11: Visualization of how a vertex f is added to the neighborhood of c . The original graph is shown in fig. 6.11a where the vertices $\{c\} \cup \{l_i\}_i$ (squares) already form a star graph and the dotted lines are arbitrary edges to the rest of the graph. The new vertex f (white circle) is made adjacent to the center c (white square) by performing pivots and local complementations along the shortest path P (black circles). After the pivots and local complementations the new vertex f is adjacent to the center c of the star graph but also to some leaves B by *bad edges* (dashed lines), see fig. 6.11b.

As mentioned above, by making f and c adjacent we might have also added bad edges between f and some of the vertices $\{l_i\}_i$, see fig. 6.11b. Let's denote the

set of vertices which are incident to a bad edge by B and the set of vertices not incident to a bad edge, apart from c , by $L = (V' \setminus \{c\}) \setminus B$. We call such a graph as the induced subgraph on V' a star-star graph, see definition 6.3.2.

Next, we must remove the bad edges in order to turn $G[V']$ into a star graph. Let G now be the graph with f and c adjacent but with possibly some bad edges.

Details of step 2:

In this step we will remove the bad edges, if we can. A situation where bad edges can be removed, as we will show, is when there exists a vertex $u \notin V'$, which is adjacent to all vertices in B but not to any vertex in L . The existence of such a vertex u is thus a sufficient condition for the removal of bad edges. When G is a distance hereditary graph, it turns out that this condition is also necessary, that is if no such vertex u exists, then the star graph on V' is not a vertex-minor of G , and we can stop the algorithm. This is shown in detail in section 6.3.1. For this statement to hold L cannot be empty, but this can always be achieved by performing a local complementation at c first if needed, which is done in line 14 in algorithm 6.1. Assume that there indeed exist such a vertex u , i.e.

$$(L \neq \emptyset) \wedge (u \notin V') \wedge (B \subseteq N_u) \wedge (L \cap N_u = \emptyset) \quad (6.28)$$

see fig. 6.12. Now u can be adjacent to c or not. Let's consider these cases separately:

- Case 1 u and c are not adjacent:

Remember that f is the center of the induced star graph $G[B]$. If a local complementation is performed at u , the bad edges are removed but new ones will be created between the vertices in $B \setminus \{f\}$. These new bad edges will then form a complete graph on $B \setminus \{f\}$ and we call such a graph on the vertices $V' \setminus \{f\}$ a complete-star graph, see definition 6.3.3.

Performing the same step again, i.e. doing a local complementation at another vertex adjacent to all vertices in $B \setminus \{f\}$, will remove all bad edges. We have then produced the star graph on V' in two steps.

- Case 2: u and c are adjacent:

In this case, if a local complementation is performed at u , some edges between c and vertices in L will be removed, which is not desired. We can solve this by finding another vertex h adjacent to both u and c but not to any other vertex in V' , by which we can remove the edge (u, c) , see fig. 6.13. In the following section we show that if there is no vertex h of this form, the star graph is not a vertex-minor of G and we can stop the algorithm.

To prove that the algorithm is correct we need to show that cases checked by algorithm 6.2 to remove the bad edges actually provides a necessary condition for $S_{V'}$ being a vertex-minor of G . To be precise, we will show that a necessary³

³This condition is not sufficient in itself, however theorem 6.3.4 provide a necessary and sufficient condition.

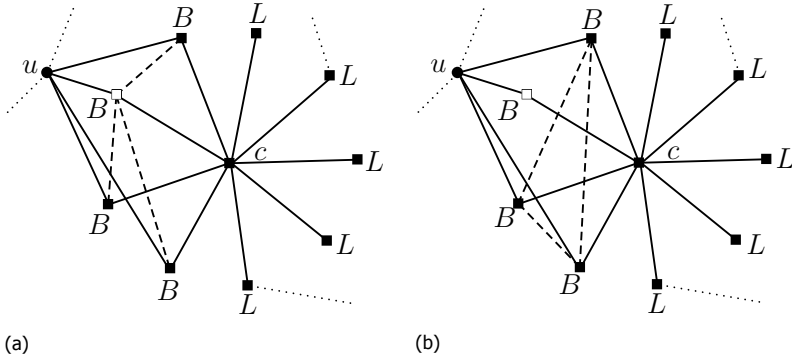


Figure 6.12: Visualization of how *bad edges* are removed. The original graph is shown in fig. 6.12a where the vertices $B \cup L \cup \{c\}$ (squares) are the desired vertices of the star graph, the dashed lines are the bad edges and vertex u (black circle) is as in eq. (6.29). Figure 6.12b shows the graph after performing a local complementation on u which produces a new leaf (white square) and makes the bad edges form a complete graph. This complete graph of bad edges can then be removed by finding a vertex u' that is adjacent to all vertices in B (and to none in L) and performing a local complementation at u' .

condition for the star graph on V' being a vertex-minor of G is

$$\mathcal{P}(B, L, c) = \exists u \in V \setminus V' : \left(B \subseteq N_u \wedge L \cap N_u = \emptyset \wedge \left((u, c) \notin E \vee \exists h : \left(h \in N_u \cap N_c \setminus \bigcup_{x \in V' \setminus \{c\}} N_x \right) \right) \right), \quad (6.29)$$

where $V' = B \cup L \cup \{c\}$ and L is assumed to be nonempty. It is important to note here that this condition is only valid if the graph is in the correct form, i.e. the induced subgraph on V' forms a star-star graph or a complete-star graph.

We formally state that eq. (6.29) is a necessary condition for the star graph on V' being a vertex-minor of G in theorem 6.3.4, which we prove in section 6.3.1. The theorem uses the notion of star-star and complete-star graphs which we formally define as:

Definition 6.3.2 (Star-star graph). A graph $G = (V, E)$ is called a star-star graph if there exist two subsets B and L and a vertex c , such that $\{B, L, \{c\}\}$ form a partition of V and $|B| > 1$. Furthermore $N_l = \{c\} \forall l \in L$ and $c \in N_b \forall b \in B$. Finally $G[B] = S_B$. Such a graph is denoted $SS_{(B,L,c)}$. \diamond

Definition 6.3.3 (Complete-star graph). A graph $G = (V, E)$ is called a complete-star graph if there exist two subsets B and L and a vertex c , such that $\{B, L, \{c\}\}$ form a partition of V and $|B| > 1$. Furthermore $N_l = \{c\}, \forall l \in L$ and $c \in N_b, \forall b \in B$. Finally $G[B] = K_B$. Such a graph is denoted $KS_{(B,L,c)}$. Note that if $|B| = 2$, G is also a star-star graph. \diamond

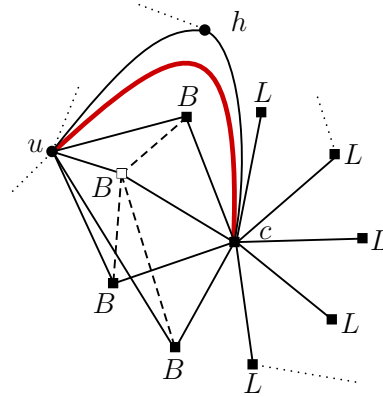


Figure 6.13: A visualization of the case where a vertex u used to remove the bad edges is also adjacent to c (red thick edge). The vertex h can be used to remove the edge (u, c) by applying a local complementation at h . Since h is not adjacent to any other vertex in V' , no edges in the induced subgraph on V' are changed by this local complementation.

Theorem 6.3.4. *Let G be a distance-hereditary graph on the vertices V and let V' be a subset of V . Furthermore, let $V' = B \cup L \cup \{c\}$ be a partition of V' and let $S_{V'}$ be a star graph on the vertices V' . Then the following statements hold*

- If $G[V'] = SS_{(B,L,c)}$ is a star-star graph and $|B| = 2$, then

$$\mathcal{P}(B, L, c) \Leftrightarrow S_{V'} < G. \tag{6.30}$$

- If $G[V'] = SS_{(B,L,c)}$ is a star-star graph then

$$\neg \mathcal{P}(B, L, c) \Rightarrow S_{V'} \not< G. \tag{6.31}$$

- If $G[V'] = SS_{(B,L,c)}$ is a star-star graph and f is the center of the star graph $G[B]$, then

$$\mathcal{P}(B, L, c) \Leftrightarrow KS_{(B \setminus \{f\}, L \cup \{f\}, c)} < G. \tag{6.32}$$

- If $G[V'] = KS_{(B,L,c)}$ is a complete-star graph then

$$\mathcal{P}(B, L, c) \Leftrightarrow S_{V'} < G. \tag{6.33}$$

◊

Theorem 6.3.4 implicitly gives a necessary and sufficient condition for when $S_{V'}$ is a vertex-minor of G , if $G[V']$ is a star-star graph. More precisely, if the induced subgraph on V' is a star-star graph and $\mathcal{P}(B, L, c)$ is true then we know that local complementations can be performed to turn the induced subgraph on $V' \setminus \{f\}$ into a complete-star graph, see eq. (6.32). Then, if $\mathcal{P}(B \setminus \{f\}, L \cup \{f\}, c)$ is again true, then a star graph can be created on V' by performing further local complementations, see eq. (6.33). If in any of these two steps, $\mathcal{P}(B, L, c)$ or $\mathcal{P}(B \setminus \{f\}, L \cup \{f\}, c)$ is false, then $S_{V'}$ is not a vertex-minor of G , see eqs. (6.31) and (6.33). In section 6.3.1 we prove these statements.

Runtime of the algorithm

The algorithm described in the previous section checks if a star graph with vertex set V' is a vertex-minor of a distance-hereditary graph G . Here we show that the runtime of this algorithm is $\mathcal{O}(|V'| |V(G)|^3)$. We will represent subsets of a base-set as unsorted binary lists⁴, where 1 indicates that an element in the base-set is in the represented set and 0 that an element in the base-set is not in the represented set. This will be the case both for sets of vertices and sets of edges. The base-set for sets of vertices will be the set of vertices $V(G)$ of the input graph G and the base-set for edges-sets will be $V(G) \times V(G)$. Thus, we assume that the input graph G is given as an unsorted binary list, of length $|V(G)|^2$, indicating which edges are in $E(G)$. This allows us to check if an edge (u, v) is in the graph or not in constant time. Furthermore, we assume that the input-set V' is also represented as an unsorted binary list, of length $|V(G)|$, indicating which of the vertices of G are in V' . We also assume that the size of $|V'|$ is given together with its representation, which allows us to faster create representations of subsets of V' .

Sets used internally by the algorithm (B , L and U) will also be represented as unsorted binary lists together with the size of the sets. The sizes of the sets will be updated accordingly whenever an element is added. Note that B and L are subsets of V' and will therefore be represented as unsorted binary lists, of length $|V'|$, indicating which elements of V' are in these sets. However, U is not a subset of V' and will therefore be represented by an unsorted binary list of length $|V|$. Thus, given a vertex v , checking if v is in a set of vertices V can be done in constant time and adding a vertex to a set can be done in constant time (flipping the bit at the corresponding position). Furthermore, iterating over elements in a set can be done in linear time with respect to the base-set, i.e. $\mathcal{O}(|V(G)|)$ for V' and U and $\mathcal{O}(|V'|)$ for B and L .

As described, the full algorithm starts by calling algorithm 6.1, which in turn calls algorithm 6.2, which again calls algorithm 6.1 and so on. We will see that the computation that dominates the runtime is updating the graph $\tau_{\mathbf{v}}(G)$ whenever \mathbf{v} is concatenated, as in line 13 of algorithm 6.1 and line 6 of algorithm 6.2. We will assume that both algorithm 6.1 and algorithm 6.2 have access to a common graph which they can update to $\tau_{\mathbf{v}}(G)$, whenever \mathbf{v} is concatenated, to prevent this from being done for the whole sequence \mathbf{v} every time. Note that $\tau_{\mathbf{v}}(G)$ takes up the same amount of space regardless of \mathbf{v} . Each local complementation in the sequence can be performed in time $\mathcal{O}(|V(G)|^2)$ [10]. Since algorithm 6.1 and algorithm 6.2 increase the length of \mathbf{v} by $\mathcal{O}(1)$ and $\mathcal{O}(|V(G)|)$ respectively each call, the runtime to update the graph $\tau_{\mathbf{v}}(G)$ is $\mathcal{O}(|V(G)|^3)$. We will now show that all other parts of both algorithm 6.1 and algorithm 6.2 takes time less than $\mathcal{O}(|V(G)|^3)$, which will imply that the total runtime is $\mathcal{O}(|V'| |V(G)|^3)$ since algorithm 6.2 is called $\mathcal{O}(|V'|)$

⁴It is possible to represent the sets in different ways, by for example (un)sorted lists containing the vertices as entries. However most reasonable data structures will not affect the total runtime of the algorithm but can reduce the memory used.

times⁵. Let's start by going through the runtime of algorithm 6.1 line by line:

- Line 6 (and 15, 24): Checking if there is only one element (or none) in V' (in B , in U) can be done in constant time, since we keep track of the sizes of these sets.
- Line 11: Finding a vertex $c \in V'$ adjacent to all vertices in V' (except itself) can be done in time $\mathcal{O}(|V'|^2)$ by checking for each vertex v in V' if $\tau_v(G)$ contains all edges in the set $\{(v, w) : w \in V' \setminus \{v\}\}$. Let c be the first such vertex v .
- Line 13 and 14: Constructing the sets B and L can be done in time $\mathcal{O}(|V'|^2)$ by checking, for each vertex v in $V' \setminus \{c\}$, if $\tau_v(G)$ contains at least one edge from the set $\{(v, w) : w \in V' \setminus \{c\}\}$. If this is the case, v will be added to the array representing B , otherwise v will be added to L .
- Line 19: Checking if $B = V' \setminus \{c\}$ can be done in time $\mathcal{O}(|V'|)$ by checking if all entries of the list representing B are 1, except at position c .
- Line 23: Constructing the set U can be done in time $\mathcal{O}(|V'| |V(G)|)$ by checking, for each u in $V(G) \setminus V'$, that $\tau_u(G)$ contains all edges in the set $\{(u, w) : w \in B\}$ and no edges in the set $\{(u, w) : w \in L\}$. If this is the case, u will be added to the array representing U .
- Line 28-38: The body of this *for*-loop will be executed $\mathcal{O}(|V(G)|)$ since there are at most $\mathcal{O}(|V(G)|)$ elements in U .
 - Line 29: Checking if (u, c) is an edge in $\tau_u(G)$ can be done in constant time.
 - Line 33: Finding a vertex h which is adjacent to both u and c but to no other vertex in V' can be done in time $\mathcal{O}(|V'| |V(G)|)$ (or determining that there is none), by first finding the neighbors of u , i.e all the vertices h such that (u, h) is an edge in $\tau_u(G)$ and then, for each neighbor h of u , checking if h is also adjacent to c but to no other vertex in V' . This is done by checking if (h, c) is an edge in $\tau_u(G)$ and that no element of the set $\{(h, w) : w \in V' \setminus \{c\}\}$ is.

Thus, the total runtime of algorithm 6.1, except for the recursive call to algorithm 6.2 in line 10, is $\mathcal{O}(|V'| |V(G)|^2)$ (from line 33 in the *for*-loop).

The runtime of each command in algorithm 6.2 is:

- Line 5: Picking the vertex f can be done in constant time (pick the first entry).
- Line 7: Finding a shortest path between f and c can be done in time $\mathcal{O}(|V(G)|^2)$ by using Dijkstra's algorithm [13].

⁵Note that algorithm 6.2 calls algorithm 6.1 with the set $V' \setminus \{f\}$, thereby reducing the size of V' in each recursive call.

- Line 8-15: The body of this *for*-loop will be executed $\mathcal{O}(|V(G)|)$ since the shortest path P is necessarily shorter than the number of vertices in $\tau_{\mathbf{v}}(G)$.
 - Line 9: Checking if f is adjacent to any vertex in $V' \setminus \{c\}$ can be done in time $\mathcal{O}(|V'|)$ by checking if any of the edges $\{(f, w) : w \in V' \setminus \{c\}\}$ are in $\tau_{\mathbf{v}}(G)$.
 - Line 10: The column with entry 1 in line 9 can be used for v here and thus only adds a constant time to the runtime.

Thus, the total runtime of algorithm 6.2, except for the recursive call to algorithm 6.1 in line 6, is $\mathcal{O}(|V(G)|^2)$ (from line 7 in the *for*-loop).

To further substantiate the efficiency of the algorithm we give actual run-times for an implementation of the algorithm in fig. 6.1.

Proof that the algorithm is correct

In this section we prove that the algorithm presented in the previous section works, i.e. it gives a sequence of local complementations \mathbf{v} such that $\tau_{\mathbf{v}}(G)[V'] = S_{V'}$, given a distance-hereditary graph G , if such a sequence exists. It is relatively easy to show that the algorithm gives the desired results when it does not return an error, which we show in section 6.3.1. The hard part is to prove that, when the algorithm gives an error-flag it is in fact not possible to produce the star graph, i.e. the star graph is not a vertex-minor of G , which is done in section 6.3.1. The notation will be the same as in the previous section, c is a vertex in V' and is adjacent to the rest of the vertices in V' . The vertices in $G[V' \setminus \{c\}]$ with degree greater than 0, i.e. the vertices incident on some bad edge, are denoted as the set B .

Algorithm succeeds In this section we show that if algorithm 6.1 returns a sequence \mathbf{v} , i.e. does not give an error-flag, then $\tau_{\mathbf{v}}(G)[V'] = S_{V'}$. We start by showing that algorithm 6.2 always succeeds and gives the desired result, assuming that algorithm 6.1 works. Recall that the task of algorithm 6.2 is to transform G using local complementations such that a star graph on V' is a subgraph of G . Algorithm 6.1 then tries to remove any *bad edges* to make the star graph an induced subgraph. We now show that the correctness of algorithm 6.2.

Note first that, after performing a pivot $\rho_{(v,u)}$, i.e. $\tau_u \circ \tau_v \circ \tau_u$, any neighbor of v will become a neighbor of u , except u itself. This means that after the first pivot in line 9 in algorithm 6.2, i.e. $\rho_{(p_1,f)}$, f and p_2 will be adjacent. We want to inductively show that this implies that after performing pivots along the whole path, f and c are adjacent. To do this we only need to make sure that a pivot does not remove edges in the later part of the path. More precisely, the pivot $\rho_{(p_i,f)}$ should not remove an edge (p_j, p_{j+1}) for $j > i$. The fact that later edges in the path are not removed follows from the properties of the pivot and that the path is a shortest path. Apart from edges incident on u or v , a pivot $\rho_{(v,u)}$ can only flip edges in the set $N_v \times N_u$. This shows that the pivot $\rho_{(p_i,f)}$ cannot remove an edge (p_j, p_{j+1}) since neither p_j or p_{j+1} is equal to f or p_i or is adjacent to f . If p_j or p_{j+1} would be adjacent to f , then this would not be a shortest path. We also need to make sure

that we do not remove the edges from $E(S_{V' \setminus \{f\}}) = \{(c, v) : v \in V' \setminus \{c, f\}\}$, when doing pivots along the path. By the same argument above we have that the pivot $\rho_{(p_i, f)}$ can only remove an edge in $E(S_{V' \setminus \{f\}})$ if f is adjacent to a vertex in $V' \setminus \{c\}$. This is the reason for the if-statement in line 5 in algorithm 6.2, where we then just perform a local complementation on the corresponding vertex in $v \in V' \setminus \{c\}$ which will make f and c adjacent. Performing the local complementation on such a vertex v will not remove edges in $E(S_{V' \setminus \{f\}})$, since v is a leaf in the induced subgraph on V' . Note that there are only two cases where *bad edges* are created. If f and c are made adjacent by a local complementation on a vertex $v \in V' \setminus \{c\}$, as in line 7, the bad edge (f, v) will be created. On the other hand, if this is not the case but the last vertex p_k is adjacent to some vertices $U \subseteq (V' \setminus \{c, f\})$, then the bad edges $\{(f, u)\}_{u \in U}$ will be created. In both of these cases $\tau_{\mathbf{v}}(G)[V']$ will be a star-star graph, see definition 6.3.2. Note that f can also be adjacent to some vertices in $V' \setminus \{f\}$, even before we perform the local complementations, but these edges will still form a star graph with f as the center. If one wants to minimize the number of local complementations and use local complementation instead of pivots along the path, this is in fact possible. The only place where a pivot is needed instead of a local complementation is towards the end of the path, when p_i is adjacent to a vertex in $V' \setminus \{c, f\}$ not on the path.

What is left to show is that if algorithm 6.1 succeeds and returns a \mathbf{v} , then $\tau_{\mathbf{v}}(G) = S_{V'}$. This is easy to see, since if we perform local complementations on such vertices we are looking for, i.e. u and possibly h in eq. (6.29), we will remove the bad edges and produce the star graph. If $|B| > 2$ this has to be done twice, as captured by the loop over i in algorithm 6.1. The reason for this is that, when doing a local complementation on such a u we complement the induced subgraph $G[B]$. Since $G[B]$ is a star graph, the induced subgraph after the local complementation will be a complete graph plus a single disconnected vertex which was the center of $G[B]$. Performing the step once more will then complement the complete graph, without the disconnected vertex, and all bad edges are thus removed.

Note that we have nowhere in this section used the assumption that the graph is distance-hereditary. This implies that if the algorithm succeeds we know that $\tau_{\mathbf{v}}(G) = S_{V'}$, independently of whether G is distance-hereditary, in fact even independently of the rank-width of G . Furthermore, since algorithm 6.2 always succeeds to make $\tau_{\mathbf{v}}(G)[V']$ connected and from the fact that any connected graph on two or three vertices is either a star graph or a complete graph, this implies that a star graph on any subset of size two or three is a vertex-minor of G , if the vertices are connected in G , which we make use of in section 6.4. On the other hand, if the algorithm stops and gives an error-flag, then we do not know in general if $S_{V'}$ is a vertex-minor of G or not. In the next section we show that if the graph is distance-hereditary and the algorithm gives an error-flag we actually do know that $S_{V'}$ is not a vertex-minor of G .

Algorithm gives error In this section we prove that if algorithm 6.1 gives an error-flag, i.e. if $\mathcal{P}(B, L, c)$ in eq. (6.29) is false, then the star graph is not a vertex-minor of the input graph. At the steps in the algorithm where the error-flag can be

raised, we know that the induced subgraph on V' is either a star-star graph (definition 6.3.2) or a complete-star (definition 6.3.3) graph as shown in section 6.3.1. The proof will follow the following sequence of steps.

1. Prove for any distance-hereditary graph G that if $\mathcal{P}(B, L, c)$ is false and $G[V']$ is a star-star graph (or complete-star graph) where $|V'| = 4$ then $S_{V'}$ is not a vertex-minor of G . This is done in theorem 6.3.5.
2. Use the case proven in step 1 to argue that if $\mathcal{P}(B, L, c)$ is false and $G[V']$ is a star-star graph where $|V'| > 4$ then $S_{V'}$ is not a vertex-minor of G . This is done in theorem 6.3.6.
3. Use the case proven in step 1 to argue that if $\mathcal{P}(B, L, c)$ is false and $G[V']$ is a complete-star graph where $|V'| > 4$ then $S_{V'}$ is not a vertex-minor of G . This is done in theorem 6.3.8.

Proof for a star-star (complete-star) graph of size 4 We will first show in theorem 6.3.5 that if $\mathcal{P}(B, L, c)$ is false and $|V'| = 4$, then $S_{V'}$ is not a vertex-minor of G . This will then allow us to prove the statement for the cases where $|V'| \geq 4$.

Theorem 6.3.5. *Let's assume that G is a distance-hereditary graph with the following induced subgraph (which is both a star-star and a complete-star-graph)*

$$G[V' = \{1, 2, 3, 4\}] = \begin{array}{c} \bullet 1 \\ | \\ \bullet 2 \\ / \quad \backslash \\ \bullet 3 \quad \bullet 4 \end{array} . \tag{6.34}$$

Furthermore assume that $\mathcal{P}(B, L, c)$ in eq. (6.29) is false, where $B = \{3, 4\}$, $L = \{1\}$ and $c = 2$. Then $S_{V'} \not\prec G$. \diamond

Proof. We will prove this by first showing that if $\mathcal{P}(B, L, c)$ is false and $|V(G)| > 4$, then there exist a removable leaf or twin.⁶ This then implies we can actually delete removable leaves and twins, i.e. vertices in $T(G) \setminus V'$, until there is only the vertices in V' left. This is because, if $\mathcal{P}(B, L, c)$ is false, then it is also false for any graph reached by deleting vertices in $V \setminus V'$. From theorem 4.8.4, i.e. the fact that deletion of removable twins or leaves does not change the property of whether a graph on a V' is a vertex-minor, we then know that $G[V']$ is in fact the only vertex-minor of G on the vertices V' . Since $S_{V'} \neq_{LC} G[V']$, the theorem follows.

Let's first look at what $\mathcal{P}(B, L, c)$ implies. Visually, $\mathcal{P}(B, L, c)$ is false if there exist no $u, h \in V \setminus V'$ such that

$$G[V' \cup \{u\}] = \begin{array}{c} \bullet 1 \\ | \\ \bullet 2 \\ / \quad \backslash \\ \bullet 3 \quad \bullet 4 \\ | \quad | \\ \bullet u \end{array} \quad \vee \quad G[V' \cup \{u, h\}] = \begin{array}{c} \bullet 1 \\ | \\ \bullet 2 \\ / \quad \backslash \\ \bullet 3 \quad \bullet 4 \\ | \quad | \\ \bullet u \quad \bullet h \end{array} . \tag{6.35}$$

⁶As in section section 4.8, removable means a vertex not belonging to the target vertices V'

There are only two ways for $\mathcal{P}(B, L, c)$ to be false; either $(N_3 \cap N_4) \setminus \{2\} = \emptyset$ or

$$(N_3 \cap N_4) \setminus \{2\} \neq \emptyset \quad \wedge \quad \forall u \in (N_3 \cap N_4) \setminus (N_1 \cup \{2\}) : \\ \left((u, 2) \in E(G) \wedge (N_u \cap N_2) \setminus (N_1 \cup N_3 \cup N_4) = \emptyset \right). \quad (6.36)$$

We will consider these two cases separately and prove that if either is true, then $T(G) \setminus V' \neq \emptyset$. In most cases below we will do this by showing that one of the four vertices in V' is not in $T(G)$ which shows that $T(G) \setminus V' \neq \emptyset$ since $T(G) \geq 4$ by theorem 4.8.10.

Case 1:

To prove that if $(N_3 \cap N_4) \setminus \{2\} = \emptyset$ and $|V(G)| > 4$, then there exists a removable leaf or twin, we consider the following cases.

- Assume that $|(N_3 \cup N_4) \setminus \{3, 4\}| > 1$. Since, by assumption $N_3 \cap N_4 = \{2\}$, 3 and 4 does not form a twin-pair. Also, neither 3 nor 4 is not a twin, since the twin-partner would have to be a common neighbor of 3 and 4. Furthermore, neither 3 or 4 is a leaf. Thus, the only way for $3 \in T(G)$, is if 3 is an axil, requiring some vertex not in V' being a leaf.⁷ Finally, if $3 \notin T(G)$, then there exist a vertex in $T(G)$ which is not in V' , since by theorem 4.8.10 we know that $|T(G)| \geq 4$.
- Assume that $(N_3 \cup N_4) \setminus \{3, 4\} = \{2\}$.
 - Assume that $|N_1| > 1$. Then 1 is not a leaf and 2 is not an axil. Furthermore, since nothing else is connected to 3 and 4, 2 is not a twin. Thus, the only way for $2 \in T(G)$, is if 2 is an axil, requiring some vertex not in V' being a leaf. Finally, if $2 \notin T(G)$, then since $|T(G)| \geq 4$ by theorem 4.8.10, there must exist a vertex in $T(G)$ which is not in V' .
 - Assume that $|N_1| = 1$. Then 2 is necessarily a cut-vertex and $G \setminus 2$ will contain a connected component with no vertices in V' , since $|V(G)| > 4$. Thus, there exist a vertex in $T(G)$ which is not in V' by corollary 4.8.10.1.

Case 2:

To prove that if eq. (6.36) is true then there exists a removable leaf or twin, we consider the following cases.

- Assume that $|N_1| > 1$. Then if 2 is an axil, the corresponding leaf cannot be in V' , since 1 is not a leaf. Furthermore, since 2 is not a leaf, if $2 \in T(G)$ then 2 has a twin-partner not in V' , which is then also in $T(G)$. On the other hand if $2 \notin T(G)$, then there exist a vertex in $T(G)$ which is not in V' , by theorem 4.8.10.
- Assume that $N_1 = \{2\}$.
 - Assume that $|(N_3 \cup N_4) \setminus \{3, 4\}| > |N_3 \cap N_4|$. In this case, 3 and 4 does not form a twin-pair. Furthermore, neither 3 or 4 is a leaf. Thus the only only way for 3(or 4) $\in T(G)$, is if 3(4) is an axil or a twin, requiring some vertex not in V' being a leaf or a twin. Finally if 3(4) $\notin T(G)$, then

⁷The same for 4.

there exist a vertex in $T(G)$ which is not in V' , since by theorem 4.8.10 we know that $|T(G)| \geq 4$.

- Assume that $(N_3 \cup N_4) \setminus \{3, 4\} = N_3 \cap N_4$. We will for this case show that $|T(G) \setminus V'| > 0$ by assuming that $T(G) = V'$ and arriving at a contradiction. Since this implies that $T(G) \neq V'$ and from the fact that $|T(G)| \geq 4$, we know that $|T(G) \setminus V'| > 0$. Consider the induced subgraph $G \setminus 4$ which is also distance-hereditary.⁸ From theorem 4.8.10 we know that $|T(G \setminus 4)| \geq 4$, since $(N_3 \cap N_4) \setminus \{2\} \neq \emptyset$ and therefore $|G \setminus 4| \geq 4$. Thus, there is a vertex $v \notin V'$ such that $v \in T(G \setminus 4)$ but $v \notin T(G)$. Note that by the assumption from eq. (6.36), any neighbor of 4, except 2, is also a neighbor of both 2 and 3. The removal of 4 cannot therefore create a new leaf in $V \setminus V'$. The only option left is if there are two vertices $v, v' \in V(G) \setminus \{4\}$, such that v, v' form a twin-pair in $G \setminus 4$ but not in G . If v and v' are such vertices, it must be the case that 4 is adjacent to exactly one of v and v' . Assume without loss of generality that 4 is adjacent to v' but not to v . The neighborhoods of these vertices are then

$$N_v = N_{v'} \setminus \{4\} \quad \wedge \quad 4 \in N_{v'}. \quad (6.37)$$

Note that the vertices adjacent to 4 are $(N_3 \cap N_4) \cup \{3\}$. Firstly, v' cannot be in $N_3 \cap N_4$, since v is then necessarily adjacent to 3 but not to 4 which contradicts the assumption that $(N_3 \cup N_4) \setminus \{3, 4\} = N_3 \cap N_4$. Secondly, v' cannot be 3, since v is then necessarily a neighbor of 2 and all vertices in $N_3 \cap N_4$, contradicting the second part of eq. (6.36). □

6

Proof for star-star graphs We are now able to prove the same statement as in theorem 6.3.5 but for $|V'| \geq 4$. The case when $G[V']$ is a star-star graph is given in theorem 6.3.6.

Theorem 6.3.6. *Let's assume that G is a distance-hereditary graph and V' is a subset $V' \subseteq V(G)$ such that the induced subgraph $G[V']$ is a star-star graph $SS_{(B', L', c')}$. Furthermore assume that $\mathcal{P}(B', L', c')$ is false, then $S_{V'} \triangleleft G$.* ◊

Proof. Pick an edge in $(b_1, b_2) \in G[B']$, which exist since $|B'| > 1$ and $G[B']$ is a star graph. We will prove this by first showing that

$$\neg \mathcal{P}(B', L', c') \quad \Rightarrow \quad \exists l \in L' : (\neg \mathcal{P}(\{b_1, b_2\}, \{l\}, c')).^9 \quad (6.38)$$

Then from theorem 6.3.5 we know that $S_{\{l, c', b_1, b_2\}} \triangleleft G$ for some $l \in L'$ and the corollary follows, because if $S_{\{l, c', b_1, b_2\}}$ is not a vertex-minor of G then neither is $S_{V'}$, since $S_{\{l, c', b_1, b_2\}} < S_{V'}$. To show that eq. (6.38) is true, we instead show the contrapositive statement, i.e.

$$\forall l \in L' : (\mathcal{P}(\{b_1, b_2\}, \{l\}, c')) \quad \Rightarrow \quad \mathcal{P}(B', L', c'). \quad (6.39)$$

⁸Induced subgraphs of a distance-hereditary graph are distance-hereditary.

⁹Remember that $L' \neq \emptyset$, by definition of a star-star graph.

Let $Q(u, B, L, c)$ be the expression on $\mathcal{P}(B, L, c)$ such that

$$\mathcal{P}(B, L, c) = \exists u \in V \setminus V' : Q(u, B, L, c) \quad (6.40)$$

For each $l \in L'$, let u_l be a vertex in $V \setminus \{l, c', b_1, b_2\}$ such that $Q(u_l, \{b_1, b_2\}, \{l\}, c')$ is true. We will now show that for at least one of these u_l , $u_l \in V \setminus V'$ and $Q(u_l, B', L', c')$ is true. To show that for at least one u_l , $u_l \in V \setminus V'$ and $Q(u_l, B', L', c')$ is true we will go through the following steps:

1. Show that

$$\forall l \in L' : (u_l \notin V') \quad (6.41)$$

2. Show that

$$\forall l \in L' : (B' \subseteq N_{u_l}) \quad (6.42)$$

3. Show that

$$\exists l \in L' : (L' \cap N_{u_l} = \emptyset) \quad (6.43)$$

4. Fix $l \in L'$ to be such that the corresponding expression in eq. (6.43) is true.

5. Show that

$$(u_l, c') \in E(G) \vee \exists h : \left(h \in N_{u_l} \cap N_{c'} \setminus \bigcup_{x \in V' \setminus \{c'\}} N_x \right) \quad (6.44)$$

If all the statements in the above steps are shown to be true we know that there exist a u_l in $V \setminus V'$ such that $Q(u_l, B', L', c')$ and therefore $\mathcal{P}(B', L', c')$ is true. It is important to note that even if we consider the statements in the above steps separately we know that there exist at least one u_l that simultaneously satisfy all. To see this, note that we only use an existential quantifier in step 3 and in step 5 we consider a u_l which satisfies the corresponding property in step 3. Let's now consider the steps 1 through 5 one by one.¹⁰ We will in these steps often claim that certain small graphs are not distance-hereditary. Verifying this can be done by hand or using our code supplied at [4].

Step 1:

Firstly, since $Q(u_l, \{b_1, b_2\}, \{l\}, c')$ is true we know that $u_l \neq c'$. Similarly, we know that $u_l \in (N_{b_1} \cap N_{b_2}) \setminus \{c'\}$, thus u_l is not in B' , since $G[B']$ is a star graph. Furthermore, since b and l are not adjacent $\forall b \in B'$ and $\forall l \in L'$, u_l is not in L' . We therefore know that $u_l \notin V'$. \diamond

Step 2:

To see that $B' \subseteq N_{u_l}$, assume first that this is not the case, i.e. $\exists \tilde{b} \in B' : \tilde{b} \notin N_{u_l}$. We will now show that this contradicts the distance-hereditary property. Note that \tilde{b} is not the center of the star graph $G[B]$, because either b_1 or b_2 is the center, since (b_1, b_2) is an edge in $G[B]$. Let's assume without loss of generality that b_1 is the center of $G[B]$. Furthermore, let's consider the cases where u_l and c' are adjacent or not separately.

¹⁰Step 4 is trivial.

- Assume that u_l and c' are not adjacent and consider the following induced subgraph

$$G[\{c', \tilde{b}, b_1, b_2, u_l\}] = \begin{array}{c} u_l \\ \tilde{b} \quad b_1 \quad b_2 \\ \cdot \\ c' \end{array} \quad (6.45)$$

The graph in eq. (6.45) is not distance-hereditary, since the distance between for example \tilde{b} and u_l increase if b_1 is removed. This is therefore a contradiction to the assumption that G is distance-hereditary.

- Assume that u_l and c' are adjacent. We then know that there exist a vertex h_l which is adjacent to u_l and c' , since $\mathcal{Q}(u_l, \{b_1, b_2\}, \{l\}, c')$ is true. First, let's show that h_l and \tilde{b} are not adjacent. In fact we will show that h_l is not adjacent to any vertex in B , which will be useful in step 5.

Assume the opposite, i.e. h_l is adjacent to some vertex $\hat{b} \in B' \setminus \{b_1, b_2\}$. We already know that h_l is not adjacent to b_1 or b_2 , since $\mathcal{Q}(u_l, \{b_1, b_2\}, \{l\}, c')$ is true. Consider therefore the following induced subgraph

$$G[\{c', \hat{b}, b_1, b_2, h_l\}] = \begin{array}{c} \hat{b} \quad b_1 \quad b_2 \\ h_l \quad \cdot \\ c' \end{array} \quad (6.46)$$

which is not distance-hereditary and we therefore know that $N_{h_l} \cap B' = \emptyset$. Consider now on the other hand the following induced subgraph, with the knowledge that h_l is not adjacent to \tilde{b}

$$G[\{c', \tilde{b}, b_1, u_l, h_l\}] = \begin{array}{c} u_l \\ \tilde{b} \quad b_1 \quad h_l \\ \cdot \\ c' \end{array} \quad (6.47)$$

which is also not distance-hereditary.

Since in all cases we arrived at a non-distance-hereditary graph we know that $B' \subseteq N_{u_l}$. ◊

Step 3:

We show that at least for one of the u_l , $L' \cap N_{u_l} = \emptyset$. We will do this by contradiction, assume therefore that $\forall l \in L' : (L' \cap N_{u_l} \neq \emptyset)$. Since $\mathcal{Q}(u_l, \{b_1, b_2\}, \{l\}, c')$ is true, we know that $\{l\} \cap N_{u_l} = \emptyset$ for all $l \in L'$. Consider now the graph $G[L' \cup \{u_l\}_l]$. From theorem 6.3.7 we know that there exist $l_1, l_2 \in L'$ and $u_{l_3}, u_{l_4} \in V \setminus V'$, such

that u_{l_3} is adjacent to l_1 but not to l_2 and u_{l_4} is adjacent to l_2 but not to l_1 .¹¹ But this is in contradiction with that the graph is distance-hereditary. To see this, consider the induced subgraph

$$G[\{c', b_1, l_1, l_2, u_{l_3}, u_{l_4}\}] = \tag{6.48}$$

which is not distance-hereditary, independently if the edges (u_{l_3}, u_{l_4}) , (c', u_{l_3}) and (c', u_{l_4}) are individually present or not. Since this contradicts the distance-hereditary property we know that $\exists l \in L' : (L' \cap N_{u_l} = \emptyset)$. \diamond

Step 5:

Let's assume that u_l is then a vertex such that $B \subseteq N_{u_l}$ and $L' \cap N_{u_l} = \emptyset$. If u_l is not adjacent to c' , then clearly $\mathcal{Q}(u_l, B', L', c')$. On the other hand if u_l and c' are adjacent we know that there exist a h_l in $N_{u_l} \cap N_{c'} \setminus \cup_{x \in \{l, b_1, b_2\}} N_x$. We thus need to show that h_l is not adjacent to any vertex in V' , other than c' . Firstly, h_l cannot be adjacent to a vertex in L' , since this would violate the distance-hereditary property. To see this, assume that h_l is adjacent to $\tilde{l} \in L'$ and consider the following induced subgraph

$$G[\{c', b_1, \tilde{l}, u_l, h_l\}] = \tag{6.49}$$

which is not distance-hereditary. This is a contradiction with the distance-hereditary property and we therefore know that $N_{h_l} \cap L' = \emptyset$. As we already shown in step 2, h_l is also not adjacent to any vertex in B' . Thus, h_l is not adjacent to any vertex in $V' = B' \cup L' \cup \{c'\}$. \diamond

We have therefore shown that eq. (6.39) is true which implies that eq. (6.38) is true. Finally, as we described in the beginning of the proof, this implies that if $\mathcal{P}(B', L', c')$ is false then $S_{V'} \not\prec G$. \square

Theorem 6.3.7. Assume G is a graph on the vertices $U \cup L$ such that $U \cap L = \emptyset$ and $U \neq \emptyset$. Furthermore, assume that for each l in L , there is at least one vertex in U not adjacent to l and for each u in U , there is at least one vertex in L adjacent to u , i.e. G satisfies the following expression

$$\mathcal{R}(U, L) = \forall l \in L : (\exists u \in U : u \notin N_l) \wedge \forall u \in U : (\exists l \in L : l \in N_u) \tag{6.50}$$

¹¹Note that for example l_1 and l_3 could be the same vertex, but not necessarily.

Then there exist two vertices u_1 and u_2 in U and two vertices l_1 and l_2 in L such that u_1 is adjacent to l_1 but not to l_2 and u_2 is adjacent to l_2 but not to l_1 . In other words the induced subgraph is of the following form

$$G[\{u_1, u_2, l_1, l_2\}] = \begin{array}{ccc} & l_1 & \text{---} & l_2 & \\ & \bullet & & \bullet & \\ & | & & | & \\ u_1 & \bullet & \text{---} & \bullet & u_2 \end{array} . \quad (6.51)$$

where the dashed edges are individually either present or not. \diamond

Proof. We will first show that $|L| \geq 2$ and $|U| \geq 2$. Pick an element $u_1 \in U$, which exists since $U \neq \emptyset$, by assumption there is a $l_1 \in L$ which is adjacent to u_1 . Furthermore there exist a $u_2 \in U$ which is not adjacent to l_1 , thus $u_1 \neq u_2$. Finally, by assumption there is a $l_2 \in L$ which is adjacent to u_2 , thus $l_1 \neq l_2$. Note, that this does not yet prove the theorem, since u_1 and l_2 might be adjacent.

We will first prove the theorem for $|L| = 2$ and then use this to prove the general case.

$|L| = 2$:

Let's denote the vertices in L by l_1 and l_2 . We first show by contradiction that both vertices in L must have at least one neighbor in U . Assume that l_1 does not have a neighbor in U , then all vertices in U must be adjacent to l_2 by the second part of eq. (6.50), but then the first part of eq. (6.50) is false. Thus l_1 has at least one neighbor in U and by symmetry the same is true for l_2 . Now choose such a neighbor of l_1 in U and denote this u_1 . We now show by contradiction that there exist another vertex $u_2 \in U$ which is adjacent to l_2 but not to l_1 . Assume that this is not the case, i.e. all vertices in $U \setminus \{u_1\}$ are adjacent to l_1 or not adjacent to l_2 . If a vertex in U is not adjacent to l_2 then it is necessarily adjacent to l_1 , thus by assumption all vertices in $U \setminus \{u_1\}$ are adjacent to l_1 . This is in contradiction with the first part of eq. (6.50) and the theorem for $|L| = 2$ follows.

$|L| > 2$:

We will show that the following is true: (1) G has an induced subgraph as in eq. (6.51) or (2) there exist a $l \in L$ such that $G \setminus l$ satisfy $\mathcal{R}(U, L \setminus \{l\})$. The theorem then follows since if (1) is true the theorem follows directly and if (2) is true we can make the same argument for $G \setminus l$ for some $l \in L$ and then possibly for $(G \setminus l) \setminus l'$ etc., which at some point will give the case $|L| = 2$, which we have proven above. Note that if the graph reached by deleting vertices from G , has the graph in eq. (6.51) as an induced subgraph, then so does G .

To prove that (1) or (2) is true, we show that if (2) is false then (1) is necessarily true. Therefore, assume now that (2) is false, which means that for every choice of l , $G \setminus l$ does not satisfy $\mathcal{R}(U, L \setminus \{l\})$. The only possibility for this to happen, i.e. the deletion of l makes the graph not satisfy eq. (6.50), is if the deletion of l makes some $u \in U$ not adjacent to any vertex in L . It is easy to see that this can only happen if $\exists u \in U : (L \cap N_u = \{l\})$. Since this should be true for all $l \in L$, we have that if (2) is false, the following is true,

$$\forall l \in L : (\exists u \in U : (L \cap N_u = \{l\})). \quad (6.52)$$

But eq. (6.52) implies that (1) is true. To see this pick two different vertices l_1 and l_2 in L . From eq. (6.52) we know that there exist a vertex $u_1 \in U$ such that $L \cap N_{u_1} = \{l_1\}$ and similarly a u_2 for l_2 . Note that $u_1 \neq u_2$ since $N_{u_1} \neq N_{u_2}$. Furthermore, since $L \cap N_{u_1} = \{l_1\}$ and $L \cap N_{u_2} = \{l_2\}$ the induced subgraph $G[\{u_1, u_2, l_1, l_2\}]$ is as in eq. (6.51). \square

Proof for a complete-star graph Here we prove that if $\mathcal{P}(B', L', c')$ is false, then $S_{V'} \not\prec G$ for the case where $G[V']$ is a complete-star graph. We have the following theorem.

Theorem 6.3.8. *Let's assume that G is a distance-hereditary graph and V' is a subset $V' \subseteq V(G)$ such that the induced subgraph $G[V']$ is a complete-star graph $KS_{(B', L', c')}$. Furthermore assume that $\mathcal{P}(B', L', c')$ is false, then $S_{V'} \not\prec G$. \diamond*

Proof. We will prove this by induction on the size of B' . The base-case, $|B'| = 2$, is true due to theorem 6.3.6, since for $|B'| = 2$, the graph G is also a star-star graph. Let's now assume that theorem 6.3.8 is true for $|B'| = k$. We will now prove that the theorem is true for $|B'| = k + 1$ by showing that

$$\neg \mathcal{P}(B', L', c') \Rightarrow \exists b \in B' : \neg \mathcal{P}(B' \setminus \{b\}, L', c'). \quad (6.53)$$

where $|B'| = k + 1 \geq 3$. Then from the induction hypothesis we know that $S_{V' \setminus \{b\}} \not\prec G$ for some $b \in B'$ and the corollary follows, because if $S_{V' \setminus \{b\}}$ is not a vertex-minor of G then neither is $S_{V'}$, since $S_{V' \setminus \{b\}} < S_{V'}$. To show that eq. (6.53) is true, we instead show the contrapositive statement, i.e.

$$\forall b \in B' : \mathcal{P}(B' \setminus \{b\}, L', c') \Rightarrow \mathcal{P}(B', L', c'). \quad (6.54)$$

Let's therefore assume that $\mathcal{P}(B' \setminus \{b\}, L', c')$ is true for all b in B' . Let $Q(u, B', L', c')$ be the expression on $\mathcal{P}(B', L', c')$ such that

$$\mathcal{P}(B', L', c') = \exists u \in V \setminus V' : Q(u, B', L', c') \quad (6.55)$$

For each $b \in B'$, let u_b be a vertex in $V \setminus (V' \setminus \{b\})$ such that $Q(u_b, B' \setminus \{b\}, L', c')$ is true. We now need to show that for at least one of these u_b , $u_b \in V \setminus V'$ and $Q(u_b, B', L', c')$ is true. Note that $L' \cap N_{u_b} = \emptyset$ for all b , since $Q(u_b, B' \setminus \{b\}, L', c')$ for all b . Thus, to show that for at least on u_b , $u_b \in V \setminus V'$ and $Q(u_b, B', L', c')$ is true we will go through the following steps:

1. Show that there can maximally be one $\tilde{b} \in B'$ such that $u_{\tilde{b}} \in B'$, i.e.

$$\exists \tilde{b} \in B' : \left(\forall b \in B' : (b = \tilde{b} \vee u_b \notin B') \right) \quad (6.56)$$

2. Fix $\tilde{b} \in B'$ to be such that the corresponding expression in eq. (6.56) is true.¹²

¹²Note that this does not imply that $u_{\tilde{b}} \in B'$.

3. Show that there can maximally be one $\hat{b} \in B' \setminus \{\tilde{b}\}$ such that $B' \not\subseteq N_{u_{\hat{b}}}$, i.e.

$$\exists \hat{b} \in B' \setminus \{\tilde{b}\} : (\forall b \in B' \setminus \{\tilde{b}\} : (b = \hat{b} \vee B' \subseteq N_{u_b})) \quad (6.57)$$

4. Fix $\hat{b} \in B'$ to be such that the corresponding expression in eq. (6.57) is true.

5. Use step 1 to 4 to show that there exist a $b \in B'$ such that $u_b \notin B'$, $B' \subseteq N_{u_b}$ and

$$(u_b, c') \notin E(G) \vee \exists h_b : \left(h_b \in N_{u_b} \cap N_{c'} \setminus \bigcup_{x \in V' \setminus \{c'\}} N_x \right) \quad (6.58)$$

i.e. $Q(u_b, B', L', c')$ is true.

Let us now consider the steps 1 through 5 one by one.¹³ We will in these steps often claim that certain small graphs are not distance-hereditary. Verifying this can be done by hand or using our code supplied at [4].

Step 1:

Here we show that eq. (6.56) is true. Firstly, if for all $b \in B'$ we have that $u_b \notin B'$, then eq. (6.56) is clearly true, since \tilde{b} can then be chosen as any element in B' .¹⁴ We now show by contradiction that there cannot exist two different vertices $\tilde{b}_1, \tilde{b}_2 \in B'$, such that $u_{\tilde{b}_1} \in B'$ and $u_{\tilde{b}_2} \in B'$. Thus, let's assume that such vertices \tilde{b}_i for $i \in \{1, 2\}$, does exist. Note that, since $Q(u_{\tilde{b}_i}, B' \setminus \{\tilde{b}_i\}, L', c')$ is true, we know that $u_{\tilde{b}_i}$ is adjacent to all vertices in $B' \setminus \{\tilde{b}_i\}$ and therefore $u_{\tilde{b}_i} = \tilde{b}_i$. Since $u_{\tilde{b}_i} = \tilde{b}_i$, we know that $u_{\tilde{b}_1} \neq u_{\tilde{b}_2}$. Furthermore, from the fact that $u_{\tilde{b}_i} \in B'$, we know that $u_{\tilde{b}_i}$ is adjacent to c' and thus, since $Q(u_{\tilde{b}_i}, B' \setminus \{\tilde{b}_i\}, L', c')$ is true, there exist a vertex h_i such that

$$h_i \in N_{u_{\tilde{b}_i}} \cap N_{c'} \setminus \bigcup_{x \in V' \setminus \{c', \tilde{b}_i\}} N_x. \quad (6.59)$$

The vertices h_i are necessarily different, since h_1 is adjacent to $b_1 = u_{\tilde{b}_1}$ but not to b_2 and vice versa for h_2 . Now consider the following induced subgraph

$$G[\{c', \tilde{b}_1, \tilde{b}_2, b, h_1, h_2\}] = \text{Diagram} \quad (6.60)$$

where b is a vertex in $B' \setminus \{b_1, b_2\}$, which exists since $|B'| \geq 3$ by assumption. The graph in eq. (6.60) is not distance-hereditary, independently if the edge (h_1, h_2) is present or not. Since this contradicts the fact that G is distance-hereditary, we know that eq. (6.56) is true. \diamond

¹³Step 2 and 4 are trivial.

¹⁴Remember that $|B'| \geq 3$.

Step 3:

Here we show that eq. (6.57) is true. Firstly, if for all $b \in B' \setminus \{\tilde{b}\}$ we have that $B' \subseteq N_{u_b}$, then eq. (6.57) is clearly true, since \hat{b} can then be chosen as any element in $B' \setminus \{\tilde{b}\}$.¹⁵ We now show by contradiction that there cannot exist two different vertices $\hat{b}_1, \hat{b}_2 \in B'$, such that $B' \not\subseteq N_{u_{\hat{b}_1}}$ and $B' \not\subseteq N_{u_{\hat{b}_2}}$. Thus, let's assume that such vertices \hat{b}_i for $i \in \{1, 2\}$, does exist. Let's for the remainder of this step denote $u_{\hat{b}_i}$ as $u^{(i)}$. From the previous steps we know that $u^{(i)} \notin B'$ and furthermore $(B' \setminus \{\hat{b}_i\}) \subseteq N_{u^{(i)}}$ since $Q(u^{(i)}, B' \setminus \{\hat{b}_i\}, L', c')$ is true. Thus, by assumption, we have that $\hat{b}_i \notin N_{u^{(i)}}$. The vertices $u^{(i)}$ are then necessarily different, i.e. $u^{(1)} \neq u^{(2)}$, since for example \hat{b}_1 is a neighbor of $u^{(2)}$ but not of $u^{(1)}$. We will now show that this contradicts the fact that G is distance-hereditary, by considering the following cases:

- Assume that $u^{(1)}$ is not adjacent to $u^{(2)}$ and consider the following induced subgraph

$$G[\{b, \hat{b}_1, \hat{b}_2, u^{(1)}, u^{(2)}\}] = \begin{array}{c} u^{(2)} \quad u^{(1)} \\ \diagdown \quad \diagup \\ b \\ \diagup \quad \diagdown \\ \hat{b}_1 \quad \hat{b}_2 \end{array} \quad (6.61)$$

where b is a vertex in $B' \setminus \{\hat{b}_1, \hat{b}_2\}$, which exists since $|B'| \geq 3$ by assumption. The graph in eq. (6.61) is not distance-hereditary.

- Assume that $u^{(1)}$ is adjacent to $u^{(2)}$.
 - Assume that neither $u^{(1)}$ or $u^{(2)}$ is adjacent to c' and consider the following induced subgraph

$$G[\{c', \hat{b}_1, \hat{b}_2, u^{(1)}, u^{(2)}\}] = \begin{array}{c} u^{(2)} \quad u^{(1)} \\ \text{---} \quad \text{---} \\ \diagdown \quad \diagup \\ \hat{b}_1 \quad \hat{b}_2 \\ \diagdown \quad \diagup \\ c' \end{array} \quad (6.62)$$

which is not distance-hereditary.

- Assume that exactly one of $u^{(1)}$ and $u^{(2)}$ is adjacent to c' and let's assume without loss of generality that it is $u^{(1)}$ that is adjacent to c' . Since $Q(u^{(1)}, B' \setminus \{\hat{b}_1\}, L', c')$ is true and $u^{(1)}$ is adjacent to c' , we know that there exist a vertex h_1 such that

$$h_1 \in N_{u^{(1)}} \cap N_{c'} \setminus \bigcup_{x \in V' \setminus \{c', \hat{b}_1\}} N_x. \quad (6.63)$$

¹⁵Remember that $|B'| \geq 3$.

Consider now the following induced subgraph

$$G[\{c', b, \hat{b}_2, u^{(1)}, u^{(2)}, h_1\}] = \text{Diagram} \quad (6.64)$$

where b is a vertex in $B' \setminus \{\hat{b}_1, \hat{b}_2\}$, which exists since $|B'| \geq 3$ by assumption. The graph in eq. (6.64) is not distance-hereditary, independently if the edge $(h_1, u^{(2)})$ is present or not.

- Assume that both $u^{(1)}$ and $u^{(2)}$ is adjacent to c' . Since $Q(u^{(i)}, B' \setminus \{\hat{b}_i\}, L', c')$ is true and $u^{(i)}$ is adjacent to c' , we know that there exist a vertex h_i such that

$$h_i \in N_{u^{(i)}} \cap N_{c'} \setminus \bigcup_{x \in V' \setminus \{c', b_i\}} N_x. \quad (6.65)$$

- ◊ Assume that $h_1 = h_2$, which implies that h_1 is not adjacent to \hat{b}_1 or \hat{b}_2 and consider the following induced subgraph

$$G[\{\hat{b}_1, \hat{b}_2, u^{(1)}, u^{(2)}, h_1\}] = \text{Diagram} \quad (6.66)$$

which is not distance-hereditary.

- ◊ Assume that $h_1 \neq h_2$ and consider the following induced subgraph

$$G[\{c', b, \hat{b}_1, \hat{b}_2, u^{(1)}, u^{(2)}, h_1, h_2\}] = \text{Diagram} \quad (6.67)$$

where b is a vertex in $B' \setminus \{\hat{b}_1, \hat{b}_2\}$, which exists since $|B'| \geq 3$ by assumption. The graph in eq. (6.67) is not distance-hereditary, independently if the edges

$$(h_1, h_2), (h_1, u^{(2)}), (h_2, \hat{b}_2), (h_1, u^{(1)}), (h_1, \hat{b}_2) \quad (6.68)$$

are individually present or not. To make this statement more transparent, we also provide the adjacency matrix of the graph in eq. (6.67). The graph in eq. (6.67) has adjacency matrix

$$\Gamma = \begin{matrix} & c' & b & \hat{b}_1 & \hat{b}_2 & u^{(1)} & u^{(2)} & h_1 & h_2 \\ \begin{matrix} c' \\ b \\ \hat{b}_1 \\ \hat{b}_2 \\ u^{(1)} \\ u^{(2)} \\ h_1 \\ h_2 \end{matrix} & \begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 1 & x_1 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & x_2 \\ 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & x_3 \\ 1 & 1 & 1 & 0 & 1 & 0 & x_4 & 0 & 1 \\ 1 & 0 & x_1 & 0 & 1 & x_4 & 0 & 0 & x_5 \\ 1 & 0 & 0 & x_2 & x_3 & 1 & x_5 & 0 & 0 \end{pmatrix} \end{matrix} \quad (6.69)$$

where $x_1, \dots, x_5 \in \{0, 1\}$. By explicit computation one can check that for any assignment of the variables x_1, \dots, x_5 , the graph with adjacency matrix as in eq. (6.69) is not distance-hereditary.

Since in all cases we arrived at a contradiction of the fact that G is distance-hereditary, we know that eq. (6.57) is true. \diamond

Step 5:

Here we show that there exist a $b \in B'$ such that $u_b \notin B'$, $B' \subseteq N_{u_b}$ and

$$(u_b, c') \notin E(G) \vee \exists h_b : \left(h_b \in N_{u_b} \cap N_{c'} \setminus \bigcup_{x \in V' \setminus \{c'\}} N_x \right). \quad (6.70)$$

Note that $B' \subseteq N_{u_b}$ implies $u_b \notin B'$, thus we can focus on the first property. We prove the statement by contradiction and assume therefore that there exist no such b , i.e. there exist no $b \in B'$ such that $B' \subseteq N_{u_b}$ and for which eq. (6.70) is true. Let's first introduce the set \mathcal{B} of vertices in B' which satisfy the first of these properties, i.e.

$$\mathcal{B} = \{b \in B' : (B' \subseteq N_{u_b})\}. \quad (6.71)$$

From the previous steps we know that \mathcal{B} is not empty. Furthermore, from our assumption we must have that for all $b \in \mathcal{B}$, eq. (6.70) is false, i.e

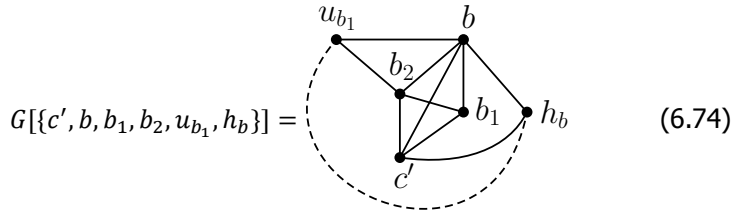
$$\forall b \in \mathcal{B} : \left((u_b, c') \in E(G) \wedge \forall h_b : \left(h_b \notin N_{u_b} \cap N_{c'} \setminus \bigcup_{x \in V' \setminus \{c'\}} N_x \right) \right). \quad (6.72)$$

Let b now be a fixed element of \mathcal{B} . Since u_b is adjacent to c' and from the fact that $\mathcal{Q}(u_b, B' \setminus \{b\}, L', c')$ is true, we know that there exist a h_b such that

$$h_b \in N_{u_b} \cap N_{c'} \setminus \bigcup_{x \in V' \setminus \{c', b\}} N_x. \quad (6.73)$$

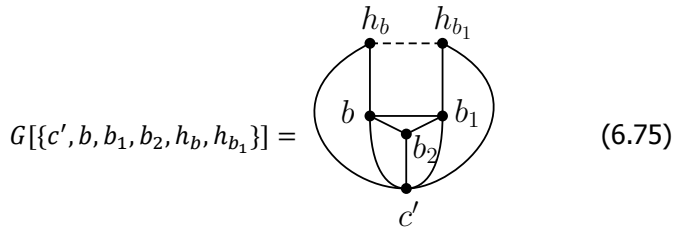
Note that, eq. (6.72) together with eq. (6.73) implies that h_b is adjacent to b but to no other vertex in B' . We will now show that this leads to a contradiction by considering a vertex $b_1 \in B' \setminus \{b\}$, such that $u_{b_1} \notin B'$, which we showed exists in step 1. Furthermore, let b_2 a vertex in $B' \setminus \{b, b_1\}$, which is necessarily adjacent to u_{b_1} , since $\mathcal{Q}(u_{b_1}, B' \setminus \{b_1\}, L', c')$ is true. Let's consider the following cases:

- Assume that u_{b_1} is not adjacent to c' . By assumption we then have that $B' \not\subseteq N_{u_{b_1}}$, which implies that u_{b_1} is not adjacent to b_1 . Consider now the following induced subgraph



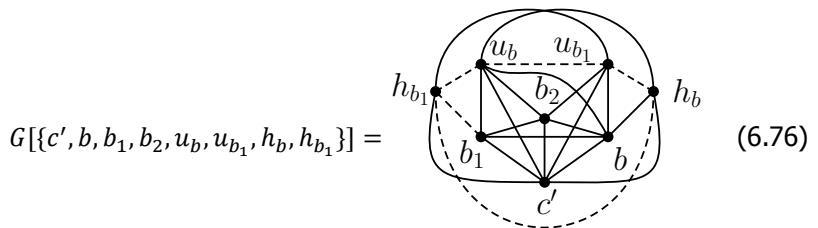
which is not distance-hereditary, independently if the edge (h_b, u_{b_1}) is present or not.

- Assume that u_{b_1} is adjacent to c' .
 - Assume that $B' \subseteq N_{u_{b_1}}$. By the same argument as for b and u_b , we know that there exist a vertex h_{b_1} which is adjacent to u_{b_1} , c' and b_1 but not to any other vertex in B' . Consider now the following induced subgraph



which is not distance-hereditary, independently if the edge (h_b, h_{b_1}) is present or not.

- Assume that $B \not\subseteq N_{u_{b_1}}$, which implies that b_1 is not adjacent to u_{b_1} since $\mathcal{Q}(u_{b_1}, B' \setminus \{b_1\}, L', c')$ is true. Furthermore, we know that there is a vertex h_{b_1} which is adjacent to u_{b_1} and c' and possibly to b_1 but no other vertex in B' . Consider now the following induced subgraph



which is not distance-hereditary, independently if the edges

$$(h_b, h_{b_1}), (h_b, u_{b_1}), (h_{b_1}, b_1), (h_{b_1}, u_b), (u_b, u_{b_1}) \quad (6.77)$$

are individually present or not. To make this statement more transparent, we also provide the adjacency matrix of the graph in eq. (6.76). The graph in eq. (6.76) has adjacency matrix

$$\Gamma = \begin{matrix} & c' & b & b_1 & b_2 & u_b & u_{b_1} & h_b & h_{b_1} \\ \begin{matrix} c' \\ b \\ b_1 \\ b_2 \\ u_b \\ u_{b_1} \\ h_b \\ h_{b_1} \end{matrix} & \begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 0 & x_1 \\ 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & x_2 & 1 & x_3 \\ 1 & 1 & 0 & 1 & x_2 & 0 & x_4 & 1 \\ 1 & 1 & 0 & 0 & 1 & x_4 & 0 & x_5 \\ 1 & 0 & x_1 & 0 & x_3 & 1 & x_5 & 0 \end{pmatrix} \end{matrix} \quad (6.78)$$

where $x_1, \dots, x_5 \in \mathbb{F}_2$. By explicit computation one can check that for any assignment of the variables x_1, \dots, x_5 , the graph with adjacency matrix as in eq. (6.78) is not distance-hereditary.

Since in all cases we arrived at a contradiction of the fact that G is distance-hereditary, we know that there exist a $b \in B'$ such that $u_b \notin B'$, $B' \subseteq N_{u_b}$ and such that eq. (6.70) is true. \diamond

We have therefore shown that eq. (6.54) is true which implies that eq. (6.53) is true. Finally, as we described in the beginning of the proof, this implies, by induction, that if $\mathcal{P}(B', L', c')$ is false then $S_{V'} \not\prec G$. \square

6.3.2. Fixed-parameter tractable algorithm for unbounded rank-width

In this section we will show that the star vertex-minor problem (StarVertexMinor) is fixed-parameter tractable for circle graphs, in terms of the size of the considered star graph. More specifically, we will show that there exists an efficient algorithm to decide if $S_{V'}$ is a vertex-minor of G , given that G is a circle graph and the subset $V' \subseteq V(G)$ is of size k . We will do this by showing that we can map this problem in polynomial time to deciding whether the 4-regular multi-graph that defines G has a SOET on the vertices V' . This is done in section 6.3.2. We will then give an algorithm that decides whether a 4-regular multi-graph has a SOET on a subset V' of its vertices where $V' = k$ is fixed. This is done in section 6.3.2. We begin by formally stating the decisions problems considered.

We first define the problem $k\text{-StarVertexMinor}$.

Problem 6.3.9 ($k\text{-StarVertexMinor}$). Let G be a graph and let V' be a subset of $V(G)$ with $|V'| = k$. Decide whether $S_{V'}$ is a vertex-minor of G . \diamond

We also define the problem k -SOET.

Problem 6.3.10 (k -SOET). Let F be a 4-regular multi-graph and let V' be a subset of $V(F)$ with $|V'| = k$. Decide whether F allows for a SOET with respect to V' \diamond

Theorem 6.3.11. k -StarVertexMinor, restricted to circle graphs, is in \mathbb{P} . \diamond

Proof. This will follow from theorem 6.3.12 which provides an efficient mapping of every circle graph instance of k -StarVertexMinor to a corresponding instance of k -SOET. By corollary 6.3.16.1 k -SOET is in \mathbb{P} and hence so is k -StarVertexMinor. An efficient algorithm for k -SOET is given in algorithm 6.4. \square

This theorem has the following corollary.

Corollary 6.3.11.1. StarVertexMinor is fixed-parameter tractable in the size of the input vertex-set V' if the input graph G is a circle graph. \diamond

The existence of this fixed-parameter tractable algorithm is theoretically interesting but it is not likely to be of practical use. This is so because while the algorithm is efficient in the size of the input graph G it suffers from a hidden constant that is of size $O(k! \cdot (f(k)^{O(kf(k))}))$, where $f(k) = 2^{2^{O(k^2)}}$ making its practical implementation unlikely.

6

Mapping k -StarVertexMinor to k -SOET

In this section we will prove that there exists an efficient mapping from instances of k -StarVertexMinor that are circle graphs to k -SOET. This is formalized in the following theorem, the proof of which also provides a prescription of the algorithm that defines the mapping.

Theorem 6.3.12. Let (G, V') be an instance of k -StarVertexMinor and let G be a circle graph. There is an efficient mapping from this instance to an instance of k -SOET and moreover the instance (G, V') is a yes-instance of k -StarVertexMinor if and only if its image under the mapping is a yes-instance of k -SOET. \diamond

Proof. We will prove this by providing an explicit mapping. An instance (G, V') of k -StarVertexMinor, where G is a circle graph and V' a vertex set, can be mapped to an instance of k -SOET by the following two steps:

- Find a double occurrence word \mathbf{X} with letters in $V(G)$ such that $G = \mathcal{A}(\mathbf{X})$. This can be done in time $O(|V(G)|^2)$ by using Spinrad's algorithm [14].
- Construct a 4-regular multi-graph F , such that $\mathbf{X} = m(U)$ for some Eulerian tour U on F . As shown in [15], this can be done in the following way:
 - Let $C_{\mathbf{X}}$ be a cycle graph with the vertices labeled as the consecutive letters of \mathbf{X} .

- Contract every pair of vertices with the same label, while keeping all the edges. Note that this step can create multi-edges or self-loops. This step can be done in time $\mathcal{O}(|V(G)|)$ by adding the corresponding rows of the adjacency matrix and deleting one row and one column.

The graph obtained from these steps is then a 4-regular multi-graph F with a Eulerian tour U , such that $\mathbf{X} = m(U)$. Therefore, constructing the 4-regular multi-graph F , given \mathbf{X} , can also be done in time $\mathcal{O}(|V(G)|^2)$.

One can see that the above algorithm runs in $\mathcal{O}(|V(G)|^2)$ and given a circle graph G outputs a 4-regular graph F that has a Eulerian tour U such that $\mathcal{A}(U) = G$. From corollary 4.7.16.1 we then know that $S_{V'} < G$ if and only if F allows for a SOET with respect to V' . Using the above two steps we see that any circle graph instance of k -StarVertexMinor can be mapped to k -SOET in time $\mathcal{O}(|V(G)|^2)$. \square

Given this mapping, the next logical step is to find an efficient algorithm for k -SOET. This is done in the next section.

k -SOET is in \mathbb{P}

In this section we will prove that k -SOET is in \mathbb{P} . We will do this by explicitly writing down an efficient algorithm. This algorithm will make use of an algorithm which solves a well known graph problem we call k' -DPP. k' -DPP, for k' -Disjoint Path Problem is formally defined as follows.

Problem 6.3.13 (k' -DPP). Let H be a multi-graph and let $L = \{(v_1, v'_1), \dots, (v_{k'}, v'_{k'})\}$ be a set of two-tuples of vertices of H such that $|L| = k'$. Decide whether there exist k' edge-disjoint paths P_i on H that start at v_i and end at v'_i for $i \in [k']$. \diamond

Robertson and Seymour proved that there exist an efficient algorithm for solving k' -DPP, if k' is fixed. Their proof of correctness is based on 23 papers named Graph minors. I, ..., Graph minors. XXIII [16]. In [17] an improved algorithm for k' -DPP is given, with a much smaller hidden constant describing the scaling in terms of k' . This hidden constant is still quite large as running time for the algorithm in [17] is $(f(k')^{\mathcal{O}(k'f(k'))})n^{\mathcal{O}(1)}$, where n is the number of vertices in the graph and $f(k') = 2^{2^{\mathcal{O}(k'^2)}}$

We will discuss an algorithm that solves k -SOET efficiently. It will use an algorithm for k' -DPP as a subroutine, calling it a constant number of times.

The algorithm for k -SOET is sketched as follows. Let F be a 4-regular multi-graph and let $V' \subset V(F)$ be of size k . Recall that a SOET is a Eulerian path that visits the vertices in V' in some order. A necessary condition for a SOET to exist is that there are, for some ordering of the vertices in V' two edge-disjoint trails from the first vertex in V' to the second vertex in V' and from the second to the third and so on. If one is given such a collection of paths it is not hard to see that one can connect these paths to each other to form a tour and moreover extend this total

tour to be Eulerian. Hence we could use the k' -DPP algorithm described above to find such a tour by finding all the edge disjoint trails that connect the vertices in V' .

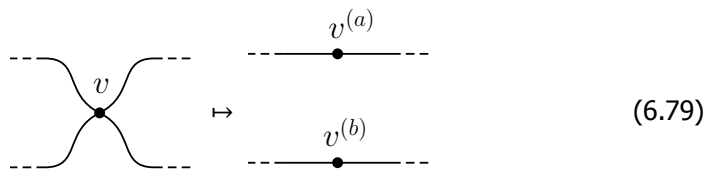
There are two problems with this. The first problem is that a SOET with respect to V' can exist with respect to any possible ordering of the set V' . This means that in order to find this tour we might have to apply k -DPP to all $k!$ different orderings. This is a large overhead but acceptable since we are only looking for an algorithm that is efficient for fixed k . The second problem is that the k' -DPP algorithm expects k' pairs of vertices and requires these pairs to be different.

This means we cannot input the same vertex-pair twice to get two edge disjoint paths. We can resolve this at some overhead by running the k' -DPP algorithm on a modified multi-graph H . This multi-graph, which we call a SOET splitting of F , is created by taking each vertex v_i in V and splitting it into two vertices $v_i^{(a)}, v_i^{(b)}$ such that the four edges e_1, \dots, e_4 originally incident on v_i are now pairwise incident on $v_i^{(a)}, v_i^{(b)}$.

As an example we could have for instance that e_1, e_2 are incident on $v_i^{(a)}$ and e_3, e_4 are incident on $v_i^{(b)}$. We must consider all possible choices of pairings here (of which there are 6) and since there are k vertices on which to perform this procedure (each vertex in V') there are 6^k such SOET splittings of F (which are not 4-regular anymore). This means we must call the $2k$ -DPP subroutine $6^k \cdot k!$ times to account for all possible orderings of the SOET.

Let's make this a little more rigorous. We begin by defining the notion of a SOET-splitting of a 4-regular multi-graph F with respect to a set V' .

Definition 6.3.14 (SOET-splitting). Let F be a 4-regular multi-graph and let V' be a subset of it's vertices. A SOET-splitting H of F with respect to V is a multi-graph created from F by performing the following operation on all vertices in V



We label the two vertices that originate from a vertex $v \in V'$ as $v^{(a)}$ and $v^{(b)}$. Note that we have not specified how to connect the edges that were originally incident on v to $v^{(a)}$ and $v^{(b)}$. There are six possible ways to do this for each $v \in V'$ and each choice leads to an a priori distinct SOET-splitting multi-graph H . \diamond

We also define the subroutine k' -DPP. Note that we have only specified the inputs and outputs of this subroutine. For details on the inner workings of this algorithm see [17]. Using this subroutine we can write down an algorithm for k -SOET.

We now prove that this algorithm returns TRUE if and only if the multi-graph F allows for a SOET with respect to the vertex-subset V' . We begin by proving that if the algorithm returns TRUE the multi-graph F allows for a SOET with respect to the vertex-subset V' . We have the following theorem.

Theorem 6.3.15. *Let F be a connected 4-regular multi-graph and let $V' \subset V(F)$ be a subset of its vertices with $|V'| = k$. If algorithm 6.4 returns TRUE then F allows for a SOET with respect to the vertex set V' .* \diamond

Proof. Let F be a connected 4-regular multi-graph and let $V' \subset V(F)$ be a subset of its vertices with $|V'| = k$. If algorithm 6.4 returns true this means there exists a SOET-splitting H of F and a permutation π of the set $[k]$ such that there exist $2k$ edge-disjoint paths in H that connect the vertices $v_{\pi(1)}^{(a)}$ and $v_{\pi(2)}^{(a)}$, $v_{\pi(2)}^{(a)}$ and $v_{\pi(3)}^{(a)}$ and so forth. Undoing the SOET-splitting operation that defines F we can see that these edge-disjoint paths can be attached to one another to form a closed trail¹⁶ (a tour) U on F that visits all vertices of V' twice in the order $v_{\pi(1)}, \dots, v_{\pi(k)}$. This is not yet a Eulerian tour however. To construct a Eulerian tour out of the tour U we consider the multi-graph \hat{H} obtained from F by deleting all vertices in V (looking at the induced subgraph $F[V \setminus V']$) and subsequently removing all edges in the tour U from the remaining multi-graph. The resulting multi-graph will consist of multiple connected components. These connected components will either consist of a single vertex or will be multi-graphs with vertices of degree two and four. This means all these connected components are Eulerian. Moreover each of these connected components will contain a vertex that is also a vertex in the tour U . Consider on each of these connected components then a Eulerian tour. These tours will thus form tours on the original multi-graph F as well and since all such tours have at least one vertex in common with the tour U we can extend U to a Eulerian tour on the multi-graph F by cutting U at such a vertex for each connected component and inserting the tour originating from the connected components of G . This means that U can be turned into a Eulerian tour and hence there exists a SOET on the multi-graph F . This proves the theorem. \square

Next we prove the converse statement.

Theorem 6.3.16. *Let F be a connected 4-regular multi-graph and let $V' \subset V(F)$ be a subset of its vertices with $|V'| = k$. If F allows for a SOET with respect to V' then algorithm 6.4 will return TRUE.* \diamond

Proof. Let U be a SOET on F with respect to V' and without loss of generality assume that U traverses the vertices of V' in the order v_1, v_2, \dots, v_k . Hence for the vertices v_1, v_2 there are 2 sub-trails $U_1^{(a)}, U_1^{(b)}$ of U that start at v_1 and end at v_2 . The sub-trail $U_1^{(a)}$ (or $U_1^{(b)}$) might not be a path, but one can easily pick a subset of the edges in $U_1^{(a)}$ which gives a path, for example as the shortest path $P_1^{(a)}$ between v_1 and v_2 in the subgraph of F given by the vertices and edges of $U_1^{(a)}$. Similarly for $U_1^{(b)}$ and $P_1^{(b)}$. We can make the same argument for the vertices

¹⁶Note that a path is also a trail.

v_2, v_3 and so forth. By the definition of SOET splittings there must thus exist a SOET splitting G of F with respect to V' such that the path $P_1^{(a)}$ starts at $v_1^{(a)}$ and ends at $v_2^{(a)}$ and such that the path $P_1^{(b)}$ starts at $v_1^{(b)}$ and ends at $v_2^{(b)}$. We can make the same argument for the vertices v_2, v_3 and so forth. Hence there must exist a SOET-splitting G of F with respect to V' such that the algorithm $2k\text{-DPP}(G, S)$ with $S = \{(v_1^{(a)}, v_2^{(a)}), \dots, (v_k^{(a)}, v_1^{(a)}), (v_1^{(b)}, v_2^{(b)}), \dots, (v_k^{(b)}, v_1^{(b)})\}$ returns TRUE. Since algorithm 6.4 runs over all possible orderings of V' and all possible SOET splittings this particular call to $2k\text{-DPP}$ will always happen and hence algorithm 6.4 will return TRUE. This proves the theorem. \square

This leads to the following corollary.

Corollary 6.3.16.1. $k\text{-SOET}$ is in \mathbb{P} . \diamond

Proof. By theorem 6.3.16 and theorem 6.3.15, algorithm 6.4 returns TRUE if and only if the tuple (F, V') , with F a 4-regular multi-graph and $V' \subset V(F)$ a subset of its vertices with $|V'| = k$, is a YES instance of $k\text{-SOET}$. Moreover the function $k'\text{-DPP}$ runs in polynomial time in the size of the input graph and we have for all SOET-splittings G of F with respect to V' that $|V(G)| = |V(F)| + k$ and $|E(G)| = |E(F)|$. Algorithm 6.4 hence performs a constant number of function calls (constant in the size of F , not in k) of $k'\text{-DPP}$ with an input multi-graph of size $O(|V(F)|, |E(F)|)$. Hence Algorithm 6.4 runs in polynomial time with respect to $|V(F)|, |E(F)|$ and thus we have that $k\text{-SOET}$ is in \mathbb{P} . \square

This corollary then leads to theorem 6.3.11.

Algorithm 6.1 Producing $S_{V'}$ from a distance-hereditary graph G

```

1: INPUT: A graph  $G$  and a subset of vertices  $V' \subseteq V(G)$ .
2: OUTPUT: A sequence  $\mathbf{v}$  such that  $\tau_{\mathbf{v}}(G)[V'] = S_{V'}$ , if  $S_{V'} < G$ .
3: ERROR, if  $S_{V'} \not< G$ .
4:
5:
6: if  $|V'| = 1$  then
7:   Return ()
8:   QUIT
9: end if
10: Find a  $\mathbf{v}$  such that  $\tau_{\mathbf{v}}(G)$  contain the star graph on  $V'$  as a subgraph by calling
    algorithm 6.2.
11: Let  $c$  be a vertex in  $V'$ , adjacent to all other in  $V'$  (except itself).
12: for  $i$  in  $\{0, 1\}$  do # Two iterations are always needed if there is more than one
    bad edge
13:   Let  $B$  be the vertices incident to a bad edge. # These are the vertices in
     $\tau_m(G)[V' \setminus \{c\}]$  of degree 1 or higher
14:   Let  $L = V' \setminus (\{c\} \cup B)$ .
15:   if  $B = \emptyset$  then # If already  $S_{V'}$ , only for  $i = 0$ 
16:     Return  $\mathbf{v}$ 
17:     QUIT
18:   else
19:     if  $B = V' \setminus \{c\}$  then # I.e. if  $L = \emptyset$ 
20:       Set  $\mathbf{v} = \mathbf{v} \parallel (c)$ 
21:       BREAK
22:     end if
23:     Let  $U$  be the set  $U = \{u \in V(G) \setminus V' : B \subseteq N_u \wedge L \not\subseteq N_u\}$  # Candidates
    for the  $u$  in eq. (6.29)
24:     if  $U = \emptyset$  then
25:       Raise ERROR( $S_{V'}$  is not a vertex-minor of  $G$ ) # Actually not
    needed, only for clarity
26:     end if
27:     Set  $found = \text{False}$ 
28:     for  $u$  in  $U$  do
29:       if  $(u, c) \notin E(\tau_{\mathbf{v}}(G))$  then
30:         Set  $\mathbf{v} = \mathbf{v} \parallel (u)$ 
31:         Set  $found = \text{True}$  # Found a  $u$  satisfying eq. (6.29)
32:         Break
33:       else if  $\exists h : (h \in N_u \cap N_c \setminus \cup_{x \in V' \setminus \{u, c\}} N_x)$  then
34:         Set  $\mathbf{v} = \mathbf{v} \parallel (h, u)$ 
35:         Set  $found = \text{True}$  # Found a  $u$  and  $h$  satisfying eq. (6.29)
36:         Break
37:       end if
38:     end for
39:     if  $\neg found$  then # I.e. condition eq. (6.29) is false
40:       Raise ERROR( $S_{V'}$  is not a vertex-minor of  $G$ )
41:     end if
42:   end if
43: end for
44: Return  $\mathbf{v}$ 
45: QUIT

```

Algorithm 6.2 Find a \mathbf{v} such that $\tau_{\mathbf{v}}(G)$ contain the star graph on V' as a subgraph

- 1: **INPUT:** A graph G and a subset of vertices $V' \subseteq V(G)$.
 - 2: **OUTPUT:** A sequence \mathbf{v} such that $\tau_{\mathbf{v}}(G)[V'] = SS_{(B,L,\{c\})}$, where $(B, L, \{c\})$ is a partition of V' .
 - 3: _____
 - 4: _____
 - 5: Pick an arbitrary vertex from V' and denote this f
 - 6: Find a \mathbf{v} such that $\tau_{\mathbf{v}}(G)[V' \setminus \{f\}] = S_{V'}$ and denote the center c by calling algorithm 6.1
 - 7: Find a shortest path $P = (p_0 = f, p_1, \dots, p_k, p_{k+1} = c)$ between f and c .
 - 8: **for** i in $(1, \dots, k)$ **do**
 - 9: **if** f is adjacent to any vertex in $V' \setminus \{c\}$ in the graph $\tau_{\mathbf{v}}(G)$ **then**
 - 10: Pick an arbitrary vertex in $N_f(\tau_{\mathbf{v}}(G)) \cap V' \setminus \{c\}$ and denote this v
 - 11: Set $\mathbf{v} = \mathbf{v}||v$
 - 12: **else**
 - 13: Set $\mathbf{v} = \mathbf{v}||(f, p_i, f)$
 - 14: **end if**
 - 15: **end for**
 - 16: **Return** \mathbf{v}
 - 17: QUIT
-

Algorithm 6.3 k' -DPP

- 1: **INPUT:** A multi-graph H and a set of 2-tuples $\{(v_1, \hat{v}_1), \dots, (v_{k'}, \hat{v}_{k'})\}$
 - 2: **OUTPUT:** TRUE if there exist edge-disjoint paths in H connecting v_i, \hat{v}_i for all $i \in [k']$.
 - 3: FALSE otherwise
-

Algorithm 6.4 k -SOET

```

1: INPUT:   A 4-regular multi-graph  $F$  and a set  $V' = \{v_1, \dots, v_k\}$  such that  $V' \subset V(F)$ 
2: OUTPUT: TRUE if there exist a SOET on  $F$  with respect to  $V'$ 
3:           FALSE otherwise
4: _____
5:
6: Generate a list  $L$  of all possible SOET-splittings of  $F$  with respect to  $V'$ 
7:
8: for all graphs  $H$  in  $L$  do
9:
10:  for all permutations  $\pi$  of the set  $[1 : k]$  do
11:    Run  $2k$ -DPP on the set
12:     $\{(v_{\pi(1)}^{(a)}, v_{\pi(2)}^{(a)}), \dots, (v_{\pi(k)}^{(a)}, v_{\pi(1)}^{(a)}), (v_{\pi(1)}^{(b)}, v_{\pi(2)}^{(b)}), \dots, (v_{\pi(k)}^{(b)}, v_{\pi(1)}^{(b)})\}$ 
13:
14:    if  $2k$ -DPP returns TRUE then
15:      RETURN TRUE
16:    QUIT
17:    end if
18:  end for
19: end for
20: RETURN FALSE
21: QUIT

```

6.4. Connected vertex-minor on three vertices or less

In this section we prove for the decision problem QubitMinor with arbitrary connected¹⁷ input graphs states $|G\rangle$ and $|G'\rangle$, that whenever $|V(G')| \leq 3$ and $V(G') \subset V(G)$ we have that $|G'\rangle$ is a qubit-minor of $|G\rangle$. Furthermore we provide an algorithm that reduces the number of operations one needs to transform $|G\rangle$ to the desired graph state. Equivalently, we have the following theorem.

Theorem 6.4.1. *Let G be a connected graph and G' be a connected graph with vertices V' , such that $|V'| \leq 3$. Then we have that*

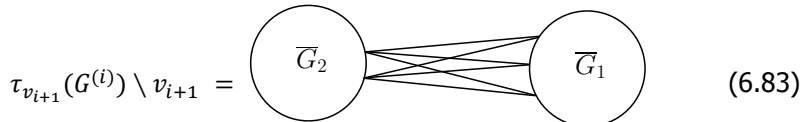
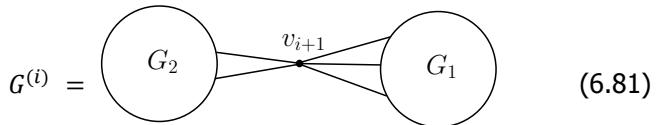
$$G' < G \iff V' \subseteq V(G) \tag{6.80}$$

◊

Proof. Note that if $G' < G$ then clearly $V' \subseteq V(G)$. Assume therefore that $V' \subseteq V(G)$. Let's denote the vertices in $V(G) \setminus V'$ as $U = (v_1, v_2, \dots, v_{n-k})$, where $n = |V(G)|$ and $k = |G'|$. We will now show that $G' < G$ by finding a sequence of operations $P = P_{v_{n-k}} \circ \dots \circ P_{v_1}$, where $P_{v_i} \in \{X_{v_i}, Y_{v_i}, Z_{v_i}\}$, such that each intermediate graph $G^{(i)} = P_{v_i} \circ \dots \circ P_{v_1}(G)$ is connected. Since any connected graph with vertices V' , for $|V'| \leq 3$, is either a star graph or a complete graph, which are LC-equivalent, this shows that $G' < G$. Let's therefore consider such an intermediate graph $G^{(i)}$ for some $i \in [n - k]$ and the next vertex v_{i+1} . We will now show that $G^{(i)} \setminus v_{i+1}$ or $\tau_{v_{i+1}}(G^{(i)}) \setminus v_{i+1}$ is connected. This will be done by considering the following two cases:

Assume that $G^{(i)} \setminus v_{i+1}$ is not connected:

This means that v_{i+1} is a cut vertex. Let G_1 be a connected component in the graph $G^{(i)} \setminus v_{i+1}$ and G_2 the rest of the vertices, see eqs. (6.81) and (6.82) for an illustration. Since v_{i+1} is a cut vertex, we know that no vertex in $N_{v_{i+1}} \cap V(G_2)$ is adjacent to any vertex in $N_{v_{i+1}} \cap V(G_1)$, in the graph $G^{(i)}$. Thus, all vertices in $N_{v_{i+1}} \cap V(G_2)$ are adjacent to all vertices in $N_{v_{i+1}} \cap V(G_1)$ in the graph $\tau_{v_{i+1}}(\hat{G}) \setminus v_{i+1}$, showing that this graph is connected, see eq. (6.83).



¹⁷Here connected means that the corresponding graphs are connected or in other words that there is entanglement between any bipartition of the qubits of the state.

Assume that $\tau_v(\hat{G}) \setminus v$ is not connected:

This case follows from the previous. To see this, let \tilde{G} be the graph $\tau_v(G^{(i)})$. From the above case we then know that $\tau_v(\tilde{G}) \setminus v_{i+1}$ is connected. But this graph is exactly $G^{(i)} \setminus v_{i+1}$ since local complementations are involutions and the theorem follows. □

From theorem 6.4.1 we can easily formulate an algorithm that finds a sequence of operations taking G to a desired connected graph G' on V' , by simply checking recursively if $\hat{G} \setminus v$ or $\tau_v(\hat{G}) \setminus v$ is connected. However, if the vertices in V' are already close in G and G is a very large, it would be practical to not have to consider all the vertices in G . Next, we present a more practical algorithm to find a sequence of local complementations and vertex-deletions that take some graph G to a star graph¹⁸ on the vertices a , b and c . The algorithm performs the following steps:

- Find the shortest path P_1 between a and b and connect these by doing τ -operations along this path. This first step already gives an algorithm for creating a star graph on vertices a and b , see algorithm 6.5.
- Then do the same with b and c by finding the shortest path P_2 between b and c . The question is now whether we removed the edge between a and b while connecting b and c . This could only happen if P_2 goes through a vertex which is a common neighbor of a and b . Furthermore, since P_2 is the shortest path from c to b this could only be the case for the last vertex on the path, before b .
 - If c is connected to a before the last local complementation on P_2 , then stop, since the induced graph is already connected.
 - Assume that c is not connected to a before the last local complementation is performed. So in this case, after performing the local complementation along P_2 , a and b will not be connected but they will both be connected to c .

The full protocol is formalized in algorithm 6.6.

Algorithm 6.5 Producing S_1 on vertices a and b

- 1: **INPUT:** A graph G and two vertices $a, b \in V(G)$.
 - 2: **OUTPUT:** A sequence of vertices \mathbf{v} such that $\tau_{\mathbf{v}}(G)[\{a, b\}]$ is a star graph.
 - 3: _____
 - 4: _____
 - 5: Find the shortest path P between a and b
 - 6: Perform τ_p for all $p \in P \setminus \{a, b\}$
-

¹⁸Note that any other connected graph on $\{a, b, c\}$ can easily be constructed.

Algorithm 6.6 Producing S_2 on vertices a, b and c

```

1: INPUT: A graph  $G$  and three vertices  $a, b, c \in V(G)$ .
2: OUTPUT: A sequence of vertices  $\mathbf{v}$  such that  $\tau_{\mathbf{v}}(G)[\{a, b, c\}]$  is a star graph.
3:
4:
5: Find the shortest path  $P_1$  between  $a$  and  $b$ 
6: Perform  $\tau_p$  for all  $p \in P_1 \setminus \{a, b\}$ 
7: Find the shortest path  $P_2 = (p_0 = b, p_1, \dots, p_n, p_{n+1} = c)$ 
8: for  $i$  in  $1, \dots, n$  do
9:   if  $a$  and  $c$  are not adjacent then
10:    Perform  $\tau_{p_i}$ 
11:   end if
12: end for

```

6.5. Conclusion

We have shown that deciding if a graph state $|G'\rangle$ can be obtained from another graph state $|G\rangle$ using LC + LPM + CC is NP-Complete , by showing that `VertexMinor` is NP-Complete . The computational complexity of `VertexMinor` was previously unknown and was posted as an open problem in [18]. It is important to note that our results are for labeled graphs, since vertices correspond to physical qubits in the case of transforming graph states.

We presented an efficient algorithm for `StarVertexMinor` when the input graph is restricted to be distance-hereditary. It would be interesting to know if the same approach generalize to qudit graph states described by weighted graphs of low rank-width.

We presented an efficient algorithm for `k-StarVertexMinor` on circle graphs, that is the problem of deciding if the star graph on a subset of vertices V' , with $|V'| = k$, is a vertex-minor of another graph. However, the computational complexity of `k-StarVertexMinor` or `k-VertexMinor` on general graphs is still unknown.

Below we list some more open questions regarding the computational complexity of deciding how graph states can be transformed using local operations:

- Is the problem still NP-Complete if one allows for arbitrary local operations and classical communication (LOCC), instead of restricting to only LC + LPM + CC? Is there an efficient algorithm for states with bounded Schmidt-rank width also in this case? It has been shown that many graph states are equivalent under LC if and only if they are equivalent under stochastic LOCC (SLOCC) [19]. An interesting question is therefore whether graph states described by distance-hereditary or circle graphs fall into this category.
- What is the computational complexity of deciding if a graph state $|G\rangle$ can be transformed in to another $|G'\rangle$ if the qubits are partitioned into multiple local sets, within which multi-qubit Clifford operations and multi-qubit Pauli measurements can be performed.

- How do the computational complexity results of this chapter relate to the complexity of *code switching* in stabilizer quantum error correction codes, e.g. fault-tolerant interconversion between two stabilizer codes [20]? More precisely, can one re-use the techniques presented here, to prove that it is computationally hard to decide whether one stabilizer code can be fault-tolerantly transformed into another. We conjecture that this could be possible, since many stabilizer codes (such as topological stabilizer codes) have a notion of 'locality' with fault-tolerant conversions being the conversions that respect this locality [21].

Acknowledgements

We thank Kenneth Goodenough, Think Le, Bas Dirkse, Victoria Lipinska, Stefan Bäuml, Tim Coopmans, Jérémy Ribeiro, Ben Criger and Jens Eisert for discussions. AD, JH and SW were supported by STW Netherlands, NWO VIDI grant and an ERC Starting grant.

References

- [1] A. Dahlberg, J. Helsen, and S. Wehner, *How to transform graph states using single-qubit operations: computational complexity and algorithms*, *Quantum Science and Technology* (2020).
- [2] M. Van den Nest, J. Dehaene, and B. De Moor, *Graphical description of the action of local clifford transformations on graph states*, *Physical Review A* **69**, 022316 (2004).
- [3] B. Courcelle and J. Engelfriet, *Graph Structure and Monadic Second-Order Logic: A Language Theoretic Approach*, 1st ed. (Cambridge University Press, New York, NY, USA, 2011).
- [4] *Git-repository with implemented code for graph state calculations*, <https://github.com/AckslD/vertex-minors> (2020).
- [5] H.-J. Bandelt and H. M. Mulder, *Distance-hereditary graphs*, *Journal of Combinatorial Theory, Series B* **41**, 182 (1986).
- [6] B. Courcelle and S. il Oum, *Vertex-minors, monadic second-order logic, and a conjecture by Seese*, *Journal of Combinatorial Theory. Series B* **97**, 91 (2007).
- [7] *SAGE*, <http://www.sagemath.org/> (2020).
- [8] S. Oum and P. Seymour, *Approximating clique-width and branch-width*, *Journal of Combinatorial Theory, Series B* **96**, 514 (2006).
- [9] B. Courcelle, *Circle graphs and monadic second-order logic*, *Journal of Applied Logic* **6**, 416 (2008).
- [10] A. Bouchet, *An efficient algorithm to recognize locally equivalent graphs*, *Combinatorica* **11**, 315 (1991), [arXiv:0702057v2 \[cs\]](https://arxiv.org/abs/0702057v2) .

- [11] M. R. Garey, D. S. Johnson, and R. E. Tarjan, *The Planar Hamiltonian Circuit Problem is NP-Complete*, [SIAM Journal on Computing](#) **5**, 704 (1976).
- [12] M. Fleury, *Deux problemes de Geometrie de situation*, *Journal de mathematiques elementaires* **2nd**, 257 (1883).
- [13] E. W. Dijkstra, *A note on two problems in connexion with graphs*, *Numerische Mathematik* , 269 (1959).
- [14] J. Spinrad, *Recognition of Circle Graphs*, (1994).
- [15] A. Bouchet, *Circle Graph Obstructions*, [Journal of Combinatorial Theory, Series B](#) **60**, 107 (1994).
- [16] N. Robertson and P. D. Seymour, *Graph minors. I - XXIII. Journal of Combinatorial Theory, Series B* **35**, 39 (1983-2010).
- [17] K. I. Kawarabayashi and Y. Kobayashi, *The edge-disjoint paths problem in Eulerian graphs and 4-edge-connected graphs*, [Combinatorica](#) **35**, 477 (2015).
- [18] K. K. Dabrowski, F. Dross, J. Jeong, M. M. Kanté, O. Kwon, S. Oum, and D. Paulusma, *Computing small pivot-minors*, in *Graph-Theoretic Concepts in Computer Science*, edited by A. Brandstädt, E. Köhler, and K. Meer (Springer International Publishing, Cham, 2018) pp. 125–138.
- [19] B. Zeng, H. Chung, A. W. Cross, and I. L. Chuang, *Local unitary versus local Clifford equivalence of stabilizer and graph states*, [Physical Review A](#) **75** (2007), 10.1103/PhysRevA.75.032325, [arXiv:0411115 \[quant-ph\]](#) .
- [20] D. Gottesman, *PhD Theses, Ph.D. thesis*, California Institute of Technology (2004), [arXiv:9705052 \[quant-ph\]](#) .
- [21] H. P. Nautrup, N. Friis, and H. J. Briegel, *Fault-tolerant interface between quantum memories and quantum processors*, *Nature Communications* **8**, 1321 (2017).

7

The complexity of the vertex-minor problem

Axel Dahlberg, Jonas Helsen, Stephanie Wehner

A graph H is a vertex-minor of a graph G if it can be reached from G by the successive application of local complementations and vertex deletions. Vertex-minors have been the subject of intense study in graph theory over the last decades and have found applications in other fields such as quantum information theory. Therefore it is natural to consider the computational complexity of deciding whether a given graph G has a vertex-minor isomorphic to another graph H . Here we prove that this decision problem is \mathbb{NP} -complete, even when restricting H, G to be circle graphs, a class of graphs that has a natural relation to vertex-minors.

7.1. Introduction

A central problem in graph theory is the study of ‘substructures’ of graphs. These substructures of some graph are usually defined as the graphs which can be reached by a given set of graph operations. An example of such a substructure to be seriously studied early on, was the graph minor, where the central question was to decide whether a graph G can be transformed into a graph H through the successive application on G of vertex deletions, edge deletions and edge contractions [1]. If this is the case we call H a minor of G . The problem (Minor) of deciding whether a graph H is a minor of G is NP -complete when both H and G are part of the input to the problem [2]. However, given a fixed H , we can define the problem ($H\text{-Minor}$) of deciding whether H is a minor of G , where only G is part of the input. As expounded in Robertson & Seymour’s seminal series of papers [3], $H\text{-Minor}$ is solvable in cubic time for any graph H . Since then a great variety of minor-relations has been defined and for many of those the complexity has been studied. Of particular interest recently are the minor-relations related to the graph operation of local complementation, namely vertex- and pivot-minors. These two minor structures have been studied within the graph theory community [4–7] but have also found surprising applications outside of it, notably in the field of quantum information science [8–13]. The complexity of the vertex- and pivot-minor decision problems was a notable open problem (see question 7 in [14]). Recently it was proven in [15] that the pivot-minor problem is NP -complete if both G and H are part of the input to the problem, but the complexity of the vertex-minor problem was left open. In chapter 6 we proved, in the context of quantum information theory, the NP -completeness of the labeled version of the vertex-minor problem, i.e. the problem of deciding if H is a vertex-minor of the graph G , taking labeling into account. The labeled version of the vertex-minor problem is relevant in the context of quantum information theory since there the vertices of the graph corresponds to physical qubits in, for example, a quantum network. However we did not discuss the complexity of the related problem of deciding whether G has a vertex-minor isomorphic to H (on any subset of the vertices). Here we close this gap, proving that the unlabeled version of the vertex-minor problem is also NP -complete. To avoid confusion with the problems studied in chapter 6 we will here call the unlabeled version of the vertex-minor problem Iso-VertexMinor . Moreover we here prove that Iso-VertexMinor remains NP -complete even when H is restricted to be a star graph and G a circle graph. Our work resolves the problem left open in [15] and provides a partial answer to the questions posed in [14]. In the process of proving hardness we make use of the concept of the semi-ordered Eulerian tour (SOET), a graph construction we introduced in definition 4.7.15 that may be of further independent interest.

The chapter is organized as follows: in section 7.2 we recall relevant graph theoretic notions such as vertex-minors and circle graphs and rephrase some of these in the unlabeled setting. We also discuss the concept of semi-ordered Eulerian tours. In section 7.3 we prove the main result: the NP -completeness of the vertex-minor problem.

7.2. Preliminaries

In this section we re-phrase some of the problems and results from chapter 4 but for unlabeled graphs.

7.2.1. Vertex-minors

In chapter 4 we introduced problem 4.5.4. Here we are interested in the unlabeled version:

Problem 7.2.1 (*Iso-VertexMinor*). Given a graph G and a graph H decide whether there exists a graph \tilde{H} such that (1) H and \tilde{H} are isomorphic, and (2) $\tilde{H} < G$. \diamond

We can restrict this problem to a special case, where the graph H is a star-graph on k vertices. We call this problem *Iso-StarVertexMinor*. Note that we must only specify k as there exists only one star-graph on k vertices up to isomorphism. Formally we have

Problem 7.2.2 (*Iso-StarVertexMinor*). Given a graph G and an integer k decide whether there exists a subset V' of $V(G)$ with $|V'| = k$ and a star graph on V' denoted $S_{V'}$ such that $S_{V'} < G$. \diamond

7.2.2. Semi-Ordered Eulerian tours

In chapter 6 we used the problem of finding a SOET, problem 4.7.16, to prove that *VertexMinor* is NP-Complete. We will use a similar approach here and thus introduce the unlabeled version of the *SOET*-problem.

Problem 7.2.3 (*Iso-SOET*). Let F be a 4-regular multi-graph and let $k \leq |V(F)|$ be an integer. Decide whether there is a $V' \subset V(F)$ with $|V'| = k$ such that there exists a SOET U on F with respect to the set V' . \diamond

In chapter 6 we proved that a version of problem 7.2.3 where V' is part of the input to the problem, is NP-complete. In the next section we prove that also the problem of deciding whether such a V' exists, i.e. problem 7.2.3, is also NP-complete.

7.3. NP-completeness of *Iso-VertexMinor*

In this section we prove the NP-completeness of *Iso-VertexMinor*. This we do in three steps. We will begin by (1) proving that *Iso-SOET* is NP-Hard. We do this by reducing the problem of deciding whether a 3-regular graph R is Hamiltonian to *Iso-SOET*. Next we (2) reduce *Iso-SOET* to *Iso-StarVertexMinor* and *Iso-StarVertexMinor* to *Iso-VertexMinor*, thus proving the NP-hardness of *Iso-VertexMinor*. Finally we (3) show that *Iso-VertexMinor* is also in NP.

7.3.1. *ISO-SOET* is NP-hard

We will make use of the definition of a Hamiltonian graph, definition 6.2.4, and the associated *CubHam* decision problem, problem 6.2.5.

The reduction of CubHam to Iso-SOET is done by going through the following steps.

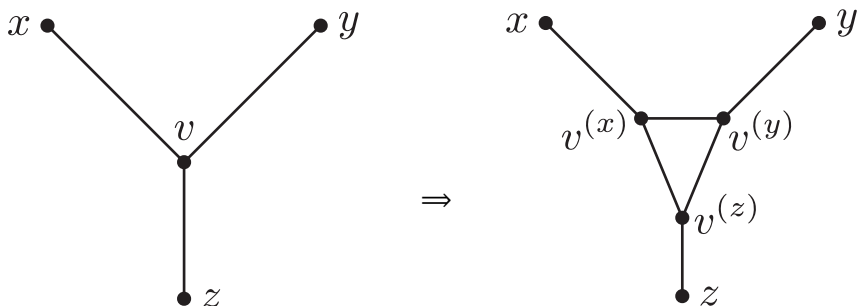
1. Introduce the notion of a (4-regular) K_3 -expansion $\Lambda(R)$ of a 3-regular graph R . This is done in definition 7.3.1.
2. Prove that if a 3-regular graph R is Hamiltonian then the K_3 -expansion $\Lambda(R)$ of R allows for a SOET of size $2|V(R)|$. This is done in lemma 7.3.2
3. Prove that if the K_3 -expansion $\Lambda(R)$ of a 3-regular graph R allows for a SOET of size $2|V(R)|$ then R is Hamiltonian. This is done in lemma 7.3.4

Note that 1. and 2. above provides necessary and sufficient condition for whether a 3-regular graph R is Hamiltonian in terms of whether $\Lambda(R)$ allows for a SOET of a certain size. This implies that CubHam reduces to Iso-SOET and hence that Iso-SOET is NP-hard .

We begin by introducing the K_3 -expansion: a mapping from 3-regular graphs to 4-regular multi-graphs. This expansion is similar to definition 6.2.7 which was used in chapter 6 but differ in the graph used to replace each vertex.

Definition 7.3.1 (K_3 -expansion). Let R be a 3-regular graph. A K_3 -expansion $\Lambda(R)$ of a 3-regular graph R is constructed from R by performing the following two steps:

1. Replace each vertex v in R with a subgraph isomorphic to K_3 as below



(7.1)

where x, y and z are the neighbors of v . We will denote the K_3 -subgraph associated to the vertex v with T_v , i.e. $T_v = G[\{v^{(x)}, v^{(y)}, v^{(z)}\}]$.

2. For all $v, v' \in R$ such that $v \neq v'$, double the edge that is incident on two subgraphs $T_v, T_{v'}$.

◊

The graph $\Lambda(R)$ will be called a K_3 -expansion of R . A multi-graph F that is the K_3 -expansion of some 3-regular graph R will also be referred to as a K_3 -expanded graph. Furthermore, the number of vertices in $\Lambda(R)$ is $3 \cdot |V(R)|$ and the number

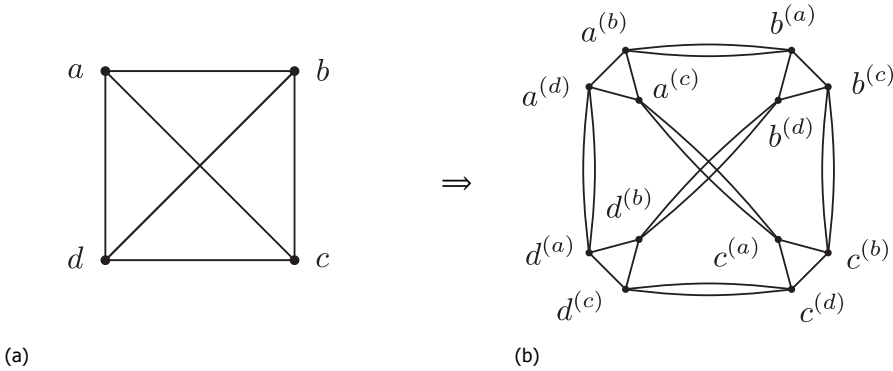


Figure 7.1: Figure showing (a) the complete graph K_V on vertices $V = \{a, b, c, d\}$ and (b) its associated K_3 -expansion $\Lambda(K_V)$.

of edges is $2 \cdot |E(R)| + 3 \cdot |V(R)|$. In figure 7.1 we show an example of a 3-regular graph and its K_3 -expansion.

We now argue that if a 3-regular graph R is Hamiltonian then its K_3 -expansion allows for a SOET on $2|V(R)|$ vertices and thus providing a necessary condition for a 3-regular graph being Hamiltonian.

Lemma 7.3.2. *Let R be a triangular graph with k vertices and let $\Lambda(R)$ be its K_3 -expansion. If R is Hamiltonian then $\Lambda(R)$ allows for a SOET of size $2k$.* \diamond

Proof. Let M be a Hamiltonian tour on R . Choose $x_0 \in V(R)$ and let $\mathbf{L} = x_0 x_1 \dots x_{k-1}$ be the word formed by walking along M when starting on x_0 . Note that x_i and $x_{(i+1) \pmod k}$ are adjacent in R for all $i \in [k]$. Now consider the K_3 -expansion $\Lambda(R)$ of R . We will argue that $\Lambda(R)$ allows for a SOET with respect to the set

$$V' = \{x_0^{(x_{k-1})}, x_0^{(x_1)}, x_1^{(x_0)}, x_1^{(x_2)}, \dots, x_{k-1}^{(x_{k-2})}, x_{k-1}^{(x_0)}\}. \tag{7.2}$$

For all $i \in [k]$ let v_i be the unique vertex adjacent to x_i in $\Lambda(R)$ that is not $x_{(i-1) \pmod k}$ or $x_{(i+1) \pmod k}$. Now consider the following words on $V(\Lambda(R))$.

$$\mathbf{V} := x_0^{x_{k-1}} x_0^{(x_1)} x_1^{(x_0)} x_1^{(x_2)} x_2^{(x_1)} x_2^{(x_3)} \dots x_{k-1}^{(x_{k-2})} x_{k-1}^{(x_0)} \tag{7.3}$$

$$\mathbf{W} := x_0^{x_{k-1}} x_0^{(v_0)} x_0^{(x_1)} x_1^{(x_0)} x_1^{(v_1)} x_1^{(x_2)} x_2^{(x_1)} x_2^{(v_2)} x_2^{(x_3)} \dots x_{k-1}^{(x_{k-2})} x_{k-1}^{(v_{k-1})} x_{k-1}^{(x_0)} \tag{7.4}$$

These words describe disjoint trails on $\Lambda(R)$ as illustrated for an example graph in figure 7.2.

Now consider the word \mathbf{VW} . This word describes a trail U_{VW} on $\Lambda(R)$ that visits every vertex in V' exactly twice in the same order. This means U_{VW} is a semi-ordered tour. It is however not Eulerian. To make it Eulerian we have to extend the tour U_{VW} to include all edges in $\Lambda(R)$. Note that these edges are precisely the edges connecting the vertices $x_i^{(v_i)}, v_i^{(x_i)}$ for all $i \in [k]$. We can lift U_{VW} to a Eulerian tour by adding vertices to W by the following algorithm. It is easy to see that the

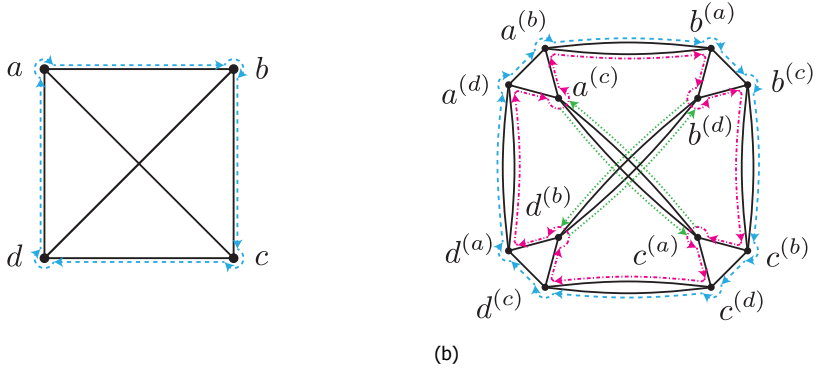


Figure 7.2: Figure showing (a) a Hamiltonian path (blue dashed arrows) on the complete graph on vertices $V = \{a, b, c, d\}$ and (b) the corresponding disjoint trails described by the words \mathbf{V} (blue dashed arrows) and \mathbf{W} (pink dashed-dotted arrows) from eqs. (7.3) and (7.4) on the associated K_3 -expansion $\Lambda(K_V)$. The edges used to extend the tour to a Eulerian tour as captured by algorithm 7.1 are show as green dotted arrows.

Algorithm 7.1 Algorithm for lifting the tour U_{VW} to a Eulerian tour on $\Lambda(R)$

```

for  $i \in [k]$  do
  if  $x_i^{(v_i)} v_i^{(x_i)} x_i^{(v_i)} \notin \mathbf{X}'$  then
    Insert  $v_i^{(x_i)} x_i^{(v_i)}$  into  $\mathbf{X}'$  right after  $x_i^{(v_i)}$ 
  end if
end for
    
```

7

tour described by \mathbf{VW} after running algorithm 7.1 is also Eulerian and is hence a SOET with respect to the set V' . This completes the lemma. □

Next we prove a necessary condition (lemma 7.3.4) for the existence of a SOET on a subset V' of the vertices of a 4-regular graph F .

Lemma 7.3.3. *Let F be a 4 regular graph and $V' \geq 4$ be a subset of its vertices. If the fully connected graph K_3 is an induced subgraph of $F[V']$ then F does not allow for a SOET with respect to V'* ◇

Proof. Assume that F has three vertices $V' = \{u, v, w\}$ such that $F[V'] = K_3$. Let U be a Eulerian tour on F (note that U always exists). Assume by contradiction that U is a SOET with respect to V' . It is easy to see that since u and v are adjacent in F they must also be consecutive in U . However the same is true for u and w and also w and v . This means a tour starting at u and must traverse v , then w , and then immediately u again (up to interchanging u and w). Since $\{u, v, w\}$ is a strict subset of V' (since by assumption $|V'| \geq 4$) this means that, when starting at u , the tour U does not traverse all vertices in V' before returning to u . This gives a contradiction with the definition of SOET from which the lemma follows. □

Now we will leverage lemma 7.3.3 to prove that if the K_3 -expansion $\Lambda(R)$ of a 3-regular graph R allows for a SOET with respect to a vertex-set V' with $|V'| = 2|V(R)|$ then the graph R must be Hamiltonian.

Lemma 7.3.4. *Let R be a 3-regular graph and $\Lambda(R)$ its K_3 -expansion. If there exists a set $V' \subset V(\Lambda(R))$ with $|V'| = 2|V(R)|$ such that $\Lambda(R)$ allows for a SOET with respect to V' then R must be Hamiltonian. \diamond*

Proof. Assume that there exists a subset V' of $V(\Lambda(R))$ with $|V'| = 2|V(R)|$ such that $\Lambda(R)$ allows for a SOET U with respect to V' .

Note first that since $|V(\Lambda)| = 3|V(R)|$ and $|V'| = 2|V(R)|$ we must have, by lemma 7.3.3 that $|V[T_u] \cap V'| = 2$ for all $u \in V(R)$. This is easiest seen by contradiction. Assume that there exists a $u \in V(R)$ such that $|V[T_u] \cap V'| < 2$. Since $V(T_v) \cap V(T_{v'}) = \emptyset$ for all $v, v' \in V(R)$, $|V(\Lambda)| = 3|V(R)|$, and $|V'| = 2|V(R)|$ this implies there must also exist a $u' \in V(R)$ such that $|V(T_{u'}) \cap V'| = 3$. This means that $V(T_{u'}) \subset V'$. However the induced subgraph $\Lambda(R)[T_{u'}]$ is isomorphic to K_3 (this is easily seen from the definition of K_3 -expansion). By lemma 7.3.3 we must thus conclude that $\Lambda(R)$ does not allow for a SOET with respect to V' leading to a contradiction. Hence we must have that $|V(T_u) \cap V'| = 2$ for all $u \in V(R)$.

Now consider two vertices $x, x' \in V'$ such that x, x' are consecutive in the SOET U . Note that, by definition of $\Lambda(R)$, there must exist $w, w' \in V(R)$ such that $x \in T_w$ and $x' \in T_{w'}$. We will now argue that we must have either $w = w'$ or w, w' are adjacent in R . We argue this by contradiction. Assume thus that w, w' are neither equal nor adjacent in R . Now consider the maximal sub-word \mathbf{Y} of $m(U)$ associated to x, x' . Since w, w' are neither equal nor adjacent in R , the trail described by the word \mathbf{Y} must pass through a triangle subgraph different from T_w and $T_{w'}$, i.e. there exist a vertex $w'' \in V(R)$ such that $|\mathbf{Y} \cap V(T_{w''})| \geq 2$. However since by construction $|V(T_{w''}) \cap V'| = 2$ (as shown above) and $|V(T_{w''})| = 3$ we must have that $|V' \cap \mathbf{Y}| \geq 1$. This is however in contradiction with the maximality of the sub-word \mathbf{Y} . Hence we must have that $w = w'$ or that $(w, w') \in E(R)$. Now consider the word $m(U)$ associated to the SOET U and the induced sub-word $m(U)[V']$. By the above, and the fact that if two vertices in V' are adjacent in $\Lambda(R)$, they must also be consecutive in U (this is a consequence of U being Eulerian and thus having to traverse the edge connecting these vertices), we have that $m(U)[V']$ must be of the form

$$m(U)[V'] = x_0 x'_0 x_1 x'_1 x_2 x'_2 \dots x_{k-1} x'_{k-1} x_0 x'_0 \dots x_{k-1} x'_{k-1} \quad (7.5)$$

where $x_i, x'_i \in T_{w_i}$ and $\{w_1, \dots, w_k\} = V(R)$ and moreover that $(w_i, w_{i+1}) \in E(R)$ for all $i \in [k]$ and also $(w_k, w_0) \in E(R)$. This immediately implies that the word $\mathbf{M} = w_1 w_2 \dots w_k$ describes a Hamiltonian tour on R , and hence that R is Hamiltonian. \square

Since lemma 7.3.4 and 7.3.2 provide necessary and sufficient conditions for a 3-regular graph being Hamiltonian in terms of whether a K_3 -expanded graph allows for a SOET, we can now easily prove the hardness of problem 7.2.3.

Theorem 7.3.5. *ISO-SOET is NP-Hard. \diamond*

Proof. Let R be an instance of CubHam , that is, a 3-regular graph on k vertices. From R we can construct the 4-regular K_3 -expansion $\Lambda(R)$. Note that this can be done in poly-time in k . Now note that $(\Lambda(R), 2k)$ is an instance of Iso-SOET . If R is a YES instance of CubHam , that is, R is Hamiltonian, then by lemma 7.3.2 we have that $(\Lambda(R), 2k)$ is a YES instance of Iso-SOET . On the other hand, if $(\Lambda(R), 2k)$ is a YES instance of Iso-SOET , then R is a YES instance of CubHam by lemma 7.3.4. By contra-position this means that if R is a NO instance of CubHam , then $(\Lambda(R), 2k)$ is a NO instance of Iso-SOET . This means CubHam is Karp-reducible to SOET . Since CubHam is NP-complete [16], this implies that SOET is NP-hard. \square

7.3.2. ISO-VERTEXMINOR is NP-Hard

Note first that $\text{Iso-StarVertexMinor}$ trivially reduces to Iso-VertexMinor , as it is a strict sub-problem. This means that if $\text{Iso-StarVertexMinor}$ is NP-hard then so is Iso-VertexMinor . In this section we show that the Iso-SOET reduces to Iso-VertexMinor . For this we will make use of the properties of circle graphs, discussed in section 7.2.

Corollary 4.7.16.1 states that a 4-regular multi-graph F allows for a SOET with respect to a subset of its vertices $V' \subseteq V(F)$ if and only if an alternance graph $\mathcal{A}(U)$ (which is a circle graph), induced by some Eulerian tour on F , has a star graph $S_{V'}$ on V' as a vertex-minor.

Since circle graphs are a subset of all simple graphs we can then decide whether a 4-regular graph F allows for a SOET with respect to some subset V' of its vertices by constructing the circle graph induced by an Eulerian tour on F and checking whether it has a star-vertex-minor on the vertex set V' . This leads to the following theorem.

Theorem 7.3.6. *The decision problem Iso-SOET reduces to $\text{Iso-StarVertexMinor}$.* \diamond

Proof. Let (F, k) be an instance of Iso-SOET , where F is a 4-regular multi-graph and $k \leq |V(F)|$ some integer. Also let G be a circle graph induced by some Eulerian tour U on F . From corollary 4.7.16.1 we see that G has $S_{V'}$ as a vertex-minor for some subset of vertices V' of G if and only if F allows for a SOET with respect to this vertex set V' . Since an Eulerian tour U can be found in polynomial time [17] and since G can be efficiently constructed given U , see chapter 6, considering the case of $|V'| = k$ concludes the reduction. \square

7.3.3. ISO-VERTEXMINOR is in NP

Next we argue that the problem Iso-VertexMinor is in NP. This just follows from the fact that the non-isomorphic vertex-minor problem is in NP.

Theorem 7.3.7. *The decision problem Iso-VertexMinor is in NP.* \diamond

Proof. From chapter 6 we know that there exists a polynomial-length witness for the problem of deciding if a labeled graph G has a vertex-minor equal to another graph H on some fixed subset of its vertices. Since GRAPHISOMORPHISM is in NP we can construct a polynomial-length for Iso-VertexMinor , i.e. to decide if

G has a vertex-minor isomorphic to H . We thus conclude that `Iso-VertexMinor` is in `NP`. \square

Acknowledgements

AD, JH and SW were supported by an ERC Starting grant, and NWO VIDI grant, and Zwaartekracht QSC.

References

- [1] K. Wagner, *Über eine eigenschaft der ebenen komplexe*, [Mathematische Annalen](#) **114**, 570 (1937).
- [2] J. Matoušek and R. Thomas, *On the complexity of finding iso- and other morphisms for partial k -trees*, [Discrete Mathematics](#) **108**, 343 (1992).
- [3] N. Robertson and P. D. Seymour, *Graph minors. I - XXIII. Journal of Combinatorial Theory, Series B* **35**, 39 (1983-2010).
- [4] S. I. Oum, *Rank-width and vertex-minors*, [Journal of Combinatorial Theory. Series B](#) **95**, 79 (2005).
- [5] J. Jeong, O. Kwon, and S. Oum, *Excluded vertex-minors for graphs of linear rank-width at most k* , [European Journal of Combinatorics](#) **41**, 242 (2014).
- [6] O. Kwon and S. Oum, *Graphs of small rank-width are pivot-minors of graphs of small tree-width*, [Discrete Applied Mathematics](#) **168**, 108 (2014).
- [7] J. Geelen and S. Oum, *Circle graph obstructions under pivoting*, [Journal of Graph Theory](#) **61**, 1 (2009).
- [8] A. Dahlberg and S. Wehner, *Transforming graph states using single-qubit operations*, [Phil. Trans. R. Soc. A](#) **376**, One contribution of 15 to a discussion meeting issue 'Foundations of quantum mechanics and their impact on contemporary society' (2018), [10.1098/rsta.2017.0325](#).
- [9] F. Hahn, A. Pappa, and J. Eisert, *Quantum network routing and local complementation*, [npj Quantum Information](#) **5**, 76 (2019).
- [10] M. Mhalla and S. Perdrix, *Graph states, pivot minor, and universality of (x, z) -measurements*, arXiv preprint [arXiv:1202.6551](#) (2012).
- [11] R. Duncan and S. Perdrix, *Pivoting makes the zx -calculus complete for real stabilizers*, arXiv preprint [arXiv:1307.7048](#) (2013).
- [12] M. Van den Nest, J. Dehaene, and B. De Moor, *Efficient algorithm to recognize the local clifford equivalence of graph states*, [Physical Review A](#) **70**, 034302 (2004), [arXiv:0405023 \[quant-ph\]](#) .

- [13] M. Van den Nest, J. Dehaene, and B. De Moor, *Graphical description of the action of local clifford transformations on graph states*, [Physical Review A](#) **69**, 022316 (2004).
- [14] S. Oum, *Rank-width: Algorithmic and structural results*, [Discrete Applied Mathematics](#) **231**, 15 (2017), algorithmic Graph Theory on the Adriatic Coast.
- [15] K. K. Dabrowski, F. Dross, J. Jeong, M. M. Kanté, O. Kwon, S. Oum, and D. Paulusma, *Computing small pivot-minors*, in *Graph-Theoretic Concepts in Computer Science*, edited by A. Brandstädt, E. Köhler, and K. Meer (Springer International Publishing, Cham, 2018) pp. 125–138.
- [16] R. M. Karp, *Reducibility among combinatorial problems*, in *Complexity of computer computations* (Springer, 1972) pp. 85–103.
- [17] M. Fleury, *Deux problemes de Geometrie de sitation*, *Journal de mathematiques elementaires* **2nd**, 257 (1883).

8

Transforming graph states to Bell-pairs is NP-Complete

Axel Dahlberg, Jonas Helsen, Stephanie Wehner

Critical to the construction of large scale quantum networks, i.e. a quantum internet, is the development of fast algorithms for managing entanglement present in the network. One fundamental building block for a quantum internet is the distribution of Bell pairs between distant nodes in the network. Here we focus on the problem of transforming multipartite entangled states into the tensor product of bipartite Bell pairs between specific nodes using only a certain class of local operations and classical communication. In particular we study the problem of deciding whether a given graph state, and in general a stabilizer state, can be transformed into a set of Bell pairs on specific vertices using only single-qubit Clifford operations, single-qubit Pauli measurements and classical communication. We prove that this problem is NP-Complete.

8.1. Introduction

Entanglement takes center stage in the modern understanding of quantum mechanics. Apart from its usefulness as a theoretical tool, entanglement can also be seen as a resource that can be harnessed for secure communication and many other tasks, see e.g. [1], not achievable by any protocol using only classical communication. One can imagine a network of quantum-enabled nodes, a quantum internet, generating entanglement and harnessing it to perform tasks. However, entanglement over large distances is very difficult to produce and even with rapidly improving technology, the amount of entanglement available in a network will for the foreseeable future be the limiting factor when performing the tasks mentioned above. Moreover entanglement comes in different classes that are not necessarily mutually inter-convertible by operations performed locally on the nodes. These two considerations: (1) the scarcity of entanglement as a resource and (2) the lack of local inter-convertibility of classes of entanglement sets the stage for the present work. In this chapter we assume that we already have some existing shared entangled state in a quantum network and we ask the question of whether this state can be transformed into a set of Bell pairs between specific nodes, using only a restricted set of local operations. Examples of such a situation can be found in [2, 3], where an approach is presented of first probabilistically generating a large graph state and then transforming this to the desired target state using local operations. In [2] this target state is precisely a set of Bell pairs between multiple pairs of nodes.

Any decision on this transformation process must be made fast, since entanglement decays over time [4]. Hence there is a need for fast algorithms to decide whether different entangled states can be converted into each other by local operations. In [5], measurement-based quantum network coding was introduced, where one step in the procedure includes transforming general graph states into Bell-pairs using single-qubit Clifford operations, single-qubit Pauli measurement and classical communication: LC + LPM + CC. However, the computational complexity of finding the correct operations or even deciding if it can be done, was never mentioned. In this work we answer this question. Specifically we investigate the computational complexity of the question of whether a given graph state can be decomposed into a series of Bell pairs on specific vertices, using only LC + LPM + CC. We consider this set of operations since they are on many hardware platforms the simplest and fastest operations to perform. For example on Nitrogen-Vacancies in diamond, single-qubit gates on one of the qubits (electron) is many orders faster than two-qubits gates and single-qubit gates on the other qubits (nuclear spins) [6]. Furthermore, restricting to only Cliffords, instead of allowing general unitary operations, is natural on systems where, for example, the qubits are logical qubits of some error-correcting code. Since in this case, doing non-Clifford operations is a costly process involving, in many cases, the consumption of magic states. Moreover, over the years many techniques from graph theory have been put to bear on specifically the behavior of graph states under LC + LPM + CC [7–12]. This gives us the ability to ask very precise questions and also answer them.

For discussion on previous work we refer to section 4.1.1.

Our main result is that we prove that the problem of deciding whether a given graph state can be converted into Bell Pairs using only LC + LPM + CC (BellVM) is in general NP-Complete. This means that unless $\mathbb{P} = \text{NP}$, there will be no efficient algorithm for solving this problem on arbitrary graph states. In order to prove our results we make heavy use of results in algorithmic graph theory, and in fact we prove new graph theoretical results in the process of proving our main theorem.

Overview

In section 8.2 we phrase some of the concepts introduced in the previous sections with focus on transforming graph states to tensor product of Bell pairs. In section 8.3 we discuss the Edge-Disjoint path problem, a well-studied computational problem in algorithmic graph theory. In section 8.4 we show that a certain version of the Edge Disjoint Path problem can be polynomially reduced to BellVM and in appendix D.1 that this version of the Edge Disjoint Path problem is NP-Complete, implying that BellVM is NP-hard. Finally, in section 8.5 we discuss the implications of our result.

8.2. Bell vertex-minors

In this chapter we are interested in the question of whether a given graph state $|G\rangle$ can be transformed into some number of Bell pairs using LC + LPM + CC between specific vertices. We will denote the following Bell pair on qubits a and b as

$$|\Phi^+\rangle_{ab} = \frac{1}{\sqrt{2}}(|0\rangle_a \otimes |0\rangle_b + |1\rangle_a \otimes |1\rangle_b). \quad (8.1)$$

We formally define the following main problem of this chapter.

Problem 8.2.1 (BellQM). Given a graph state $|G\rangle$ and a set of disjoint pairs $B = \{\{p_1, p'_1\}, \dots, \{p_k, p'_k\}\}$. Let $|G_B\rangle$ be the state following state consisting of Bell pairs between each pair of B

$$|G_B\rangle = \bigotimes_{\{p,p'\} \in B} |\Phi^+\rangle_{pp'} \quad (8.2)$$

Decide if $|G_B\rangle$ is a qubit-minor of G . \diamond

The graph state described by the complete graph on two vertices K_2 is single-qubit Clifford equivalent to each of the four Bell pairs since

$$|K_2\rangle = \frac{1}{\sqrt{2}}(|0\rangle_a \otimes |+\rangle_b + |1\rangle_a \otimes |-\rangle_b) = H_b |\Phi^+\rangle_{ab} \quad (8.3)$$

where $|+\rangle = (|0\rangle + |1\rangle)/\sqrt{2}$, H_b is a Hadamard gate on qubit b .

Using theorem 4.4.2 we can turn the question of transforming graph states to Bell pairs using LC + LPM + CC i.e. BellQM , into the question of whether a disjoint union of K_2 's is a vertex-minor of some graph. Formally we have the following problem graph problem.

Problem 8.2.2 (BellVM). Given a graph G and a set of disjoint pairs $B = \{\{p_1, p'_1\}, \dots, \{p_k, p'_k\}\}$. Let G_B be the graph with vertices $V(G_B) = \cup_{i \in [k]} \{p_i, p'_i\}$ and edges $E(G_B) = \cup_{i \in [k]} \{(p_i, p'_i)\}$. Decide if G_B is a vertex-minor of G . \diamond

To be precise (G, B) is a YES-instance of BellQM , if and only if (G, B) is a YES-instance of BellVM .

In section 4.7 we described how vertex-minors of circle graphs can be characterized by certain Eulerian tours on 4-regular multi-graphs. Using theorem 4.7.14, we can now ask what property a 4-regular multi-graph should have, such that it's induced alternance graphs are YES-instances to BellVM , given a set of disjoint pairs P . As we show in lemma 8.4.2, this question will be directly related to a restricted version of the edge-disjoint path problem, which we define in the next section.

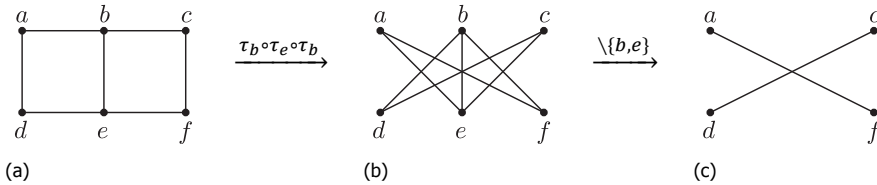


Figure 8.1: An example of a graph being transformed to a union of two K_2 graphs, using local completions and vertex-deletions. The original graph in fig. 8.1a get transformed to the graph in fig. 8.1b by performing local completions on the vertices b, e and then b again. Finally, when vertices b and e are deleted, the disjoint union of two K_2 graphs, on vertices $\{a, f\}$ and $\{c, d\}$ respectively, is reached (fig. 8.1c).

8.3. The Edge-disjoint path problem

In section 8.4 we show that BellVM is NP-Complete by reducing the 4-regular EDPDT (Edge Disjoint Paths with Disjoint Terminals) problem (see below) to BellVM . We then show that that 4-regular EDPDT is NP-Complete, see corollary 8.3.4.1, which therefore implies that the same is true for BellVM . This is done by reducing the EDP (Edge Disjoint Path) problem to EDPDT and then EDPDT to 4-regular EDPDT . We begin by formally defining all problems just mentioned. We will denote the set of edges in a path $P = v_0 e_1 v_1 \dots e_l v_l$ as $E(P) = \{e_1, \dots, e_l\}$. Moreover, given two graphs $G = (V(G), E(G))$ and $D = (V(D), E(D))$ we denote by $G \cup D$ the graph formed by the vertices $V(G \cup D) = V(G) \cup V(D)$ and the edge multi-set $E(G \cup D) = E(G) \cup E(D)$. A path on a graph is a walk without repeated vertices. A closed path is called a circuit.

We first define the EDP (Edge Disjoint Path) problem.

Problem 8.3.1 (EDP). Let G and D be graphs such that $V(D) \subseteq V(G)$. Decide whether there exists a set of edge-disjoint circuits \mathcal{C} on the graph $G \cup D$ such that every edge $e \in E(D)$ is part of exactly one circuit in \mathcal{C} . \diamond

Let (G, D) be a YES-instance of EDP and let \mathcal{C} be the edge-disjoint circuits of problem 8.3.1. For each edge e in $E(D)$ denote the circuit in \mathcal{C} which e is part

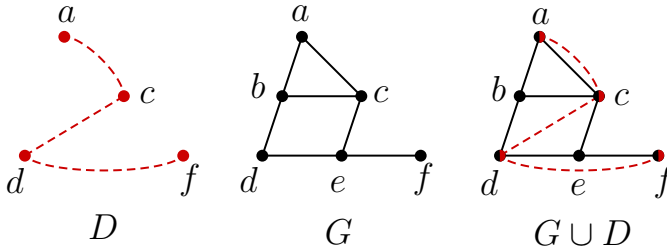


Figure 8.2: The EDP problem concerns deciding if there exists a set of edge-disjoint circuits in $G \cup D$ such that each edge in D is in part of exactly one of the circuits. In this example a satisfying solution is a set consisting of the three circuits $a(a, c)c(c, a)a$, $d(d, b)b(b, c)c(c, d)d$ and $d(d, e)e(e, f)f(f, d)d$.

of as C_e . These edge disjoint circuits C_e on $G \cup H$ correspond to edge disjoint paths on the graph G with *terminals*, i.e. beginning and ending points of the path, precisely at the vertices u, v s.t. $e = (u, v)$ for all $e \in D$. The EDP problem is known to be NP-Complete even in the case where $G \cup D$ is a Eulerian graph [13]. The EDPDT problem is now the EDP problem but with the demand graph D restricted to be a disjoint union of connected graphs on two vertices, i.e. of the form $K_2^{\times k}$ for some k , such that the terminals are distinct.

Problem 8.3.2 (EDPDT). Let G and $D = K_2^{\times k}$ be graphs such that $V(D) \subseteq V(G)$. Decide EDP with the instance (G, D) . \diamond

The 4-regular EDPDT problem is then a further restriction of this problem to the case where $G \cup D$ is 4-regular (and $D = K_2^{\times k}$). Formally we have

Problem 8.3.3 (4-regular EDPDT). Let G and $D = K_2^{\times k}$ be graphs such that $V(D) \subseteq V(G)$ and $G \cup D$ is 4-regular. Decide EDP with the instance (G, D) . \diamond

Note that the 4-regularity of $G \cup D$ means that all vertices in G which are not in D must have degree 4, while all vertices also in D must have degree 3. We can equivalently formulate the 4-regular EDPDT problem as a problem involving only G and a set of terminal pairs on G , which will be a more useful definition for some of the proofs.

Problem 8.3.4 (4-regular EDPDT (equivalent formulation)). Let G be a multi-graph where each vertex has degree either 3 or 4. Let $T = \{\{t_1, t'_1\}, \dots, \{t_k, t'_k\}\}$ be the set of disjoint terminal pairs, such that $t_i, t'_i \in V(G)$ and $d_G(t_i) = d_G(t'_i) = 3$ for all $i \in [k]$. Furthermore, assume that there are no other vertices of degree 3 in G , i.e. $\bigcup_{i \in [k]} \{t_i, t'_i\} = \{v \in V(G) : d_G(v) = 3\}$. Decide if there exists k edge-disjoint paths P_i for $i \in [k]$ such that the ends of P_i are t_i and t'_i . \diamond

In the appendix (theorem D.1.3) we show that EDP can be reduced to 4-regular EDPDT in polynomial time. A corollary to this theorem is therefore.

Corollary 8.3.4.1. 4-regular EDPDT is NP-Complete. \diamond

8.4. BellVM is NP-Complete

Now we move on to proving the main result of this chapter. The 4-regular EDPDT problem can be reduced to the BellVM problem as shown below in theorem 8.4.1. Since we also show that BellVM is in NP we conclude that BellVM is NP-Complete.

Corollary 8.4.0.1. BellVM is NP-Complete. \diamond

Proof. Theorem 8.4.1 states that there exists a Karp reduction from 4-regular EDPDT to BellVM . This implies that BellVM is NP-hard, by corollary 8.3.4.1. Since any instance of BellVM is also an instance of the general vertex-minor problem, which is in NP, see chapter 6, also BellVM is in NP and hence NP-Complete. \square

As a directly corollary we then also have that.

Corollary 8.4.0.2. BellQM is NP-Complete. \diamond

Proof. Follows directly from corollary 8.4.0.1 and theorem 4.4.2. \square

The remaining part of this section will be used to prove theorem 8.4.1. The main part of the proof consists of proving lemma 8.4.2 which provides an explicit reduction from 4-regular EDPDT to BellVM . This is done by constructing a 4-regular multi-graph $H_{(G,T)}$ from an instance (G,T) of 4-regular EDPDT , together with the graph $G_B = K_2^{\times |T|}$ on a certain subset B of the vertices of $H_{(G,T)}$. The construction is such that G_B is a vertex-minor of any alternance graph $\mathcal{A}(U)$ induced by an Eulerian tour U on $H_{(G,T)}$ if and only if (G,T) is a YES-instance of 4-regular EDPDT . In other words, $(\mathcal{A}(U), B)$ is a YES-instance of BellVM if and only if (G,T) is a YES-instance of 4-regular EDPDT . What is left to show is that reduction can be done in polynomial time, which is done in the proof of theorem 8.4.1 below.

Theorem 8.4.1. The 4-regular EDPDT problem is polynomially reducible to BellVM . \diamond

Proof. From lemma 8.4.2 below we see that any yes(no)-instance of 4-regular EDPDT can be mapped to a yes(no)-instance of BellVM . What remains to be shown is that this mapping can be performed in polynomial time. The reduction consist of the following to three steps:

1. Construct the multi-graph $H_{(G,T)}$ as defined in lemma 8.4.2.
2. Find an Eulerian tour U on $H_{(G,T)}$.
3. Construct the alternance graph $\mathcal{A}(U)$ induced by U .

Computing the graph $H_{(G,T)}$ can be done in polynomial time by simply adding the vertices and edges described in lemma 8.4.2. Note that the number of vertices and vertices in $H_{(G,T)}$ is $|V(G)| + 2 \cdot |T|$ and $|E(G)| + 4 \cdot |T|$. Finding an Eulerian tour U on $H_{(G,T)}$ can be done in polynomial time [14] in the size of $H_{(G,T)}$. Furthermore, constructing the alternance graph $\mathcal{A}(U)$ can be done in polynomial time as shown in [15]. \square

Lemma 8.4.2. Let $(G, T = \{\{t_1, t'_1\}, \dots, \{t_k, t'_k\}\})$ be an instance of 4-regular ED-PDT (see problem 8.3.4). As illustrated in fig. 8.3a, let $H_{(G,T)}$ be the multi-graph constructed from G by adding the distinct vertices $V_B = \cup_{i \in [k]} \{p_i, p'_i\}$ and the edges (t_i, p_i) , (t'_i, p'_i) , (p_i, p'_i) and¹ (p_i, p'_i) for $i \in [k]$ and the edges (p'_i, p_{i+1}) for $i \in [k - 1]$ and the edge (p'_k, p_1) . Let G_B be the graph with vertices V_B and edges $\cup_{i \in [k]} \{(p_i, p'_i)\}$. Let $\mathcal{A}(U)$ be a circle graph described by the Eulerian tour U on $H_{(G,T)}$. Then G_B is a vertex-minor of $\mathcal{A}(U)$ if and only if (G, T) is a yes-instance of 4-regular EDPDT. \diamond

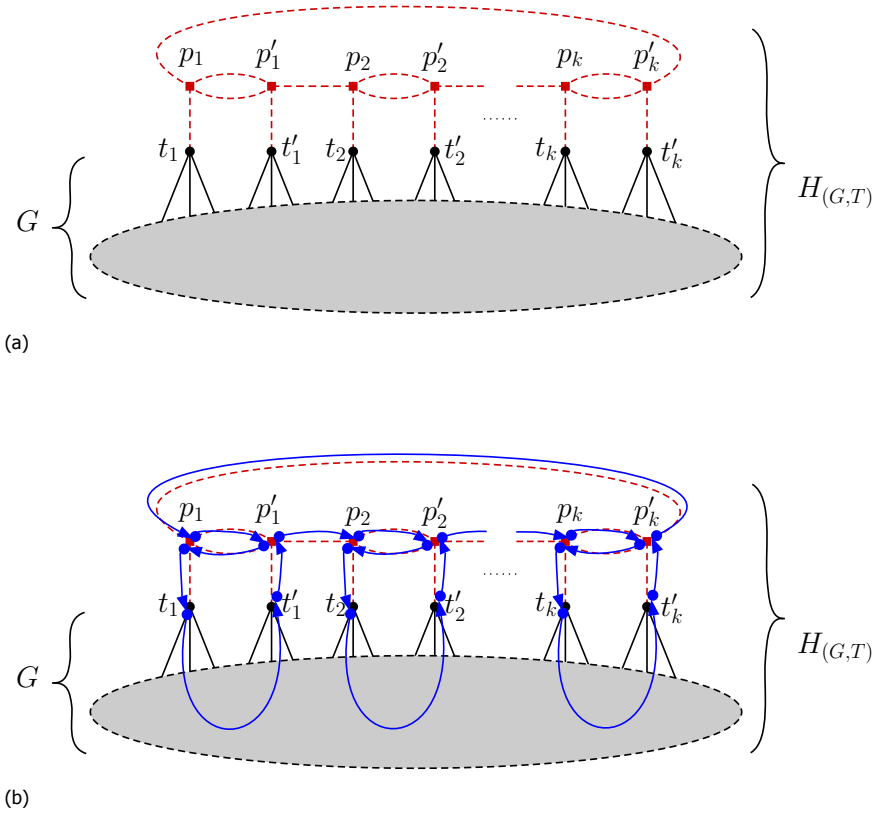


Figure 8.3: Figure 8.3a shows the graph construction used to reduce 4-regular EDPDT to BellVM. The graph G is a graph with only vertices of degree 4 except the vertices $\{t_1, t'_1, t_2, t'_2, \dots, t_k, t'_k\}$ which have degree 3. The vertices of G not in $\{t_1, t'_1, t_2, t'_2, \dots, t_k, t'_k\}$ are visualized as a grey solid area in order to show that we put no further restrictions on G . $H_{(G,T)}$ is constructed from G by adding the red square vertices and red dashed edges. In fig. 8.3b the tour U_0 described by the word in eq. (8.4) is illustrated using solid blue arrows.

¹Note that (p_i, p'_i) is added twice.

Proof. Let's first assume that (G, T) is a YES-instance of 4-regular EDPDT . This implies that there exist k edge-disjoint paths P_i for $i \in [k]$ such that the ends of P_i are t_i and t'_i . These paths also exist in $H_{(G, T)}$, since G is a subgraph of $H_{(G, T)}$. Consider then the tour U' , see fig. 8.3b, described by the word

$$m(U') = p_1 p'_1 p_1 m(P_1) p'_1 p_2 \dots p'_{k-1} p_k p'_k p_k m(P_k) p'_k. \quad (8.4)$$

Note U' is not necessarily an Eulerian tour but can be extended to one using the efficient Hierholzer's algorithm [16]. Denote the Eulerian tour obtained from U' by U . Consider now the induced subgraph $\mathcal{A}(U)[V_B]$. Using eq. (4.66) we know that this induced subgraph is the same as the alternance graph given by the induced word

$$m(U)[V_B] = p_1 p'_1 p_1 p'_1 p_2 \dots p'_{k-1} p_k p'_k p_k p'_k. \quad (8.5)$$

We then have that $\mathcal{A}(m(U)[V_B])$, and therefore $\mathcal{A}(U)[V_B]$, is the graph G_B . To see this, note that the only alternances of $m(U)[V_B]$ are $p_i p'_i p_i p'_i$ for $i \in [k]$ and the edges of $\mathcal{A}(m(U)[V_B])$ are therefore (p_i, p'_i) for $i \in [k]$. Since $\mathcal{A}(U)[V_B]$ is an induced subgraph of $\mathcal{A}(U)$ it is by definition also a vertex-minor of $\mathcal{A}(U)$. As we saw above, G_B is equal to $\mathcal{A}(U)[V_B]$ and is therefore also a vertex-minor of $\mathcal{A}(U)$.

Let's now instead assume that (G, T) is a NO-instance of 4-regular EDPDT . We will show that G_B is not a vertex-minor of $\mathcal{A}(U)$. Let U be an Eulerian tour on $H_{(G, T)}$, which exist since $H_{(G, T)}$ is 4-regular. Consider the sub-trails of U for which both ends are in V_B and that use no edges between vertices in V_B . Note that there are k such sub-trails since each vertex in V_B only has one edge which does not go to another vertex in V_B . Let's denote these sub-trails as \tilde{P}_i for $i \in [k]$ and their ends as $\{p_i, \tilde{p}'_i\}$. From the assumption that (G, T) is a *no*-instance of 4-regular EDPDT , we know that $\{p_i, \tilde{p}'_i\} \neq \{p_i, p'_i\}$ for at least one $i \in [k]$. Consider now the graph $\mathcal{A}(U)[V_B]$. We will now show that $\mathcal{A}(U)[V_B]$ cannot be G_B . Since U was an arbitrary Eulerian tour on $H_{(G, T)}$, we then know that G_B is not a vertex-minor of U , by theorem 4.7.14.

To show that $\mathcal{A}(U)[V_B]$ is not G_B , consider the induced subgraph $H_{(G, T)}[V_B]$. Note that all vertices in $H_{(G, T)}[V_B]$ have degree 3, see fig. 8.3a. Let $H_{(U, B)}$ be the graph obtained from $H_{(G, T)}[V_B]$ by adding the edges (p_i, \tilde{p}'_i) . Note now that $m(U)[V_B]$ also describes an Eulerian tour on $H_{(U, B)}$. Thus, $\mathcal{A}(U)[V_B]$ is the alternance graph $\mathcal{A}(U')$ for some Eulerian tour U' on $H_{(U, B)}$. Therefore, by using lemma 8.4.3 we know that $\mathcal{A}(U)[V_B]$ is not G_B . \square

Lemma 8.4.3. *Let B be the set $\{p_1, p'_1, \dots, p_k, p'_k\}$. Let C_B be the cycle graph on vertices $\cup_{i \in [k]} \{p_i, p'_i\}$ and with edges $\{(p_1, p'_1), (p'_1, p_2), \dots, (p_k, p'_k), (p'_k, p_1)\}$. Let \tilde{C}_B be the multi-graph obtained from C_B by duplicating the edges $\{(p_i, p'_i)\}_{i \in [k]}$. Let $\tilde{E}_B = \{(p_i, \tilde{p}'_i)\}_{i \in [k]}$ be edges such that if they are added to \tilde{C}_B , the graph obtained is 4-regular. Denote the graph obtained from \tilde{C}_B by adding the edges \tilde{E}_B by $H_{(B, \tilde{E}_B)}$. Let G_B be the graph $(B, \cup_{i \in [k]} \{(p_i, p'_i)\})$. Then for any Eulerian tour U on $H_{(B, \tilde{E}_B)}$, the graph $\mathcal{A}(U)$ is equal to G_B if and only if $\{p_i, \tilde{p}'_i\} = \{p_i, p'_i\}$ for all $i \in [k]$. \diamond*

Proof. First note that if $\mathcal{A}(U) = G_B$ for some Eulerian tour U on $H_{(B, \tilde{E}_B)}$ then $\mathcal{A}(U') = G_B$ for any Eulerian tour U' on $H_{(B, \tilde{E}_B)}$. This is because the graph G_B is invariant under local complementations.

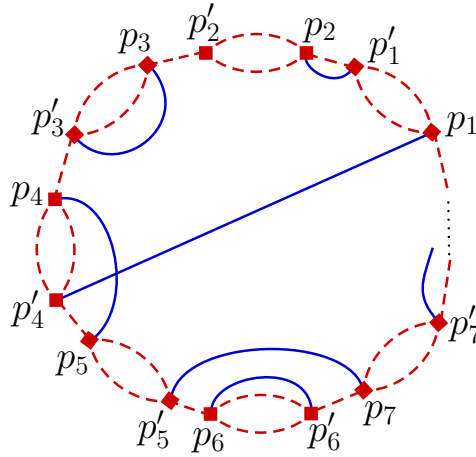


Figure 8.4: The graph $H_{(B, \tilde{E}_B)}$ used in lemma 8.4.3, where \tilde{E}_B are the solid blue edges. Lemma 8.4.3 states that there exist an Eulerian tour U on $H_{(B, \tilde{E}_B)}$ such that $\mathcal{A}(U) = G_B$ if and only if all the solid blue edges \tilde{E}_B are between vertices p_i and p'_i for $i \in [k]$.

Assume now that $\{p_i, \tilde{p}'_i\} = \{p_i, p'_i\}$ for all $i \in [k]$. Then the tour U described by the double occurrence word

$$m(U) = p_1 p'_1 p_1 p'_1 p_2 \dots p'_{k-1} p_k p'_k p_k p'_k. \tag{8.6}$$

is an Eulerian tour on $H_{(B, \tilde{E}_B)}$. Furthermore, $\mathcal{A}(U) = G_B$.

Assume now on the other hand that $\{p_i, \tilde{p}'_i\} \neq \{p_i, p'_i\}$ for some $i \in [k]$. Let (p_i, \tilde{p}'_i) be a pair for which $\{p_i, \tilde{p}'_i\} \neq \{p_i, p'_i\}$. Consider now the tour

$$U' = p_i (p_i, \tilde{p}'_i) P_C \tag{8.7}$$

on $H_{(B, \tilde{E}_B)}$ where P_C is a path in C_B which ends at p_i , i.e. $E(P_C) \cap \tilde{E}_B = \emptyset$. By Hierholzer's algorithm let U be an Eulerian tour on $H_{(B, \tilde{E}_B)}$ obtained by extending U' . Note now that (p_i, \tilde{p}'_i) is an alternance in $m(U)$ and therefore an edge in $\mathcal{A}(U)$. But since (p_i, \tilde{p}'_i) is not an edge in G_B , we know that $\mathcal{A}(U)$ is not equal to G_B . \square

8.5. Conclusion

The problem of transforming graph states to Bell-pairs using local operations is a problem with direct applications to the development of quantum networks or distributed quantum processors. Solutions to special cases of this problem have been considered in for example [2] and [5]. However, at least to our knowledge, the computational complexity of this problem was previously unknown. Here we show that deciding whether a given graph state $|G\rangle$ can be transformed into a set of Bell pairs on a given set of vertices, using only single-qubit Clifford operations, single-qubit Pauli measurements and classical communication is NP-Complete. In fact, we show that the problem remains NP-Complete if G is a circle graph.

Acknowledgements

AD, JH and SW were supported by an ERC Starting grant, and NWO VIDI grant, and Zwaartekracht QSC.

References

- [1] S. Wehner, D. Elkouss, and R. Hanson, *Quantum internet: A vision for the road ahead*, *Science* **362**, eaam9288 (2018).
- [2] M. Pant, H. Krovi, D. Towsley, L. Tassiulas, L. Jiang, P. Basu, D. Englund, and S. Guha, *Routing entanglement in the quantum internet*, *npj Quantum Information* **5**, 25 (2019).
- [3] M. Pant, D. Towsley, D. Englund, and S. Guha, *Percolation thresholds for photonic quantum computing*, *Nature Communications* **10**, 1070 (2019).
- [4] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*, 10th ed. (Cambridge University Press, Cambridge, 2010).
- [5] T. Matsuo, T. Satoh, S. Nagayama, and R. Van Meter, *Analysis of measurement-based quantum network coding over repeater networks under noisy conditions*, *Physical Review A* **97**, 13 (2017).
- [6] N. Kalb, A. A. Reiserer, P. C. Humphreys, J. J. W. Bakermans, S. J. Kamerling, N. H. Nickerson, S. C. Benjamin, D. J. Twitchen, M. Markham, and R. Hanson, *Entanglement distillation between solid-state quantum network nodes*, *Science* **356**, 928 (2017), arXiv:1703.03244 .
- [7] M. Hein, W. Dür, J. Eisert, R. Raussendorf, M. V. den Nest, H. J. Briegel, M. V. den Nest, and H. J. Briegel, *Entanglement in Graph States and its Applications*, *Quantum Computers, Algorithms and Chaos* **162**, 1 (2006), arXiv:0602096 [quant-ph] .
- [8] M. Van den Nest, J. Dehaene, and B. De Moor, *Graphical description of the action of local clifford transformations on graph states*, *Physical Review A* **69**, 022316 (2004).
- [9] A. Dahlberg and S. Wehner, *Transforming graph states using single-qubit operations*, *Phil. Trans. R. Soc. A* **376**, One contribution of 15 to a discussion meeting issue 'Foundations of quantum mechanics and their impact on contemporary society' (2018), 10.1098/rsta.2017.0325.
- [10] A. Dahlberg, J. Helsen, and S. Wehner, *How to transform graph states using single-qubit operations: computational complexity and algorithms*, *Quantum Science and Technology* (2020).
- [11] F. Hahn, A. Pappa, and J. Eisert, *Quantum network routing and local complementation*, *npj Quantum Information* **5**, 76 (2019).

- [12] A. Dahlberg, J. Helsen, and S. Wehner, *The complexity of the vertex-minor problem*, arXiv preprint arXiv:1906.05689 (2019).
- [13] J. Vygen, *Np-completeness of some edge-disjoint paths problems*, *Discrete Applied Mathematics* **61**, 83 (1995).
- [14] M. Fleury, *Deux problemes de Geometrie de sitation*, *Journal de mathematiques elementaires* **2nd**, 257 (1883).
- [15] A. Bouchet, *Circle Graph Obstructions*, *Journal of Combinatorial Theory, Series B* **60**, 107 (1994).
- [16] S. Even, *Graph Algorithms*, 2nd ed. (Cambridge University Press, 2011).

9

Counting single-qubit Clifford equivalent graph states is $\#\mathbb{P}$ -Complete

Axel Dahlberg, Jonas Helsen, Stephanie Wehner

Graph states, which include Bell states, Greenberger-Horne-Zeilinger (GHZ) states, and cluster states, form a well-known class of quantum states with applications ranging from quantum networks to error-correction. Whether two graph states are equivalent up to single-qubit Clifford operations is known to be decidable in polynomial time and have been studied both in the context of producing certain required states in a quantum network but also in relation to stabilizer codes. The reason for the latter is that single-qubit Clifford equivalent graph states exactly correspond to equivalent stabilizer codes. We here consider the computational complexity of, given a graph state $|G\rangle$, counting the number of graph states, single-qubit Clifford equivalent to $|G\rangle$. We show that this problem is $\#\mathbb{P}$ -Complete. To prove our main result we make use of the notion of isotropic systems in graph theory. We review the definition of isotropic systems and point out their strong relation to graph states. We believe that these isotropic systems can be useful beyond the results presented in this chapter.

Parts of this chapter have been published in Journal of Mathematical Physics [1].

9.1. Introduction

Graph states form a well-studied class of quantum states and are applied in many fields such as in quantum networks and quantum computers. In a quantum network, graph states are a resource used by applications such as secret sharing [2], anonymous transfer [3] and others. In a quantum computer, graph states are the logical codewords of many quantum error-correcting codes [4] and form a universal resource for measurement-based quantum computing [5]. The action of single-qubit Clifford operations on graph states is well-understood and can be characterized completely in terms of operations called *local complementations*, acting on the corresponding graph [6]. When faced with a class of objects and an action on them it is natural to consider the orbits induced by this action. The orbit of graph states under single-qubit Clifford operations can be studied by considering the orbits of simple graphs under local complementations, through the mapping mentioned above. The orbits of graph states have been studied in for example [7]. There the motivation came from quantum error correction: graph states can be mapped to stabilizer codes and moreover, the number of orbits for a given number of qubits is equal to the number of *equivalent* stabilizer codes. This gives a method to count the number of inequivalent stabilizer codes on a fixed number of qubits. In [7] the number of inequivalent stabilizer codes is computed for up to 12 qubits by counting the number of orbits of graphs under local complementations.

Furthermore, when studying entanglement measures and the equivalence of quantum states under local operations, the orbits of graphs states under single-qubit Cliffords is naturally an important question. In the excellent survey on graph states [8] it is stated that the computational complexity of generating the orbit of a given graph states is unknown. Here we show that given a graph G , counting the number of graph states equivalent to $|G\rangle$ under single-qubit Clifford operations, i.e. deciding the size of the orbit, is $\#\mathbb{P}$ -Complete. $\#\mathbb{P}$ -Complete problems are of great interest in the field of quantum computing. The reason being that the problem of boson-sampling [9], efficiently solvable using a quantum computer, has very strong similarities with the $\#\mathbb{P}$ -Complete problem of computing the permanent of a matrix [10].

From previous chapters we know that the action of single-qubit Clifford operations on graph states can be completely described by the action of local complementations on the corresponding graphs. This fact allows us to instead focus on the problem of counting the number of graphs equivalent under local complementations, also called *locally equivalent* graphs.

We point out that, since the property of whether a graph is locally equivalent to another is also expressible in MS, see chapter 5, Courcelle's machinery can also be applied to this problem. In fact, Courcelle's theorem also holds for counting the number of satisfying solutions [11], which is what we are interested in here. The details for how to apply Courcelle's theorem to the problem at hand, we leave for another thesis. Here, we instead show that the problem is $\#\mathbb{P}$ -Complete, and thus has no efficient algorithm in the general case, unless $\mathbb{P} = \mathbb{NP}$.

Furthermore, we review the concept of isotropic systems which captures equivalence classes of graphs under local complementations, introduced by Bouchet

in [12]. We focus on how these concept can be used when studying equivalence of graph states

Overview

In section 9.2 we show that graph states are equal if and only if there corresponding graphs are equal and that there therefore is a bijective mapping from simple graphs to graph states. In section 9.3 we review the graph theoretical notion of an isotropic system and relate this to stabilizer and graph states. In section 9.4 we review the complexity class $\#\mathbb{P}$ -Complete. In section 9.5 we prove our main result that counting the number of graph states equivalent under single-qubit Cliffords is $\#\mathbb{P}$ -Complete.

Notation

We use the same notation introduced in section 4.2 and in particular the notion of a *neighborhood* of a vertex u in a graph $G = (V, E)$ as

$$N_G(v) = \{u \in V : (u, v) \in E\}. \tag{9.1}$$

Furthermore, given a subset $X \subseteq V$ we use the following notation for the symmetric difference of the neighborhoods of the vertices in X

$$N_G(X) = \bigtriangleup_{v \in X} N_G(v). \tag{9.2}$$

9.2. Graph states

In chapter 4 we discussed graph states which form a subset of stabilizer states. Importantly here is that the above in fact gives a bijective mapping from graphs to graph states. Formally we have the following theorem.

Lemma 9.2.1. *Two graphs states $|G\rangle$ and $|G'\rangle$ are equal if and only if their corresponding graphs G and G' are equal.* \diamond

Proof. Let $|G\rangle$ and $|G'\rangle$ be two graph states. If G and G' have differing vertex-sets then clearly $|G\rangle$ and $|G'\rangle$ are different since they are states on different sets of qubits. Assume now that G and G' are graphs with the same vertex-set V . The inner product between $|G\rangle$ and $|G'\rangle$ will then be given as¹

$$\langle G|G'\rangle = \left(\bigotimes_{v \in V} \langle +|_v \right) \prod_{(u,v) \in E(G)} C_Z^{(u,v)} \prod_{(u,v) \in E(G')} C_Z^{(u,v)} \left(\bigotimes_{v \in V} |+\rangle_v \right). \tag{9.3}$$

Using the fact that $C_Z^{(u,v)}$ commute and square to identity for any (u, v) we find that the above equation evaluates to

$$\langle G|G'\rangle = \left(\bigotimes_{v \in V} \langle +|_v \right) |G + G'\rangle \tag{9.4}$$

¹We assume here that when iterating over a set, the order of the elements is always the same.

where $G + G'$ is the graph with vertex-set V and edge-set $E(G) \Delta E(G')$ with Δ being the symmetric difference. The state $|G + G'\rangle$ is equal to $\bigotimes_{v \in V} |+\rangle_v$ if and only if $G + G'$ is the empty graph. One can see this by for example considering the Schmidt-rank for a bipartition which separates some adjacent vertices in $G + G'$, since this would be one for $\bigotimes_{v \in V} |+\rangle_v$ and greater than one for $|G + G'\rangle$. We therefore have that $\langle G | G' \rangle$ is one if and only if G and G' are equal. \square

We have seen in chapter 4 that single-qubit Clifford operations on graph states can be completely described by local complementations on graphs. As a direct corollary of lemma 9.2.1 and theorem 4.4.4, we therefore have the following result.

Corollary 9.2.1.1. *Let G be a graph with its corresponding graph state $|G\rangle$. The number of graph states which are single-qubit Clifford equivalent to $|G\rangle$ is equal to the number of locally equivalent graphs to G .* \diamond

Using corollary 9.2.1.1 we can now restrict ourselves to the problem of counting locally equivalent graphs.

9.3. Isotropic systems

Our main result of this chapter makes great use of the concept of an *isotropic system*. In this section we review the definition of an isotropic system and its relation to locally equivalent graphs and graph states. What is interesting to point out, and perhaps never before noted, is that an isotropic system is in fact equivalent to a stabilizer group, see below. For this reason, results obtained for isotropic systems can be of great use when studying stabilizer states and graph states.

Isotropic systems were introduced by Bouchet in [12]. The power of isotropic systems is that they exactly capture the equivalence classes of graphs under local complementation or equivalently equivalence classes of graphs states under single-qubit Clifford operations. Any isotropic system has a set of *fundamental graphs* which are all locally equivalent. As shown in [13], any graph G is a fundamental graph of some isotropic system S . Furthermore, given a isotropic system S with a fundamental graph G , another graph G' is a fundamental graph of S if and only if G and G' are locally equivalent².

In section 9.3.2 we review the formal definition of an isotropic system. In the sections leading up to this, we first set the notation and introduce certain concepts needed.

9.3.1. Finite fields and Pauli groups

Let $\{0, 1, \omega, \omega^2\}$ be the elements of the finite field of four elements \mathbb{F}_4 . Under addition we have that $x + x = 0$ for any element x of \mathbb{F}_4 and furthermore we have that $1 + \omega = \omega^2$. Under multiplication we have that $x^i \cdot x^j = x^{i+j \pmod{3}}$ for any element $x \neq 0$. An useful inner product on \mathbb{F}_4 is the trace inner product, defined as

$$\langle a, b \rangle = a \cdot b^2 + a^2 \cdot b \tag{9.5}$$

²That is, if and only if $|G\rangle$ and $|G'\rangle$ are equivalent under single-qubit Clifford operations.

What is interesting in relation to quantum information theory is that addition in \mathbb{F}_4 corresponds to matrix multiplication in the Pauli group, up to a global phase. Furthermore, the trace inner product captures whether two elements of the Pauli group commute or not. To see this, consider the following mapping α from \mathbb{F}_4 to the Pauli group.

$$\alpha(0) = I, \alpha(1) = X, \alpha(\omega) = Y, \alpha(\omega^2) = Z. \tag{9.6}$$

One can then check that

$$\alpha(a)\alpha(b) = i^k \alpha(a + b) \quad \text{where } k \in \mathbb{Z}_4. \tag{9.7}$$

Furthermore, we have that

$$[\alpha(a), \alpha(b)] = 0 \iff \langle a, b \rangle = 0. \tag{9.8}$$

where $[\cdot, \cdot]$ is the commutator. Similarly we can also define a map from the elements of the vector space \mathbb{F}_4^n . Let \mathbf{v} be a vector of \mathbb{F}_4^n and \mathbf{v}^i be the i 'th element of \mathbf{v} . We then define a map from \mathbb{F}_4^n to the Pauli group on n qubits as follows.

$$\alpha(\mathbf{v}) = \bigotimes_{i=0}^n \alpha(\mathbf{v}^i). \tag{9.9}$$

Both eq. (9.6) and eq. (9.8) hold for α also acting on vectors of \mathbb{F}_4^n .

9.3.2. Isotropic systems

Formally an isotropic system is defined as follows³.

Definition 9.3.1 (isotropic system). A subspace S of \mathbb{F}_4^n is said to be an isotropic system if: (1) for all $\mathbf{v}, \mathbf{w} \in S$ it holds that $\langle \mathbf{v}, \mathbf{w} \rangle = 0$ and (2) S has dimension n . \diamond

Now note that $\alpha(S) \equiv \{\alpha(\mathbf{v}) : \mathbf{v} \in S\}$ forms a stabilizer group (ignoring global phases). This is because condition (1) of the above definition says that all the elements of $\alpha(S)$ commute, by eq. (9.8), as required by a stabilizer group.

9.3.3. Complete and Eulerian vectors

Here we review some further concepts related to isotropic systems which we need for the proof of our main result. Certain isotropic systems can be represented as Eulerian tours on 4-regular multi-graphs (see section 9.3.5). These Eulerian tours correspond to what are called Eulerian vectors of the isotropic system. The definition of a Eulerian vector also generalizes to all isotropic systems, even those not representable as Eulerian vectors on 4-regular multi-graphs. In order to give the definition of an Eulerian vector we must first define what are called complete vectors.

³The definition here is equivalent to the original one in [12], however we use a slightly different notation than Bouchet used 30 years ago.

Definition 9.3.2 (complete vector). A vector \mathbf{v} of \mathbb{F}_4^n such that $\mathbf{v}^i \neq 0$ for all $i \in [n]$ is called *complete*. \diamond

We will also need to notion of supplementary vectors.

Definition 9.3.3 (supplementary vectors). Two vectors \mathbf{v}, \mathbf{w} of \mathbb{F}_4^n are called supplementary if (1) they are complete and (2) $\mathbf{v}^i \neq \mathbf{w}^i$ for all $i \in [n]$. \diamond

Complete vectors come equipped with a notion of *rank*. To define the rank of a complete vector we need some further notation. Let \mathbf{v} be a complete vector of \mathbb{F}_4^n . Let X be a subset of $[n]$ and let $\mathbf{v}[X]$ be a vector such its elements are

$$(\mathbf{v}[X])^i = \begin{cases} \mathbf{v}^i & \text{if } i \in X \\ 0 & \text{else} \end{cases} \quad (9.10)$$

We can now define the following set

$$V_{\mathbf{v}} = \{\mathbf{v}[X] : X \subseteq [n]\}. \quad (9.11)$$

Note that $V_{\mathbf{v}}$ forms a subspace of \mathbb{F}_4^n . The rank of \mathbf{v} with respect to S is now defined as the dimension of the intersection of $V_{\mathbf{v}}$ and S .

Definition 9.3.4 (rank of a complete vector). Let \mathbf{v} be a complete vector of \mathbb{F}_4^n . The rank of \mathbf{v} , $r_S(\mathbf{v})$, with respect to S is the dimension of the intersection of $V_{\mathbf{v}}$ and S , i.e.

$$r_S(\mathbf{v}) = \dim(V_{\mathbf{v}} \cap S) \quad (9.12)$$

We are now ready to formally define an Eulerian vector of an isotropic system.

Definition 9.3.5 (Eulerian vector). A complete vector \mathbf{v} of \mathbb{F}_4^n , such that $r_S(\mathbf{v}) = 0$ is called an Eulerian vector of S . \diamond

9.3.4. Fundamental graphs

As mentioned, the power of isotropic systems is that their *fundamental graphs* are exactly the graphs in an equivalence class under local complementations. Here we review the definition of fundamental graphs of an isotropic system, which is defined by a Eulerian vector through a *graphic description*.

Definition 9.3.6 (graphic presentation). Let G be a graph with vertices⁴ $V(G) = [n]$ and \mathbf{v}, \mathbf{w} be supplementary vectors of \mathbb{F}_4^n . The following is then an isotropic system

$$S = \{\mathbf{v}[N_X] + \mathbf{w}[X] : X \subseteq V(G)\}. \quad (9.13)$$

The tuple $(G, \mathbf{v}, \mathbf{w})$ is called a graphic presentation of the isotropic system S . Furthermore, G is called a fundamental graph of S . \diamond

⁴Note that we can always choose such a labeling of the vertices of G .

Note that

$$\{\mathbf{v}[N_v] + \mathbf{w}[\{v\}] : v \in V(G)\} \tag{9.14}$$

forms a basis for S . In [13] it is shown that if $(G, \mathbf{v}, \mathbf{w})$ is a graphic presentation of S then \mathbf{v} is a Eulerian vector of S . Furthermore, it is shown that given an Eulerian vector \mathbf{v} of S , there exists a unique graphic presentation $(G, \hat{\mathbf{v}}, \mathbf{w})$ of S such that $\mathbf{v} = \hat{\mathbf{v}}$. Note that two Eulerian vectors can still represent the same fundamental graph.

The observant reader will notice the close similarity between eq. (9.14) and eq. (4.12). Indeed, consider the two supplementary vectors $\mathbf{v}_{\omega^2} = (\omega^2, \dots, \omega^2)$ and $\mathbf{w}_1 = (1, \dots, 1)$. Now, let G be an arbitrary graph and S be the isotropic system with $(G, \mathbf{v}_{\omega^2}, \mathbf{w}_1)$ as a graph presentation, as by eq. (9.13). We will here call S_G the *canonical* isotropic system of G . We then have that $\alpha(S_G)$ is exactly the stabilizer group of the graph state $|G\rangle$. To see this note that

$$g_v = \alpha(\mathbf{v}[N_v] + \mathbf{w}[\{v\}]) \quad \forall v \in V(G) \tag{9.15}$$

using eq. (9.9) and eq. (4.12).

As mentioned before, any two graphs G and G' are locally equivalent if and only if they are fundamental graphs of the same isotropic system [13]. Furthermore, any graph is a fundamental graph of some isotropic system. We therefore see that, for any isotropic system S , there exists a surjective map from the set of Eulerian vectors of S to the graphs in an equivalence class of graphs under local completions. As described in the section 9.3.6, for certain isotropic systems, the number of Eulerian vectors equals the number of Eulerian tours on some 4-regular multi-graph. We will make use of this fact to prove our main result.

9.3.5. Graphic systems

Certain isotropic systems, called *graphic* systems, can be represented as a 4-regular multi-graphs. There is then a surjective map from the Eulerian tours on the 4-regular multi-graph to the fundamental graphs of the graphic system [13]. The set of fundamental graphs for graphic systems is exactly the set of circle graphs [13]. We will briefly describe this relation here, however leaving out some details which are out of scope for this chapter. For details on graphic systems see [13], for circle graphs see [14, 15] and it's relation to graph states see chapter 4.

In section 4.7 we recalled the definition of circle graphs, also called alternance graphs, and their relation to Eulerian tours on 4-regular graphs. It turns out that the set of alternating graphs induced by the Eulerian tours on some 4-regular multi-graph F are exactly the fundamental graphs of some isotropic system S . We then say that S is *associated* to F . An isotropic system that is associated to some 4-regular multi-graph is called *graphic*. There is a formal mapping λ from a 4-regular multi-graph F together with an ordering T of its edges to an isotropic system $S = \lambda_T(F)$. However, this mapping is rather complex and the interested reader can find the details in [13]. What is important here is that, for any T , there is a bijective mapping from the Eulerian tours of F to the Eulerian vectors of $S = \lambda_T(F)$ [13]. This statement is implied by the results developed in [13], however in a non-trivial way. For this reason, we here point out why this follows in section 9.3.6.

9.3.6. Eulerian decompositions

An Eulerian decomposition D of a 4-regular multi-graph F is a set of tours on F such that each edges of F is in exactly one of the tours. As shown in [13], given a 4-regular multi-graph on n vertices, any Eulerian decomposition can be describe by a complete vector of \mathbb{F}_4^n . To see this, note that an Eulerian decomposition on a 4-regular multi-graph F can be described by, for each vertex v in F , a pairing of the incident edges on v . For example, let e_v^1, e_v^2, e_v^3 and e_v^4 be the four edges incident on the vertex v and consider now a pairing where e_v^1 is paired with e_v^2 and e_v^3 with e_v^4 , written as $((e_v^1, e_v^2), (e_v^3, e_v^4))$. We can then construct an Eulerian decomposition by walking along the vertices and edges of F and when we reach v through the edge e_v^1 we should exit through the edge e_v^2 and vice versa. Note that there are exactly three different ways to pair the four edges of a vertex and we can thus represent this pairing by a non-zero element of \mathbb{F}_4 as

$$1 \mapsto ((e_v^1, e_v^2), (e_v^3, e_v^4)) \quad (9.16)$$

$$\omega \mapsto ((e_v^1, e_v^3), (e_v^2, e_v^4)) \quad (9.17)$$

$$\omega^2 \mapsto ((e_v^1, e_v^4), (e_v^2, e_v^3)). \quad (9.18)$$

Furthermore we can represent the pairings of all the vertices of F as a complete vector of \mathbb{F}_4^n . Note that the Eulerian decomposition for a given complete vector depends on the ordering of the edges incident on a vertex. However this ordering simply changes which Eulerian decomposition is related to which complete vector, but not the fact that we now have a mapping from complete vectors of \mathbb{F}_4^n to Eulerian decompositions of F . This ordering T is exactly the ordering mentioned in section 9.3.5, which can be used to map F to an isotropic system $S = \lambda_T(F)$. Let now $D_T(\mathbf{v})$ be the Eulerian decomposition induced by the complete vector \mathbf{v} as described above.

Importantly here, as stated in [13], is that, for any Eulerian decomposition D of F there is a unique complete vector $\mathbf{v} \in \mathbb{F}_4^n$ such that $D = D_T(\mathbf{v})$, for a fixed T . Furthermore, the Eulerian decomposition $D_T(\mathbf{v})$ consists of an Eulerian tour if and only if \mathbf{v} is an Eulerian vector of $S = \lambda_T(F)$. We therefore have the following corollary.

Corollary 9.3.1.1 (Implied by [13]). *Let F be a 4-regular multi-graph with n vertices. Let T be an ordering of its vertices as described above and formally defined in [13]. The number of Eulerian tours on F equals the number of Eulerian vectors of $S = \lambda_T(F)$.* \diamond

Proof. From above we already know that a Eulerian decomposition of F is described by exactly one complete vector of \mathbb{F}_4^n through the mapping D_T . Furthermore, the Eulerian decomposition $D_T(\mathbf{v})$ consists of exactly one Eulerian tour if and only if \mathbf{v} is a Eulerian vector of $S = \lambda_T(F)$. Finally the number of Eulerian decompositions of F that consists of exactly one Eulerian tour are clearly equal to the number of Eulerian tours on F . \square

9.3.7. Number of locally equivalent graphs

In [16] Bouchet showed that $l(G)$, the number of graphs locally equivalent to some graph G , is given by

$$l(G) = \frac{e(S)}{k(S)} \tag{9.19}$$

where S is an isotropic system with G as a fundamental graph and $e(S)$ is the number of Eulerian vectors of S and $k(S)$ is an index of S . We also have that if S and S' are isotropic systems which both have G as a fundamental graph, then $e(S) = e(S')$ and $k(S) = k(S')$. Using the canonical isotropic system we introduced in section 9.3.4 we can therefore also define

$$e(G) \equiv e(S_G), \quad k(G) \equiv k(S_G), \tag{9.20}$$

such that

$$l(G) = \frac{e(G)}{k(G)}. \tag{9.21}$$

Below, we review the definition of $k(G)$ as presented in [16]. The index $k(G)$ of a graph is given as

$$k(G) = \begin{cases} |v(G)^\perp| + 2 & \text{if } G \text{ is in the class } \mu \\ |v(G)^\perp| & \text{else} \end{cases} \tag{9.22}$$

where the bineighborhood space $v(G)$ and the graph class μ are defined below and $^\perp$ denotes the orthogonal complement. Firstly, we introduce the following notation that will help simplify some later expressions.

Definition 9.3.2. Let $S = \{s_1, \dots, s_k\}$ be a set and $P \subseteq S$ a subset of S . We will associate to P a binary vector \vec{P} of length k as follows:

$$\vec{P}^{(i)} = \begin{cases} 1 & \text{if } s_i \in P \\ 0 & \text{else} \end{cases} \tag{9.23}$$

where $\vec{P}^{(i)}$ is the i -th element of \vec{P} . We denote the number of nonzero elements of \vec{P} as $|\vec{P}|$, such that $|\vec{P}| = |P|$. ◊

Here, the base-set S will here be the vertices V of a graph G and from the context it will always be clear which graph. We will also use \cdot to denote the element-wise product between two binary vectors, such that

$$\overrightarrow{P_1 \cap P_2} = \vec{P}_1 \cdot \vec{P}_2. \tag{9.24}$$

To define the graph class μ we first need to review the notion of a bineighborhood space.

Definition 9.3.3 (bineighborhood space). Let $G = (V, E)$ be a simple graph and $\overline{G} = (V, \overline{E})$ the complementary graph of G . For any $u, v \in V$ let

$$v_G(e) = \overline{N_G(u) \cap N_G(v)}. \tag{9.25}$$

For any subset $E' \subseteq E \cup \bar{E}$, let

$$v_G(E') = \sum_{e \in E'} v_G(e). \tag{9.26}$$

We will sometimes write $v(e)$ or $v(E')$ if it is clear which graph is considered. A subset $C \subseteq E$ such that the number of edges in C incident to any vertex in G is even is called a cycle. We denote the set of cycles of G as $\mathcal{C}(G)$. Let $\mathfrak{B} = \mathbb{Z}_2^{|V|}$ be the binary vector space of dimensions $|V|$ and consider the two subspaces

$$\bar{\mathfrak{C}} = \{v(E') : E' \subseteq \bar{E}\}, \quad \mathfrak{C} = \{v(C) : C \subseteq \mathcal{C}(G)\} \tag{9.27}$$

The bineighborhood space $v(G)$ is defined as the sum of the two subspaces $\bar{\mathfrak{C}}$ and \mathfrak{C} , i.e.

$$v(G) = \bar{\mathfrak{C}} + \mathfrak{C}. \tag{9.28}$$

◊

Finally the graph class μ is defined as follows.

Definition 9.3.4 (graph class μ). A simple graph $G = (V, E)$ is said to be in the class μ if:

1. $d_G(v) = 1 \pmod{2}$ for every vertex $v \in V$. I.e. all vertices in G should have an odd degree.
2. $|v(e)| = 0 \pmod{2}$ for all edges $e \in \bar{E}$. I.e. for every edge (u,v) , not in G , the symmetric difference of the neighborhoods of u and v should have an even size.
3. $|v(C)| = |C| \pmod{2}$ for all cycles $C \in \mathcal{C}(G)$. I.e., for all cycles C of G , the number of non-zero elements of the $v(C)$ and the number of edges of C should both be even or both be odd.

◊

9

9.4. Complexity

The problems in \mathbb{NP} are decision problems where YES-instances to the problem have proofs that can be checked in polynomial time. For example the SAT-problem is in \mathbb{NP} , where one is asked to decide if a given boolean formula has a satisfying assignment of its variables [17]. On the other hand, problems where the NO-instances have proofs that can be checked in polynomial time are the problems in $\text{co-}\mathbb{NP}$. A problem is said to be \mathbb{NP} -Complete if (1) it is in \mathbb{NP} and (2) any other problem in \mathbb{NP} can be reduced to this problem in polynomial time. \mathbb{NP} -Complete problems are therefore informally the hardest problem in \mathbb{NP} .

$\#\mathbb{P}$ problems are the *counting* versions of the \mathbb{NP} problems. For example, the *counting* version of SAT ($\#\text{SAT}$) is to compute how many satisfying assignments a given boolean formula has. $\#\mathbb{P}$ -Complete problems are the problems in $\#\mathbb{P}$ for

which any other problem in #P can be polynomially reduced to. For example #SAT is #P-Complete [10]. Note that #P-Complete is at least as hard as NP-Complete, since if we know the number of satisfying assignments we know if at least one exists. Other well-known problems #P-Complete are for example computing the permanent of a given boolean matrix or finding how many perfect matchings a given bipartite graph has [10].

Recently, #P-Complete problems have been the interest of the quantum computing community due to the problem of boson sampling [9]. The boson sampling problem can be solved efficiently on a quantum computer. Furthermore, the boson sampling problem can be related to the problem of estimating the permanent of a complex matrix. Since computing the permanent is in general a #P-Complete problem and thus believed to be infeasible to solve efficiently on a classical computer, the boson sampling problem is therefore a strong candidate for a problem showing 'quantum supremacy'.

9.5. Counting the number of locally equivalent graphs is #P-Complete

Here we show our following main result.

Theorem 9.5.1 (main). *Counting the number, $l(G)$, of locally equivalent graphs to a given graph G is #P-Complete.* \diamond

We do this by showing that counting the number of Eulerian tours of a 4-regular multi-graph can be reduced in polynomial time to computing $l(G)$, where G is a circle graph. Since counting the number of Eulerian tours of a 4-regular multi-graph is #P-Complete [18], the result follows. By corollary 9.2.1.1 we have the following corollary.

Corollary 9.5.1.1. *Counting the number of graph states which are single-qubit Clifford equivalent to a given graph state $|G\rangle$ is #P-Complete.* \diamond

Proof. Directly implied by theorem 9.5.1 and corollary 9.2.1.1. \square

9.5.1. Reducing # of Eulerian tours to # of local equivalent graphs

Here we show how the problem of computing the number of Eulerian tours on a 4-regular multi-graph can be reduced in polynomial time to the problem of computing the number of locally equivalent graphs to some circle graph and thus provide the proof for theorem 9.5.1.

Proof of theorem 9.5.1. From corollary 9.3.1.1 we know that for any 4-regular multi-graph F , there exists an isotropic system $S = \lambda_T(F)$ such that the number of Eulerian vectors $e(S)$ equals the number of Eulerian tours on F . Let now G be a fundamental graph of S . We then have that $e(G) = e(S)$, by eq. (9.20) and [13]. Furthermore, recall the G is necessarily also an alternance graph induced by some Eulerian tour on F , see section 9.3.5. We can therefore compute the number of Eulerian tours

on F by computing $l(G) \cdot k(G)$, as by eq. (9.21). As we show below, we can both find G and compute $k(G)$ in polynomial time from which the theorem follows.

We can find G in polynomial time as follows.

1. Find an Eulerian tour U on F , can be done in polynomial time by Fleury's algorithm [19].
2. Construct the alternance graph $G = \mathcal{A}(U)$ induced by U , can be done in polynomial time, see chapter 6.

In the rest of this section we show that $k(G)$ can be computed in polynomial time from which the main result follows. We will start by showing that determining if a graph G is in the class μ , see definition 9.3.4, can be done in time $\mathcal{O}(|V|^5)$. Note that there might be even faster ways to compute this, but we are here only interested to show that this can be done in polynomial time. We assume that the graph $G = (V, E)$ is represented by its adjacency matrix. To check if G is in the class μ one needs to check the three properties in definition 9.3.4:

1. Checking if all vertices have odd degree can be done in $\mathcal{O}(|V|^2)$ time.
2. Checking if $|v(e)|$ is even for all edges can be done in $\mathcal{O}(|V|^3)$ time, since there are $\mathcal{O}(|V|^2)$ edges and computing $|v(e)|$ can be done in linear time⁵.
3. For the last property is not directly clear whether this can be done in polynomial time since we need to a priori check the property $|v(C)| = |C| \pmod{2}$ for all cycles in G , which might be exponentially many. As we will now show, we only need to check the property for the cycles in a cycle basis of G . A cycle basis $\mathcal{CB} = \{C_1, \dots, C_k\}$, where $k = \mathcal{O}(|V|^2)$, is a set of cycles such that any cycle of G can be written as the symmetric difference of the elements of a subset of \mathcal{CB} . As shown in [20] a cycle basis of an undirected graph can be found in $\mathcal{O}(|V|^2)$ time. Thus any cycle of G can be written as

$$\bigtriangleup_{C' \in \mathcal{C}} C' \tag{9.29}$$

where \mathcal{C} is a subset of the cycle basis \mathcal{CB} . We then have that

$$v \left(\bigtriangleup_{C' \in \mathcal{C}} C' \right) = \sum_{C' \in \mathcal{C}} v(C'). \tag{9.30}$$

Thus we need to show that for any $\mathcal{C} \subseteq \mathcal{CB}$

$$\left| \sum_{C' \in \mathcal{C}} v(C') \right| = \left| \bigtriangleup_{C' \in \mathcal{C}} C' \right| \pmod{2} \quad \forall \mathcal{C} \subseteq \mathcal{CB} \tag{9.31}$$

⁵By taking the inner product of the corresponding rows in the adjacency matrix.

if and only if

$$|\nu(C)| = |C| \pmod{2} \quad \forall C \in \mathcal{CB}. \tag{9.32}$$

Lets first show that eq. (9.31) implies eq. (9.32). Equation (9.31) states that the equation holds for every subset \mathcal{C} of the elements of the cycle basis \mathcal{CB} . In particular it should hold for the singletons $\mathcal{C} = \{C\}$, where $C \in \mathcal{CB}$. Note that this directly implies eq. (9.32). For the rest of this section we now prove that eq. (9.32) implies eq. (9.31). We will do this by induction on the size of \mathcal{C} . This is obviously true if $|\mathcal{C}| = 1$. Lets therefore assume that the statement is true for $|\mathcal{C}| \leq k$ which we will show implies that it is also true for $|\mathcal{C}| = k + 1$. Lets assume that \mathcal{C} is a subset of \mathcal{CB} of size $k + 1$ and that \tilde{C} is an element of \mathcal{C} . Lets then consider the left-hand side of eq. (9.31)

$$\left| \sum_{C' \in \mathcal{C}} \nu(C') \right| = \left| \sum_{C' \in \mathcal{C} \setminus \{\tilde{C}\}} \nu(C') + \nu(\tilde{C}) \right|. \tag{9.33}$$

We will now make use of the fact that the size of the symmetric difference of two sets S_1 and S_2 is $|S_1 \Delta S_2| = |S_1| + |S_2| - 2|S_1 \cap S_2|$. Expressed in terms of binary vectors this relation reads $|\vec{S}_1 + \vec{S}_2| = |\vec{S}_1| + |\vec{S}_2| - 2|\vec{S}_1 \cdot \vec{S}_2|$. We therefore have that eq. (9.33) evaluates to

$$\left| \sum_{C' \in \mathcal{C} \setminus \{\tilde{C}\}} \nu(C') + \nu(\tilde{C}) \right| = \left| \sum_{C' \in \mathcal{C} \setminus \{\tilde{C}\}} \nu(C') \right| + |\nu(\tilde{C})| - 2 \left| \left(\sum_{C' \in \mathcal{C} \setminus \{\tilde{C}\}} \nu(C') \right) \cdot \nu(\tilde{C}) \right| \tag{9.34}$$

Wr can then see that eq. (9.32) implies eq. (9.31) since when taking $\pmod{2}$, the last term in the above expression vanishes and the two first evaluate to

$$\left| \Delta_{C' \in \mathcal{C} \setminus \{\tilde{C}\}} C' \right| + |\tilde{C}| \tag{9.35}$$

where we used the induction hypothesis. By a similar argument one can see that the expression in eq. (9.35) equals $\pmod{2}$

$$\left| \Delta_{C' \in \mathcal{C}} C' \right|. \tag{9.36}$$

Thus the total time to check property 3 in definition 9.3.4 is $\mathcal{O}(|V|^5)$. To see this, note that we need to check $|\nu(C)| = |C| \pmod{2}$ for all $C \in \mathcal{CB}$, which contains $\mathcal{O}(V^2)$ elements. To compute $|\nu(C)|$, we compute $\nu(e)$, in linear time, for each of the $\mathcal{O}(V^2)$ elements of \mathcal{C} , and add these together, also in linear time.

In addition to deciding if the graph G is in the class μ , we also need to compute $|\nu(G)^\perp|$ to determine $k(G)$. This can be done by first finding bases for the subspaces $\overline{\mathcal{C}}$ and \mathcal{C} . For $\overline{\mathcal{C}}$ a basis can be found as $\{\overline{\{e\}} : e \in \overline{E}\}$. As stated above we can also find a basis for \mathcal{C} , i.e. the cycle basis, in $\mathcal{O}(|V|^2)$ time. From the bases for $\overline{\mathcal{C}}$ and \mathcal{C} we can find a basis for $\nu(G)$ in $\mathcal{O}(|V|^3)$ time, by Gaussian elimination. The number of basis vectors we found for $\nu(G)$ is then the dimension of $\nu(G)$. From the dimension of $\nu(G)$ we can find the dimension of $\nu(G)^\perp$ as

$$\dim(\nu(G)^\top) = |V| - \dim(\nu(G)) \quad (9.37)$$

and finally the size of $\nu(G)^\perp$ as

$$|\nu(G)^\perp| = 2^{\dim(\nu(G)^\perp)}. \quad (9.38)$$

Thus there exist an algorithm to compute $k(G)$ with running time $\mathcal{O}(|V|^5)$. This then implies that computing the number of Eulerian tours in a 4-regular multi-graph can be reduced in polynomial time to computing the number of locally equivalent graphs to some circle graph, by using eq. (9.21), and therefore theorem 9.5.1. \square

9.6. Conclusion

We have shown that counting the number of graph states equivalent under single-qubit Clifford operations is $\#\mathbb{P}$ -Complete. To do this we have made heavy use of certain concepts in graph theory, mainly developed by Bouchet. As it turns out these concepts, for example isotropic systems, are highly relevant for the study of stabilizer and graph states. We hope that this chapter can serve as not only a proof of our main theorem but also as a reference for those in quantum information theory interested in finding use for these graph theory concepts in their research.

Acknowledgements

AD, JH and SW were supported by an ERC Starting grant, an NWO VIDI grant, and the Zwaartekracht QSC.

References

- [1] A. Dahlberg, J. Helsen, and S. Wehner, *Counting single-qubit clifford equivalent graph states is $\#p$ -complete*, *Journal of Mathematical Physics* **61**, 022202 (2020), <https://doi.org/10.1063/1.5120591> .
- [2] D. Markham and B. C. Sanders, *Graph states for quantum secret sharing*, *Physical Review A - Atomic, Molecular, and Optical Physics* **78**, 042309 (2008), [arXiv:0808.1532](https://arxiv.org/abs/0808.1532) .
- [3] M. Christandl and S. Wehner, *Quantum anonymous transmissions*, in *Advances in Cryptology - ASIACRYPT 2005*, edited by B. Roy (Springer Berlin Heidelberg, Berlin, Heidelberg, 2005) pp. 217–235.

- [4] D. Gottesman, *PhD Theses, Ph.D. thesis*, California Institute of Technology (2004), [arXiv:9705052 \[quant-ph\]](#) .
- [5] R. Raussendorf and H. J. Briegel, *A one-way quantum computer*, [Physical Review Letters](#) **86**, 5188 (2001), [arXiv:0510135v1 \[quant-ph\]](#) .
- [6] M. Van den Nest, J. Dehaene, and B. De Moor, *Graphical description of the action of local clifford transformations on graph states*, [Physical Review A](#) **69**, 022316 (2004).
- [7] L. E. Danielsen and M. G. Parker, *On the classification of all self-dual additive codes over $gf(4)$ of length up to 12*, [Journal of Combinatorial Theory, Series A](#) **113**, 1351 (2006).
- [8] M. Hein, W. Dür, J. Eisert, R. Raussendorf, M. V. den Nest, H. J. Briegel, M. V. den Nest, and H. J. Briegel, *Entanglement in Graph States and its Applications, Quantum Computers, Algorithms and Chaos* **162**, 1 (2006), [arXiv:0602096 \[quant-ph\]](#) .
- [9] S. Aaronson and A. Arkhipov, *The computational complexity of linear optics*, [Theory of Computing](#) **9**, 143 (2013).
- [10] L. Valiant, *The complexity of computing the permanent*, [Theoretical Computer Science](#) **8**, 189 (1979).
- [11] B. Courcelle and J. Engelfriet, *Graph Structure and Monadic Second-Order Logic: A Language Theoretic Approach*, 1st ed. (Cambridge University Press, New York, NY, USA, 2011).
- [12] A. Bouchet, *Isotropic systems*, [European Journal of Combinatorics](#) **8**, 231 (1987).
- [13] A. Bouchet, *Graphic presentations of isotropic systems*, [Journal of Combinatorial Theory, Series B](#) **45**, 58 (1988).
- [14] A. Bouchet, *Circle Graph Obstructions*, [Journal of Combinatorial Theory, Series B](#) **60**, 107 (1994).
- [15] M. C. Golumbic, *Algorithmic graph theory and perfect graphs*, 2nd ed. (North-Holland Publishing Co., 2004).
- [16] A. Bouchet, *Recognizing locally equivalent graphs*, [Discrete Mathematics](#) **114**, 75 (1993).
- [17] S. A. Cook, *The complexity of theorem-proving procedures*, in [Proceedings of the Third Annual ACM Symposium on Theory of Computing](#), STOC '71 (ACM, New York, NY, USA, 1971) pp. 151–158.
- [18] Q. Ge and D. Stefankovic, *The Complexity of Counting Eulerian Tours in 4-Regular Graphs*, in [LATIN 2010: Theoretical Informatics](#) (Springer Berlin Heidelberg, Berlin, Heidelberg, 2010) pp. 638–649.

- [19] M. Fleury, *Deux problemes de Geometrie de situation*, Journal de mathematiques elementaires **2nd**, 257 (1883).
- [20] K. Paton, *An algorithm for finding a fundamental set of cycles of a graph*, [Communications of the ACM](#) **12**, 514 (1969).

10

Conclusion

This chapter summarizes the thesis and discusses future research directions building on this work.

10.1. Summary of results

The focus of this thesis has been to enable arbitrary applications in a quantum internet and in particular to tackle the question on how to generate entanglement, in an **efficient**, **robust** and **scalable** manner. In the first half of the thesis, advancements towards a software stack for a quantum internet has been made, summarized by the following results:

- We presented a functional allocation of a network stack for a quantum internet. Furthermore, we developed a protocol providing the proposed service of the link layer. The performance of this link layer protocol was benchmarked through rigorous simulations.
- We created a simulator, SimulaQron, which is intended to be used to develop software for a quantum internet.

In the second half of the thesis we explored how graph states, a certain class of entangled quantum states, can be transformed in a quantum network, arriving at the following results:

- We showed how Courcelle's theorem can be applied to problems concerning transforming graph states in a quantum network.
- We showed that deciding if a GHZ-state, on a given subset of qubits, can be reached from a given graph state, using only single-qubit Clifford operations, single-qubit Pauli measurements and classical communication is NP-Complete.
- We showed that there exists an efficient algorithm that solves the same problem when certain restrictions are met which can be of practical interest for a quantum network.
- We show that deciding if a GHZ-state, on any subset of qubits of a certain size, can be reached from a given graph state, using only single-qubit Clifford operations, single-qubit Pauli measurements and classical communication is also NP-Complete. This result also shows that a related problem in graph theory concerning vertex-minors is also NP-Complete, solving an open question in this field.
- We showed that deciding if a given tensor-product of Bell pairs can be reached from a given graph state, using only single-qubit Clifford operations, single-qubit Pauli measurements and classical communication is NP-Complete.
- We showed that counting the number of graph states which are equivalent to a given graph state under single-qubit Clifford operations is #P-Complete.

10.2. Future work

In this section we list some possible future research directions that can extend the work done in this thesis:

Scheduling of operations in a quantum network: Some quantum devices, such as for example those based on nitrogen-vacancies in diamond [1], cannot perform local operations and entanglement generation with remote nodes at the same time. There is thus a need for arbitration between local operations and networking operations. However, the networking operations, i.e. entanglement generation, needs to be synchronized with remote nodes. This creates a complicated distributed scheduling problem, which needs to be addressed in order to enable a quantum internet based on these devices.

Optimized compiler for NetQASM: In chapter 3 we introduced the interface CQC. In later work [2], not part of this thesis, we have reworked this interface to a low-level instruction set architecture which we now call NetQASM. There is currently a very simple compiler from a higher-level SDK, in Python, for NetQASM. However, this compiler does currently no optimization and only compiles an application at run-time. To be able to efficiently execute applications on real quantum hardware the compiler also need to optimize both quantum gates and classical logic. Since such optimization can be time-consuming this should furthermore be done ahead-of-time, rather than during runtime. This task is a whole research project on its own. Optimized compilers already exist for quantum computing. However, in this case the compilation and optimization need to also take care of the distributed action of entanglement generation and also closely integrate quantum operations with classical logic, which might include information communicated from a remote node.

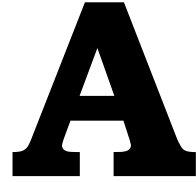
Local multi-qubit Clifford equivalence of graph states: In chapters 4 to 9 we have considered the transformation of graph states under single-qubit Clifford operations, single-qubit Pauli measurements and classical communication. These are generally the most simple operations and are therefore considered *cheap*. However, one can also envision a scenario where each node in a network has some set of qubits, and is able to perform arbitrary Clifford operations between its set of qubits. One would then have a graph representing the state, together with a partition of the vertices specifying between which qubits these operations are allowed. If Pauli measurements are allowed, the general problem of deciding if a graph state can be transformed to another is clearly NP-Complete, since the problem we discussed in 6 is then a special case. However, if only *local* multi-qubit Clifford operations are allowed, it is not so clear. What we know is that if each entry in the partition of the qubits has size 1, then the problem can be efficiently solved. The same holds if there is a single entry in the partition, containing all the qubits, since then all graph states are equivalent. The question is then if this problem remains efficiently solvable for any partition or if there is a middle ground where it becomes NP-Complete. As far as we are aware, this question is still open. An initial investigation was done in [3].

Does LULC hold for distance-hereditary (circle) graph states?: In chapters 4 to 9 we considered the transformation of graph states using single-qubit Clifford operations. It turns out that for some graph states, allowing

also for general single-qubit unitary operations does not increase the equivalence class. If this is the case, the class of graph states are said to satisfy the LULC property (local unitary operations, local Clifford operations). It is known that for example graph states described by trees and complete graphs satisfy the LULC property. The class of distance-hereditary graphs considered in previous chapter in this thesis is a super-class of both trees and complete graphs. It is then a natural question to ask if distance-hereditary graph states satisfy the LULC property, which is to our knowledge not known. If this would be the case, then some of the results presented here could also be applied to the more general question of transforming graph states under single-qubit *unitary* operations, single-qubit Pauli measurements and classical communication. The same question could then be asked also for circle graph states, since these are again a superclass of distance-hereditary graph states.

References

- [1] H. Bernien, *Control, measurement and entanglement of remote quantum spin registers in diamond*, [Phd thesis](#), TU Delft (2014).
- [2] A. Dahlberg, B. van der Vecht, C. Delle Donne, M. Skrzypczyk, W. Kozłowski, and S. Wehner, *Netqasm - a low-level instruction set architecture for hybrid quantum-classical programs in a quantum internet*, (2020), in preparation.
- [3] C. Molengraaf, *Local multi-qubit Clifford equivalence of graph states*, Master's thesis, Delft University of Technology (2019).



A Link Layer Protocol for Quantum Networks

**Axel Dahlberg, Matthew Skrzypczyk, Tim
Coopmans, Leon Wubben, Filip Rozpędek, Matteo
Pompili, Arian Stolk, Przemysław Pawełczak,
Robert Knegjens, Julio de Oliveira Filho, Ronald
Hanson, Stephanie Wehner**

In this Appendix, we provide further background and details, as well as more in-depth simulation results from chapter 2.

- In Section A.1, we explain how to estimate the fidelity by interspersing test rounds.
- In Section A.2, we provide further simulation results to illustrate protocol performance and further validate our simulation against hardware.
- In Section A.3, we provide further details of the NV platform as relevant for the design considerations of the proposed protocol. Furthermore, we provide details the physical modeling as well as numerical methods used to conduct the simulation.
- In Section A.4, we provide a complete description of the proposed protocol.

A.1. Testing

We now explain the test used to gain confidence in transmission quality, specifically to estimate the quality parameter fidelity F used in the FEU (section 2.5.2). The following is a standard procedure in quantum information to estimate the fidelity $F = F[|\Psi^-\rangle]$ of a state ρ to the entangled target state $|\Psi^-\rangle$. We emphasize that it is not possible to measure F from a single copy of the state ρ . The matrices ρ are a mathematical description of an underlying quantum system, and not a matrix that one can read or access like classical information.

We first describe the standard procedure in the way that it is normally used. We then outline how this protocol can be extended to the case of interest here, and why we can draw conclusions even in real world scenarios in which we can experience arbitrary correlated errors.

Let us first assume a simpler scenario, in which n identical noisy entangled states ρ are produced in succession and we want to estimate F . We remark that when using imperfect quantum devices it is evidently a highly idealized situation that all states ρ are exactly identical. We can see from Eq. (1.23) that we can express F in terms of the quantum bit error rates QBER_X , QBER_Z , and QBER_Y , which immediately suggests a protocol: specifically, we will estimate the QBERs in bases X , Z and Y to obtain F . We sketch such a protocol in a specific way to build intuition for the more general procedure below:

- Node A randomly chooses an n element string $r = r_1, \dots, r_n \in \{X, Z, Y\}$ and sends it to Node B .
- Nodes A and B now perform the following procedure for $1 \leq j \leq n$ rounds:
 - Node A produces one entangled pair ρ with Node B .
 - Nodes A and B both measure their respective qubits in the basis r_j and record outcomes x_j^A (Node A) and x_j^B (Node B) respectively.
- Node B (A) transmits the outcome string $x^B = x_1^B, \dots, x_n^B$ ($x^A = x_1^A, \dots, x_n^A$) to Node A (B).

- Both nodes estimate the error rates

$$QBER_Z \approx \frac{\#\{j \mid x_j^A = x_j^B, r_j = Z\}}{\#\{j \mid r_j = Z\}}, \quad (\text{A.1})$$

$$QBER_X \approx \frac{\#\{j \mid x_j^A = x_j^B, r_j = X\}}{\#\{j \mid r_j = X\}}, \quad (\text{A.2})$$

$$QBER_Y \approx \frac{\#\{j \mid x_j^A = x_j^B, r_j = Y\}}{\#\{j \mid r_j = Y\}}, \quad (\text{A.3})$$

$$(\text{A.4})$$

where $\#\{j \mid \text{condition}\}$ is the number of indices $1 \leq j \leq n$ satisfying the stated condition.

Using Eq. (1.23) then yields an estimate of F .

Before we continue it may be instructive to compare the procedure above to the classical world. Evidently, classically, one way to gain confidence in a channel's ability to transmit classical bits would be rather similar: Instead of preparing states ρ , we choose n random bits and send them. In the end, we estimate the error rate. Translated to the quantum setting, we would be preparing random bits $|0\rangle$ and $|1\rangle$ at node A , and sending them to node B which measures them in the Z basis to obtain an estimate of the bit error rate, similarly to $QBER_Z$. Such an estimate can give us confidence that also future bits are likely to be transmitted with roughly the same amount of errors as the test bits. This of course does *not* allow the same level of confidence as error detection in the quality of transmission. Specifically, a CRC is a check for a *specific* piece of data (e.g. one frame in 100 Base T), whereas such a test only yields a confidence in transmission quality.

Creating an analogous quantum CRC is theoretically possible by using a quantum error correcting code [2], but technologically highly challenging and highly infeasible for many years to come. Yet, we remark that also in a future in which such methods would become feasible we may not want to employ them because the requirements of our use cases are different. Since many protocols for our use cases are probabilistic, or make many pairs (especially NL and MD use cases), we often do not require more confidence on the exact quality of a single pair. Indeed, we can pass errors all the way up to the application level (such as for example in QKD [3]), where errors are then corrected using classical instead of quantum error correction. In such cases, fluctuations in quality are indeed expected at the application level. Here, using fast and easy to produce test rounds may remain preferable over more time consuming quantum CRCs.

The protocol above is limited in two ways: (1) all states were assumed to be both identical and independent from each other. I.e., there are no memory effects in the noise. Such memory effects are non-trivial in the quantum regime since they may inadvertently create (some amount of) entanglement not only between A and B , but between subsequent pairs produced. (2) We measured all n rounds, consuming all entangled pairs. Instead, we would like a protocol in which only test rounds are interspersed, and we can draw an inference about the pairs which

we did *not* measure. Again, the possibility of quantum correlated noise between subsequent rounds makes this non-trivial.

To achieve this, we use a slight variant of the above as in [4]. Precise statistical statements are relatively straightforward - but very lengthy - to obtain using the techniques in [4, 5] and are out of scope of this chapter. Here, we focus on the practical protocol and intuition without the need for mathematical tools from quantum information, which is a direct extension of the one above:

- Nodes A and B agree on a sampling window N .
- Nodes A and B randomly pick an N bit string $t = t_1, \dots, t_N$ where $\Pr[t_j = 1] = q$ for some parameter q determining the frequency of using test rounds. A and B periodically refresh t as needed.
- Nodes A and B randomly pick an N element basis string $r = r_1, \dots, r_N \in \{X, Z, Y\}$. A and B periodically refresh r as needed.
- The QEGP uses t to determine when to intersperse a test round. When producing the j -th response to the MHP, the QEGP checks whether $t_j = 1$. If so, it uses a standard test response instead to attempt to produce a test pair ρ , and takes as the measurement basis the next available in the random basis string r .
- A and B record their measurement outcomes.
- A and B estimate QBER_X , QBER_Z , QBER_Y over the past N rounds of producing entanglement (tested and untested 'data' rounds)

The key insight in the analysis of this procedure is that we can (with some amount of confidence depending on N and q) use the QBER measured on the test rounds to determine the QBER on the untested - i.e. data - rounds [5, Inequality 1.3]. Using Eq. (1.23), then again allows one to draw conclusions about the average fidelity of the untested rounds to inform the FEU.

A.2. Simulation and modeling

Here we provide additional simulation results, and further verification against the quantum hardware.

For our simulations we make use of a purpose built discrete event simulator for quantum networks: NetSquid¹. By utilizing the discrete event paradigm NetSquid is capable of efficiently simulating the transmission and decay of quantum information in combination with the complex and stochastic nature of communication protocols. NetSquid can simulate both arbitrary quantum operations and Clifford-only gates, the former allowing for a precise simulation of small networks, while the latter allowing for networks containing thousands of nodes and qubits to be studied. Complete libraries of base classes enable users to simulate protocols and model physical devices at different levels of abstraction; for instance, (quantum)

¹NetSquid is an acronym for Network Simulator for Quantum Information using Discrete events.

channels with modular noise, loss and delay models, or quantum processing devices with configurable gate topologies. NetSquid thus provides an ideal tool to validate network design choices and verify the performance of quantum network protocols in a physically-realistic setting.

The core simulation engine used by NetSquid is based on DynAA[6–8], a computer-aided analysis and design tool for the development of large, distributed, adaptive, and networked systems. It combines the best of network and system simulation technologies in a discrete-event modeling framework. DynAA provides a simple, yet powerful language able to describe large and complex system architectures, and innovative constructs to express adaptation mechanisms of the system, such as dynamic parameterization, and functional and architectural reconfiguration. A DynAA model can be simulated and/or analyzed to reveal system wide performance indicators, such as – but not limited to – throughput, power consumption, connectivity, reliability, and availability.

A.2.1. Validation of simulation

We compare our simulation model against further data gathered from the NV platform Labscenario. Node A rotates its qubit over the Z-axis of the Bloch sphere by a fixed angle, followed by measuring its communication qubit in a basis (X, Y or Z) that the nodes agreed upon beforehand. Node B only performs the measurement on its communication qubit, in the same pre-agreed basis. Regardless of the signal from the heralding station, both nodes initialize their qubit in $|0\rangle$ before the start of the next round.

We compute the correlations of the measurement outcomes ($m_A, m_B = \pm 1$) as shown in Figure A.1 using

$$\Pr(m_A \neq m_B) = \frac{1 - \langle B \otimes B \rangle}{2}$$

where $\langle B \otimes B \rangle$ is the expectation value of the product of joint measurement outcomes $m_A \cdot m_B$ with $B \in \{X, Y, Z\}$ the measurement basis after rotation.

The fidelity with the target state $|\Psi^\pm\rangle$ (where \pm denotes the heralding detector) can be expressed as a function of the correlations as

$$\frac{1}{4} [1 \pm \langle X \otimes X \rangle \pm \langle Y \otimes Y \rangle - \langle Z \otimes Z \rangle].$$

Assuming independence between the different rounds, propagation of standard deviations can be computed using standard techniques.

A.2.2. Simulation data

In this section we present further results from the simulations of our proposed link layer protocol. Simulation data is available at [11]. In total 1618 simulation runs were performed: 2×169 long runs (120 h wall time each) with 169 scenarios and 1280 shorter runs (24 h wall time each) with varying request load and minimal requested fidelity. Out of the 169 scenarios used in the long runs, $2 \times 5 \times 3 = 30$ concerned scenarios where the entanglement generation requests were a mix of

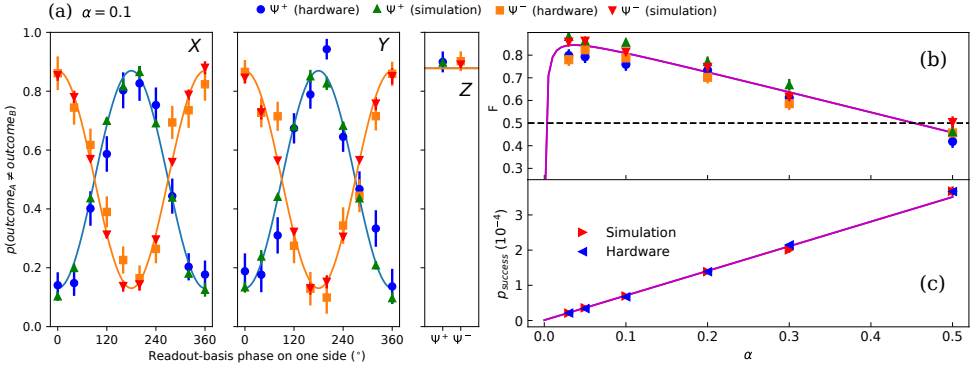


Figure A.1: Comparison of simulation results with data from NV hardware from [9] (Lab scenario), showing good agreement. (a) Probability of success that the two nodes' measurements in basis $X/Y/Z$ on the state after a one-sided Z -rotation are unequal, at $\alpha = 0.1$ and (b) fidelities, both computed from correlations in the measurement outcomes (see text). (c) Probability that a single generation attempt succeeds, which is computed as $1/\bar{N}$ where \bar{N} is the average number of runs up to and including successful heralding of entanglement. Solid line is the theoretical model from [9]. Error bars indicate 1 standard deviation. The simulation data was extracted by running our model implemented in NetSquid on the supercomputer *Cartesius* at SURFsara[10] for 122 hours of wall clock time using 63 cores. A single data point is the average over at least (a) 100 pairs, (b) 300 pairs and (c) 600 pairs, which took between 500k (for $\alpha = 0.5$) and 10.000k (for $\alpha = 0.03$) entanglement generation attempts, with elapsed simulated time between 5 and 117 seconds.

the priorities NL , CK and MD . For these mixed scenarios we considered (1) two physical setups, Lab and QL2020, (2) five usage patterns (described below) and (3) three different schedulers, FCFS, LowerWFQ and HigherWFQ.

We implement different usage patterns of the link layer by, in every MHP cycle, issuing a new CREATE request for a random number of pairs k (max k_{\max}) with probability $f \cdot p_{\text{succ}}/(E \cdot k)$, where p_{succ} is the probability of an attempt being successful, f is a fraction determining load of our system and E is the expected number of MHP cycles to make one attempt. For Lab(QL2020) $E = 1$ ($E = 1$) for M requests and $E \approx 1.1$ ($E \approx 16$) for K requests. We consider five different use patterns with f and k_{\max} defined in table A.1.

Table A.1

| Usage pattern | <i>NL</i> | <i>CK</i> | <i>MD</i> |
|---------------|------------------------------------|------------------------------------|--------------------------------------|
| Uniform | $f = 0.99 \cdot 1/3, k_{\max} = 1$ | $f = 0.99 \cdot 1/3, k_{\max} = 1$ | $f = 0.99 \cdot 1/3, k_{\max} = 1$ |
| MoreNL | $f = 0.99 \cdot 4/6, k_{\max} = 3$ | $f = 0.99 \cdot 1/6, k_{\max} = 3$ | $f = 0.99 \cdot 1/6, k_{\max} = 256$ |
| MoreCK | $f = 0.99 \cdot 1/6, k_{\max} = 3$ | $f = 0.99 \cdot 4/6, k_{\max} = 3$ | $f = 0.99 \cdot 1/6, k_{\max} = 256$ |
| MoreMD | $f = 0.99 \cdot 1/6, k_{\max} = 3$ | $f = 0.99 \cdot 1/6, k_{\max} = 3$ | $f = 0.99 \cdot 4/6, k_{\max} = 256$ |
| NoNLMoreCK | $f = 0, k_{\max} = 3$ | $f = 0.99 \cdot 4/5, k_{\max} = 3$ | $f = 0.99 \cdot 1/5, k_{\max} = 256$ |
| NoNLMoreMD | $f = 0, k_{\max} = 3$ | $f = 0.99 \cdot 1/5, k_{\max} = 3$ | $f = 0.99 \cdot 4/5, k_{\max} = 256$ |

Table A.2: Relative difference (Rel. Diff.) for the metrics: fidelity (Fid.), throughput (Throughp.) scaled latency (Laten.) and number of generated entangled pairs (Nr. pairs), between two identical scenarios except that the probability of losing a classical message (p_{loss}) is zero for one and between 10^{-10} and 10^{-4} for the other. The relative difference is maximized over three simulation runs (281 - 3973 s simulated time and 70 h wall time each) with requests of priority either *NL*, *CK* or *MD* ($f = 0.99, k_{\max} = 3$), with equal p_{loss} .

| p_{loss} | Max Rel. Diff. Fid. | Max Rel. Diff. Throughp. | Max Rel. Diff. Laten. | Max Rel. Diff. Nr pairs |
|-------------------|---------------------|--------------------------|-----------------------|-------------------------|
| 10^{-4} | 0.005 | 0.027 | 0.629 | 0.026 |
| 10^{-5} | 0.004 | 0.012 | 0.469 | 0.008 |
| 10^{-6} | 0.016 | 0.037 | 0.332 | 0.047 |
| 10^{-7} | 0.040 | 0.026 | 0.576 | 0.020 |
| 10^{-8} | 0.007 | 0.023 | 0.623 | 0.020 |
| 10^{-9} | 0.004 | 0.026 | 0.338 | 0.021 |
| 10^{-10} | 0.018 | 0.075 | 0.742 | 0.077 |

We make use of the following three scheduling strategies:

- FCFS: First-come-first-serve with a single queue.
- LowerWFQ: NL are always service first (strict priority) and a weighted fair queue (WFQ) is used between *CK* (weight 2) and *MD* (weight 1).
- HigherWFQ: NL are always service first (strict priority) and a weighted fair queue (WFQ) is used between *CK* (weight 10) and *MD* (weight 1).

Figures A.2-A.7 show scaled latencies and request latencies as functions of simulated time for 20 of the first long simulations runs using the different scenarios of mixed request types. Furthermore, Figures A.8-A.13 show the throughput as a function of simulated time for the same runs. We also ran a second batch with the same scenarios which produced similar plots. When using FCFS the request latency for the different requests are highly correlated, which is to be expected since all requests are in the same queue. On the other hand the scaled latency for the different request priorities, in particular *MD*, deviates from the others, which is due to the varying number of pairs in the requests. From Figures A.8-A.13 one can observe that the type of scheduler, at least for these simulated scenarios, have a relatively small effect on the throughput. In table A.3 and A.4 the average throughput, scaled latency and request latencies are collected for the same 20 simulation runs.

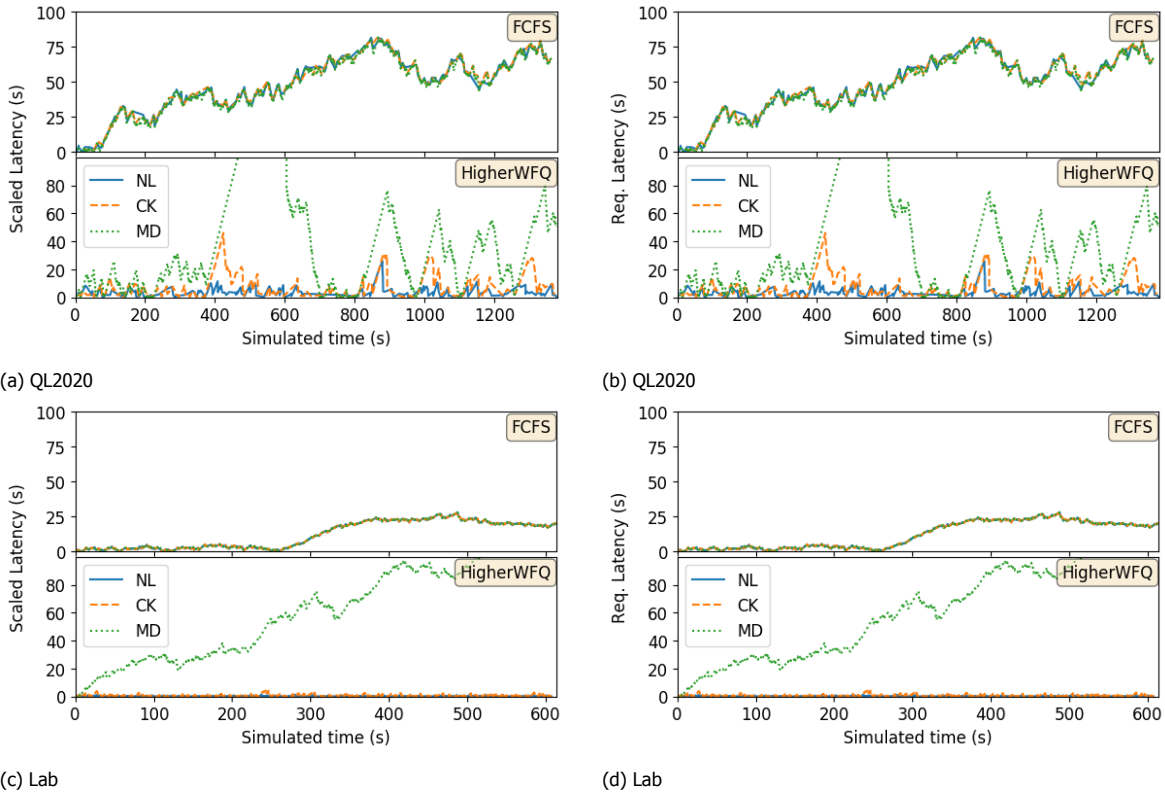


Figure A.2: Latencies for Uniform

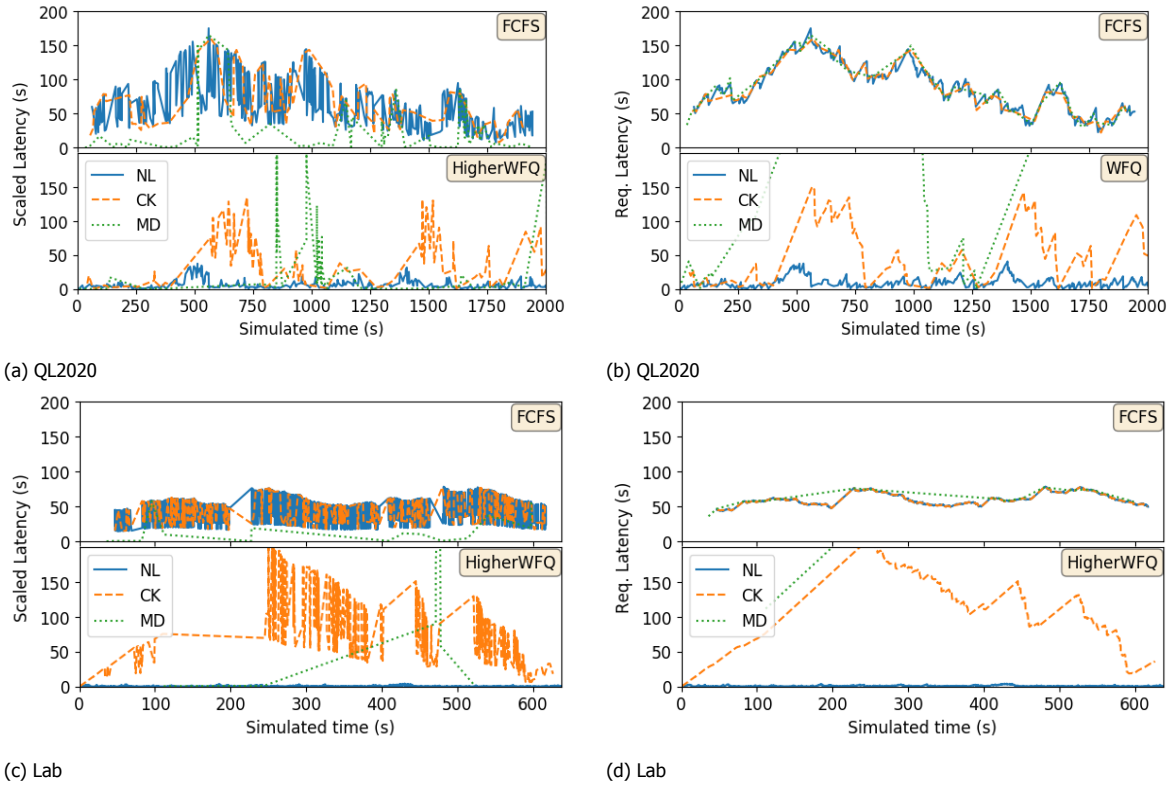


Figure A.3: Latencies for MoreNL

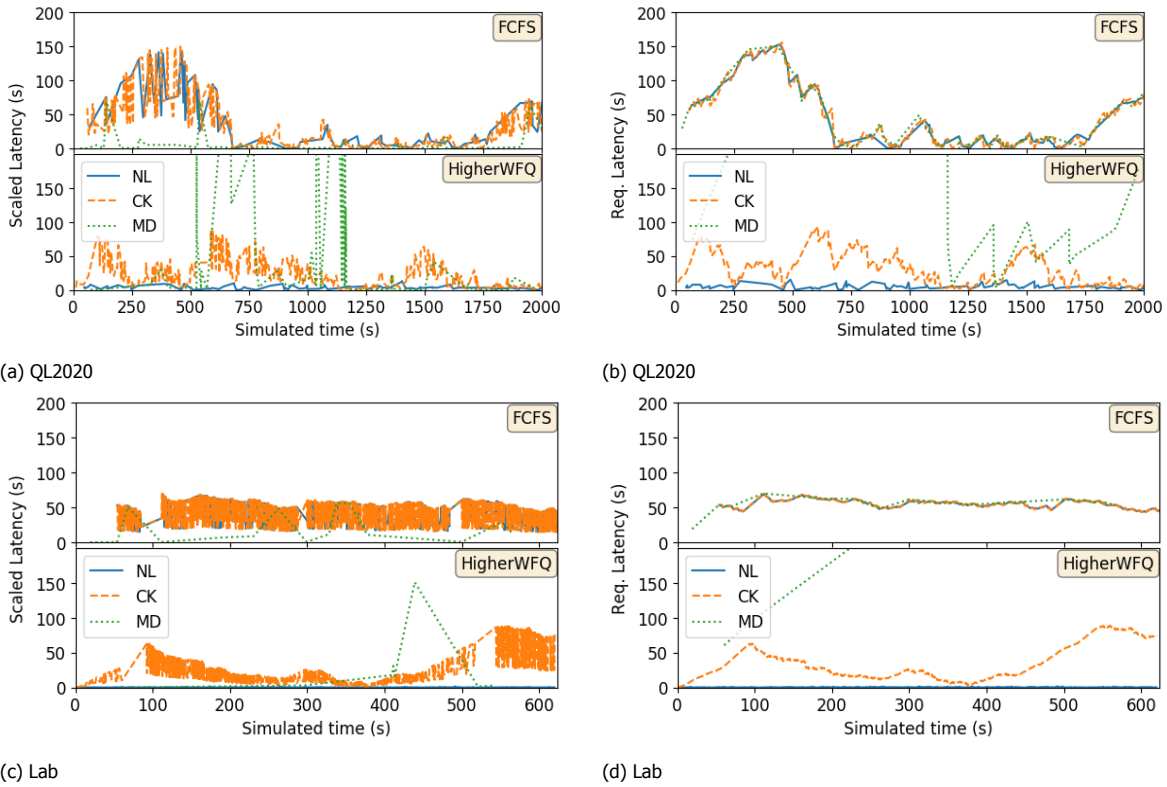


Figure A.4: Latencies for MoreCK

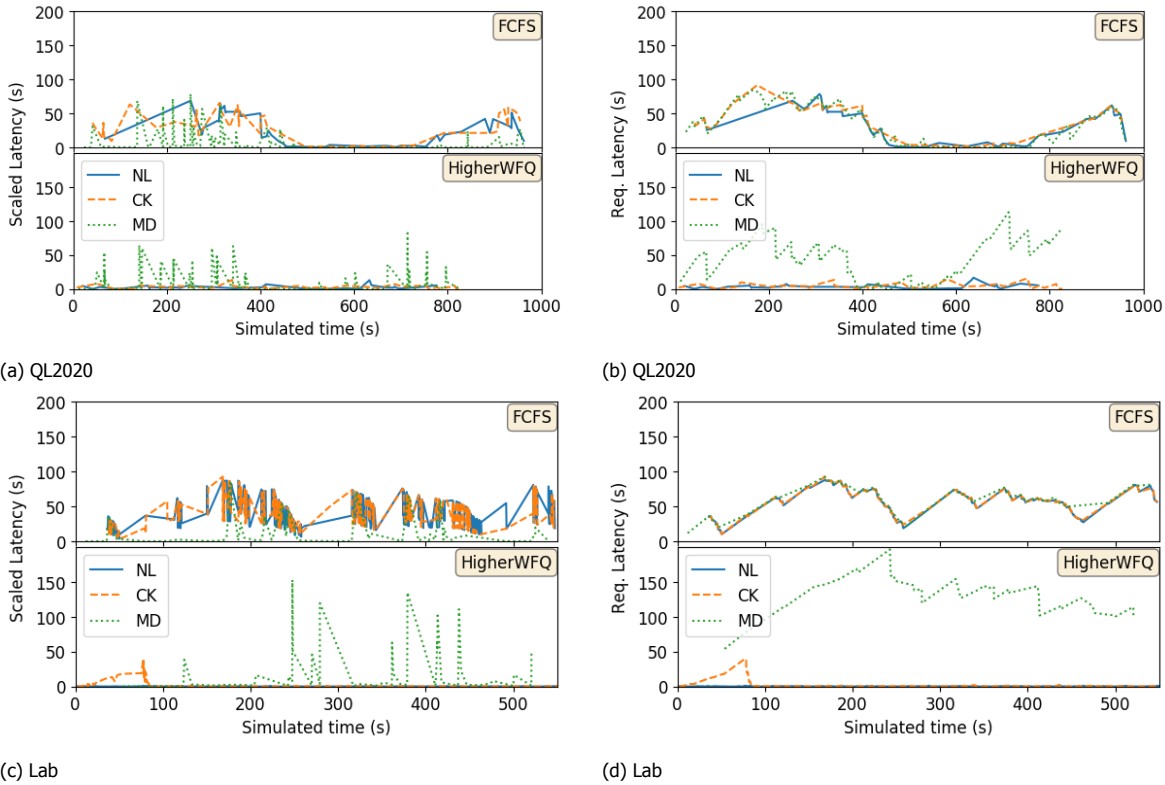


Figure A.5: Latencies for MoreMD

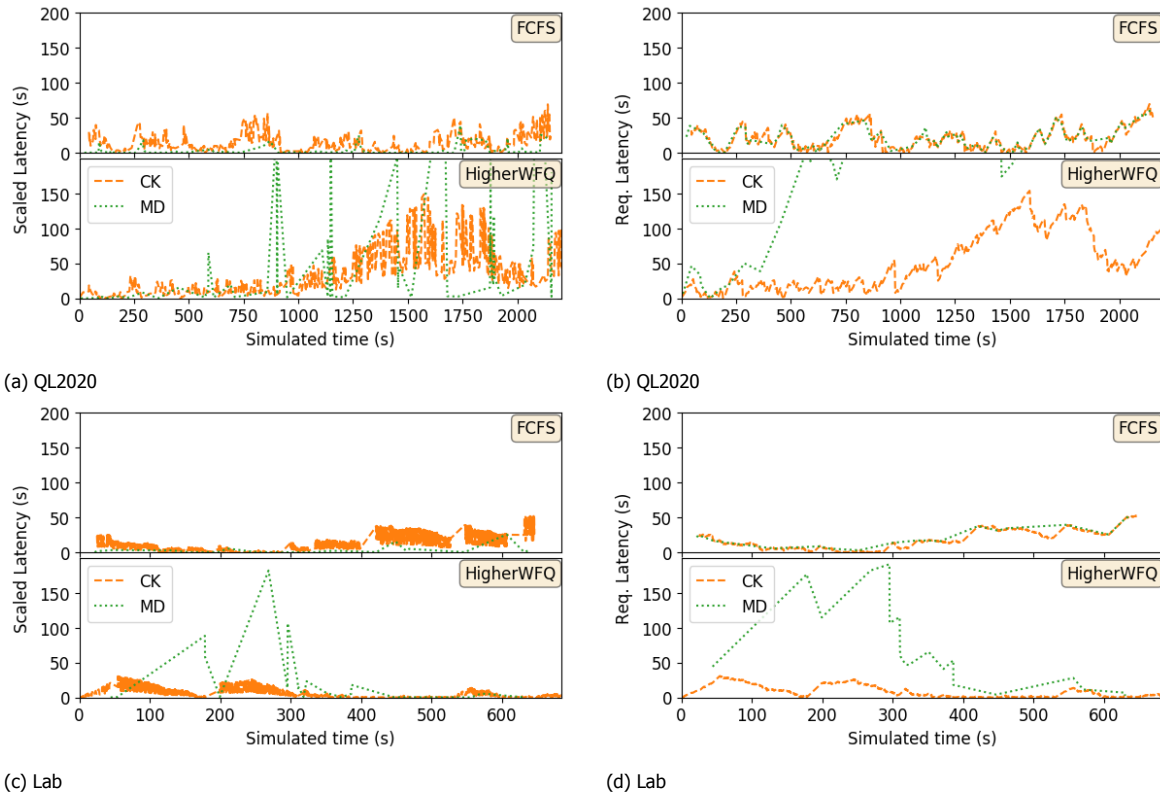


Figure A.6: Latencies for NoNLMoreCK

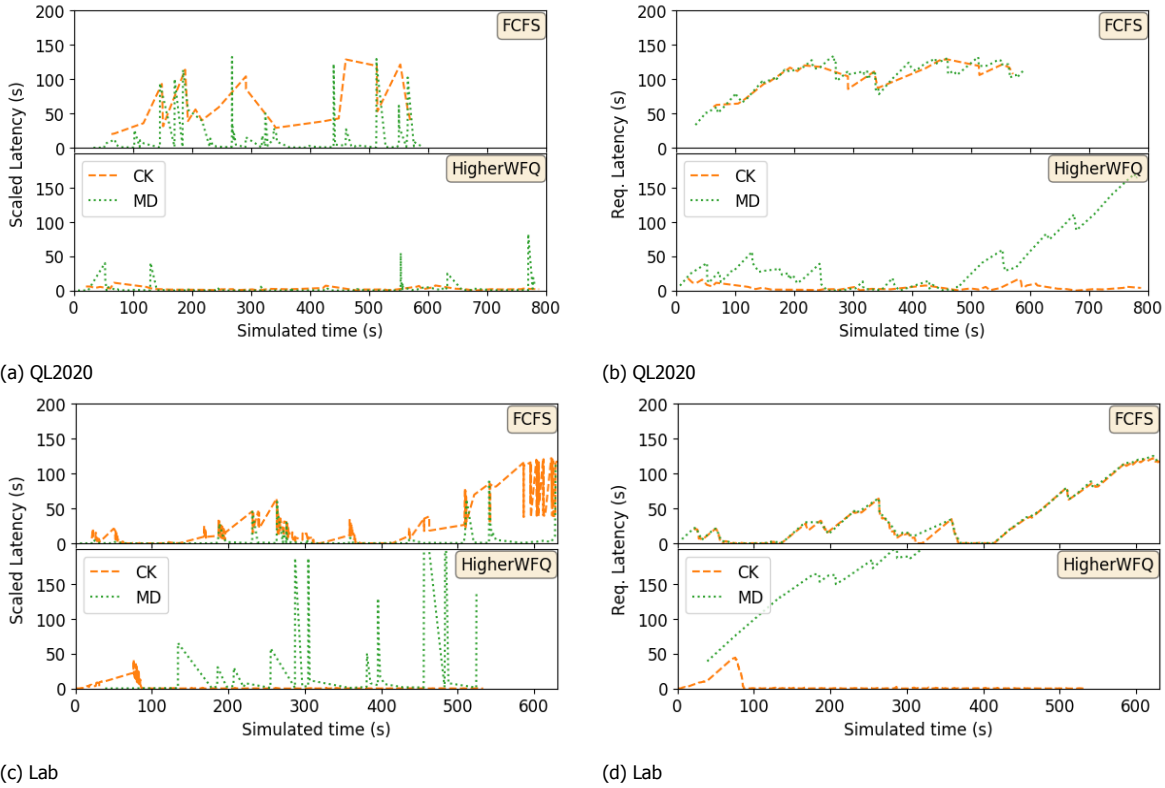
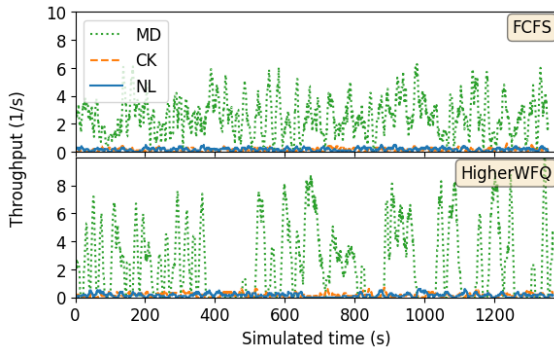
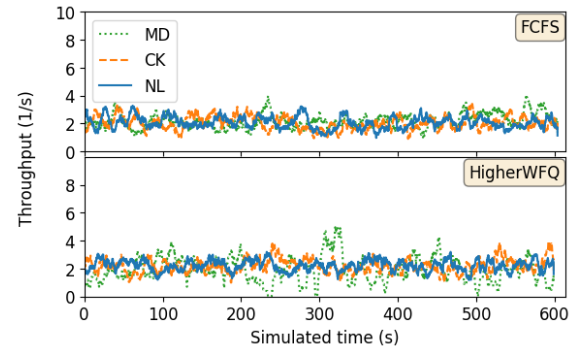


Figure A.7: Latencies for NoNLMoreMD

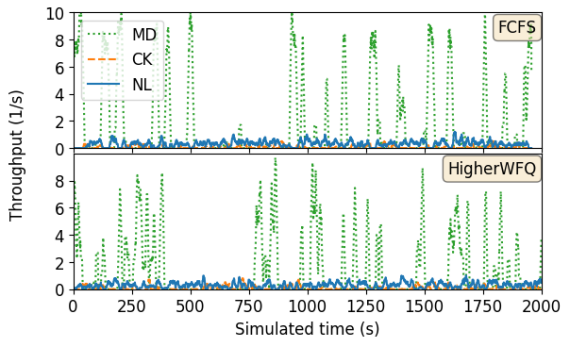


(a) QL2020

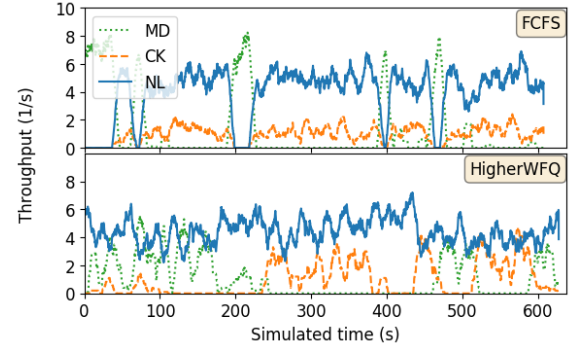


(b) Lab

Figure A.8: Throughputs for Uniform

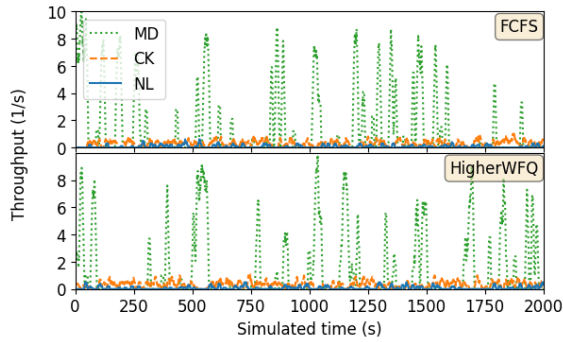


(a) QL2020

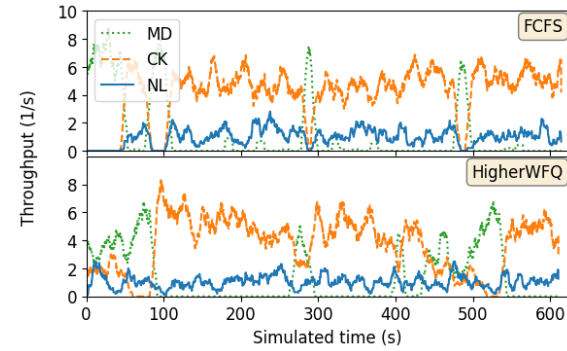


(b) Lab

Figure A.9: Throughputs for MoreNL

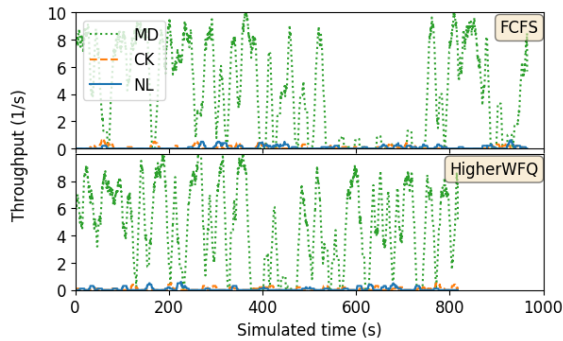


(a) QL2020

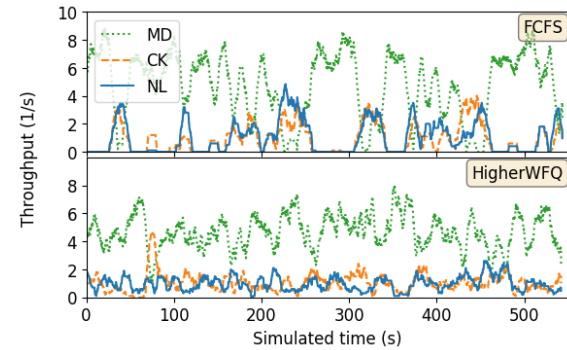


(b) Lab

Figure A.10: Throughputs for MoreCK

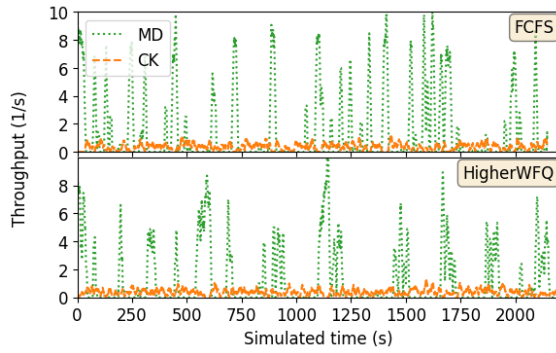


(a) QL2020

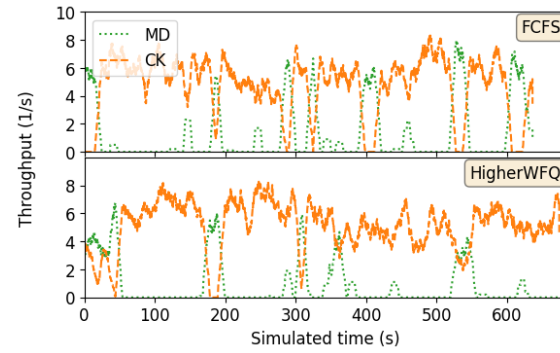


(b) Lab

Figure A.11: Throughputs for MoreMD

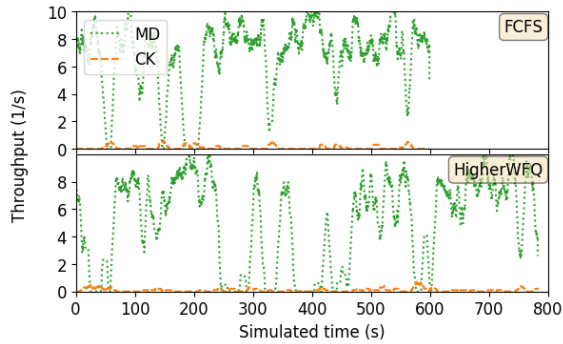


(a) QL2020

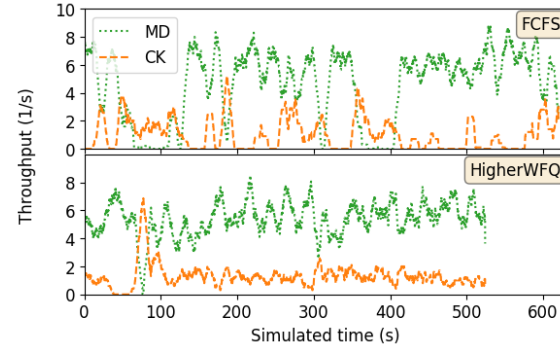


(b) Lab

Figure A.12: Throughputs for NoNLMoreCK



(a) QL2020



(b) Lab

Figure A.13: Throughputs for NoNLMoreMD

Table A.3: Average throughput for 20 simulation scenarios, as defined above, of requests with mixed priorities. Each value is computed as $(\#pairs/t_{sim_max})$ where t_{sim_max} is the reached simulated time (93 - 2355 s) of a single simulation run (24 h wall time).

| Scenario | Throughput_NL (1/s) | Throughput_CK (1/s) | Throughput_MD (1/s) |
|-----------------------------|---------------------|---------------------|---------------------|
| Lab_MoreCK_FCFS | 0.976 | 4.126 | 1.187 |
| Lab_MoreCK_HigherWFQ | 1.046 | 3.719 | 1.408 |
| Lab_MoreMD_FCFS | 1.025 | 0.905 | 4.771 |
| Lab_MoreMD_HigherWFQ | 0.981 | 1.058 | 4.659 |
| Lab_MoreNL_FCFS | 3.975 | 0.950 | 1.375 |
| Lab_MoreNL_HigherWFQ | 4.447 | 0.986 | 1.117 |
| Lab_NoNLMoreCK_FCFS | - | 4.696 | 1.366 |
| Lab_NoNLMoreCK_HigherWFQ | - | 5.101 | 0.916 |
| Lab_NoNLMoreMD_FCFS | - | 1.044 | 4.600 |
| Lab_NoNLMoreMD_HigherWFQ | - | 1.300 | 5.408 |
| Lab_Uniform_FCFS | 2.066 | 2.035 | 2.170 |
| Lab_Uniform_HigherWFQ | 2.210 | 2.186 | 1.908 |
| QL2020_MoreCK_FCFS | 0.064 | 0.302 | 1.398 |
| QL2020_MoreCK_HigherWFQ | 0.078 | 0.329 | 1.146 |
| QL2020_MoreMD_FCFS | 0.075 | 0.078 | 4.139 |
| QL2020_MoreMD_HigherWFQ | 0.066 | 0.073 | 4.793 |
| QL2020_MoreNL_FCFS | 0.312 | 0.066 | 1.667 |
| QL2020_MoreNL_HigherWFQ | 0.292 | 0.084 | 1.374 |
| QL2020_NoNLMoreCK_FCFS | - | 0.355 | 1.480 |
| QL2020_NoNLMoreCK_HigherWFQ | - | 0.374 | 1.180 |
| QL2020_NoNLMoreMD_FCFS | - | 0.084 | 6.756 |
| QL2020_NoNLMoreMD_HigherWFQ | - | 0.091 | 5.036 |
| QL2020_Uniform_FCFS | 0.175 | 0.143 | 2.538 |
| QL2020_Uniform_HigherWFQ | 0.154 | 0.166 | 2.483 |

Table A.4: Average scaled latencies (SL) and request latencies (RL) for 20 simulation scenarios, as defined above, of requests with mixed priorities of a single simulation run (93 - 2355 s simulated time and 24 h wall time). Values in parentheses are estimates of standard errors, computed as s_n/\sqrt{n} where s_n is the sample standard deviation and n is the number of data points used for averaging.

| Scenario | SL_NL (s) | SL_CK (s) | SL_MD (s) | RL_NL (s) | RL_CK (s) | RL_MD (s) |
|-----------------------------|--------------|--------------|---------------|--------------|---------------|----------------|
| Lab_MoreCK_FCFS | 40.18 (0.90) | 41.09 (0.42) | 19.64 (5.52) | 55.50 (0.30) | 55.58 (0.14) | 55.11 (2.83) |
| Lab_MoreCK_HigherWFQ | 0.30 (0.01) | 25.63 (0.61) | 24.95 (15.20) | 0.46 (0.02) | 33.88 (0.66) | 264.28 (35.75) |
| Lab_MoreMD_FCFS | 40.25 (1.15) | 41.70 (1.28) | 13.87 (2.91) | 55.63 (1.00) | 57.24 (1.00) | 62.55 (2.43) |
| Lab_MoreMD_HigherWFQ | 0.23 (0.01) | 2.09 (0.29) | 27.53 (6.00) | 0.36 (0.02) | 2.65 (0.35) | 129.30 (3.51) |
| Lab_MoreNL_FCFS | 45.59 (0.47) | 46.44 (0.95) | 14.70 (4.65) | 60.36 (0.21) | 60.59 (0.42) | 61.80 (2.76) |
| Lab_MoreNL_HigherWFQ | 0.69 (0.02) | 83.79 (2.64) | 98.34 (46.15) | 0.97 (0.02) | 114.89 (2.57) | 299.28 (37.25) |
| Lab_NoNLMoreCK_FCFS | - | 13.05 (0.27) | 3.55 (1.36) | - | 17.58 (0.30) | 23.04 (3.21) |
| Lab_NoNLMoreCK_HigherWFQ | - | 6.70 (0.16) | 26.14 (9.05) | - | 9.04 (0.19) | 76.02 (12.51) |
| Lab_NoNLMoreMD_FCFS | - | 23.45 (1.65) | 10.97 (2.51) | - | 30.86 (1.94) | 39.33 (4.09) |
| Lab_NoNLMoreMD_HigherWFQ | - | 2.26 (0.31) | 44.33 (11.92) | - | 3.34 (0.43) | 204.78 (9.54) |
| Lab_Uniform_FCFS | 11.41 (0.27) | 11.46 (0.27) | 12.38 (0.26) | 11.41 (0.27) | 11.46 (0.27) | 12.38 (0.26) |
| Lab_Uniform_HigherWFQ | 0.35 (0.01) | 0.73 (0.02) | 61.19 (0.97) | 0.35 (0.01) | 0.73 (0.02) | 61.19 (0.97) |
| QL2020_MoreCK_FCFS | 40.65 (4.43) | 37.46 (1.93) | 12.22 (3.82) | 52.20 (5.21) | 51.34 (2.33) | 43.41 (5.79) |
| QL2020_MoreCK_HigherWFQ | 4.11 (0.24) | 26.66 (0.95) | 76.29 (16.50) | 5.72 (0.34) | 35.91 (1.05) | 238.09 (21.00) |
| QL2020_MoreMD_FCFS | 25.45 (3.03) | 28.34 (3.01) | 9.30 (1.63) | 32.94 (3.43) | 38.90 (3.65) | 37.97 (2.67) |
| QL2020_MoreMD_HigherWFQ | 2.62 (0.42) | 3.04 (0.42) | 14.44 (1.92) | 3.79 (0.60) | 4.51 (0.62) | 47.64 (2.75) |
| QL2020_MoreNL_FCFS | 65.92 (1.93) | 64.05 (4.18) | 26.16 (5.84) | 89.83 (1.79) | 85.99 (3.70) | 85.98 (5.24) |
| QL2020_MoreNL_HigherWFQ | 7.04 (0.35) | 45.87 (3.73) | 50.55 (12.63) | 9.78 (0.41) | 59.92 (4.15) | 236.03 (21.62) |
| QL2020_NoNLMoreCK_FCFS | - | 15.98 (0.64) | 4.63 (0.87) | - | 21.90 (0.72) | 21.27 (1.63) |
| QL2020_NoNLMoreCK_HigherWFQ | - | 39.31 (1.64) | 70.64 (16.03) | - | 55.94 (1.93) | 277.08 (19.97) |
| QL2020_NoNLMoreMD_FCFS | - | 60.12 (6.95) | 22.28 (4.04) | - | 102.23 (4.17) | 104.88 (2.69) |
| QL2020_NoNLMoreMD_HigherWFQ | - | 3.01 (0.35) | 6.21 (1.58) | - | 5.18 (0.70) | 38.91 (4.89) |
| QL2020_Uniform_FCFS | 49.13 (1.29) | 50.85 (1.38) | 47.39 (0.33) | 49.13 (1.29) | 50.85 (1.38) | 47.39 (0.33) |
| QL2020_Uniform_HigherWFQ | 4.16 (0.26) | 8.22 (0.58) | 34.39 (0.61) | 4.16 (0.26) | 8.22 (0.58) | 34.39 (0.61) |

A.3. Under the hood

We now provide some more details on the simulation, numerical methods and the underlying NV platform. We remark that physical models for different parts of the NV platform are well established and validated [12]. The validation performed here is thus only about how the combined system performing entanglement generation matched with our simulation.

A.3.1. The simulated network

To understand our simulation we perform a full implementation of the MHP and QEGP, running on simulated quantum hardware. To achieve this, we start by defining basic components in NetSquid, which, inspired by NS-3, is entirely modular and can be used to construct complex simulation scenarios by combining component elements. The components in the simulation are as follows, and our simulation could easily be configured to examine the performance of our protocol on other underlying hardware platforms such as Ion Traps.

- A QuantumProcessingDevice, which is a general component we defined in NetSquid. Abstractly, such a QuantumProcessingDevice is described by the following:
 1. A number of communication and memory qubits. Each such qubit is associated with a noise-model that describes how quantum information decoheres over time when kept in the memory itself. Concrete parameters for the NV platform are given in Section A.3.2.
 2. Possible one or two-qubit quantum gates to be performed on each (pair of) qubit(s), the time required to execute the gate, as well as a noise-model associated with each such gate that may differ from qubit to qubit. For the NV platform we will only need to consider the gates given in Section A.3.2.
 3. With each communication qubit we associate a trigger that allows the generation of entanglement between this communication qubit and a traveling qubit (a photon). Such an operation only succeeds with some probability of success, requires a certain amount of time, and can also be noisy. For the NV platform, we explain this in Section A.3.4.
 4. Readout - i.e. measurement of a qubit. This takes a certain amount of time, and does again carry a noise-model. For NV, we explain this in Section A.3.3.
- A FiberConnection, which is a general NetSquid component that allows us to simulate optical fibers, including photon loss per km.
- A HeraldingStation, which automatically measures incoming photons in a certain time interval. This process is again subject to several possible errors explained in Section A.3.5.

- `ClassicalConnection`, which we use to transmit classical messages allowing us to simulate, for example, losses on the channel. Section [A.3.6](#) explains the model considered here.
- `Node`, which includes a `QuantumProcessingDevice`, and enjoys fiber connections with the `HeraldingStation` or other nodes. Each `Node` can run programs in the same way that they could be run on e.g. a computer or microcontroller, allowing these `Programs` to make use of - for example - the `QuantumProcessingDevice`. This allows us to perform a full implementation (here in Python) of the MHP and QEGP including all subsystems in the same way as the program will later run on an actual microcontroller (however, in C).

We briefly review properties of the nitrogen-vacancy (NV) center in diamond [12] and how they affect the design and performance of our protocol. We will also highlight how this can be modeled in simulation. Important to the design and performance of our protocol is how long operations on qubits stored in the NV-center take. Additionally, the coherence time, i.e. how long a qubit can be stored, has an impact on our protocol. We summarize typical values for the noise-level and execution time of the allowed operations of a NV-center together with coherence times. These are the values we used in our simulation. Note however that these values can vary significantly between samples.

Table A.5: Gates and coherence times used in simulation. Values used in the simulation corresponding to Lab. We remark that since these are custom chips, no two are exactly identical. Individual values have since seen significant improvements (Experimentally realized), but have not been realized simultaneously for producing entanglement that would allow a direct comparison to simulation. We have thus focused in simulation only what enables a comparison to data gathered from entanglement generation on hardware.

| | (Unsqured) fidelity | Duration/time | Experimentally realized |
|--|---|---------------|--|
| Electron T_1 | - | 2.86 ms | > 1h[13] |
| Electron T_2^* | - | 1.00 ms | 1.46 s[13] |
| Carbon T_1 | - | ∞ | > 6m [14] |
| Carbon T_2^* | - | 3.5 ms | \approx 10ms [14] |
| Electron single-qubit gate | 1.0 | 5 ns | > 0.995 (100 ns) [15] |
| E-C controlled- \sqrt{X} -gate (E=control) | 0.992 | 500 μ s | 0.992 (500-1000 μ s) fig 2 in [15] |
| Carbon Rot-Z-gate | 0.999 | 20 μ s | 1.0 (20 μ s) [16] |
| Electron initialization in $ 0\rangle$ | 0.95 | 2 μ s | 0.99 (2 μ s) [17] |
| Carbon initialization in $ 0\rangle$ | 0.95 | 310 μ s | 0.95 (300 μ s) [18] |
| Electron readout | 0.95 ($ 0\rangle$), 0.995 ($ 1\rangle$) | 3.7 μ s | 0.95 ($ 0\rangle$), 0.995 ($ 1\rangle$) (3-10 μ s) [9] |

A.3.2. Qubits on the NV Platform

A NV center is formed by replacing a carbon atom in a diamond lattice with a nitrogen atom and removing a neighboring carbon (vacancy). This structure traps electrons which together form a spin-1 system. Two of the levels of the collective spin-1 state can be used as a communication qubit in a quantum network. Around the NV center there is also a natural abundance of carbon-13 atoms which interact with the communication qubit (electron spin). The surrounding carbon spins can be addressed using the communication qubit and can thus be used as memory qubits. Here we consider a situation with only one communication qubit, and one memory qubit.

Noise model - Free evolution of the electronic and nuclear spins

Noise in experimental implementations is described in terms of T_1 , T_2 , T_2^* times, where Section 1.4.5 serves to provide intuition on how our quantity of interest - the fidelity to the maximally entangled target state $|\Psi^+\rangle$ - depends on their values. Table A.5 lists values used in simulation (reflecting Lab), and state of the art for the communication qubit (Electron), and memory qubit (Carbon).

Quantum gates

Procedure and parallelism constraints Quantum gates can be realized by applying microwave pulses. Of specific interest that affects the throughput is the duration of such operations given in Table A.5. While not absolutely necessary for the understanding of the simulation, we briefly sketch how operations are performed also on the carbons to illustrate one feature of this system that is relevant for the performance of our protocols - namely the parallelism of the allowed gate operations. We remark that operations on the carbon spins are performed using the following pulse sequence

$$(\tau - \pi - 2\tau - \pi - \tau)^{N/2}, \quad (\text{A.5})$$

where π is a microwave- π -pulse on the electron spin, 2τ is the time between the pulses and N is the total number of pulses. The target carbon spin can be chosen by picking τ such that it is precisely in resonance with the target carbon spin's hyperfine interaction with the electron spin. If the electron spin is in the state $|0\rangle$ ($|1\rangle$) the target carbon spin will rotate around the X -axis of the Bloch sphere in the positive (negative) direction, with an angle θ which depends on the total number of pulses N . This means that one can perform quantum gates on the carbon that are controlled by the state of the electron spin. The effective unitary operation (E =control) on the electron and the target carbon spin is then given as

$$\begin{pmatrix} R_X(\theta) & 0 \\ 0 & R_X(-\theta) \end{pmatrix}, \quad (\text{A.6})$$

where $R_X(\theta) = \exp(i\theta/2X)$ denotes a rotation around the X -axis of an angle θ . Not only does the pulse sequence (A.5) manipulate the carbon spin, but it also decouples the electron from its environment, thereby prolonging its coherence time and is thus

also called *dynamical decoupling*, allowing longer memory lifetimes (Figure 1.5a). We thus see a limit to the amount of parallelism when operating on the carbon and the electron spin.

Other quantum gates are however simpler: Since the carbon continuously precesses around the Z axis of the Bloch sphere, rotations around Z (Carbon Rot- Z) are simply done by waiting a correct amount of time. Thus, also controlled rotations around other axes than X can be performed by correctly interspersed waiting times during the pulse sequence above.

A.3.3. Gates and their noise

In this section we collect parameters for noise and delays of gates used in our simulation. Table A.5 summarize the possible gates that can be performed on the electron and carbon spins in the NV system, together with decoherence times. Section A.3.4 describes how the noise occurring from entanglement generation attempts is modeled.

Here the E-C controlled- X rotates the carbon spin around the X -axis in the positive (negative) direction if the electron is in the $|0\rangle$ ($|1\rangle$) state. Furthermore, note that there is an asymmetry in reading out the $|0\rangle$ -state and the $|1\rangle$ -state of the electron.

Modeling noisy operations

Noise in gate operations is modeled by applying noise after a perfect gate (a standard method):

$$U_{\text{noisy}}(\rho) = \mathcal{N}_{\text{dephas}}^f \circ U_{\text{perfect}}(\rho),$$

where

$$\mathcal{N}_{\text{dephas}}^p : \rho \mapsto f\rho + (1-f)Z\rho Z$$

is the dephasing channel in Z and f is the gate fidelity as given in table A.5. States are initialized as $\mathcal{N}_{\text{depol}}^p(|0\rangle\langle 0|)$, where

$$\mathcal{N}_{\text{depol}}^p : \rho \mapsto f\rho + \frac{1-f}{3} [X\rho X + Y\rho Y + Z\rho Z]$$

denotes the depolarization channel by.

How the electron spin is initialized

Initialization of the electron spin means setting the state to $|0\rangle$ [17]. Initialization of the electron spin is done by performing *optical pumping*, in which light shines onto the electron, thereby bringing its quantum state in a higher energy level, given it was in $|1\rangle$, after which it falls back to either $|0\rangle$ or $|1\rangle$ with a given probability. If the electron falls down to the state $|0\rangle$ it will stay there, thus after many repetitions of this process, the electron is with high confidence in the state $|0\rangle$. For our discussion here, it will be relevant to remark that this operation takes time (Table A.5), and we will need to perform it repeatedly as the first step in producing entanglement.

Moving a qubit to memory

When moving a qubit from the communication qubit to the memory qubit, the memory qubit needs to already be initialized. Initialization of the carbon is done by effectively swapping the $|0\rangle$ state from the electron to the carbon and cannot therefore be performed while having an entangled state in the electron. For this reason, initialization of the carbon ($310 \mu\text{s}$) needs to be performed before a photon is emitted from the electron during an entanglement generation attempt. However, it is not necessary to re-initialize the carbon before every entanglement generation attempt but simply periodically depending on the coherence time. In our simulation we assumed T_1 to be $3500 \mu\text{s}$ and thus re-initialize the carbon every $3500 \mu\text{s}$ (every 350th MHP cycle).

Swapping a state in the electron to the carbon can be done by 2 E-C controlled- $\sqrt{\mathcal{X}}$ -gates and single qubit gates (total time $1040 \mu\text{s}$) [15].

How a measurement (readout) is performed

Readout of the communication qubit First, we are again interested in the time to perform this operation given in Table A.5 which will be relevant in the MD use case. Evidently a readout can be noisy, where we remark that the noise is asymmetric in that the probability of incorrectly obtaining measurement outcome 0 is much lower than incorrectly getting outcome 1.

Reading out a memory qubit We again remark that next to timing constraints (Table A.5), we have limited parallelism on the current NV platform, since we need the electron spin to readout the memory qubit. Reading out the nuclear spin is done by performing the following steps:

1. initialize the electron spin,
2. apply an effective controlled NOT operation with the nuclear spin as control (consisting of one E-C controlled- $\sqrt{\mathcal{X}}$ -gate and single-qubit gates),
3. measure (readout) the electron spin.

The reason why a controlled NOT is sufficient, rather than a full swap, is the following: If the nuclear spin is in state $\alpha |0\rangle + \beta |1\rangle$, then after the CNOT, the combined state is $\alpha |00\rangle_{EC} + \beta |11\rangle_{EC}$. The reduced state [2] of the electron is then $|\alpha|^2 |0\rangle\langle 0| + |\beta|^2 |1\rangle\langle 1|$, so measuring in the standard basis yields the same statistics as measuring $\alpha |0\rangle + \beta |1\rangle$ in the same basis.

Readout noise Readout is modeled by performing a POVM measurement with the following Kraus operators (see [2] for definition)

$$M_0 = \begin{pmatrix} \sqrt{f_0} & 0 \\ 0 & \sqrt{1-f_1} \end{pmatrix}, \quad M_1 = \begin{pmatrix} \sqrt{1-f_0} & 0 \\ 0 & \sqrt{f_1} \end{pmatrix} \quad (\text{A.7})$$

where f_0 (f_1) is the readout fidelity of the $|0\rangle$ -state ($|1\rangle$) as given in table A.5

A.3.4. Physical Entanglement Generation and Noise

We here consider the single-click scheme of Lab. To understand timing and quality as well as parameter choices, let us give a high-level overview of the single-click scheme: A microwave pulse is used to prepare the communication qubit in the state $\sqrt{\alpha}|0\rangle + \sqrt{1-\alpha}|1\rangle$ (max. $5.5\mu\text{s}$ for A and B), where $|0\rangle$ is also called the bright state, and α the bright state population. A resonant laser pulse is then used to cause emission of a photon, if the state was in the bright state, preparing the joint state of the communication qubit (C) and an emitted photon (P) in the state $\sqrt{\alpha}|0\rangle_C|1\rangle_P + \sqrt{1-\alpha}|1\rangle_C|0\rangle_P$, where $|0\rangle_P$ ($|1\rangle_P$) denotes the absence (presence) of a photon. This process succeeds with probability $p_{\text{em}} \approx 0.03$ without Purcell enhancement using optical cavities. To ensure phase-stabilization only one laser may be used for both nodes, combined with local shutters to allow node control. We remark that local control at the node is still desirable at a distance due to aligning with other operations such as performing gates. to keep qubits stable. The heralding station interferes both incoming photons on a beam splitter, thereby performing a probabilistic entanglement swap. Intuitively, this can be understood as a measurement of the incoming qubits in the Bell basis, where we can only obtain outcomes $|\Psi^+\rangle$, $|\Psi^-\rangle$ or "other". Since "other" is more than one possible state, we declare failure in this case.

Depending on the clicks observed in the detectors, we have projected the state of the communication qubits at A and B in the state $|\Psi^+\rangle = \frac{1}{\sqrt{2}}(|0\rangle_A|1\rangle_B + |1\rangle_A|0\rangle_B)$ (only left detector clicks), $|\Psi^-\rangle = \frac{1}{\sqrt{2}}(|0\rangle_A|1\rangle_B - |1\rangle_A|0\rangle_B)$ (only right detector clicks), or we declare failure (either none or both of the detectors click). Success occurs with probability $p_{\text{succ}} \approx 2\alpha p_{\text{det}}$, where p_{det} is the probability of detecting an emitted photon.

During entanglement generation, a variety of noise processes occur:

1. Dephasing of the nuclear spins (memory qubits) due to resetting the electron during entanglement generation attempts.
2. Effective dephasing of the photon's state due to uncertainty in the phase difference between the paths the photons travel to the beam-splitter.
3. Effective dephasing of the photon's state due to non-zero probability of emitting two photons.
4. Effective amplitude damping of the photon state due to coherent emission, i.e. the photon is in a superposition of being emitted at different times.
5. Effective amplitude damping due to collection losses from non-unity probability of emitting the photon in the zero-phonon line and non-unity collection efficiency into the fiber.
6. Effective amplitude damping due to losses in fiber.
7. Non-perfect beam-splitter measurement at the heralding stations due to photons not being perfectly indistinguishable.

8. Errors in the classical outcome from the detectors due to non-unity detection efficiency and dark counts.

Dephasing mechanism on nuclear spins during entanglement attempts

Between every entanglement attempt, the electron spin (communication qubit) needs to be reset. The dominant source of noise on the nuclear spins (memory qubits) during the entanglement attempts is due to this re-initialization of the electron spin, as described in [19]. We model the noise on the nuclear spins as a fixed amount of dephasing noise

$$\mathcal{D}_{p_d}(\rho_n) = (1 - p_d)\rho_n + p_d Z \rho_n Z \quad (\text{A.8})$$

for every entanglement attempt. The dephasing parameter depends on: the bright state population α , the coupling strength $\Delta\omega$ and a decay constant τ_d as follows

$$p_d = \frac{\alpha}{2} (1 - \exp(-\Delta\omega^2 \tau_d^2 / 2)), \quad (\text{A.9})$$

see equation (2) in [19]. If the length of the Bloch vector in the equatorial plane of the state in the nuclear spin is before the entanglement attempts r_{XY} , then after N attempts the length will be

$$(1 - p_d)^N r_{XY}. \quad (\text{A.10})$$

The bright state population can be chosen per experiment but the coupling strength and decay constant for a nuclear spin are fixed but vary between different spins. The decay constant can also vary by performing different microwave control pulses of the electron spin. As an example of these parameters, for the nuclear spin C_1 and the standard microwave control pulses, the coupling strength is $\Delta\omega = 2\pi \times 377$ kHz and the decay constant is $\tau_d = 82$ ns, see [19]. In the simulations we use these values for the coupling strength and decay constant.

Phase uncertainty for photon paths

There is uncertainty in the phase between the paths the photon travels from the nodes to the beam-splitter, due to for example uncontrolled stretching of the fiber. If this phase difference is $\Delta\phi$ then the state after a successful measurement at the heralding station (conditioned on there being only one photon) is

$$|0\rangle_{e_A} |1\rangle_{e_B} \pm e^{i\Delta\phi} |1\rangle_{e_A} |0\rangle_{e_B}. \quad (\text{A.11})$$

where e_A is the electron spin at node A and e_B is the electron spin at node B .

We model this by performing dephasing noise to both qubits encoding the presence/absence of photon from A and B . As shown in [20], if we chose the dephasing parameter to be

$$p_d = \left(1 - \frac{I_1(\sigma(\phi)^{-2})}{I_0(\sigma(\phi)^{-2})}\right) / 2 \quad (\text{A.12})$$

then the standard deviation of the phase ϕ in the state between the electron (e_A/e_B) and the photon (p_A/p_B)

$$|0\rangle_{e_A} |1\rangle_{p_A} \pm e^{i\phi} |1\rangle_{e_A} |0\rangle_{p_A} \quad (\text{A.13})$$

is precisely $\sigma(\phi)$. To efficiently compute quotients of modified Bessel functions we implemented the algorithm described in [21]. Note that the variance of the phase in equation (A.11) is twice the variance of the phase in equation (A.13). In experiments the standard deviation of the phase for the state between the electrons, i.e. (A.11), has in [9] shown to be 14.3° . Thus, we set the standard deviation in equation (A.13) to be $14.3^\circ/\sqrt{2}$.

Two-photon emission

There is a probability that two photons are emitted from the electron during the entanglement generation attempt at a node. For the physical setup we assume the probability of there being two photons emitted, given that at least one is emitted, to be approximately 4% [9]. As argued in [9], the two-photon event can effectively be modeled as applying dephasing noise to the electron qubit which is part of the generated entanglement.

Coherent emission, superposition of time-modes

The emission of the photon is a coherent process and the photon is actually in a superposition of being emitted at different times. As shown in [20], the effect of a finite detection window can be seen as effective amplitude damping noise to the qubit encoding the presence/absence of a photon. The amplitude damping parameter is then given by

$$p_a = \exp(-t_w/\tau_e), \quad (\text{A.14})$$

where t_w is the detection time window and τ_e is the characteristic time of the NV emission (12 ns without cavity [22] and 6.48 ns with cavity [20]).

Collection losses

We model non-unity collection efficiencies by effective amplitude damping noise on the qubit encoding the presence/absence of a photon. The amplitude damping parameter is given by

$$p_a = (1 - p_{\text{zero_phonon}} \cdot p_{\text{collection}}), \quad (\text{A.15})$$

where $p_{\text{zero_phonon}}$ is the probability of emitting a photon in the zero phonon line (3% without cavity and 46% with cavity [22]) and $p_{\text{collection}}$ is the probability of collection the photon into the fiber. From [9] we know that the total detection efficiency of the system is $4 \cdot 10^{-4}$, which can be decomposed as

$$p_{\text{total}} = p_{\text{zero_phonon}} \cdot p_{\text{collection}} \cdot p_{\text{transmission}} \cdot p_{\text{detection}}, \quad (\text{A.16})$$

where $p_{\text{transmission}}$ is the probability that the photon is not lost during transmission in the fiber and $p_{\text{detection}}$ is the probability that the detector clicks, given that there was a photon. Using equation (A.16) we find that $p_{\text{collection}} = 0.014$ given the numbers in [9]. Frequency conversion succeeds with probability 30% [23], so in this case we use $p_{\text{collection}} = 0.014 \cdot 0.3$.

Transmission losses

Since the qubit sent from the node to the heralding station is encoded in the presence/absence of a photon, the losses during transmission over fiber are modeled as amplitude damping noise. We use an amplitude damping parameter p_{t_loss} given as

$$p_{t_loss} = 1 - 10^{-L \cdot \gamma / 10}, \quad (\text{A.17})$$

where L is the length of the fiber (in km) and γ is assumed to be 5 dB/km without frequency conversion and 0.5 dB/km with frequency conversion.

Distinguishable photons

Entanglement is generated between the electrons of the two nodes since the beam-splitter in the heralding station effectively deletes the information of which node a detected photon came from. This information is only perfectly detected if the photons emitted from the nodes are completely indistinguishable. In reality however, the photons properties (spectral, temporal etc.) can be slightly different and they are therefore not completely indistinguishable. In section A.3.4 we derive effective measurement operators of a beam-splitter measurement, taking photon indistinguishability into account, which we make use of in our simulation. For the physical setup we simulate, the overlap (visibility) of the photons coming from the nodes is approximately 0.9 [9], i.e. $|\mu|^2 = 0.9$ where μ is defined in equation (A.50).

Detection losses and dark counts

Detection losses and dark counts are modeled by probabilistically changing the ideal classical outcome from the detectors at the heralding station. For each detector, if the ideal detector clicked the noisy detector also clicks with probability $p_{\text{detection}}$ and otherwise not. In the simulations we use $p_{\text{detection}} = 0.8$, as measured in [24].

If the ideal detector did not click the noisy detector does click with probability p_{dark} . The parameter used for the dark count is the dark count rate $\lambda_{\text{dark}} = 20$ per second [9]. The dark counts follow a Poisson distribution and we have that

$$p_{\text{dark}} = 1 - \exp(-t_w \cdot \lambda_{\text{dark}}), \quad (\text{A.18})$$

where t_w is the detection time window.

A.3.5. Heralding station

Let us now consider the measurement at the Heralding Station in more detail in order to understand its error models.

Distinguishable photons

We here describe how we model a beam-splitter measurement of two photons which are not perfectly indistinguishable. This is relevant for many heralding entanglement generation schemes, since if photons are distinguishable the beam-splitter will not erase the information of where the photons came from. Two perfectly indistinguishable photons incident on a beam-splitter will always go to the same output arm, as captured by the Hong-Ou-Mandel effect [25]. However, if the photons are

distinguishable they do not necessarily go to the same output arm, which can be detected in experiment. For a given setup, let's denote the probability that two incident photons on the beam-splitter go to different output arms as χ .

We will in this section derive the effective POVM and Kraus operators correspond to detecting photons at the ends of the output arms of the beam-splitter in terms of χ , under the assumptions described below and using ideas from the paper [26] where χ is computed.

Model

Assume that there is a 50:50 beam-splitter with input arms a and b and output arms c and d . At the end of the output arms there are photon detectors that can click. We will assume that the detectors have a flat frequency response and at first that the detectors can count photons, i.e. there are different measurement outcomes for there being one or two photons incident on a detector. However we will show below how one can easily consider detectors which cannot count photons from the analysis in this note.

Photons In many simulations we model the presence or absence of a photon as a two-level system, i.e. a qubit $\alpha |0\rangle + \beta |1\rangle$, where $|0\rangle$ means no photon and $|1\rangle$ one photon. We would then describe the state before the beam-splitter as a state living in a 2-qubit Hilbert space spanned by the following four basis vectors:

$$|00\rangle_{lr}, \quad |01\rangle_{lr}, \quad |10\rangle_{lr}, \quad |11\rangle_{lr} \quad (\text{A.19})$$

describing 0 photons, photon on the right, photon on the left and two photons. Here l and r corresponds to arm a and b of the beam-splitter, but we distinguish these since we will denote a (and b) as the infinite dimensional Hilbert space describing the spectral property of the photon. Note that we assume that there are never more than one photon per arm.

Describing the presence and absence of a photon as a qubit masks the fact that a photon can have many other degrees of freedom, such as polarization, spectral and temporal properties. We will here focus on spectral and temporal properties and will therefore model a photon in arm a with a spectral amplitude function ϕ as the state

$$\int d\omega \phi(\omega) a^\dagger(\omega) |0\rangle_a, \quad (\text{A.20})$$

where $a^\dagger(\omega)$ is the creation operator of a photon in arm a of frequency ω and $|0\rangle_a$ is the vacuum and ϕ is normalized such that

$$\int d\omega |\phi(\omega)|^2 = 1. \quad (\text{A.21})$$

Furthermore, the state of arm b will be described by a spectral amplitude function ψ as

$$\int d\omega \psi(\omega) b^\dagger(\omega) |0\rangle_b. \quad (\text{A.22})$$

Two photons arriving at the beam-splitter can have different spectral properties, captured by ϕ and ψ being different. We will also include a possible temporal shift τ between the arrival times of the two photons. As described in equation (16) of [26], a temporal shift of a photon in arm b induces the following action on the creation operators

$$b^\dagger(\omega) \rightarrow b^\dagger(\omega)e^{-i\omega\tau}. \quad (\text{A.23})$$

Beam-splitter The 50:50 beam-splitter acts on the creation operators in the following way:

$$a^\dagger(\omega) \rightarrow \frac{1}{\sqrt{2}}(c^\dagger(\omega) + d^\dagger(\omega)) \quad (\text{A.24})$$

$$b^\dagger(\omega) \rightarrow \frac{1}{\sqrt{2}}(c^\dagger(\omega) - d^\dagger(\omega)). \quad (\text{A.25})$$

Thus the state of a photon described as in equation (A.20), i.e. one photon in the input arm a , will after the beam-splitter become

$$|\phi\rangle_{cd} = \frac{1}{\sqrt{2}} \int d\omega \phi(\omega)(c^\dagger(\omega) + d^\dagger(\omega)) |0\rangle_{cd}. \quad (\text{A.26})$$

Furthermore, the three other cases of no photon, one photon in the input arm b and one photon in each input arm becomes after the beam-splitter:

$$|0\rangle_{cd} \quad (\text{A.27})$$

$$|\psi\rangle_{cd} = \frac{1}{\sqrt{2}} \int d\omega \psi(\omega)e^{-i\omega\tau}(c^\dagger(\omega) - d^\dagger(\omega)) |0\rangle_{cd} \quad (\text{A.28})$$

$$|\phi, \psi\rangle_{cd} = \frac{1}{2} \int d\omega_1 \int d\omega_2 \phi(\omega_1)\psi(\omega_2)e^{-i\omega_1\tau} (c^\dagger(\omega_1) + d^\dagger(\omega_1))(c^\dagger(\omega_2) - d^\dagger(\omega_2)) |0\rangle_{cd}. \quad (\text{A.29})$$

Where the states $|0\rangle_{cd}$, $|\phi\rangle_{cd}$, $|\psi\rangle_{cd}$ and $|\phi, \psi\rangle_{cd}$ should be thought of as the corresponding states to the states in equation (A.19). Below, we will in fact formally define an isometry between these two Hilbert spaces.

Detectors As mentioned we assume that the detectors have a flat frequency response. The event that the detector in arm c detected one photon can then be described by the projector

$$P_{1,0} = \int d\omega c^\dagger(\omega) |0\rangle_{cd} \langle 0|_{cd} c(\omega) \quad (\text{A.30})$$

Since we assume that there is maximally one photon arriving at each input arm of the beam-splitter the only other possible measurement outcomes are described by

the following projectors:

$$P_{0,0} = |0\rangle\langle 0|_{cd} \quad (\text{A.31})$$

$$P_{0,1} = \int d\omega d^\dagger(\omega) |0\rangle\langle 0|_{cd} d(\omega) \quad (\text{A.32})$$

$$P_{1,1} = P_{1,0} \otimes P_{0,1} = \int d\omega_1 \int d\omega_2 c^\dagger(\omega_1) d^\dagger(\omega_2) |0\rangle\langle 0|_{cd} c(\omega_1) d(\omega_2) \quad (\text{A.33})$$

$$P_{2,0} = \frac{1}{2} \int d\omega_1 \int d\omega_2 c^\dagger(\omega_1) c^\dagger(\omega_2) |0\rangle\langle 0|_{cd} c(\omega_1) c(\omega_2) \quad (\text{A.34})$$

$$P_{0,2} = \frac{1}{2} \int d\omega_1 \int d\omega_2 d^\dagger(\omega_1) d^\dagger(\omega_2) |0\rangle\langle 0|_{cd} d(\omega_1) d(\omega_2) \quad (\text{A.35})$$

where $P_{0,0}$ corresponds to no photon, $P_{0,1}$ one photon in arm d , $P_{1,1}$ one photon in each arm, $P_{2,0}$ two photons in arm c and $P_{0,2}$ two photons in arm d . Note that the factors of $\frac{1}{2}$ are needed for $P_{2,0}$ such that $P_{2,0}^2 = P_{2,0}$ and similarly with $P_{0,2}$.

Deriving effective POVM on presence/absence description The goal of this note is to derive the effective POVM on the Hilbert space lr , spanned by vectors in equation (A.19), induced by the projective measurements in equations (A.30)-(A.35) on the infinite-dimensional Hilbert space cd . To do this we will first define an isometry $U_{lr \rightarrow cd}$ from the Hilbert space lr to cd , using the states in equation (A.19) and equations (A.26)-(A.29). This isometry will have the following action on the basis states of lr :

$$|00\rangle_{lr} \rightarrow |0\rangle_{cd} \quad (\text{A.36})$$

$$|01\rangle_{lr} \rightarrow |\psi\rangle_{cd} \quad (\text{A.37})$$

$$|10\rangle_{lr} \rightarrow |\phi\rangle_{cd} \quad (\text{A.38})$$

$$|11\rangle_{lr} \rightarrow |\phi, \psi\rangle_{cd} \quad (\text{A.39})$$

and will therefore be given as

$$U_{lr \rightarrow cd} = |0\rangle_{cd} \langle 00|_{lr} + |\psi\rangle_{cd} \langle 01|_{lr} + |\phi\rangle_{cd} \langle 10|_{lr} + |\phi, \psi\rangle_{cd} \langle 11|_{lr}. \quad (\text{A.40})$$

One can easily check that the states $|0\rangle_{cd}$, $|\phi\rangle_{cd}$, $|\psi\rangle_{cd}$ and $|\phi, \psi\rangle_{cd}$ are mutually orthogonal and that $U_{lr \rightarrow cd}$ is therefore indeed an isometry, i.e.

$$(U_{lr \rightarrow cd})^\dagger U_{lr \rightarrow cd} = \mathbb{1}_{lr}. \quad (\text{A.41})$$

Let us assume that $|\Phi\rangle_{lr}$ is a state in lr and we wish to compute the probability of receiving a measurement outcome corresponding to the projector $P \in \{P_{0,0}, P_{1,0}, P_{0,1}, P_{1,1}, P_{2,0}, P_{0,2}\}$ for the state $U_{lr \rightarrow cd} |\Phi\rangle_{lr}$. Using Born's rule we find that this probability is given as

$$\langle \Phi|_{lr} (U_{lr \rightarrow cd})^\dagger P U_{lr \rightarrow cd} |\Phi\rangle_{lr} = \text{tr}[(U_{lr \rightarrow cd})^\dagger P U_{lr \rightarrow cd} |\Phi\rangle\langle \Phi|_{lr}]. \quad (\text{A.42})$$

From the above equation we find that the effective POVM on lr is given as

$$\{(U_{lr \rightarrow cd})^\dagger P U_{lr \rightarrow cd} : P \in \{P_{0,0}, P_{1,0}, P_{0,1}, P_{1,1}, P_{2,0}, P_{0,2}\}\}. \quad (\text{A.43})$$

whose elements we will denote as M_{00} , M_{10} , M_{01} , M_{11} , M_{20} and M_{02} . In section A.3.5 we compute what these POVM-elements are and find a choice of Kraus operators in section A.3.5 for both the case when the detector can count photons and when it cannot.

Effective POVMs Here we compute the POVM-elements in equation (A.43) one-by-one.

M_{11} :

Let us start with M_{11} since this will allow us to relate these POVM-elements to χ , i.e. the probability that both detectors click, given that there was one photon in each input arm. The operator P_{11} only has non-zero overlap with the term $|\phi, \psi\rangle_{cd} \langle 11|_{lr}$ of $U_{lr \rightarrow cd}$ and is therefore given as

$$M_{11} = (U_{lr \rightarrow cd})^\dagger P_{11} U_{lr \rightarrow cd} = |11\rangle_{lr} \langle \phi, \psi|_{cd} P_{11} |\phi, \psi\rangle_{cd} \langle 11|_{lr}. \quad (\text{A.44})$$

Let us evaluate the factor $\langle \phi, \psi|_{cd} P_{11} |\phi, \psi\rangle_{cd}$. Using equation (A.29) and equation (A.33) we find that the above expression evaluates to

$$\begin{aligned} \langle \phi, \psi|_{cd} P_{11} |\phi, \psi\rangle_{cd} &= \frac{1}{2} \int d\omega_1 \int d\omega_2 \phi^*(\omega_1) \psi^*(\omega_2) e^{i\omega_2 \tau} \\ &\quad \langle 0|_{cd} (c(\omega_1) + d(\omega_1))(c(\omega_2) - d(\omega_2)) \\ &\quad \times \int d\omega_3 \int d\omega_4 c^\dagger(\omega_3) d^\dagger(\omega_4) |0\rangle\langle 0|_{cd} c(\omega_3) d(\omega_4) \\ &\quad \times \frac{1}{2} \int d\omega_5 \int d\omega_6 (c^\dagger(\omega_5) + d^\dagger(\omega_5))(c^\dagger(\omega_6) - d^\dagger(\omega_6)) |0\rangle_{cd} \\ &\quad \phi(\omega_5) \psi(\omega_6) e^{-i\omega_6 \tau} \end{aligned} \quad (\text{A.45})$$

$$\begin{aligned} &= \frac{1}{4} \int d\omega_1 \int d\omega_2 \int d\omega_3 \int d\omega_4 \int d\omega_5 \int d\omega_6 \\ &\quad \phi^*(\omega_1) \psi^*(\omega_2) \phi(\omega_5) \psi(\omega_6) e^{i\omega_2 \tau} e^{-i\omega_6 \tau} \left(\right. \\ &\quad + \delta(\omega_2 - \omega_3) \delta(\omega_3 - \omega_6) \delta(\omega_1 - \omega_4) \delta(\omega_4 - \omega_5) \\ &\quad - \delta(\omega_2 - \omega_3) \delta(\omega_3 - \omega_5) \delta(\omega_1 - \omega_4) \delta(\omega_4 - \omega_6) \\ &\quad - \delta(\omega_1 - \omega_3) \delta(\omega_3 - \omega_6) \delta(\omega_2 - \omega_4) \delta(\omega_4 - \omega_5) \\ &\quad \left. + \delta(\omega_1 - \omega_3) \delta(\omega_3 - \omega_5) \delta(\omega_2 - \omega_4) \delta(\omega_4 - \omega_6) \right) \end{aligned} \quad (\text{A.46})$$

where we used the fact that $\langle 0|_{cd} c(\omega_1) c^\dagger(\omega_2) |0\rangle_{cd} = \delta(\omega_1 - \omega_2)$ and similarly for arm d . Using that

$$\int d\omega_2 f(\omega_2) \delta(\omega_1 - \omega_2) = f(\omega_1) \quad (\text{A.47})$$

we find that equation (A.46) evaluates to

$$\begin{aligned} &\frac{1}{2} \int d\omega_1 \int d\omega_2 (\phi^*(\omega_1) \psi^*(\omega_2) \phi(\omega_1) \psi(\omega_2) \\ &\quad e^{i\omega_2 \tau} e^{-i\omega_2 \tau} - \phi^*(\omega_1) \psi^*(\omega_2) \phi(\omega_2) \psi(\omega_1) e^{i\omega_2 \tau} e^{-i\omega_1 \tau}) \end{aligned} \quad (\text{A.48})$$

Finally using equation (A.21) we find that M_{11} evaluates to

$$M_{11} = \frac{1}{2}(1 - |\mu|^2) |11\rangle\langle 11|_{lr} \quad (\text{A.49})$$

where

$$\mu = \int d\omega \phi^*(\omega)\psi(\omega)e^{-i\omega\tau}. \quad (\text{A.50})$$

From equation (A.49) we can relate $|\mu|$ to χ as

$$\chi = \frac{1}{2}(1 - |\mu|^2). \quad (\text{A.51})$$

M_{20} :

The operator P_{20} only has non-zero overlap with the term $|\phi, \psi\rangle_{cd} \langle 11|_{lr}$ of $U_{lr \rightarrow cd}$ and is therefore given as

$$M_{11} = (U_{lr \rightarrow cd})^\dagger P_{20} U_{lr \rightarrow cd} = |11\rangle_{lr} \langle \phi, \psi|_{cd} P_{20} |\phi, \psi\rangle_{cd} \langle 11|_{lr}. \quad (\text{A.52})$$

Lets evaluate the factor $\langle \phi, \psi|_{cd} P_{20} |\phi, \psi\rangle_{cd}$. Using equation (A.29) and equation (A.34) we find that the above expression evaluates to

$$\begin{aligned} \langle \phi, \psi|_{cd} P_{20} |\phi, \psi\rangle_{cd} &= \frac{1}{2} \int d\omega_1 \int d\omega_2 \phi^*(\omega_1)\psi^*(\omega_2) \\ &\quad e^{i\omega_2\tau} \langle 0|_{cd} (c(\omega_1) + d(\omega_1))(c(\omega_2) - d(\omega_2)) \\ &\quad \times \frac{1}{2} \int d\omega_3 \int d\omega_4 c^\dagger(\omega_3)c^\dagger(\omega_4) |0\rangle\langle 0|_{cd} c(\omega_3)c(\omega_4) \\ &\quad \times \frac{1}{2} \int d\omega_5 \int d\omega_6 (c^\dagger(\omega_5) + d^\dagger(\omega_5))(c^\dagger(\omega_6) - d^\dagger(\omega_6)) \\ &\quad |0\rangle_{cd} \phi(\omega_5)\psi(\omega_6)e^{-i\omega_6\tau} \end{aligned} \quad (\text{A.53})$$

$$\begin{aligned} &= \frac{1}{8} \int d\omega_1 \int d\omega_2 \int d\omega_3 \int d\omega_4 \int d\omega_5 \int d\omega_6 \\ &\quad \phi^*(\omega_1)\psi^*(\omega_2)\phi(\omega_5)\psi(\omega_6)e^{i\omega_2\tau}e^{-i\omega_6\tau} \\ &\quad \times \left(\delta(\omega_1 - \omega_4)\delta(\omega_2 - \omega_3) + \delta(\omega_1 - \omega_3)\delta(\omega_2 - \omega_4) \right) \\ &\quad \times \left(\delta(\omega_3 - \omega_6)\delta(\omega_4 - \omega_5) + \delta(\omega_3 - \omega_5)\delta(\omega_4 - \omega_6) \right) \end{aligned} \quad (\text{A.54})$$

where we used the fact that $\langle 0|_{cd} c(\omega_1)c(\omega_2)c^\dagger(\omega_3)c^\dagger(\omega_4) |0\rangle_{cd} = \delta(\omega_1 - \omega_3)\delta(\omega_2 - \omega_4) + \delta(\omega_2 - \omega_3)\delta(\omega_1 - \omega_4)$. Then similarly to M_{11} we find that equation (A.54) evaluates to

$$\langle \phi, \psi|_{cd} P_{20} |\phi, \psi\rangle_{cd} = \frac{1}{4}(1 + |\mu|^2) \quad (\text{A.55})$$

and we thus find M_{20} to be

$$M_{20} = \frac{1}{4}(1 + |\mu|^2) |11\rangle\langle 11|_{lr}. \quad (\text{A.56})$$

$\overline{M_{20}}$:

Similarly to M_{20} we find that M_{02} evaluates to

$$M_{02} = \frac{1}{2}(1 + |\mu|^2) |11\rangle\langle 11|_{lr}. \quad (\text{A.57})$$

$\overline{M_{10}}$:

The operator P_{10} only has non-zero overlap with the terms $|\phi\rangle_{cd} \langle 10|_{lr}$ and $|\psi\rangle_{cd} \langle 01|_{lr}$ of $U_{lr \rightarrow cd}$ and is therefore given as

$$M_{10} = (U_{lr \rightarrow cd})^\dagger P_{10} U_{lr \rightarrow cd} = \left(|10\rangle_{lr} \langle \phi|_{cd} + |01\rangle_{lr} \langle \psi|_{cd} \right) P_{10} \left(|\phi\rangle_{cd} \langle 10|_{lr} + |\psi\rangle_{cd} \langle 01|_{lr} \right). \quad (\text{A.58})$$

Lets evaluate the factors $\langle \phi|_{cd} P_{10} |\phi\rangle_{cd}$, $\langle \psi|_{cd} P_{10} |\psi\rangle_{cd}$, $\langle \phi|_{cd} P_{10} |\psi\rangle_{cd}$ and $\langle \psi|_{cd} P_{10} |\phi\rangle_{cd}$ one-by-one. First we have that:

$$\begin{aligned} \langle \phi|_{cd} P_{10} |\phi\rangle_{cd} &= \frac{1}{\sqrt{2}} \int d\omega_1 \phi^*(\omega_1) \langle 0|_{cd} (c(\omega_1) + d(\omega_1)) \\ &\quad \times \int d\omega_2 c^\dagger(\omega_2) |0\rangle\langle 0|_{cd} c(\omega_2) \\ &\quad \times \frac{1}{\sqrt{2}} \int d\omega_3 (c^\dagger(\omega_3) + d^\dagger(\omega_3)) |0\rangle_{cd} \phi(\omega_3) \\ &= \frac{1}{2} \int d\omega_1 \int d\omega_2 \int d\omega_3 \phi^*(\omega_1) \phi(\omega_3) \delta(\omega_1 - \omega_2) \delta(\omega_2 - \omega_3) \\ &= \frac{1}{2} \int d\omega |\phi(\omega)|^2 \\ &= \frac{1}{2}. \end{aligned} \quad (\text{A.59})$$

and similarly that

$$\langle \psi|_{cd} P_{10} |\psi\rangle_{cd} = \frac{1}{2}. \quad (\text{A.60})$$

Furthermore, we find that

$$\begin{aligned} \langle \phi|_{cd} P_{10} |\psi\rangle_{cd} &= \frac{1}{\sqrt{2}} \int d\omega_1 \phi^*(\omega_1) \langle 0|_{cd} (c(\omega_1) + d(\omega_1)) \\ &\quad \times \int d\omega_2 c^\dagger(\omega_2) |0\rangle\langle 0|_{cd} c(\omega_2) \\ &\quad \times \frac{1}{\sqrt{2}} \int d\omega_3 (c^\dagger(\omega_3) - d^\dagger(\omega_3)) |0\rangle_{cd} \psi(\omega_3) e^{-i\omega_3 \tau} \\ &= \frac{1}{2} \int d\omega_1 \int d\omega_2 \int d\omega_3 \phi^*(\omega_1) \psi(\omega_3) e^{-i\omega_3 \tau} \delta(\omega_1 - \omega_2) \delta(\omega_2 - \omega_3) \\ &= \frac{1}{2} \int d\omega \phi^*(\omega) \psi(\omega) e^{-i\omega \tau} \\ &= \frac{1}{2} \mu. \end{aligned} \quad (\text{A.61})$$

where μ is defined in equation (A.50). One easily then finds that

$$\langle \psi |_{cd} P_{10} | \phi \rangle_{cd} = (\langle \phi |_{cd} P_{10} | \psi \rangle_{cd})^* = \frac{1}{2} \mu^*. \quad (\text{A.62})$$

Combining the above results, we find that M_{10} is given as

$$M_{10} = \frac{1}{2} \left(|10\rangle\langle 10|_{lr} + |01\rangle\langle 01|_{lr} + \mu |10\rangle\langle 01|_{lr} + \mu^* |01\rangle\langle 10|_{lr} \right) \quad (\text{A.63})$$

M_{10} :

Similarly to M_{10} one finds that M_{01} evaluates to

$$M_{01} = \frac{1}{2} \left(|10\rangle\langle 10|_{lr} + |01\rangle\langle 01|_{lr} - \mu |10\rangle\langle 01|_{lr} - \mu^* |01\rangle\langle 10|_{lr} \right) \quad (\text{A.64})$$

M_{00} :

Its easy to see that

$$M_{00} = |00\rangle\langle 00|_{lr}. \quad (\text{A.65})$$

POVM for photon-counter detectors To summarize we found that the POVM-elements are given as

$$M_{00} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \quad (\text{A.66})$$

$$M_{10} = \frac{1}{2} \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & \mu & 0 \\ 0 & \mu^* & 1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \quad (\text{A.67})$$

$$M_{01} = \frac{1}{2} \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & -\mu & 0 \\ 0 & -\mu^* & 1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \quad (\text{A.68})$$

$$M_{11} = \frac{1}{2} \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 - |\mu|^2 \end{pmatrix} \quad (\text{A.69})$$

$$M_{20} = \frac{1}{4} \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 + |\mu|^2 \end{pmatrix} \quad (\text{A.70})$$

$$M_{02} = \frac{1}{4} \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 + |\mu|^2 \end{pmatrix} \quad (\text{A.71})$$

where the rows and columns of the above matrices are ordered as $|00\rangle_{\chi 00}|_{lr}$, $|10\rangle_{\chi 10}|_{lr}$, $|01\rangle_{\chi 01}|_{lr}$, $|11\rangle_{\chi 11}|_{lr}$ and μ is given as

$$\mu = \int d\omega \phi^*(\omega)\psi(\omega)e^{-i\omega\tau}. \quad (\text{A.72})$$

and is related by to the probability that both detectors click, given that there were one photon in each input arm χ as

$$\chi = \frac{1}{2}(1 - |\mu|^2). \quad (\text{A.73})$$

POVM for non-photon-counter detectors If the detectors used cannot distinguish between one and two photons we can simply add the POVM elements M_{10} and M_{20} to get a new POVM given as

$$\tilde{M}_{00} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \quad (\text{A.74})$$

$$\tilde{M}_{10} = \frac{1}{2} \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & \mu & 0 \\ 0 & \mu^* & 1 & 0 \\ 0 & 0 & 0 & (1 + |\mu|^2)/2 \end{pmatrix} \quad (\text{A.75})$$

$$\tilde{M}_{01} = \frac{1}{2} \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & -\mu & 0 \\ 0 & -\mu^* & 1 & 0 \\ 0 & 0 & 0 & (1 + |\mu|^2)/2 \end{pmatrix} \quad (\text{A.76})$$

$$\tilde{M}_{11} = \frac{1}{2} \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 - |\mu|^2 \end{pmatrix} \quad (\text{A.77})$$

Effective Kraus operators

Given the POVMs in equation (A.66)-(A.71) and equation (A.74)-(A.77) one can choose corresponding Kraus operators for these measurements by taking the matrix square root of the corresponding POVM-elements. Assuming that μ is real one finds

a set of Kraus operators of the POVM $\{\tilde{M}_{00}, \tilde{M}_{10}, \tilde{M}_{01}, \tilde{M}_{11}\}$ to be

$$\tilde{E}_{00} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \quad (\text{A.78})$$

$$\tilde{E}_{10} = \frac{1}{2} \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & (\sqrt{1+\mu} + \sqrt{1-\mu})/\sqrt{2} & (\sqrt{1+\mu} - \sqrt{1-\mu})/\sqrt{2} & 0 \\ 0 & (\sqrt{1+\mu} - \sqrt{1-\mu})/\sqrt{2} & (\sqrt{1+\mu} + \sqrt{1-\mu})/\sqrt{2} & 0 \\ 0 & 0 & 0 & \sqrt{1+|\mu|^2} \end{pmatrix} \quad (\text{A.79})$$

$$\tilde{E}_{01} = \frac{1}{2} \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & (\sqrt{1+\mu} + \sqrt{1-\mu})/\sqrt{2} & (\sqrt{1-\mu} - \sqrt{1+\mu})/\sqrt{2} & 0 \\ 0 & (\sqrt{1-\mu} - \sqrt{1+\mu})/\sqrt{2} & (\sqrt{1+\mu} + \sqrt{1-\mu})/\sqrt{2} & 0 \\ 0 & 0 & 0 & \sqrt{1+|\mu|^2} \end{pmatrix} \quad (\text{A.80})$$

$$\tilde{E}_{11} = \frac{1}{\sqrt{2}} \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \sqrt{1-|\mu|^2} \end{pmatrix}. \quad (\text{A.81})$$

A.3.6. Classical communication

Optical Link Error Model

We claimed that we highly inflated losses in the simulation to stress test our protocol. We now consider more realistic values for such errors by considering a realistic *packet-level* error model for the non-quantum optical link. For this we have assumed that two quantum internet end nodes are connected by a legacy 100BASE-ZX single-mode 1550 nm wavelength Gigabit Ethernet link. The reason for choosing 100BASE-ZX interface is (i) its achievable long-distance transmission at least up to 70 km with no dependency on optical repeaters and (ii) decades of its successful deployment within magnitude of networks worldwide.

To be conservative, our optical Gigabit Ethernet model assumes a typical worst-case optical link budget (0.5 dB/km attenuation², 0.7 dB/connector loss, 0.1 dB/splice (joint) loss, and 3 dB safety margin) [27]. We also assume a typical worst-case -1 dBm optical transmission power and -24 dB receiver sensitivity of a 100BASE-ZX small form-factor hot pluggable transceiver, see e.g. [28]. For a maximum realism of link error over an optical link we model a *IEEE 802.3 frame errors*, instead of modeling individual bit errors of every message sent across the network. The latter would require a software implementation of a complete modulation and coding

²Fibers measured for QL2020 have been found to have this loss level.

layer of IEEE 802.3 which is beyond the scope of this work. Using measurement trace-driven packet-level Gigabit Ethernet frame error data from [29, Table 6.1] we have mapped the received SNR per transmitter/receiver distance to the respective frame error probability, which was then applied to every classical message sent over an optical link between quantum end nodes. SNR values that were not represented in the measurements of [29] have been linearly interpolated. We have not distinguished between the lengths of each classical message as the model of [29] has aggregated over all messages captured over a measured campus Ethernet link (cf. [29, Fig. 6.1]). We note that our modeling approach is equivalent to the frame error models applied in e.g. NS-3 [30] for WiFi frame errors.

For two example long-distance Quantum Internet typologies (node-to-node distance of 15 km and 20 km, respectively) we have ended up in a perfect frame error probability, with the assumption that amount of splices is zero³ (we only start to observe frame errors only at transmitter/receiver distance exceeding 40 km for the above model variables, with a very narrow transition error between no frame error rate and disconnected interface, i.e. frame error rate of one). Even when we increase the number of splices to an exaggerated level, say 30 splices for a 15 km interface (with 0.3 dB loss/splice), we still observe a very low frame error probability of 4×10^{-8} . Therefore, to test the effect of frame errors on the non-quantum optical link on the Quantum Internet protocol stack—in the cases of extreme frame loss—we have increased the value of frame error to 10^{-4} (and tested frame error rate up to 10^{-10} —an error rate level of a 20 km link with 21 splices—in steps of 10^{-1}). If our protocol would work in such a high (but unrealistic) condition then it would also work on a realistic low-error optical link.

Optical Link CRC Error Model

Additionally, we have investigated a non-zero probability of CRC not being able to detect a frame error. Assuming the same optical link type (e.g. 100BASE-ZX) we have used a model of [32] to calculate the respective probability of not detecting a CRC frame error within a IEEE 802.3 frame. For this we have mapped the transmitter/receiver distance to the respective SNR (the same way as described in Section A.3.6). Then we mapped the SNR to the respective BER using [29, Table 6.3] (performing the same process of interpolating SNR between the points not measured by [29] as for the optical link error model, see again Section A.3.6) and then using [32, Fig. 1] mapped this resulting SNR to the respective probability of undetected error. We have assumed a worst case scenario of the longest IEEE 802.3 frame (i.e. $n = 12144$ bits, that is a maximum MTU). Again, for any of the two Quantum Internet lengths mentioned above, we do not find any CRC errors. At the highly-spliced case, considered in Section A.3.6, we obtain an *extremely low* CRC error rate of 1.4×10^{-23} . Therefore such errors were decided to be ignored in our implementation. Another reason for not considering these errors: it would require a full implementation of en- and decoding of classical frames which outside the scope of this work.

³Which is consistent with the measurements, e.g. in [31, Section 4].

A.4. Protocols

Here we give details of the implementation of the physical and link layer protocols in our simulations. In the following we assume that packets obey network byte ordering (Big Endian).

A.4.1. Distributed Queue Protocol

In order to track the individual applications that the entangled qubits belong to, the QEGP makes use of a *distributed queue* which shares request information between peers. Management of the distributed queue is performed by the Distributed Queue Protocol (DQP). In addition to storing the parameters supplied with a CREATE request generated by the layers above the link layer⁴, DQP will keep additional information about each entanglement request including its *create_time*, *min_time* at which the request may be executed and *MHP timeout cycle* by which the entanglement request will time out. We proceed with the introduction of the DQP by describing the structure of priority queues, followed by the queue establishment process, DQP message sequence diagram and DQP associated messages.

Priority Queues

Priorities are necessary to fulfill the use case requirements outlined in section 2.3. This is accomplished by adding requests to different types of queues $Q = \{Q_1, \dots, Q_L\}$, where L is the total number of queues in the distributed queue. Each queue can contain a maximum of x items simultaneously (in other words x is the maximum size of each individual queue), where an item is an individual entanglement request with its associated metadata, e.g. *create_time*, *min_time*, MHP timeout cycle. Each CREATE request is assigned a queue number by the scheduler (see Section A.4.3 below), and receives an absolute queue ID which is a tuple (j, i_j) where j indicates the designated queue Q_j (or, more abstractly, the *queue ID* of the entanglement request) and i_j is a unique ID within Q_j . We will denote (j, i_j) as the *absolute queue ID* or a_{ID} , and use $(j, i_j) \in Q$ to indicate that a request with the absolute queue ID (j, i_j) is in the queue Q .

The absolute queue ID must obey the following properties:

- *Total order*: Items on each queue follow a total order of items waiting in the queue determined by i_j .
- *Arrival time*: ID of an entanglement (CREATE) request is a function of its arrival time. Let t_1 and t_2 denote the *create_time* of entanglement requests 1 and 2, respectively. Then, let i_1 and i_2 denote their respective absolute queue ID's. If both requests are added to the same queue Q_j , and $t_1 < t_2$, then request 1 should be processed before request 2.

We will now outline the distributed queue establishment within DQP.

⁴Refer section 2.4 for the details of the CREATE request.

DQP Queue Establishment

The core objective of DQP is to obtain shared queues at both nodes, i.e. the items and the order of the elements in the queues are agreed upon. That is, both controllable end nodes A and B hold local queues $Q^A = \{Q_1^A, \dots, Q_L^A\}$ and $Q^B = \{Q_1^B, \dots, Q_L^B\}$ respectively, which are synchronized using the DQP. CREATE request additions to the queue Q_j can be made by either A or B invoking the DQP by the function $\text{ADD}(j, c_r)$, where c_r is the entanglement request by CREATE message. ADD returns a tuple (i_j, R) where R indicated success or failure. Failure can occur if

- no acknowledgments are received within a certain time frame, i.e. a timeout occurs,
- the remote node rejects addition to the queue, or
- the queue is full.

Success means that the request to create entanglement is placed into Q^A and Q^B such that the following properties are satisfied:

- *Equal queue number:* If a request is added by A as $(j, i_j) \in Q^A$, then it will (eventually) be added at B with the same absolute queue ID $(j, i_j) \in Q^B$ (and vice versa);
- *Uniqueness of queue ID:* If a request is placed into the queue by either A or B , then it is assigned a unique queue number. That is, if $(j, a) \in Q^A$ and $(j, a') \in Q^A$ reference two distinct CREATE requests, then $a \neq a'$;
- *Consistency:* If $(j, i_j) \in Q^A$ and $(j, i_j) \in Q^B$ then both absolute queue IDs refer to the same request⁵;
- *Fairness:* If A (or B) is issuing requests continuously, then also the other node B (or A) will get to add items to the queue after A (or B) in a “fair manner” as determined by the window size, denoted as W_A (W_B). More precisely, if a_1, \dots, a_N are CREATE requests submitted at A , and b_1, \dots, b_M are CREATE requests submitted at B with $N > W_A$ and $M > W_B$ —all assigned to the same queue Q_j but not yet added—then the final ordering of the requests on the queue obeys $a_1, \dots, a_m, b_1, \dots, b_k, a_{m+1}, \dots$ with $m \leq W_A$ and $k \leq W_B$.

Recall that each request receives a minimum time before it can be executed—a time buffer before the request may begin processing which takes into account the processing time to add it into the queue (denoted by the `min_time`)—which we will choose to be the expected propagation delay between A and B . The purpose of this minimum time is to decrease the likelihood A or B wants to produce an entanglement before the other node is ready. If either A or B begins processing early, no penalty other than reduced performance due to increased decoherence of the quantum memory results. Refer to Section A.4.1 on how this minimum time is passed between nodes.

⁵This is implied by the previous two conditions, but added for clarity.

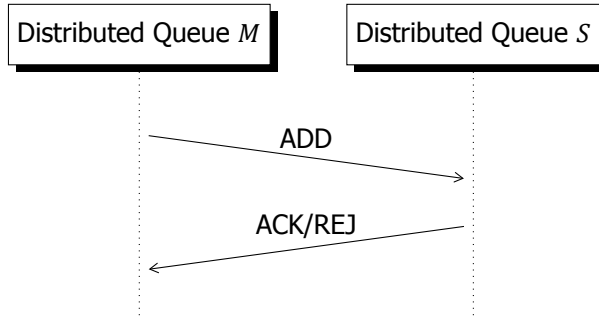


Figure A.14: DQP operation timeline. User M (master) adds an item to the distributed queue by sending an ADD message to peer node S (slave). S either acknowledges or rejects the request using ACK and REJ messages, respectively. Note that this process is symmetric when S attempts to add an item to the queue. For a definition of all messages refer to Figure A.15.

We recall that in the current implementation of quantum network we have two nodes only. This implies that the queue establishment can be realized by one node being the master controller of the queue marshaling access to the queue, and the other the slave controller. Extensions to multiple nodes are more complex, and a motivation to consider heralding station-centric protocols in the future versions of the protocol. Also, as we have two nodes only, there is no need for the introduction of leader election or a network discovery mechanism. We leave this as future work.

DQP Sequence Diagrams

Figure A.14 shows a DQP sequence diagram of adding an item to the queue containing a request to the distributed queue. Specifically, an item is an entanglement create request with its associated properties that is passed inside an ADD message within its REQ field, refer to Figure A.15 for details.

Upon receiving an ADD message from master M , a slave S may choose to acknowledge the item with an ACK message, should validation pass, or reject it with the REJ message for any of the previously mentioned reasons. In the case that master M never receives an acknowledgment (ACK) or rejection (REJ) message after a timeout, the item will not be placed in the queue and no processing will occur on the request. Loss of ADD, REJ, and ACK messages in the distributed queue protocol result in retransmissions of the original ADD to guarantee the receipt of rejection and acknowledgement messages.

When the slave S wishes to add an item to the queue, a message containing the request information and desired queue is included within the messages. Because the master controller has the final say on the state of the queue, a sequence number within the specified queue will be transmitted in return to the slave such that absolute queue IDs are consistent between the nodes.

DQP Packet Formats

Figure A.15 presents the packet format for messages exchanged in the DQP. Schedule Cycle and Timeout Cycle of 64 bits is governed by the maximum number of MHP

| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|--|--|--|-----|--|--|--|----|--|----|--|------|--|--|--|------------|--|--|--|------------|--|--|--|------|--|--|--|--|--|--|--|
| 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| OPT (reserved) | | | | | | | | | | FT | | CSEQ | | | | | | | | QID | | | | QSEQ | | | | | | | |
| Schedule Cycle | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Timeout | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Minimum Fidelity | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Purpose ID | | | | | | | | | | | | | | | | Create ID | | | | | | | | | | | | | | | |
| Number of Pairs | | | | | | | | | | | | | | | | Priority | | | | (reserved) | | | | | | | | | | | |
| Initial Virtual Finish | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Estimated Cycles/Pair | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| STR | | | | ATM | | | | MD | | | | MR | | | | (reserved) | | | | | | | | | | | | | | | |

Figure A.15: Packet format for ADD, ACK, and REJ. Explanation of the message fields—OPT: field reserved for future options, FT: frame type (00: ADD, 01: ACK, 10: REJ), CSEQ: the communication sequence number of the transmitted message (encoded as an integer in [0, 255]); QID: the ID of the queue to add the request to (encoded as an integer in [0,15]); QSEQ: the sequence number within the specified queue to assign the request (encoded as an integer in [0,255]); Schedule Cycle: the first MHP cycle when the request may begin (encoded as an integer with 64 bits, equivalent to min_time); Timeout: The MHP cycle when the request will time out (encoded as an integer with 64 bits); Minimum Fidelity: The desired minimum fidelity, between 0 and 1; Priority: The priority of this request; Initial Virtual Finish and Estimated Cycles Pair: Scheduling information for weighted fair queuing; STR: store flag, ATM: atomic flag, MD: measure directly flag, MR: master request flag.

cycles in the scheduler. Purpose ID of 16 bits enables pointing to 2^{16} different applications and the total number of uniquely addressed applications and follows from the number chosen for IPv4. Create ID defines the identifier of locally created request. Number of pairs enables to request up to 2^{16} pairs. Priority field of 4 bits is used as we enable 16 local queues composing the distributed queue and each one represents a priority lane. Initial Virtual Finish is used for weighted fair queuing.

A.4.2. Midpoint Heralding Protocol

The purpose of MHP is to create entanglement using a midpoint heralding protocol. The operation of the MHP is defined by Protocol A.4.2. We provide table A.6 as a reference to the reader when tracing the protocol.

| Variable/Parameter | Possible Values |
|--------------------------------|---|
| Pulse Sequence Identifier PSEQ | A bit string identifier. |
| Absolute Queue ID a_{ID} | Pair (j, i_j) where $j \in [0, 15]$ and $i_j \in [0, 255]$. |
| MHP Error mhp_{err} | QUEUE_MISMATCH (1), TIME_MISMATCH (2), NO_MESSAGE_OTHER (3), GEN_FAIL (4) |

Table A.6: Reference table for variables and parameters in the MHP protocol description.

Protocol 1 MHP for use with the Node-Centric QEGP

Definition of functions and variables.

- POLLQEGP: process to poll for entanglement parameters from QEGP; it returns:
 - flag: true/false indicating whether entanglement should be attempted or not;
 - PSEQ: The pulse sequence identifier that should be issued to the hardware to initialize communication qubit and produce spin-photon entanglement. May also instruct the hardware to store the spin state within a storage qubit.
 - a_{ID} : Absolute queue ID, i.e. (j, i_j) , of the request entanglement is being attempted for ($a_{ID,A}$ and $a_{ID,B}$ for nodes A and B , respectively);
 - params: parameters to use for the entanglement attempt such as bright state population α ;
 - mhp_{err} : error in MHP reported to QEGP through REPLY message (REPLY_A and REPLY_B sent to nodes A and B , respectively), which can take the following values:
 - GEN_FAIL: general failure that occurs locally at the MHP (failed qubit initialization; other errors). Note: this error message is passed to QEGP locally and not included in the REPLY message (see Figure A.19);
 - QUEUE_MISMATCH: an error sent by the midpoint when a_{ID} included in frame from A does not match a_{ID} included in frame from B ;
 - TIME_MISMATCH: when messages from A and B does not arrive at midpoint within the same time interval;
 - NO_MESSAGE_OTHER: when the midpoint receives a message from only one of A or B ;
 - GEN: the frame sent by A and B to the midpoint requesting entanglement. The contents include:
 - a_{ID} : same as a_{ID} above;
-

Protocol 1 (cont.) MHP for use with the Node-Centric QEGP

- REPLY_A and REPLY_B : the REPLY frames sent to A and B respectively. The contents include:
 - outcome: the outcome of the attempted entanglement at the midpoint, also encodes the error that occurred for the attempt at entanglement (see errors listed above); stored locally at mid-point.
 - seq_{MHP} : the sequence number from the MHP;
 - $a_{\text{ID,receiver}}$: The absolute queue ID that was submitted by the node receiving the REPLY.
 - $a_{\text{ID,peer}}$: The absolute queue ID that was submitted by peer node.

Initialization. Initialize sequence numbers (set initial $\text{seq}_{\text{MHP}} = 0$ at H). Start timer using a global synchronized clock.

The protocol, executed at each time step:

1. Executed at Node A or Node B :

- (a) Poll QEGP, i.e. $\text{POLLQEGP} = (\text{flag}, \text{PSEQ}, a_{\text{ID}}, \text{params})$
 - (b) If $\text{flag} = \text{true}$, i.e., we want to make entanglement:
 - i. Issue PSEQ to hardware to initialize communication qubit and produce spin-photon entanglement. PSEQ may also instruct the hardware to store spin state in a storage qubit. If any failures occur, send $\text{mhp}_{\text{err}} = \text{GEN_FAIL}$ back to the QEGP and skip to next time step.
 - ii. Use $\text{GEN} = (a_{\text{ID}})$ and transmit to midpoint upon photon emission.
 - iii. If params specifies a sequence of rotations and a measurement:
 - A. Perform the measurement basis rotation specified using $\text{ROTX1}, \text{ROTY}, \text{ROTX2} \in \text{params}$.
 - B. Measure the communication qubit and obtain result b .
 - C. Set $\text{RESULT} = (\text{outcome} = \text{null}, \text{seq}_{\text{MHP}} = \text{null}, a_{\text{ID}}, \text{err} = 000, b = b)$ and pass to QEGP.
-

Protocol 1 (cont.) MHP for use with the Node-Centric QEGP**2. Heralding station H :**

(a) Perform the following upon receipt of GEN messages:

- i. If messages from A and B do not arrive within the same time interval, let $\text{mhp}_{\text{err}} = \text{TIME_MISMATCH}$ and send $\text{REPLY}_A = (\text{mhp}_{\text{err}}, \text{seq}_{\text{MHP}}, a_{\text{ID},A}, a_{\text{ID},B})$ to A and $\text{REPLY}_B = (\text{mhp}_{\text{err}}, \text{seq}_{\text{MHP}}, a_{\text{ID},B}, a_{\text{ID},A})$ to B .
- ii. If $a_{\text{ID},A} \neq a_{\text{ID},B}$, then set $\text{mhp}_{\text{err}} = \text{QUEUE_MISMATCH}$ and send $\text{REPLY}_A = (\text{mhp}_{\text{err}}, \text{seq}_{\text{MHP}}, a_{\text{ID},A}, a_{\text{ID},B})$ to A and $\text{REPLY}_B = (\text{mhp}_{\text{err}}, \text{seq}_{\text{MHP}}, a_{\text{ID},B}, a_{\text{ID},A})$ to B .
- iii. If GEN arrives only from A , set $\text{mhp}_{\text{err}} = \text{NO_MESSAGE_OTHER}$ and send $\text{REPLY} = (\text{mhp}_{\text{err}}, \text{seq}_{\text{MHP}}, a_{\text{ID},A}, a_{\text{ID},B}=\text{null})$ to A , where $a_{\text{ID},B}=\text{null}$ indicates leaving the field as the zero string. Perform vice versa if GEN arrives only from B .
- iv. If no errors occurred then execute quantum swap. Inspect detection result within corresponding time window with $r \in \{0, 1, 2\}$ where 0 denotes failure and 1 and 2 denote the creation of states one and two respectively. If $r \in \{1, 2\}$, an increasing sequence number seq_{MHP} is chosen by the heralding station (incrementing a counter) to be sent to both A and B . Midpoint sends $\text{REPLY} = (\text{outcome}, \text{seq}_{\text{MHP}}, a_{\text{ID}}, a_{\text{ID}})$ to A and B .

3. Executed at Node A or Node B (here $a_{\text{ID},\text{local}}=a_{\text{ID},A}$, $a_{\text{ID},\text{peer}}=a_{\text{ID},B}$ in A and vice versa in B):

- (a) If $\text{REPLY} = (\text{outcome}, \text{seq}_{\text{MHP}}, a_{\text{ID},\text{local}}, a_{\text{ID},\text{peer}})$ returns from midpoint:
 - i. Set $\text{RESULT} = (\text{outcome}, \text{seq}_{\text{MHP}}, a_{\text{ID},\text{local}}, \text{err}=000, a_{\text{ID},\text{peer}}, b=\text{null})$ and pass to QEGP.
- (b) Else if $\text{REPLY} = (\text{mhp}_{\text{err}}, \text{seq}_{\text{MHP}}, a_{\text{ID},\text{local}}, a_{\text{ID},\text{peer}})$ returns from the midpoint:
 - i. Set $\text{RESULT} = (\text{outcome}=0, \text{seq}_{\text{MHP}}, a_{\text{ID},\text{local}}, \text{err}=\text{mhp}_{\text{err}}, a_{\text{ID},\text{peer}}, b=\text{null})$ and pass to QEGP.

MHP Sequence Diagrams

The MHP sequence diagram is defined by two cases: the successful—see Figure A.16, and unsuccessful one—see Figure A.17. Specifically, there are three failure scenarios that may occur in the MHP protocol: queue mismatch error (Figure A.17a)—where the message consistency check fails at the midpoint—, single-sided transmission error (Figure A.17b) and time mismatch of the messages arriving at H .

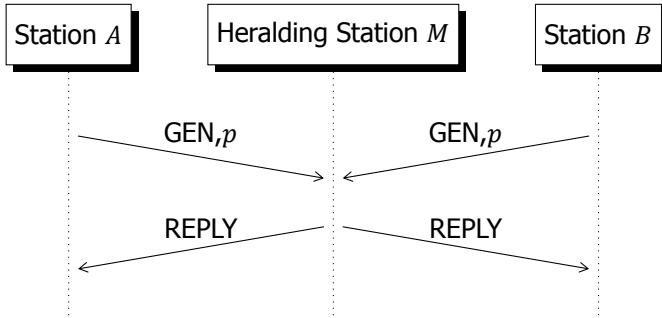


Figure A.16: Timeline of the MHP message exchange with a successful reply by the heralding station; p is a photon associated with the GEN message. For a definition of GEN and REPLY message refer to Figure A.18 and Figure A.19, respectively.

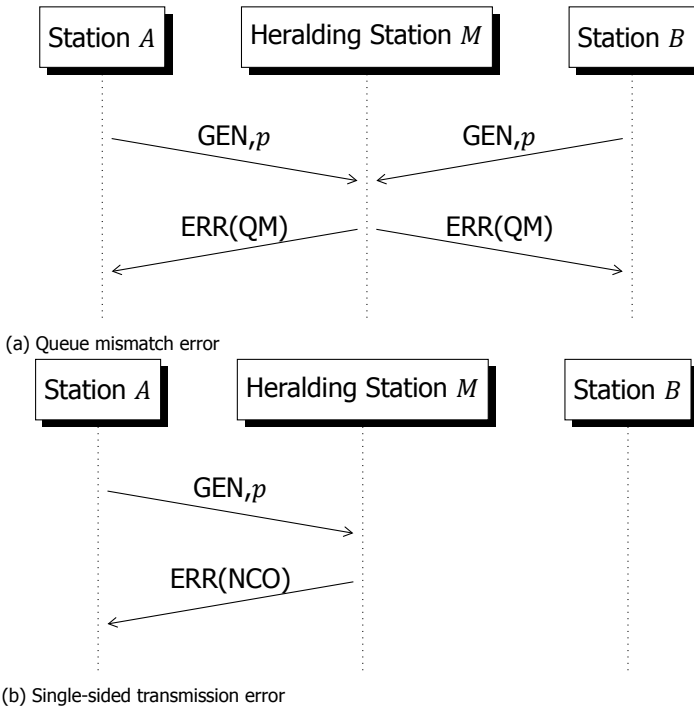


Figure A.17: Timeline of two types of errors within MHP. For a definition of GEN and REPLY message refer to Figure A.18 and Figure A.19, respectively. QM and NCO refer to specific fields of the REPLY message (i.e. OT field), i.e. QUEUE_MISMATCH and NO_MESSAGE_OTHER, respectively; both error types are explained in Protocol A.4.2.

MHP Packet Formats

MHP relies on the exchange of the packets listed in the MHP sequence diagrams, see Figure A.16 and Figure A.17: GEN and REPLY. Table A.7 shows the header encoding

for the messages (we detail the packet specification for MHP_UP in section A.4.3).

| Type | Encoding |
|--------|--------------|
| GEN | 000000000001 |
| RESULT | 000000000010 |
| REPLY | 000000000011 |
| ERROR | 000000000100 |
| MHP_UP | 000000000101 |

Table A.7: MHP packet header encodings.

GEN packet (Figure A.18) is used by the midpoint to determine whether the nodes are consistent in their local information regarding their knowledge of the attempt at entanglement.

REPLY packet (Figure A.19) is sent by the midpoint in the case of no error. It will include the senders' submitted absolute queue ID (i.e. QID and QSEQ) and additionally pass on the submitted queue ID of the peer node (i.e. QIDP and QSEQP). The sequence number, SEQ, denotes the number of successful heralded entanglement generations that have occurred at the midpoint heralding station and allows the end nodes to keep track of the number of entangled pairs that have been generated. OT encodes the heralding signal from the midpoint upon successful operation and encodes errors in case of failures.

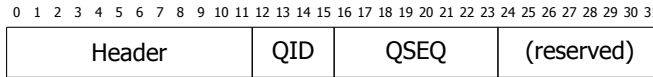


Figure A.18: GEN packet format (used in MHP) sent by end stations to heralding station (midpoint). The pair (QID, QSEQ) represents the absolute queue ID where QID and QSEQ are encoded as integers, in other words they map to (j, i_j) —see Section A.4.1.

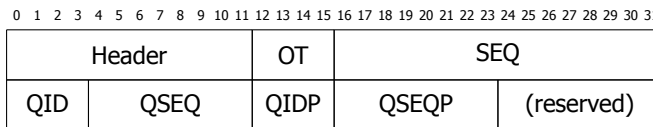


Figure A.19: RESULT/REPLY/ERR packet format (used in MHP) for replies by midpoint with no error. OT: outcome/error reported by midpoint, where 0000 encodes non-success, 0001 and 0010 encode the two different successes, and error codes include 1001: QUEUE_MISMATCH; 1010: TIME_MISMATCH, 1100: NO_MESSAGE_OTHER, (refer to Protocol A.4.2 for the above error description); SEQ: sequence number (integer in $[0, 65535]$); QIDP: QID Peer; QSEQP: QSEQ Peer; QID and QSEQ are defined the same as in for GEN message—see Figure A.18.

A.4.3. Entanglement Generation Protocol

The role of the Entanglement Generation Protocol (QEGP) is to produce the required entanglement between two end nodes or otherwise declare failure.

Entanglement Generation Scheduler

We now proceed with the description of the scheduler—refer to Protocol A.4.3 for details. The QEGP scheduler fulfills the following arbitrage functions, where we remark that for CREATE requests that demand multiple EPR pairs, only one request is added to the queue, and hence NEXT (function to select the next request from the local set of queues, see below) will return multiple pairs to be produced for the same request when called successively.

- GET_QUEUE(creq): Once a request has been submitted, GET_QUEUE chooses which queue Q_j to assign the CREATE request creq to. This may depend on the details of the request, such as for example t_{\max} , or F_{\min} as well as the purpose ID and priority.
- NEXT: Selects the next request from the local set of queues \mathcal{Q} to serve, if any. Specifically, NEXT will determine:
 - *Flag*, set to `true` when a request is ready to be served;
 - *Absolute queue ID* (and corresponding request details) of request to be served;
 - *Parameters to use in the MHP* depending on the number of type of outstanding requests;
 - *Communication and storage qubits*, determined in cooperation with QMM.

Protocol Description

Protocol A.4.3 presents a description of the Entanglement Generation Protocol. We additionally provide a reference table in table A.8 indicating values that various variables and parameters may take throughout the protocol to assist the reader.

| Variable/Parameter | Possible Values |
|--|---|
| Request Type | K (Create and Keep, 1), MD (Measure Directly, 2), RSP (Remote State Prep., 3) |
| Number of pairs n | Integer in [1,65535] |
| Minimum/Estimated Fidelity F_{\min} | Decimal number in [0,1] |
| Maximum Time t_{\max} | Integer in [0,16384] |
| Purpose ID | Integer in [0, 65535] |
| Priority | Integer in [0,15] |
| Sequence Numbers ($seq_{\text{expected}}, seq_{\text{MHP}}$) | Integer in [0,65535] |
| Queue ID | Integer in [0,15] |
| Protocol Error <code>proto_err</code> | ERR_NOSUPP (1), ERR_NOTIME (2), ERR_REJECTED (3), ERR_EXPIRE (4), ERR_TIMEOUT (5) |
| Scheduler flag | True/False |
| Qubit IDs (<code>logical_id</code>) | Integer in [0,15] |
| Rotation Angles (ROTX1,ROTY,ROTX2) | Integers in [0,255] |

Table A.8: Reference table for variables and parameters used in the QEGP.

Protocol 2 QEGP - Node A (B analogous exchanging A and B)

Definition of functions and variables.

- Node ID: the ID of the peer we want to create entanglement with;
- n : the number of entangled pairs we wish to create;
- F_{\min} : the minimum acceptable fidelity required for the generated pairs;
- t_{\max} : the maximum amount of time the higher layer is willing to wait for the entanglement to be created
- Purpose ID: port/application ID the requested this entanglement—used for forwarding OK messages to the appropriate application;
- priority: The priority of the request;
- $\text{seq}_{\text{expected}}$: The expected sequence number from the midpoint. Initially set to 1;
- $j = \text{GET_QUEUE}(c_r)$: The call to the scheduler to obtain the queue ID of Q_j where the request should be placed in the distributed queue.
- $(i_j, ok) = \text{ADD}(j, \text{creq})$: The call to the distributed queue to add the create request c_r to Q_j . i_j is the unique ID of the request within Q_j and ok is a status code of performing the ADD to the distributed queue. Can take the values `success` (item added), `timeout` (communication timeout with peer while adding), or `reject` (the peer rejected adding the item to the queue);
- ERR_NOTIME: Error issued to higher layers by the QEGP upon receiving `OK=timeout` from adding item to queue;
- ERR_REJECT: Error issued to higher layers by the QEGP upon receiving `OK=reject` from adding item to queue;
- $(\text{flag}, (j, i_j), \text{PSEQ}, \text{params}) = \text{NEXT}()$: The call to the scheduler to obtain information for the next entanglement generation attempt where `flag=True/False` indicates whether entanglement should be attempted (same in MHP outline), (j, i_j) is the absolute queue ID of the create request being served, PSEQ is a pulse sequence identifier encoding the communication and storage qubit information to use for entanglement attempts, and `params` encodes the parameters to use for entanglement attempts (same as `params` in MHP outline);
- `proto_err`: Status of the attempt at entanglement, encodes the `mhp_err` from the MHP outline if an error occurred, 0 if no errors happened;
- `create_time`: timestamp of when the entanglement was generated;
- F_{est} : the goodness passed in the frame, the estimate of the fidelity of the entangled qubits;

Protocol 2 (cont.) QEGP - Node A (B analogous exchanging A and B)

- `logical_id`: The storage qubit ID where the entangled qubit is stored, for use by higher layers;
- `tGoodness` - A timestamp of when F_{est} was record.
- `k`: the number of pairs left to generate for the request.
- `RB`: A buffered reply from the MHP initialized to null.
- `MB` and `BB`: Buffers containing measurement results and basis choices respectively, initialized to be empty.
- `TRSP`: A table containing locally stored basis information for Remote State Preparation requests, initialized to be empty.
- `ROTX1(Y,X2)`: Rotation angles (multiples of $\frac{\pi}{255}$) used to perform basis rotations for requests of type Measure Directly and Remote State Preparation.

Initialization. Query Physical Translation Unit (PTU) and load rotation angles (ROTX1, ROTY, ROTX2) for random bases $\{X, Y, Z, \frac{Z+X}{2}, \frac{Z-X}{2}\}$. Establish the distributed queue with peer and negotiate master-slave relationship. Exchange timing, device capability, and permitted purposeID information with peer for request validation.

The protocol.

1. Adding to Queue:

- (a) Validate the request against exchanged device capabilities of A and B. If validation fails, issue `ERR_NOSUPP` and stop.
 - (b) Ask scheduler which queue this request should be added to: $j = \text{GET_QUEUE}(c_r)$.
 - (c) If the request is of type RSP, strip the ROTX1, ROTY, ROTX2 basis information and store in a local variable I_b . Try to add request to the queue using the DQP: $(i_j, ok) = \text{ADD}(j, c_r)$.
 - (d) If $ok = \text{timeout error}$, issue `ERR_NOTIME` and stop.
 - (e) If $ok = \text{reject error}$, issue `ERR_REJECTED` and stop.
 - (f) Otherwise the request has been added to the Distributed Queue. If the request was of type RSP, add the (key,value) pair $((j, i_j), I_b)$ to T_{RSP} .
-

Protocol 2 (cont.) QEGP - Node A (B analogous exchanging A and B)**2. POLLQEGP (triggers pair generation, polled by MHP):**

- (a) Ask the scheduler whether entanglement should be made and for which request: $\text{NEXT} = (\text{flag}, (j, i_j) \equiv \text{req}, \text{params})$. For this end, the scheduler will employ its priority policy, as well as perform flow control depending on whether B is likely to produce entanglement.
- (b) If there is a generation waiting to be satisfied:
 - i. Query the FEU for the parameter α to be used for the generation attempt, obtain translation to hardware instruction PSEQ.
 - ii. If the request is of type MD and specifies a random basis, store a sampled basis B following the request's probability distribution in B_B and obtain preloaded $\text{ROTX1}, \text{ROTY}, \text{ROTX2}$ angles for B . Otherwise, use the request specified $\text{ROTX1}, \text{ROTY}, \text{ROTX2}$. Query the PTU with the selected angles and obtain gate instruction sequence G . Add G to params.
 - iii. If the request is of type RSP and this is the creator node of the request, obtain values $\text{ROTX1}, \text{ROTY}, \text{ROTX2}$ from T_{RSP} using (j, i_j) and query the PTU to obtain gate instruction sequence G . Add G to params.
 - iv. Construct response for MHP $\text{POLLQEGP}() = (\text{flag}, (j, i_j), \text{PSEQ}, \text{params})$.
 - v. Provide the response to the MHP.
- (c) Otherwise if there are no generations to perform provide $\text{POLLQEGP}() = (\text{flag} = \text{False}, a_{\text{ID}, \text{local}} = \text{null}, \text{PSEQ} = \text{null}, \text{params} = \text{null})$.

3. Handle Measurement reply (message from MHP):

- (a) Retrieve message from MHP including: absolute queue id (j, i_j) , and measurement result b .
- (b) Store the result m and absolute queue id (j, i_j) in M_B .
- (c) If $R_B \neq \text{null}$ then Handle Midpoint reply with R_B , set $R_B = \text{null}$. Otherwise wait for reply.

Protocol 2 (cont.) QEGP - Node A (B analogous exchanging A and B)

4. Handle Midpoint reply (message from MHP):

- (a) Retrieve message from MHP including: result of generation $r \in \{0, 1, 2\}$, seq_{MHP} , absolute queue id (j, i_j) , and protocol error flag proto_err .
 - (b) If the specified absolute queue ID is not found locally then the request may have timed out or expired:
 - i. Free the reserved communication/storage qubit in the Quantum Memory Manager.
 - ii. Update $\text{seq}_{\text{expected}}$ with $\text{seq}_{\text{MHP}}+1$ and stop handling reply.
-

Protocol 2 (cont.) MHP for use with the Node-Centric QEGP**4. (cont.) Handle Midpoint reply (message from MHP):**

- (c) Otherwise there is an absolute queue ID included in the response that is associated with an active request:
- i. If the absolute queue id (j, i_j) belongs to an MD request and M_B is empty, set $R_B = (r, \text{seq}_{\text{MHP}}, (j, i_j), \text{proto_err})$, and stop handling reply.
 - ii. Otherwise, remove b and (j, i_j) from the head of M_B and continue.
 - iii. If proto_err is not OK, a non quantum error occurred and no entanglement was produced. Update expected sequence number with seq_{MHP} and stop handling reply.
 - iv. If $r = 0$ we failed to produce entanglement, stop handling reply.
 - v. Process seq_{MHP} :
 - A. If the seq_{MHP} is larger than the expected sequence number, (partially) ERR_EXPIRE the request to higher layer and send EXPIRE message to peer. Stop handling reply.
 - B. Else. if seq_{MHP} is smaller than the expected sequence number, ignore reply. Stop handling reply.
 - C. Else, update next expected seq_{MHP} sequence number (increment current one modulo 2^{16}).
 - vi. A pair is established. If $r = 2$ and we are the origin of this request, apply correction information to transform state $|\Psi^-\rangle$ to state $|\Psi^+\rangle$, if we are the peer then we instruct the scheduler to suspend subsequent generation attempts until we believe the request originator has completed correction.
 - vii. Look up queue item (j, i_j) :
 - A. If $\text{create_time} + t_{\text{max}} > \text{current time}$ or the request is not stored locally anymore, issue ERR_TIMEOUT to higher layer and remove item from queue. Stop handling reply.
 - B. Get fidelity estimate F_{est} from Fidelity Estimation Unit.
 - C. Construct OK with the CreateID, Entanglement ID: $(A, B, \text{seq}_{\text{MHP}})$, Directionality, Remote Node ID, and Goodness F_{est} . If the request is of type K, include logical_id , $t_{\text{Goodness}} = \text{now}$. If the request is of type MD, include the oldest measurement b from M_B and measurement basis B from B_B if a random basis was specified, otherwise $B = \text{null}$. If the request is of type R and we are the state preparer, include the measurement b .
 - D. If $k = 1$, i.e. this was the last pair to be produced for this request, remove item from queue.
 - E. If $k > 2$, decrement k on queue item.

QEGP Sequence Diagrams

We proceed with the introduction of all message passing sequence diagrams for the QEGP.

Figure 2.4 presents a sequence diagram for the QEGP/MHP operation when performing emission multiplexing. In some cases (such as the M use case) it is not necessary to wait for the REPLY message from the midpoint before attempting entanglement generation again if one simply desires to generate correlated bit streams.

Figure A.20 shows a sequence diagram detailing the message flow should station A receive a message from H such that seq_{MHP} is less than $\text{seq}_{\text{MHP}_r}$, as well as a timeline of the message exchange when QEGP processes at both nodes exchange available quantum memory information. Sharing this information allows both nodes to know whether there are available resources in order to proceed with satisfying an entanglement request. In the absence of resources of either peer there is no use in photon emission. Simply, both nodes must be able to emit photons for the protocol to operate properly.

Imperfect message transmission may cause any of the GEN, REPLY, EXPIRE, REQ(E), and ACK messages to become lost or corrupted in transit between nodes. Depending on which messages are lost in the protocol, different actions are taken to prevent deadlock. A lost EXPIRE or corresponding ACK results in a retransmission of the EXPIRE to ensure that OK messages are properly revoked at the peer node. Loss of a REQ(E) message or its corresponding ACK results in a retransmission of the REQ(E) to make sure that both nodes have up-to-date information of available resources.

Losing a GEN message is handled by the midpoint heralding station when only one GEN message arrives from an end node. In this case the REPLY message containing NO_CLASSICAL_OTHER (see Protocol A.4.2) is issued to alert the nodes of the failure. In this case no attempt at entanglement is made and the sequence number at the midpoint remains the same.

Losing a REPLY message from the midpoint that contains an outcome of 0 has no impact on A or B as $\text{seq}_{\text{expected}}$ is only updated when a successful attempt at entanglement occurs. When a REPLY message containing an outcome of 1 or 2 (for successful entanglement) is lost, the end node(s) that lost the message will continue attempting entanglement generation in subsequent polls by the MHP as there are outstanding pairs to be generated for the request. Upon successful receipt of any message from the midpoint (REPLY), the included SEQ will be ahead of the receiving node(s) $\text{seq}_{\text{expected}}$ and the loss will be detected. The detecting node will then transmit an EXPIRE message to its peer containing the old $\text{seq}_{\text{expected}}$ that did not agree with the SEQ received from the midpoint along with its new $\text{seq}_{\text{expected}_r}$, so that any OK messages containing the missing set of sequence numbers are revoked at the peer.

QEGP Packet Formats

Finally, we present the definitions of all messages being used by the QEGP. Figure A.25 describes the information passed from the QEGP to the MHP during each

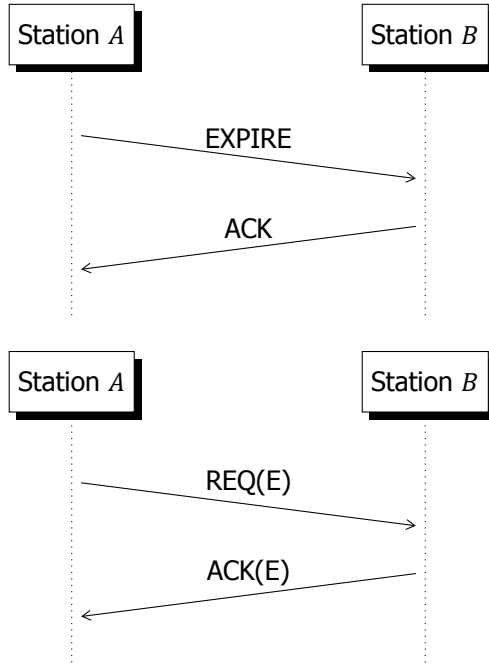


Figure A.20: (Above) Timeline of request expiration within QEGP. Definitions of EXPIRE and ACK message are given in Figure A.22 and Figure A.23, respectively. (Below) Timeline of memory advertisement requests within QEGP. Definition of REQ(E) and ACK(E) message is given in Figure A.24.

periodic cycle, while Figure A.26 shows the packet format for replies from the physical layer MHP to the QEGP. Figure A.22 and Figure A.23, define EXPIRE and ACK messages, respectively, exchanged in entanglement request expiration sequence diagram described in Figure A.20. Figure A.24 shows the REQ(E) and ACK(E) packet formats exchanged by memory advertisement requests made by the QEGP sequence diagram described in Figure A.20. Figure A.27 and Figure A.28 present the format of OK messages passed from the QEGP to higher layers, in case of *create and keep request* and *measure directly request*, respectively. Table A.9 presents an encoding of the headers in the messages.

| Type | Encoding |
|----------|--------------|
| EXPIRE | 000000001001 |
| ACK | 000000001010 |
| REQ(E) | 000000001011 |
| ACK(E) | 000000001100 |
| POLLQEGP | 000000001101 |

Table A.9: QEGP packet header encoding.



| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|----|-----|------|--------|--|--|--------|--|--|--|-------|--------|----------|-------|--|--|--|--------|--|--|--|--|--|--|--|--|--|
| 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Remote Node ID | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Minimum Fidelity | | | | | | | | | | | | | | | tu | Max Time | | | | | | | | | | | | | | |
| Purpose ID | | | | | | | | | | | | | | | Number | | | | | | | | | | | | | | | |
| Priority | T | A | C | RL | RR | rsvd | PL1 | | | | | | | PL2 | | | | | | | | | | | | | | | | |
| PR1 | | | | | PR2 | | | | | ROTX1L | | | | | | | ROTYL | | | | | | | | | | | | | |
| ROTX2L | | | | | | | ROTX1R | | | | | | | ROTYR | | | | | | | ROTX2R | | | | | | | | | |

Figure A.21: Packet format for CREATE message to QEGP. Explanation of the message fields— Remote Node ID— Used if the node is directly connected to multiple nodes. Indicates which node to generate entanglement with; Minimum Fidelity—The desired minimum fidelity, between 0 and 1, of the generated entangled pair; tu—The time units to use for interpreting Max Time, where (00,01,10) each indicate (μ s,ms,s) respectively and 11 is unused; Max Time —The maximum number of specified time units the higher layer is willing to wait for the request to be fulfilled; Purpose ID—Allows the higher layer to tag the request for a specific purpose; Number—The number of entangled pairs to generate; Priority—Can be used to indicate if this request is of high priority and should ideally be fulfilled early; T—the type of request, one of create and keep (K, encoded as 01), measure directly (M, encoded as 10), or remote state preparation (RSP, encoded as 11), where K stores the generated entanglement in memory, M measures the entanglement directly, and RSP performs a rotation on the qubit before measuring it to prepare the state at the remote node; A—atomic flag, indicates that the request should be satisfied as a whole, i.e. that all entangled pairs are available in memory at the same time; C—consecutive flag, indicates that an OK is returned for each pair made for a request; RL(R)—random basis choice for the local (remote) node, where 00 indicates omission, 01 indicates sampling from $\{X, Z\}$, 10 indicates sampling from $\{X, Y, Z\}$, and 11 indicates sampling from $\{\frac{Z+X}{2}, \frac{Z-X}{2}\}$; PL(R)1,PL(R)2—Probability distribution to use for the random basis choice at the local (remote) node that takes values in $[0,255]$. In the case that the specified random basis has 2 elements, the distribution obeys the probabilities $(\frac{PL(R)1}{255}, 1 - \frac{PL(R)1}{255})$ whereas a random basis of 3 elements obeys $(\frac{PL(R)1}{255}, \frac{PL(R)2}{255}, 1 - \frac{PL(R)1}{255} - \frac{PL(R)2}{255})$. The supported precision for PL(R)1/2 are multiples of $\frac{1}{255}$; ROTX1L(R),ROTYL(R),ROTX2L(R)—specify a discrete multiple of $\frac{2\pi}{255}$ to be used for performing the basis rotation (see equations 1.10 and 1.11) $R_X(\frac{ROTX2*2\pi}{255})R_Y(\frac{ROTY*2\pi}{255})R_X(\frac{ROTX1*2\pi}{255})$ before measurement for M requests at the local (remote) node.

| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|--|--|--|--|--|--|--|--|--|-----|--|--|--|--|-----|--|--|--|--|------|--|--|--|--|--|--|--|--|--|------------|
| 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Header | | | | | | | | | | QID | | | | | | | | | | QSEQ | | | | | | | | | | (reserved) |
| Origin ID | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Create ID | | | | | | | | | | | | | | | SEQ | | | | | | | | | | | | | | | |

Figure A.22: Packet format for EXPIRE message. Explanation of the message fields—Origin ID: ID of the node the request originated from; Create ID: creation ID of the request. SEQ: up-to-date expected MHP sequence number at the node the EXPIRE originates from. Recall that (QID, QSEQ) represent the absolute queue ID.

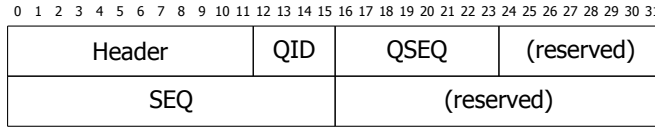


Figure A.23: Packet format for ACK message. Recall that (QID, QSEQ) is the absolute queue ID and SEQ is the acknowledger's up-to-date expected MHP sequence number (the same as in the case of EXPIRE message, see Figure A.22).

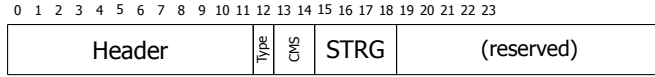


Figure A.24: REQ(E)/ACK(E) Packet format for QEGP memory requests. Explanation of the message fields—Type: message type (0: REQ(E), 1: ACK); CMS: the number of available communication qubits, STRG: the number of available storage qubits.

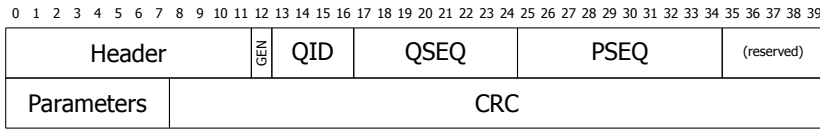


Figure A.25: Packet format of POLLQEGP messages sent from QEGP to MHP. Explanation of the message fields—GEN: emit photon flag; PSEQ: pulse sequence identifier which instructs the underlying quantum communication device of the parameters to use when emitting the photon. See fig. 2.5.

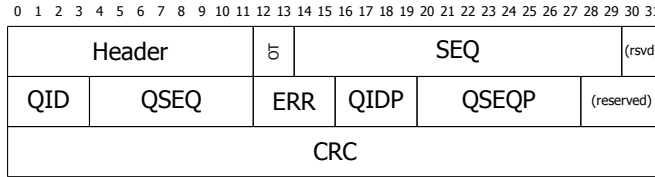


Figure A.26: Packet format of messages from MHP to QEGP. Explanation of the message fields—OT: measurement outcome (Header specifies whether OT contains the message from the heralding station or from a local message such as a measurement). Error codes in ERR field encode the errors described in the MHP protocol (refer to Protocol A.4.2). See fig. 2.5.

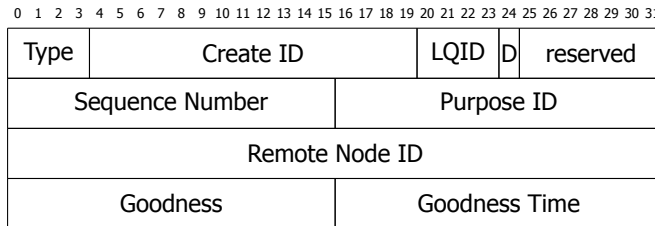


Figure A.27: Packet format for OK message corresponding to a create and keep request. Explanation of the message fields—Type: Indicates that this is a create and keep OK; Create ID: The same as the Create ID returned to the requester; LQID: Logical Qubit ID where the entanglement is stored (encoded as an integer); D: Directionality flag indicating the source of the request where 0 encodes the local node and 1 the remote node; Sequence Number: A sequence number for identifying the entangled pair (encoded as an integer); Purpose ID: The purpose ID of the request (encoded as an integer); Remove Node ID: Used if connected to multiple nodes; Goodness: An estimate of the fidelity of the generated pair (encoded as a half-precision floating point number); Goodness Time: Time of the goodness estimate. See fig. 2.5.



| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|-----------|--|--|--|--|--|--|--|--|--|--|--|--|------------|---|-------|---|----------|--|--|--|--|--|--|--|--|--|--|--|--|--|
| 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Type | Create ID | | | | | | | | | | | | | | M | Basis | D | reserved | | | | | | | | | | | | | |
| Sequence Number | | | | | | | | | | | | | | Purpose ID | | | | | | | | | | | | | | | | | |
| Remote Node ID | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Goodness | | | | | | | | | | | | | | (reserved) | | | | | | | | | | | | | | | | | |

Figure A.28: Packet format for OK message corresponding to a measure directly request. Explanation of the message fields—Type: Indicates this is an OK for a measure directly request; Create ID: The same Create ID given to the requester; M: Outcome of the measurement performed on local qubit of the entangled pair; Basis: Which basis the local qubit of the entangled pair was measured in, used if the basis is random. This field adheres to the following encoding scheme: 000: Z, 001: X, 010: Y, 011: $\frac{Z+X}{2}$, 100: $\frac{Z-X}{2}$, 101: Unused, 110: Unused, 111: Unused. The remainder of the fields are explained in Figure A.27. See fig. 2.5.

| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|-----------|--|--|--|--|--|--|--|--|--|--|--|--|------------|---|---|----------|--|--|--|--|--|--|--|--|--|--|--|--|--|--|
| 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Type | Create ID | | | | | | | | | | | | | | M | D | reserved | | | | | | | | | | | | | | |
| Sequence Number | | | | | | | | | | | | | | Purpose ID | | | | | | | | | | | | | | | | | |
| Remote Node ID | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Goodness | | | | | | | | | | | | | | (reserved) | | | | | | | | | | | | | | | | | |

Figure A.29: Packet format for OK message corresponding to a remote state preparation request. Explanation of the message fields—Type: Indicates this is an OK for a remote state preparation request; Create ID: The same Create ID given to the requester; M: Outcome of the measurement performed on local qubit of the entangled pair (only for the creator node); The remainder of the fields are explained in Figure A.27. See fig. 2.5.

| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---------------------|-----------|--|--|--|--|--|--|--|--|--|--|--|--|----------------------|-----|---|----------|--|--|--|--|--|--|--|--|--|--|--|--|--|--|
| Type | Create ID | | | | | | | | | | | | | | ERR | S | reserved | | | | | | | | | | | | | | |
| Sequence Number Low | | | | | | | | | | | | | | Sequence Number High | | | | | | | | | | | | | | | | | |
| Origin Node ID | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

Figure A.30: Packet format for ERR messages containing errors from QEGP. Explanation of the message fields—Type: Indicates this is an ERR message; ERR: The error that occurred in the QEGP, may be any of ERR_NOSUPP (0001), ERR_NOTIME (0010), ERR_REJECTED (0011), ERR_EXPIRE (0100), ERR_TIMEOUT (0101); S: Used by ERR_EXPIRE, to specify whether a range of sequence numbers should be expired (S=1) or all sequence numbers associated with the given Create ID and Origin Node (S=0); Sequence Number Low/High: Use together to specify a range of sequence numbers to expire. The remaining fields are explained in figure A.27.

References

- [1] A. Dahlberg, M. Skrzypczyk, T. Coopmans, L. Wubben, F. Rozpędek, M. Pompili, A. Stolk, P. Pawełczak, R. Knegjens, J. de Oliveria Filho, R. Hanson, and S. Wehner, *A link layer protocol for quantum networks*, in *ACM SIGCOMM 2019 Conference*, SIGCOMM '19 (ACM, New York, NY, USA, 2019) p. 15.
- [2] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*, 10th ed. (Cambridge University Press, Cambridge, 2010).
- [3] C. H. Bennett and G. Brassard, *Quantum Cryptography: Public Key Distribution, and Coin-Tossing*, in *Proc. 1984 IEEE International Conference on Computers, Systems, and Signal Processing* (1984) pp. 175–179.
- [4] C. Pfister, M. Adriaan Rol, A. Mantri, M. Tomamichel, and S. Wehner, *Capacity estimation and verification of quantum channels with arbitrarily correlated errors*, *Nature Communications* **9**, 27 (2016).
- [5] M. Tomamichel, C. C. W. Lim, N. Gisin, and R. Renner, *Tight finite-key analysis for quantum cryptography*, *Nature Communications* **3**, 634 (2012).
- [6] J. Filho, Z. Papp, R. Djapic, and J. Oostveen, *Model-based design of self-adapting networked signal processing systems*, in *Proc. SASO* (IEEE, Philadelphia, PA, USA, 2013) pp. 41–50.
- [7] C. van Leeuwen, J. de Gier, J. de Oliveira Filho, and Z. Papp, *Model-based architecture optimization for self-adaptive networked signal processing systems*, in *Proceedings of the 2014 IEEE Eighth International Conference on Self-Adaptive and Self-Organizing Systems*, SASO '14 (IEEE Computer Society, Washington, DC, USA, 2014) pp. 187–188.
- [8] J. Filho, T. Vogel, and J. de Gier, *Runtime services and tooling for reconfiguration*, in *Runtime Reconfiguration in Networked Embedded Systems: Design and Testing Practices* (Springer Singapore, Singapore, 2016) pp. 69–92.
- [9] P. C. Humphreys, N. Kalb, J. P. Morits, R. N. Schouten, R. F. Vermeulen, D. J. Twitchen, M. Markham, and R. Hanson, *Deterministic delivery of remote entanglement on a quantum network*, *Nature* **558**, 268 (2018).
- [10] SURFsara, *Cartesius*, <https://userinfo.surfsara.nl/systems/cartesius> (2018).
- [11] A. Dahlberg, M. Skrzypczyk, T. Coopmans, L. Wubben, F. Rozpędek, M. Pompili, A. Stolk, P. Pawełczak, R. Knegjens, J. de Oliveira Filho, R. Hanson, and S. Wehner, *Data from simulations*, <https://dataverse.nl/dataverse/QLinkLayer> (2019).
- [12] H. Bernien, *Control, measurement and entanglement of remote quantum spin registers in diamond*, *Phd thesis*, TU Delft (2014).

- [13] M. H. Abobeih, J. Cramer, M. A. Bakker, N. Kalb, M. Markham, D. J. Twitchen, and T. H. Taminiau, *One-second coherence for a single electron spin coupled to a multi-qubit nuclear-spin environment*, [Nature Communications](#) **9**, 2552 (2018), [arXiv:1801.01196](#) .
- [14] C. E. Bradley, J. Randall, M. H. Abobeih, R. Berrevoets, M. Degen, M. A. Bakker, R. F. L. Vermeulen, M. Markham, D. J. Twitchen, and T. H. Taminiau, *A 10-qubit solid-state spin register with quantum memory up to one minute*, [arXiv preprint](#) (2019), [arXiv:1905.02094 \[quant-ph\]](#) .
- [15] N. Kalb, A. A. Reiserer, P. C. Humphreys, J. J. W. Bakermans, S. J. Kamerling, N. H. Nickerson, S. C. Benjamin, D. J. Twitchen, M. Markham, and R. Hanson, *Entanglement distillation between solid-state quantum network nodes*, [Science](#) **356**, 928 (2017), [arXiv:1703.03244](#) .
- [16] T. H. Taminiau, J. Cramer, T. van der Sar, V. V. Dobrovitski, and R. Hanson, *Universal control and error correction in multi-qubit spin registers in diamond*, [Nature nanotechnology](#) **9**, 171 (2014).
- [17] A. Reiserer, N. Kalb, M. S. Blok, K. J. van Bemmelen, T. H. Taminiau, R. Hanson, D. J. Twitchen, and M. Markham, *Robust Quantum-Network Memory Using Decoherence-Protected Subspaces of Nuclear Spins*, [Physical Review X](#) **6**, 021040 (2016), [arXiv:1603.01602](#) .
- [18] J. Cramer, N. Kalb, M. A. Rol, B. Hensen, M. S. Blok, M. Markham, D. J. Twitchen, R. Hanson, and T. H. Taminiau, *Repeated quantum error correction on a continuously encoded qubit by real-time feedback*, [Nature communications](#) **7**, 11526 (2016).
- [19] N. Kalb, P. C. Humphreys, J. J. Slim, and R. Hanson, *Dephasing mechanisms of diamond-based nuclear-spin memories for quantum networks*, [Physical Review A](#) **97**, 1 (2018).
- [20] F. Rozpędek, R. Yehia, K. Goodenough, M. Ruf, P. C. Humphreys, R. Hanson, S. Wehner, and D. Elkouss, *Near-term quantum-repeater experiments with nitrogen-vacancy centers: Overcoming the limitations of direct transmission*, [Phys. Rev. A](#) **99**, 052330 (2019).
- [21] D. E. Amos, *Computation of Modified Bessel Functions and Their Ratios*, [Mathematics of Computation](#) **28**, 239 (1974).
- [22] D. Riedel, I. Söllner, B. J. Shields, S. Starosielec, P. Appel, E. Neu, P. Maletinsky, and R. J. Warburton, *Deterministic enhancement of coherent photon generation from a nitrogen-vacancy center in ultrapure diamond*, [Physical Review X](#) **7**, 031040 (2017).
- [23] S. Zaske, A. Lenhard, C. A. Keßler, J. Kettler, C. Hepp, C. Arend, R. Albrecht, W.-M. Schulz, M. Jetter, P. Michler, and C. Becher, *Visible-to-Telecom Quantum Frequency Conversion of Light from a Single Quantum Emitter*, [Physical Review Letters](#) **109**, 147404 (2012).

- [24] B. Hensen, H. Bernien, A. E. Dréau, A. Reiserer, N. Kalb, M. S. Blok, J. Ruitenbergh, R. F. L. Vermeulen, R. N. Schouten, C. Abellán, W. Amaya, V. Pruneri, M. W. Mitchell, M. Markham, D. J. Twitchen, D. Elkouss, S. Wehner, T. H. Taminiau, and R. Hanson, *Loophole-free Bell inequality violation using electron spins separated by 1.3 kilometres*, *Nature* **526**, 682 (2015), [arXiv:1508.05949](https://arxiv.org/abs/1508.05949) .
- [25] C. K. Hong, Z. Y. Ou, and L. Mandel, *Measurement of subpicosecond time intervals between two photons by interference*, *Physical Review Letters* **59**, 2044 (1987).
- [26] A. M. Branczyk, *Hong-Ou-Mandel Interference*, *arXiv preprint* (2017), [arXiv:1711.00080](https://arxiv.org/abs/1711.00080) [quant-ph] .
- [27] Cisco, *Calculating the Maximum Attenuation for Optical Fiber Links*, (2005).
- [28] G. V. G. . C. KG, *SFP-ZX 1.25 Gb/s Single-Mode SFP Transceiver 1000BASE-ZX*, (2013).
- [29] L. B. James, *Error Behaviour in Optical Networks*, *Ph.D. thesis*, University of Cambridge (2005).
- [30] N.-. Consortium, *NS-3 Documentation: ns3::WifiRemoteStationInfo Class Reference*, (2019).
- [31] E. Corndorf, C. Liang, G. S. Kanter, P. Kumar, and H. P. Yuen, *Quantum-noise-protected data encryption for WDM fiber-optic networks*, *SIGCOMM Comput. Commun. Rev.* **34**, 21 (2004).
- [32] T. Fujiwara, T. Kasami, and S. Lin, *Error detecting capabilities of the shortened Hamming codes adopted for error detection in IEEE Standard 802.3*, *IEEE Trans. Commun.* **37**, 986 (1989).

B

Transforming graph states using single-qubit operations

Axel Dahlberg, Stephanie Wehner

Parts of this chapter have been published in Phil. Trans. R. Soc. [1].

B.1. Measuring without disconnections

In this section we show that if it is possible to go from one graph state $|G\rangle$ to another $|G'\rangle$ by a sequence of Pauli measurements on some nodes, where G and G' are both connected and labelled, then one can always do this in way such that the graphs at each step are always connected.¹ The formal result is stated in theorem B.1.1 together with the proof below. The theorem is something that can be used to improve the runtime of we make use of in algorithm 5.1. The algorithm can be changed so that whenever the recursion hits a disconnected graph it stops searching that branch. This allows for searching less branches will still guaranteeing that a solution will be found if it exists, since as the theorem state, if a solution exists there is certainly one which has no disconnected intermediate graphs.

We will use the same notation as defined in definition 4.5.2. Additionally, we will denote a subsequence of a sequence \mathbf{v} as $\mathbf{v}_i^j = (v_i, v_{i+1}, \dots, v_{j-1}, v_j)$, where the subscript and superscript denotes the start- and end-index, respectively. Let \mathbf{u} be a sequence of vertices such that each vertex in $V(G) \setminus V(G')$ occurs exactly once in \mathbf{u} . We can now define the set of measurement operation sequences that take $|G\rangle$ to $|G'\rangle$ by measuring the vertices $V(G) \setminus V(G')$ in the order \mathbf{u} as

$$\mathcal{S}_{\mathbf{u}}(G \rightarrow G') \equiv \{P_{\mathbf{u}} \in \mathcal{P}_{\mathbf{u}} \mid P_{\mathbf{u}}(G) \leftrightarrow_{\text{LC}} G'\}. \quad (\text{B.1})$$

To simplify notation we will sometimes write $P_{\mathbf{u}_i^j}$ to mean

$$P_{\mathbf{u}_i^j} = \left(P_{\mathbf{u}_i^j} \right)_i^j. \quad (\text{B.2})$$

Next follows the main theorem of this section.

Theorem B.1.1. *Assume that G and G' are labelled connected graphs such that $V(G') \subseteq V(G)$. Furthermore, assume that the set $U = V(G) \setminus V(G')$ is given some ordering $\mathbf{u} = (u_1, \dots, u_k)$, where $k = |U|$. If $|\mathcal{S}_{\mathbf{u}}(G \rightarrow G')| > 0$, then there exist a $P_{\mathbf{u}} \in \mathcal{S}_{\mathbf{u}}(G \rightarrow G')$ such that $P_{\mathbf{u}_i}(G)$ is connected $\forall i, 1 \leq i \leq k$. \diamond*

Proof. We will prove this by induction on the variable k , the number of measurements performed. Lets denote the statement in theorem B.1.1 for a fixed k by $P[k]$. That $P[1]$ is true is easy to see, since in this case there is only one measurement to be performed. If $|\mathcal{S}_{\mathbf{u}}(G \rightarrow G')| > 0$, for $k = 1$, then $P_{\mathbf{u}_1}(G) = G'$ is connected, $\forall P_{\mathbf{u}} \in \mathcal{S}_{\mathbf{u}}(G \rightarrow G')$, since G' is assumed to be connected. Let's now fix a k and assume that $P[k]$ is true. Let G and G' be labelled connected graphs such that $V(G') \subseteq V(G)$ and $|V(G) \setminus V(G')| = k + 1$. Furthermore, let \mathbf{u} be an ordering of $U = V(G) \setminus V(G')$ such that² $|\mathcal{S}_{\mathbf{u}}(G \rightarrow G')| > 0$. Now take some $P_{\mathbf{u}} \in \mathcal{S}_{\mathbf{u}}(G \rightarrow G')$, which is possible since the set is assumed to be non-empty. What we need to show is that there exist a sequence $P'_{\mathbf{u}} \in \mathcal{S}_{\mathbf{u}}(G \rightarrow G')$, not necessarily different from $P_{\mathbf{u}}$,

¹Throughout this section we will always mean connected except for the vertex which is actually measured.

²Note that the fact that the set is non-empty does not depend on \mathbf{u} , according to corollary 4.5.3.1.

such that each intermediate graph in the measurement operation sequence is connected. Lets split this into two cases. Firstly, if $P_{u_1}(G)$ is connected, then we can directly use $P[k]$ since

$$P_{\mathbf{u}_2^{k+1}} \in \mathcal{S}_{\mathbf{u}_2^{k+1}}(P_{u_1}(G) \rightarrow G') \quad (\text{B.3})$$

which tells us that there exist a sequence of measurement that takes $P_{u_1}(G)$ to G' where each intermediate graph is connected.

In the other case we assume that $P_{u_1}(G)$ is not connected. We will now further split this into three cases where we consider P_{u_1} being Z , Y or X .

- $s_1 = Z$: We now have that a Z -measurement on vertex u_1 gives a disconnected graph³. Since we know that by performing the rest of the measurements in $P_{\mathbf{u}}$ we can reach the connected graph G' and that the measurements operations cannot connect disconnected components, it must be the case that all the vertices in $V(G')$ are in a single connected component of Z^{u_1} . Lets denote this connected component by G_1 and the rest of the vertices by G_2 , see figs. B.1a and B.1b for an illustration. This implies that it is possible to go from G_1 to G' by the corresponding measurements in $P_{\mathbf{u}}$. To find an $P'_{\mathbf{u}}$ which does not disconnect but still gives G_1 and finally G' , we instead measure u_1 in the Y -basis. This gives the graph in fig. B.1c, where \bar{G}_i denotes the complementation that occurs for the neighbors of u_1 when this is measured in the Y -basis. Since the Z -measurement on u_1 disconnected G , we know that no vertex in $N_{u_1} \cap G_2$ is adjacent to any vertex in $N_{u_1} \cap G_1$, in the original graph G . Therefore all vertices in $N_{u_1} \cap G_2$ are connected to all vertices in $N_{u_1} \cap G_1$ after the Y -measurement, showing that this graph is connected. If we after the Y -measurement also measure all vertices in \bar{G}_2 in the Z -basis, except a single vertex which was in $N_{u_1} \cap G_2$ denoted \tilde{g}_2 , we reach the graph in fig. B.1d. Finally \tilde{g}_2 can be measured in the Y -basis to remove the complementation which occurred in G_1 and we again get the graph G_1 . We have therefore showed that we can go from the connected graph $Y^{u_1}(G)$ to G' by k measurements. Although this measurement operation sequence might have a different ordering than \mathbf{u} , we know from corollary 4.5.3.1 that a sequence for \mathbf{u} then also exists. From the assumption $P[k]$ we know this can also be done without disconnected intermediate graphs.
- $s_1 = Y$: This case is in some sense the complement of the previous. Similarly to before we know that all vertices in $V(G')$ has to be in a single connected component of $Y^{u_1}(G)$ which we denote \bar{G}_1 and the rest of the vertices by \bar{G}_2 , see figs. B.1a and B.1b. Since the Y -measurement is assumed to disconnect G_1 and G_2 , we know that all vertices in $N_{u_1} \cap G_2$ must be connected to all the vertices in $N_{u_1} \cap G_1$, in the original graph G . This is then still true if we instead perform a Z -measurement on u_1 , as in fig. B.1c. As for the case $s_1 = Z$, we then measure all vertices in $G_2 \setminus \{\tilde{g}_2\}$ in the Z -basis and finish

³In graph-theoretical terms this means that u_1 is a cut vertex, since the operation Z^{u_1} just removes the vertex u_1 .

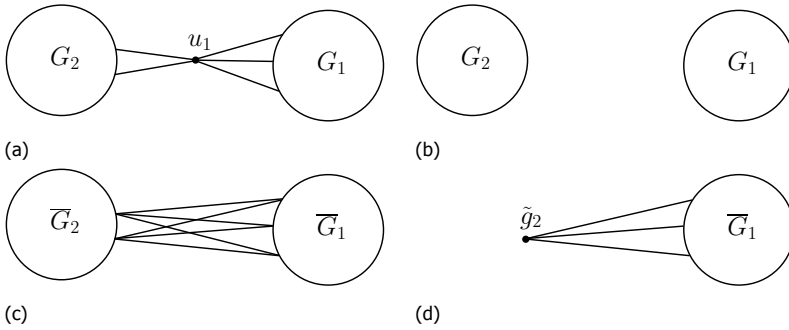


Figure B.1: The original graph G considered for the case $s_1 = Z$ is depicted in fig. B.1a. The graphs after a Z - and a Y -measurement on vertex u_1 are shown in figs. B.1b and B.1c, respectively. Finally, fig. B.1d shows the graph reached by measuring all vertices in \bar{G}_2 in the Z -basis except one of the original neighbors of u_1 , starting from the graph in fig. B.1c. This shows that instead of a Z -measurement, a Y -measurement can be done to not disconnect the rest of the graph but still reach same graph by the end of the measurement-sequence.

with a Y -measurement on \tilde{g}_2 , see fig. B.1d. We then reach \bar{G}_1 from which it is possible to go to G' with the corresponding measurements in s . From the same reasoning as in $s_1 = Z$ together with $P[k]$ we know that this can be done with no disconnected intermediate graphs.

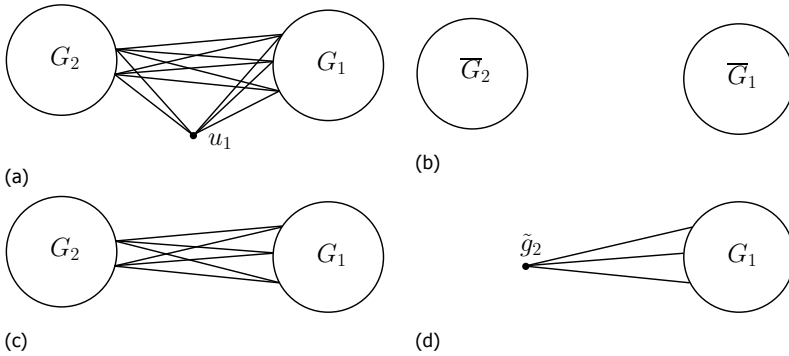


Figure B.2: The original graph G considered for the case $s_1 = Y$ is depicted in fig. B.2a. The graphs after a Y - and a Z -measurement on vertex u_1 are shown in figs. B.2b and B.2c, respectively. Finally, fig. B.2d shows the graph reached by measuring all vertices in G_2 in the Z -basis except one of the original neighbors of u_1 , starting from the graph in fig. B.2c. This shows that instead of a Y -measurement, a Z -measurement can be done to not disconnect the rest of the graph but still reach same graph by the end of the measurement-sequence.

- $s_1 = X$: The final case is when the measurement is in the X -basis and which is the most complicated. A general graph for this case is visualized in fig. B.3a. Note that the groups of vertices C, D, D' and F might include other vertices which are not neighbors of either u_1 or b and that the edges going between for example u_1 and F might be multiple, but only one is drawn for simplic-

ity. Vertices in for example c might be connected to vertices in f , but not necessarily, hence the dotted lines. Recall that we define the sets c , d and f as

$$c = N_{u_1} \cap N_b, \quad d = N_b \setminus N_{u_1}, \quad f = N_{u_1} \setminus N_b. \quad (\text{B.4})$$

After a X -measurement on u_1 with b as a special neighbor, the graph depicted in fig. B.3b is reached. Since the X -measurement flips the adjacency property of two vertices in different sets from c , d and f , these edges are now drawn as dashed. From the properties of the X -measurement we see that there will always be a connected component containing $\{b\} \cup c \cup f$ and possibly together with some vertices in d . Other connected components can only include vertices in d if these are connected to all vertices in $c \cup f$, lets denote these components by D' . The vertices in G' can then either be in D' or the connected component containing b . If they are in D' we can instead do a Z -measurement on u_1 , which will not give a disconnected graph since we know there exist a vertex in d which is connected to all vertices in $c \cup f$. Then all vertices in $\{b\} \cup c \cup d \cup f$ can also be measured in the Z -basis producing the graph D' , from which it is possible to go to G' by the corresponding measurements in s . Again by the reasoning as for $s_1 = Z$ we can use $P[k]$ to conclude that we can reach G' with no disconnected intermediate graphs. On the other hand if the vertices $V(G')$ are in the connected component containing b , we will instead measure u_1 in the Y -basis, producing the connected graph in fig. B.3c. Then pick a vertex \tilde{d} in D' which is connected to all vertices in $c \cup f$, which we know exists from the reasoning above. Measure all vertices in D' in the Z -basis except \tilde{d} which is measured in the Y -basis last. This produces the graph in fig. B.3d. By applying a local complementation on b we see that this graph is LC-equivalent to the connected component in fig. B.3b containing b . Since we can go to G' from this connected component, this is then also true for the graph in fig. B.3d, because the LC-operations relating these graphs just gives a different measurement sequence compared to s . This shows that also in this case it is possible to go to G' with no disconnected intermediate graphs, using $P[k]$. Note that this is also valid if c or f is empty.

□

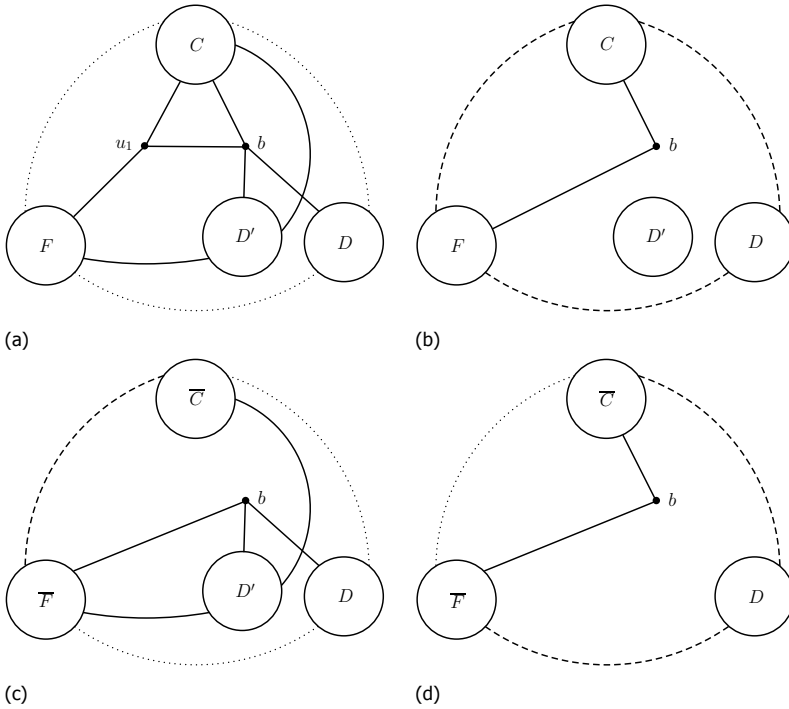


Figure B.3: The original graph G considered for the case $s_1 = X$ is depicted in fig. B.3a. The graphs after a X - and a Y -measurement on vertex u_1 , with b as special neighbor for the X -measurement, are shown in figs. B.3b and B.3c, respectively. Finally, fig. B.3d shows the graph reached by measuring all vertices in a' in the Z -basis except one of the original neighbors of b and then measuring this in the X -basis with b as the special neighbor, starting from the graph in fig. B.3c. This shows that instead of a X -measurement, a Y -measurement can be done to not disconnect the rest of the graph but still reach same graph by the end of the measurement-sequence.

B.2. Corrections from sequence of Pauli Z measurement

By performing a measurement in the standard basis of a qubit v which is part of a graph state $|G\rangle$, one can effectively disconnect qubit v from the rest of the state and produce the state $|0\rangle_v \otimes |G \setminus v\rangle$. Depending on the measurement outcome, certain single-qubit Clifford operations need to be performed to map the post-measurement state to $|0\rangle_v \otimes |G \setminus v\rangle$, as described in section 4.4.2. One can therefore effectively cut out a graph state on a subset of the qubits V' , i.e. producing the state $|G[V']\rangle$, by measuring the qubits in $V(G) \setminus V'$ in the standard basis and performing certain single-qubit Clifford operations. Here we show what corrections need to be applied to the qubits V' , such that the post-measurement state is mapped to the state $|G[V']\rangle$.

Let's assume that $|G\rangle$ is a graph state and we wish to transform this to $|G[V']\rangle$, by measuring the qubits $U = V(G) \setminus V'$ in the standard basis. Let's denote the qubits

in U as $\{v_1, v_2, \dots, v_{n-k}\}$ and the measurement outcome of qubit v_i by⁴ $x_i \in \{0, 1\}$. Furthermore, let's denote the projectors in the Z basis as $P_v^{(0)} = P_v^{(Z,+)}$ and $P_v^{(1)} = P_v^{(Z,-)}$. The post-measurement state is then given by

$$|\psi_{\text{post}}^{n-k}\rangle = Z^{n-k} P_{v_{n-k}}^{(x_{n-k})} \cdot \dots \cdot P_{v_2}^{(x_2)} P_{v_1}^{(x_1)} |G\rangle. \quad (\text{B.5})$$

By acting with the projectors on $|G\rangle$ we find by induction on $n - k$ that the post-measurement state can be evaluated to

$$|\psi_{\text{post}}^{n-k}\rangle = \left(\bigotimes_{i=1}^{n-k} Z^{\sum_{j=1}^{i-1} x_j \text{adj}(v_i, v_j)} |x_i\rangle_{v_i} \right) \otimes \left(\left(\prod_{i=1}^{n-k} (Z[N_{v_i} \cap V'])^{x_i} \right) |G[V']\rangle \right), \quad (\text{B.6})$$

where $\text{adj}(u, v)$ is 1 if (u, v) is an edge in G and zero otherwise and where $Z[X]$ is $\prod_{x \in X} Z_x$. One can see this by checking that indeed

$$Z P_{v_{n-k}}^{(x_{n-k})} |\psi_{\text{post}}^{n-k-1}\rangle = |\psi_{\text{post}}^{n-k}\rangle. \quad (\text{B.7})$$

by using equations (4.29) and (4.32). The operations on the qubits in $V(G) \setminus V'$ in the left part of equation B.6 will only give a global phase as follows

$$\begin{aligned} \left(\bigotimes_{i=1}^{n-k} Z^{\sum_{j=1}^{i-1} x_j \text{adj}(v_i, v_j)} |x_i\rangle_{v_i} \right) &= \left(\bigotimes_{i=1}^{n-k} (-1)^{x_i \sum_{j=1}^{i-1} x_j \text{adj}(v_i, v_j)} |x_i\rangle_{v_i} \right) \\ &= (-1)^{\sum_{i=1}^{n-k} \sum_{j=1}^{i-1} x_i x_j \text{adj}(v_i, v_j)} \bigotimes_{i=1}^{n-k} |x_i\rangle_{v_i}. \end{aligned} \quad (\text{B.8})$$

The exponent in the global phase $\sum_{i=1}^{n-k} \sum_{j=1}^{i-1} x_i x_j \text{adj}(v_i, v_j)$ is in fact the number of edges in the induced graph $G_x = G[\{v_i : x_i = 1\}]$. Let's now consider the correction operators in the right part of equation B.6. A qubit $v \in V'$ will have a Z contribution from the i th factor if $v \in N_{v_i}$ and $x_i = 1$. Thus, we see that the total contribution on qubit v is given by

$$Z^{y_v} \quad \text{where} \quad y_v = \sum_{i \in \{i : v_i \in N_v \setminus V'\}} x_i \quad (\text{B.9})$$

Finally we find that the post-measurement state from equation B.6 is given by

$$|\psi_{\text{post}}^{n-k}\rangle = (-1)^{|E(G_x)|} \left(\bigotimes_{i=1}^{n-k} |x_i\rangle_{v_i} \right) \otimes \left(\left(\prod_{v \in V'} Z_v^{y_v} \right) |G[V']\rangle \right). \quad (\text{B.10})$$

⁴We identify 0 and 1 with the measurement outcomes +1 and -1, respectively.

B.3. Vertex-minor formula

Here we provide the C_2 MS formulas⁵ from [2] which we make use of in section 5.4.3. We state what the formula expresses and its quantifier rank in table B.1.

Table B.1: The C_2 MS formulas used in section 5.4.3, what they express and their quantifier rank. b_v is the unique vector in $S(G)$ with respect to (X_e, Y_e, Z_e) as defined in [2].

| Formula | True if and only if | qr |
|----------------------------------|---|----|
| Disjoint(X, Y, Z) | X, Y and Z are pairwise disjoint | 1 |
| Part(X, Y, Z) | (X, Y, Z) is a tripartition | 1 |
| EvenInter(Q, v) | $ N_v \cap Q = 0 \pmod{2}$ | 2 |
| Member(X, Y, Z) | (X, Y, Z) is a vector of $S(G)$ | 4 |
| Eul(X_e, Y_e, Z_e) | (X_e, Y_e, Z_e) is a Eulerian vector of $S(G)$ | 7 |
| Base(X, Y, Z, X_e, Y_e, Z_e) | (X, Y, Z) is b_v wrt. (X_e, Y_e, Z_e) in $S(G)$ | 4 |
| Adj(u, v, X_e, Y_e, Z_e) | (u, v) is an edge of $\mathcal{G}(X_e, Y_e, Z_e)$ | 7 |

$$\text{Disjoint}(X, Y, Z) = \forall x : \left(\neg(x \in X \wedge x \in Y) \wedge \neg(x \in X \wedge x \in Z) \wedge \neg(x \in Y \wedge x \in Z) \right) \quad (\text{B.11})$$

$$\text{Part}(X, Y, Z) = \left(\underbrace{\forall x : (x \in X \vee x \in Y \vee x \in Z)}_{"V=X \cup Y \cup Z"} \right) \wedge \text{Disjoint}(X, Y, Z) \quad (\text{B.12})$$

$$\text{EvenInter}(Q, v) = \forall R : \left(\underbrace{\forall u : u \in R \Leftrightarrow (\text{adj}(u, v) \wedge u \in Q)}_{"R=N_v \cap Q"} \right) \Rightarrow \text{Even}(R) \quad (\text{B.13})$$

$$\text{Member}(X, Y, Z) = \text{Disjoint}(X, Y, Z) \wedge \left[\exists Q : \left(\forall v : \left(\right. \right. \right. \quad (\text{B.14})$$

$$\left. \left. \left. \begin{aligned} &(v \in X \Leftrightarrow (v \notin Q \wedge \neg \text{EvenInter}(Q, v))) \wedge \\ &(v \in Y \Leftrightarrow (v \in Q \wedge \text{EvenInter}(Q, v))) \wedge \\ &(v \in Z \Leftrightarrow (v \in Q \wedge \neg \text{EvenInter}(Q, v))) \wedge \\ &\left(\underbrace{\neg(v \in X \vee v \in Y \vee v \in Z)}_{"v \in V(G) \setminus (X \cup Y \cup Z)} \Leftrightarrow (v \notin Q \wedge \text{EvenInter}(Q, v)) \right) \right) \right) \right] \quad (\text{B.15})$$

⁵Note that there seems to be a typo in [2] since they use the formula $V = X_e \cup Y_e \cup Z_e$ to express that the vector (X_e, Y_e, Z_e) is complete, i.e. that each element of the vector is non-zero. This is however not true, consider for example the sets $X_e = Y_e = Z_e = V$ which corresponds to the zero-vector since $1 + \omega + \omega^2 = 0$ and is therefore not complete. Their formula for whether (X_e, Y_e, Z_e) is a Eulerian vector is on the other hand still correct since the second part of the formula can never be true for a non-complete vector (X_e, Y_e, Z_e) for which $V = X_e \cup Y_e \cup Z_e$.

$$\begin{aligned} \text{Eul}(X_e, Y_e, Z_e) = & (\text{Part}(X_e, Y_e, Z_e)) \wedge \\ & \left[\forall X, Y, Z : (X \subseteq X_e \wedge Y \subseteq Y_e \wedge Z \subseteq Z_e \wedge \text{Member}(X, Y, Z)) \right. \\ & \left. \Rightarrow \underbrace{(\forall v : \neg(v \in X \vee v \in Y \vee v \in Z))}_{"X=Y=Z=\emptyset"} \right] \end{aligned} \quad (\text{B.16})$$

$$\begin{aligned} \text{Base}(X, Y, Z, X_e, Y_e, Z_e, v) = & \text{Member}(X, Y, Z) \wedge \overbrace{(v \in X \vee v \in Y \vee v \in Z)}{"v \in X \cup Y \cup Z"} \wedge \\ & \left[\forall u : \neg(v = u) \Rightarrow \right. \\ & \left. \left((u \in X \Rightarrow u \in X_e) \wedge (u \in Y \Rightarrow u \in Y_e) \wedge (u \in Z \Rightarrow u \in Z_e) \right) \right] \end{aligned} \quad (\text{B.17})$$

$$\begin{aligned} \text{Adj}(u, v, X_e, Y_e, Z_e) = & \neg(u = v) \wedge \\ & \left[\exists X, Y, Z : \left(\text{Base}(X, Y, Z, X_e, Y_e, Z_e, v) \wedge \underbrace{(u \in X \vee u \in Y \vee u \in Z)}{"u \in X \cup Y \cup Z"} \right) \right] \end{aligned} \quad (\text{B.18})$$

B.4. Local complementations from Eulerian vector

Let's assume G is a graph, $S(G)$ its canonical isotropic system and (U_e, V_e, W_e) an Eulerian vector describing the graph $\mathcal{G}(U_e, V_e, W_e)$. We here consider the question of how to find a sequence of local complementations m , such that $\tau_m(G) = \mathcal{G}(U_e, V_e, W_e)$. In this section we will represent Eulerian vectors as vectors in \mathbb{F}_4^n instead of tripartitions of $V(G)$, where $n = |G|$. A tripartition (U_e, V_e, W_e) induces the vector

$$A(v) = \begin{cases} 1 & \text{if } v \in U_e \\ \omega & \text{if } v \in V_e \\ \omega^2 & \text{if } v \in W_e \end{cases} \quad (\text{B.19})$$

where $A(v)$ is element v of the vector $A \in \mathbb{F}_4^n$ and ω is a primitive element of \mathbb{F}_4 . In [3] it is shown that for any Eulerian vector A of an isotropic system there exists exactly one other Eulerian vector A' which differ from A in only the element v . This other Eulerian vector A' is denoted $A * v$ and is called a switching of A . Furthermore, the switching induces a local complementation on the graphs the Eulerian vectors describe. More precisely, if the Eulerian vector A describe the graph G , then the Eulerian vector $A * v$ describe the graph $\tau_v(G)$. Thus, if we find a sequence of switchings taking the Eulerian vector describing G to the Eulerian vector describing $\mathcal{G}(U_e, V_e, W_e)$, we have also directly found a sequence of local complementations taking G to $\mathcal{G}(U_e, V_e, W_e)$. The Eulerian vector of $S(G)$ describing G is given as $A_0 = (\omega, \dots, \omega)$ and the one describing $\mathcal{G}(U_e, V_e, W_e)$ is given as in equation (B.19). A sequence of switchings taking A_0 to A can be found in linear time similarly to the method described in section 4 of [4]. The idea is to go over the vectors A_0 and A element by element and make these equal one by one. Let's consider a vertex

$v \in V(G)$ and the four vectors $A_0, A_0 * v, A$ and $A * v$. These four vectors cannot all differ in the element v , since there are only three non-zero elements of \mathbb{F}_4 , i.e. $\{1, \omega, \omega^2\}$. Repeating this process for all elements of $V(G)$ will give two sequences of switchings, m_1 and m_2 , one for A_0 and one for A such that

$$A_0 * m_1 = A * m_2. \quad (\text{B.20})$$

Since the switchings are involutions we have that

$$A_0 * (m_1 \overline{m_2}) = A \quad (\text{B.21})$$

and therefore that

$$\tau_{m_1 \overline{m_2}}(G) = \mathcal{G}(U_e, V_e, W_e), \quad (\text{B.22})$$

where $m_1 \overline{m_2}$ is the sequence m_1 followed by the reversal of m_2 . Finding $m = m_1 \overline{m_2}$ thus takes time $\mathcal{O}(n)$.

References

- [1] A. Dahlberg and S. Wehner, *Transforming graph states using single-qubit operations*, *Phil. Trans. R. Soc. A* **376**, One contribution of 15 to a discussion meeting issue 'Foundations of quantum mechanics and their impact on contemporary society' (2018), [10.1098/rsta.2017.0325](https://doi.org/10.1098/rsta.2017.0325).
- [2] B. Courcelle and S. il Oum, *Vertex-minors, monadic second-order logic, and a conjecture by Seese*, *Journal of Combinatorial Theory. Series B* **97**, 91 (2007).
- [3] A. Bouchet, *Recognizing locally equivalent graphs*, *Discrete Mathematics* **114**, 75 (1993).
- [4] A. Bouchet, *An efficient algorithm to recognize locally equivalent graphs*, *Combinatorica* **11**, 315 (1991), [arXiv:0702057v2 \[cs\]](https://arxiv.org/abs/0702057v2) .

C

How to transform graph states using single-qubit operations: computational complexity and algorithms

Axel Dahlberg, Jonas Helsen, Stephanie Wehner

C.1. Circle graphs induced by Eulerian tours on triangular-expanded graphs are not distance-hereditary

In this section we show that circle graphs induced by Eulerian tours on triangular expanded graphs (CETEx graphs) are not distance-hereditary. We showed in section 6.2 that `StarVertexMinor` is NP-Complete on CETEx graphs and in section 6.3.1 that `StarVertexMinor` is in \mathbb{P} for distance-hereditary graphs. The main result of this section is therefore to show that these two graph classes are in fact disjoint. Furthermore, the results of this section validate fig. 1.3 as the class of CETEx graphs (red) are drawn as being disjoint from the distance-hereditary graphs (green).

We formally state the main result of this section as the following theorem:

Theorem C.1.1. *Let U be an Eulerian tour on a triangular expansion $\mathcal{T}(R)$ of some 3-regular graph R . Then the alternance graph $\mathcal{A}(U)$ is not distance-hereditary, i.e. $\text{rwd}(\mathcal{A}(U)) > 1$.* \diamond

To simplify the notation in this section, let's introduce a definition for the set of circle graphs which are induced by Eulerian tours on a 4-regular multi-graph:

Definition C.1.2. Let F be a 4-regular multi-graph. The set of circle graphs induced by the set of Eulerian tours on F will be denoted $\mathfrak{A}(F)$. \diamond

We will prove theorem C.1.1 by showing that all graphs which are in $\mathfrak{A}(\mathcal{T}(R))$ for some 3-regular graph R have a certain vertex-minor which has rank-width two. Since rank-width cannot increase under local complementations and vertex-deletions, the theorem follows. Below is a step-by-step overview on how we will prove theorem C.1.1:

1. Introduce transition-minors of 4-regular multi-graphs.
2. Show that if F' is a transition-minor of F , then any graph in $\mathfrak{A}(F')$ is a vertex-minor of any graph in $\mathfrak{A}(F)$.
3. Introduce subcubic minors of subcubic graphs, i.e. graphs where $N_v \leq 3$ for any vertex v .
4. Extend the notion of triangular-expansion to also work for subcubic graphs. We will call this *extended triangular expansion* and denote this as $\tilde{\mathcal{T}}$ to distinguish it from the triangular expansion in section 6.2.2.
5. Show that if R' is a subcubic minor of R , then $\tilde{\mathcal{T}}(R')$ is a transition-minor of $\tilde{\mathcal{T}}(R)$.
6. Step 2 and 5 then imply that if R' is a subcubic minor of R , then any graph in $\mathfrak{A}(\tilde{\mathcal{T}}(R'))$ is a vertex-minor of any graph in $\mathfrak{A}(\tilde{\mathcal{T}}(R))$.
7. Finally we show that there exist a subcubic graph R_0 such that R_0 is a subcubic minor of any 3-regular graph R and furthermore that all the graphs in $\mathfrak{A}(\tilde{\mathcal{T}}(R_0))$ have rank-width two.

8. Step 7 imply that all graphs in $\mathfrak{A}(\mathcal{T}(R_0))$ are vertex-minors of any graph in $\mathfrak{A}(\mathcal{T}(R))$, where R is any 3-regular graph.

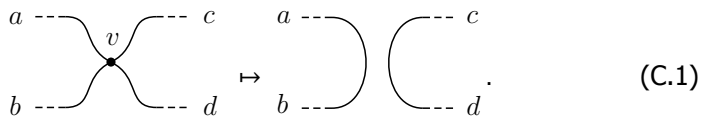
9. Theorem C.1.1 then follows.

Step 1:

In the following definition we introduce the notion of transition-minors of 4-regular multi-graphs.

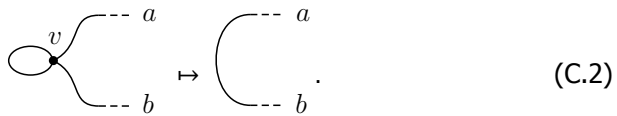
Definition C.1.3. Let F be a connected 4-regular multi-graph. We call F' a *transition-minor* of F if F' is connected and can be reached from F by some sequence of the following two operations:

- Let v be a vertex in F without self-loops. Denote the four vertices incident on the four edges incident on v by a, b, c and d . Note that these vertices are not pairwise different if multi-edges are incident on v . Let $F \setminus_{(v,a,b)}$ be the multi-graph obtained from F by deleting v and adding the edges (a,b) and (c,d) , as shown in the following equation



There are a priori three¹ possible ways to perform this operation by choosing which neighbors of v to connect. Note that if for example the edge (a,b) is already present in the multi-graph F , then the multiplicity of (a,b) increases by one.

- Let v be a vertex in F with one self-loop. Denote the two vertices, other than v , incident on the two edges incident on v which are not the self-loop by a and b . Note that a are not different if (v,a) is a multi-edge. Let $F \setminus_{(v)}$ be the multi-graph obtained from F by deleting v and adding the edge (a,b) , as shown in the following equation



◊

Step 2:

To prove theorem C.1.5, which relates transition-minors and vertex-minors, we need the following lemma:

¹If there is a multi-edge with multiplicity two, there are two choices and for multiplicity three there is only one.

Lemma C.1.4. *Let F be a connected 4-regular multi-graph. Let v be a vertex in F without self-loops. Denote the four vertices incident on the four edges incident on v by a, b, c and d . If the graph $F \setminus_{(v,a,b)}$ is connected then there exist an Eulerian tour U on F such that avb is a subword of $m(U)$.* \diamond

Proof. Due to Fleury's algorithm [2] for finding Eulerian tours we know that an Eulerian tour can be found by traversing a graph in an arbitrary way, while deleting edges which are traversed, as long as a cut-edge² is not traversed unless necessary. Let's therefore start Fleury's algorithm at vertex a and traverse the edge (a, v) , which is a valid move since 4-regular graphs have no cut-edges³. The question is now if traversing the edge (v, b) is a valid move in Fleury's algorithm. The only possibility for this not being a valid move is if the deletion of both (a, v) and (v, b) disconnects the graph. But this contradicts the assumption that $F \setminus_{(v,a,b)}$ is connected. \square

Theorem C.1.5. *Let F be a connected 4-regular multi-graph. Let F' be a transition-minor of F . Furthermore let G' and G be graphs in $\mathfrak{A}(F')$ and $\mathfrak{A}(F)$ respectively. Then we have that*

$$G' < G \tag{C.3}$$

\diamond

Proof. This follows implicitly from the work of Bouchet [4]. We will prove this for F' being $F \setminus_{(v,a,b)}$ or $F \setminus_{(v)}$ as in definition C.1.3. The proof then follows by induction. Let's start with $F' = F \setminus_{(v,a,b)}$. Let U be a Eulerian tour on F such that avb is a sub-word of $m(U)$. Such a U exists by lemma C.1.4, since F' is assumed to be connected. In [4] it is shown⁴ that any graph in $\mathfrak{A}(F \setminus_{(v,a,b)})$ is LC-equivalent to $\mathcal{A}(U) \setminus v$. Showing that all graphs in $\mathfrak{A}(F \setminus_{(v,a,b)})$ are a vertex-minors of $\mathcal{A}(U)$ and therefore vertex-minors of all graphs in $\mathfrak{A}(F)$. Consider now the case where $F' = F \setminus_{(v)}$. Since v has a self-loop, for any Eulerian tour U the induced double occurrence word $m(U)$ will contain the sub-word $avvb$. Furthermore, there exist a Eulerian tour U' on $F \setminus_{(v)}$ such that $m(U') = m(U) \setminus v$. As for the previous case, this shows that all graphs in $\mathfrak{A}(F \setminus_{(v)})$ are vertex-minors of $\mathcal{A}(U)$ and therefore vertex-minors of all graphs in $\mathfrak{A}(F)$. \square

Step 3:

Next, we extend the notion of triangular expansions to also allow for subcubic graphs, rather than only 3-regular graphs.

²An edge whose deletion increases the number of connected components of the graph.

³In fact, any graph with only vertices of even degree has no cut-edge. An easy way to see this is to assume that such an edge exists. Removing this edge creates two connected components F_1 and F_2 . Consider the sum of the degrees of the vertices in F_1 , which should be even since this is twice the number of edges, by the handshaking lemma [3]. However, since only one incident edge was removed from a vertex in F_1 the sum of the degrees should have decreased by one. This contradicts the assumption that all vertices in the original graph had even degree.

⁴Note that Bouchet did not consider the case where $a = b$ but it is easy to see that the statement also applies for this case.

Definition C.1.6 (extended triangular expansion). An extended triangular expansion $\tilde{\mathcal{T}}(R)$ of a graph R will work very much like the triangular expansion in definition 6.2.7. However we will not restrict R to be 3-regular here, but rather be connected, subcubic and contain more than one vertex. A extended triangular expansion of R is reached by the following three steps:

- Replace any vertex in R of degree three, with a triangle subgraph as in eq. (6.3).
- Add a self-loop to any vertex of degree one.
- Double every original edge form the graph R .

◊

Note that if R is 3-regular then $\mathcal{T}(R) = \tilde{\mathcal{T}}(R)$. Furthermore, note that $\tilde{\mathcal{T}}(R)$ is necessarily 4-regular. The following equation shows an example of an extended triangular expansion of a subcubic graph:

$$\tilde{\mathcal{T}} \left(\begin{array}{c} \bullet \\ | \\ \bullet \\ / \ \backslash \\ \bullet \ \bullet \end{array} \right) = \begin{array}{c} \bullet \\ \circ \\ \bullet \\ / \ \backslash \\ \bullet \ \bullet \\ \circ \ \circ \\ \bullet \ \bullet \\ \circ \ \circ \\ \bullet \ \bullet \end{array} . \tag{C.4}$$

Step 4:

We introduce the notion of subcubic minors which we relate to transition-minors and thus vertex-minors in step 5 below.

Definition C.1.7 (subcubic minor). Let R be a 3-regular graph and let R' be a graph obtained from R by some sequence of the following operations:

- Vertex-deletion: $R \mapsto R \setminus v$, where $v \in V(R)$.
- Edge-deletion: $R \mapsto R \setminus (u, v)$, where $(u, v) \in E(R)$ and $|N_u| = |N_v| = 3$.
- Shrink path: $R \mapsto \tau_v(R) \setminus v$, where $v \in V(R)$ and $|N_v| = 2$.

Note that each vertex of R' has three or less neighbors⁵.

◊

Step 5:

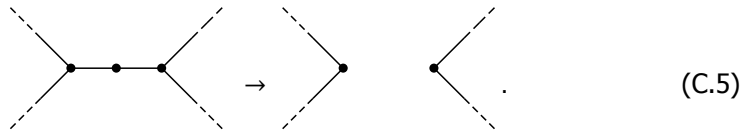
Next, we relate subcubic minors and transition minors.

Theorem C.1.8. Let R be a connected 3-regular graph and R' be a connected subcubic minor of R . Let $\tilde{\mathcal{T}}(R)$ and $\tilde{\mathcal{T}}(R')$ be extended triangular expansions of R and R' respectively. Then we have that $\tilde{\mathcal{T}}(R')$ is a transition-minor of $\tilde{\mathcal{T}}(R)$.

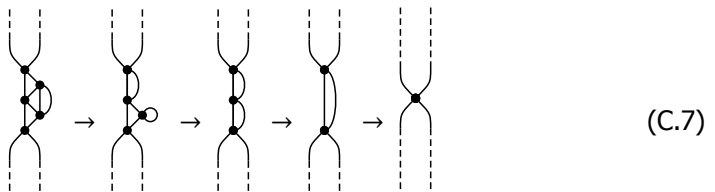
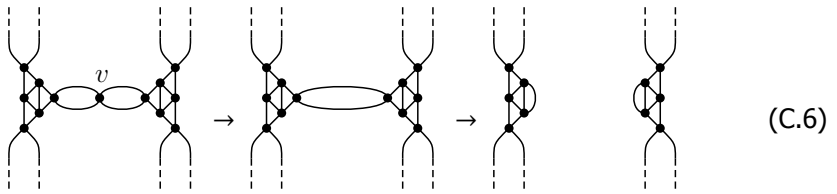
◊

⁵i.e. R' is subcubic.

Proof. Clearly, both $\tilde{\mathcal{T}}(R)$ and $\tilde{\mathcal{T}}(R')$ are connected. We therefore need to show that $\tilde{\mathcal{T}}(R')$ can be reached from $\tilde{\mathcal{T}}(R)$ by some sequence of the operations defined in definition C.1.3. We will prove this by showing that if R' is a subcubic minor reached from R by applying one of the operations in definition C.1.7 once, then $\tilde{\mathcal{T}}(R')$ is a transition-minor of $\tilde{\mathcal{T}}(R)$. The theorem then follows by induction. Let's first assume that R' is reached from R by deleting the vertex v . Firstly, if v has degree one then clearly $\tilde{\mathcal{T}}(R')$ is the multi-graph $\tilde{\mathcal{T}}(R) \downarrow_{(v)}$ and therefore a transition-minor of $\tilde{\mathcal{T}}(R)$. Assume now that v has degree two and furthermore that the neighbors of v are of degree three. This situation is visualized in the following equation



The corresponding reduction using transition-minor operations can be seen in eq. (C.6) followed by eq. (C.7). Showing that indeed $\tilde{\mathcal{T}}(R \setminus v)$ is a transition-minor of $\tilde{\mathcal{T}}(R)$.

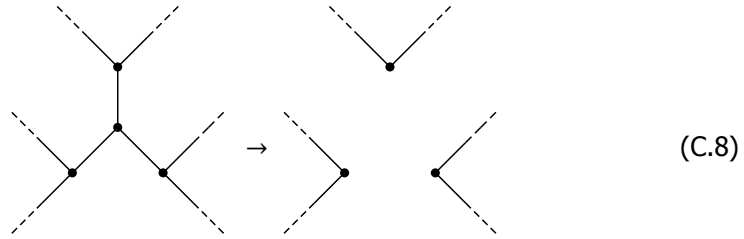


The case when at least one of the two neighbors of v have degree less than three in R can easily be checked in a similar way.

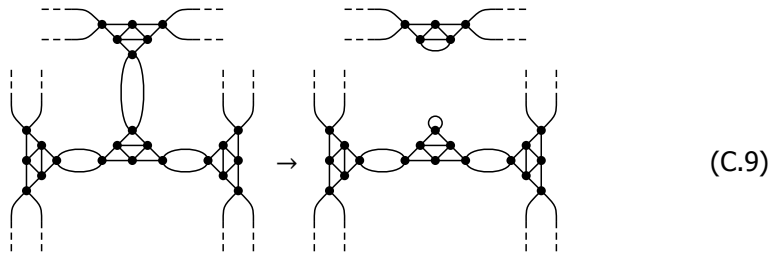
Note that eq. (C.6) also show the case when R' is reached by deleting an edge or by applying a local complementation followed by a vertex-deletion. To see this note that the multi-graph in the second step in eq. (C.6) is $\tilde{\mathcal{T}}(\tau_v(R) \setminus v)$. Furthermore, the multi-graph in the third step of eq. (C.6) is the extended triangular expansion of the graph reached from $\tau_v(R) \setminus v$ by deleting the edge created by the local complementation.

Assume now that v has degree three and that the three neighbors of v also

have degree three. This situation is visualized in the following equation



That $\tilde{\mathcal{T}}(R \setminus v)$ is a transition-minor of $\tilde{\mathcal{T}}(R)$ can be seen from eq. (C.9), followed by eq. (C.6) and eq. (C.7).



Similarly, the case when some of the neighbors of v have degree less than three can be checked. This concludes the proof. \square

Step 6:

From the theorems in step 2 and 5, we have the following corollary.

Corollary C.1.8.1. *Let R be a 3-regular graph and R' be a subcubic minor of R . Let $\tilde{\mathcal{T}}(R)$ and $\tilde{\mathcal{T}}(R')$ be the extended triangular expansions of R and R' respectively. Furthermore, let G and G' be graphs in $\mathfrak{A}(\tilde{\mathcal{T}}(R))$ and $\mathfrak{A}(\tilde{\mathcal{T}}(R'))$ respectively. Then we have that*

$$G' < G. \tag{C.10}$$

\diamond

Proof. This follows directly from theorem C.1.5 and theorem C.1.8. \square

Step 7:

Lemma C.1.9. *The diamond graph D_4 , i.e.*



is a subcubic minor of any connected 3-regular graph. Furthermore, circle graphs induced by Eulerian tours on the extended triangular expansion of D_4 , i.e. graphs in the set $\mathfrak{A}(\tilde{\mathcal{T}}(D_4))$ have rank-width two. \diamond

Proof. We start by proving that D_4 is a subcubic minor of any connected 3-regular graph R . Let's first introduce a notation which will be useful for this proof:

Let $P = v_0 e_1 v_1 \dots e_{k-1} v_{k-1} e_k v_k$ be a path in a graph R . Denote by $V(P) = \{v_0, v_1, \dots, v_{k-1}, v_k\}$ and $E(P) = \{e_1, e_2, \dots, e_k\}$ the vertices and the edges in the path respectively. Furthermore, let $R \setminus E(P)$ be the graph obtained from R by deleting all the edges in $E(P)$.

We will first prove, by contradiction, that R contains a cycle C such that there exist at least two distinct vertices in $V(C)$ which are not disconnected in $R \setminus E(C)$. Assume therefore that there exist no such C :

Then let $C^{(0)} = v_0 e_1 v_1 \dots e_{k-1} v_{k-1} e_k v_0$ be a cycle in R , which exists since R is not a tree⁶. Furthermore, let $R^{(1)}$ be the graph $R \setminus E(C^{(0)})$. From the assumption we therefore have that $R^{(1)}$ consist of k connected components $\{R_i^{(1)}\}_i$ such that $v_i \in R_i^{(1)}$ for all $i \in [k]$, see fig. C.1.

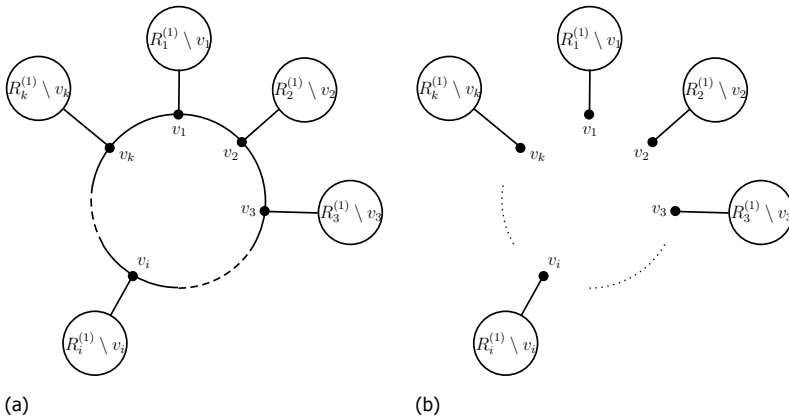


Figure C.1: Visualizations of the contradictory assumption that R contains no cycle C such that at least two vertices in $V(C)$ are connected in $R \setminus E(C)$. The graph R is shown in fig. C.1a and the graph $R \setminus E(C)$ in fig. C.1b.

Consider now one of these connected components $R_i^{(1)}$. Note that all vertices in $R_i^{(1)}$ have degree three, except v_i and furthermore that $|R_i^{(1)}| > 1$ since v_i has a neighbor in $R_i^{(1)}$. This implies that $R_i^{(1)}$ is not a tree and therefore contains a cycle $C^{(1)} = v'_0 e'_1 v'_1 \dots e'_{k'-1} v'_{k'-1} e'_{k'} v'_0$.

Consider then the graph $R_i^{(2)} = R_i^{(1)} \setminus E(C^{(1)})$. Once more, from the assumption we know that $R_i^{(2)}$ contain k' connected components $\{R_{i,i'}^{(2)}\}_{i'}$. Consider now one of these connected components $R_{i,i'}^{(2)}$ such that $v_i \notin R_{i,i'}^{(2)}$. Note that such a connected component exists since $k' \geq 3$ if $C^{(1)}$ is to be a cycle.

Continue the same process for $R^{(3)}, R^{(4)}, \dots$, etc. But note that at each step, edges are deleted and since the graph is finite this process must stop at some point, which is in contradiction with the assumption. \square

⁶A tree has leaves which are not of degree three.

We therefore know that there exists a cycle $C = v_0 e_1 v_1 \dots e_{k-1} v_{k-1} e_k v_0$ and two distinct vertices v_i and v_j in $V(C)$ which are connected in the graph $R \setminus E(C)$. Thus, let C be such a cycle and $v_i \neq v_j$ be two connected vertices in $R \setminus E(C)$. Furthermore, let $P = v_i \hat{e}_1 \hat{v}_1 \dots \hat{e}_{k-1} \hat{v}_{k-1} v_j$ be a path from v_i to v_j in $R \setminus E(C)$. Consider now the induced subgraph $R[V(C) \cup V(P)]$, i.e. the induced subgraph on the vertices in C and P , see fig. C.2. It is important to note that the graph $R[V(C) \cup V(P)]$ could

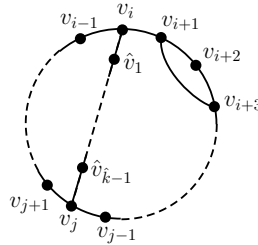


Figure C.2: The induced subgraph $R[V(C) \cup V(P)]$.

contain more edges than $E(C) \cup E(P)$, see for example the edge between v_{i+1} and v_{i+3} in fig. C.2. Let's denote the set of edges in $R[V(C) \cup V(P)]$ which are not in $E(C) \cup E(P)$ by \hat{E} . We now claim that the edges in \hat{E} can be removed with the edge-deletion operation used for subcubic minors. An edge can only be deleted with this operation if both incident vertices have degree three. All edges in R have degree three by assumption. We therefore need to show that there are no two edges in \hat{E} , which are incident on a common vertex in $V(C) \cup V(P)$. This is easily seen by the fact that each vertex in $V(C) \cup V(P)$ is incident on at least two edges in $R[V(C) \cup V(P)]$ and can therefore only be incident to maximally one edge in \hat{E} , since the vertices are of degree three. After the deletion of the vertices in \hat{E} , all the vertices not in $V(C) \cup V(P)$ can be deleted. This creates a subcubic minor in which there are exactly three paths from v_i to v_j , namely

$$P = v_i \hat{e}_1 \hat{v}_1 \dots \hat{e}_{k-1} \hat{v}_{k-1} v_j \tag{C.12}$$

$$P_1 = v_i e_{i+1} v_{i+1} \dots e_{j-1} v_{j-1} e_j v_j \tag{C.13}$$

$$P_2 = v_i e_i v_{i-1} \dots e_{j+2} v_{j+1} e_{j+1} v_j \tag{C.14}$$

Note that P , P_1 and P_2 are pairwise edge-disjoint and all vertices on these paths, except for v_i and v_j have degree two. By applying the operation $R \mapsto \tau_v(R) \setminus v$ to all the vertices in $V(C) \cup V(P)$ except v_i, v_j and two vertices on distinct paths gives the diamond graph D_4 as in eq. (C.11). Note that at least two of the paths P , P_1 and P_2 contain at least three vertices since the graph is simple.

Finally the extended triangular expansion of D_4 is given by

$$\tilde{\mathcal{T}}(D_4) = \left[\begin{array}{c} \text{Diagram of } \tilde{\mathcal{T}}(D_4) \end{array} \right]. \tag{C.15}$$

By explicit computation the rank-width of a circle graph induced by a Eulerian tour the graph $\hat{\mathcal{T}}(D_4)$, as in eq. (C.15), is found to be two. We have implemented this calculation of the rank-width in SAGE [5] and the code for this can be found at [6]. For completeness, one can verify that the graph in fig. C.3 is the circle graph induced by a Eulerian tour on the extended triangular expansion of D_4 in eq. (C.15) and that this graph is not distance-hereditary. \square

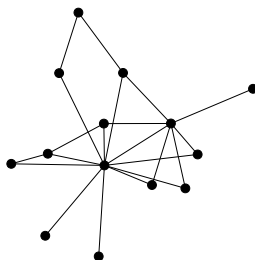


Figure C.3: The circle graph induced by a Eulerian tour on the extended triangular expansion of the diamond graph D_4 , see eq. (C.11) and eq. (C.15).

References

- [1] A. Dahlberg, J. Helsen, and S. Wehner, *How to transform graph states using single-qubit operations: computational complexity and algorithms*, [Quantum Science and Technology](#) (2020).
- [2] M. Fleury, *Journal de mathématiques élémentaires* (C. Delagrave., 1883).
- [3] L. Euler, *Solutio problematis ad geometriam situs pertinentis*, *Comment. Acad. Sci. U. Petrop.* 8 , 128 (1741).
- [4] A. Bouchet, *Circle Graph Obstructions*, [Journal of Combinatorial Theory, Series B](#) **60**, 107 (1994).
- [5] SAGE, <http://www.sagemath.org/> (2020).
- [6] *Git-repository with implemented code for graph state calculations*, <https://github.com/AckslD/vertex-minors> (2020).

D

Transforming graph states to Bell-pairs is NP-Complete

Axel Dahlberg, Jonas Helsen, Stephanie Wehner

D.1. The 4-regular EDPDT problem is NP-Complete

From [1] we know that the EDP problem is NP-Complete even when the graph $G \cup D$ is Eulerian. Here we prove that this problem remains NP-Complete if we restrict the demand graph to be of the form $D = K_2^{\times k}$ and restrict $G \cup D$ to be 4-regular. We call this problem 4-regular EDPDT, see problem 8.3.4. To do so we will first introduce the notion of a grid graph gadget, an essential tool for reducing the Eulerian EDP problem to the 4-regular EDP problem. We make use of these results to show BellVM (and BellQM) is NP-Complete in section 8.4.

Definition D.1.1. Let G be an Eulerian multi-graph and let v be a vertex of G of degree $2n$ with incident edges labeled $1, \dots, n, 1', \dots, n'$. The grid gadget gadget GG_v associated to v , illustrated in fig. D.1, is a graph on $n^2 + 2n$ vertices labeled

$$V(GG_v) = \{v_i : i \in [n]\} \cup \{v_{i'} : i \in [n]\} \cup \{v_{i,j'} : i, j \in [n]\} \setminus \{v_{0,0'}\} \quad (\text{D.1})$$

with edge set

$$\begin{aligned} E(GG_v) = & \{(v_i, v_{i+1}) : i \in [n-1]\} \cup \{(v_n, v_1)\} \\ & \cup \{(v_{i'}, v_{(i+1)'}) : i \in [n-1]\} \cup \{(v_{n'}, v_{1'})\} \\ & \cup \left[\bigcup_{i \in [n]} \{(v_i, v_{i,i'})\} \cup \{(v_{i,j'}, v_{i,(j+1)'}) : j \in [n-1]\} \right] \\ & \cup \left[\bigcup_{i \in [n]} \{(v_{i'}, v_{1,i'})\} \cup \{(v_{j,i'}, v_{j+1,i'}) : j \in [n-1]\} \right] \\ & \cup \{(v_{i,n'}, v_{n,i'}) : i \in [n]\} \cup \{(v_1, v_{1'})\}. \end{aligned} \quad (\text{D.2})$$

◊

For later convenience we also define the following subsets of edges for $i \in [n]$

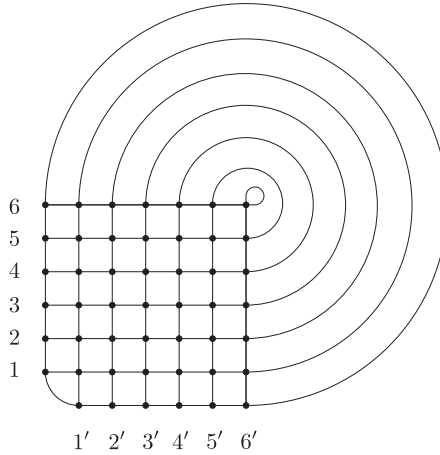
$$E_i^{\text{hor}}[j', k'] = \{(v_{i,l'}, v_{i,(l+1)'}) : l \in [j-1, k-1]\} \quad \text{for } i, j, k \in [n] \quad (\text{D.3})$$

$$E_{i'}^{\text{ver}}[j, k] = \{(v_{l,i'}, v_{l+1,i'}) : l \in [j-1, k-1]\} \quad \text{for } i, j, k \in [n]. \quad (\text{D.4})$$

where $v_{i,0'} = v_i$ and $v_{0,i'} = v_{i'}$. Informally, with respect to fig. D.1 the edges in $E_i^{\text{hor}}[j', k']$ are the horizontal edges at level i which are right of j' and left of k' . Similarly, the edges in $E_{i'}^{\text{ver}}[j, k]$ are the vertical edges at level i' which are above j below k .

Note that all the vertices in the grid graph gadget GG_v have degree 4 or 3, with only the vertices on the 'outside' (associated to the incident edges of the vertex v) having degree 3.

We will now prove that the grid gadget, in a specified sense, provides all-to-all connectivity.



D

Figure D.1: An example of the grid gadget defined in definition D.1.1.

Lemma D.1.2. *Let v be a vertex in a graph with $2n$ incident edges labeled $1, \dots, n, 1', \dots, n'$ and let GG_v be the associated grid graph gadget as defined in definition D.1.1. Consider an arbitrary pairing of the vertices $v_1, \dots, v_n, v_{1'}, \dots, v_{n'}$ consisting of tuples of the form (v_i, v_j) (unprimed pairs), $(v_k, v_{l'})$ (mixed pairs) and $(v_{m'}, v_{n'})$ (primed pairs). Then there exist edge-disjoint paths P_1, \dots, P_n that connect all pairs and moreover contain all edges in GG_v . \diamond*

Proof. Assume without of loss of generality that $i < j$ for the unprimed pairs and $m' < n'$ for the primed pairs (i.e. the first index is always smaller). We will prove the lemma by explicit construction of the paths P_1, \dots, P_n . Let M be the list of mixed pairs, L be the list of unprimed pairs and L' be the list of primed pairs. Note that L and L' are necessarily of equal length. Consider first the list of mixed pairs M . Construct the paths $P_{k,l'}$ by the following algorithm. An example of these paths is shown in fig. D.2.

Algorithm D.1 Algorithm to construct the paths $P_{k,l'}$.

for $x \in [K]$ where K is the length of M **do**

Create the path P_{k_x, l'_x} associated to the x 'th pair in M by:

- (1) Walking horizontally rightwards from v_{k_x} to v_{k_x, l'_x}
- (2) Walking vertically downwards from v_{k_x, l'_x} to $v_{l'_x}$

end for

Note that for every pair we have that

$$E(P_{k,l'}) = E_k^{\text{hor}}[0, l'] \cup E_{l'}^{\text{ver}}[0, k] \tag{D.5}$$

where $E_k^{\text{hor}}[0, l']$ and $E_{l'}^{\text{ver}}[0, k]$ are defined in eqs. (D.3) and (D.4). Note also that since these sets are all disjoint the paths $P_{k,l'}$ are all mutually edge-disjoint.

Now consider the lists of primed and unprimed pairs L and L' . We will construct paths $P_{i,j}$ (unprimed) and $P_{m',n'}$ (primed) using the following algorithm.

Algorithm D.2 Algorithm to construct the paths $P_{i,j}$ and $P_{i',j'}$.

for $x \in [K]$ where K is the length of L **do** # L and L' are necessarily of the same length.

 Create the path P_{i_x, j_x} associated to the k 'th pair in L by:

- (1) Walking horizontally rightwards from v_{i_x} to v_{i_x, m'_x}
- (2) Walking vertically upwards from v_{i_x, m'_x} to v_{j_x, m'_x}
- (3) Walking horizontally leftwards from v_{j_x, m'_x} to v_{j_x}

 Create the path $P_{m'_x, n'_x}$ associated to the x 'th pair in L' by:

- (1) Walking vertically upwards from $v_{m'_x}$ to v_{i_x, m'_x}
- (2) Walking horizontally rightwards from v_{i_x, m'_x} to v_{i_x, n'_x}
- (3) Walking vertically downwards from v_{i_x, n'_x} to $v_{n'_x}$

end for

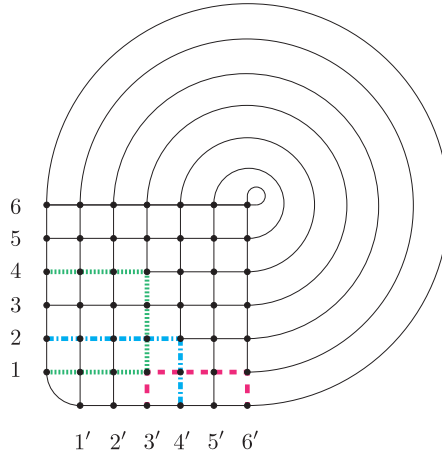


Figure D.2: An example of paths produced by algorithms D.1 and D.2 on the grid gadget defined in definition D.1.1. In this example one of the mixed pairs in M is $(2, 4')$ giving rise to the blue dashed-dotted path. Furthermore, the first unprimed pair in L is $(1, 4)$ and the first primed pair in L' is $(3', 6')$, giving rise to the green dotted path and the red dashed path respectively, meeting at $v_{2,3'}$. Note that the x 'th primed and unprimed paths always meet at the vertex v_{i_x, m'_x} where v_{i_x} and $v_{m'_x}$ is the first vertex in the x 'th primed and unprimed pair respectively.

Now note that the for all $x \in [K]$ (where K is the length of L) we have that

$$E(P_{i_x, j_x}) = E_{i_x}^{\text{hor}}[0, m'_x] \cup E_{m'_x}^{\text{ver}}[i_x, j_x] \cup E_{j_x}^{\text{hor}}[0, m'_x] \quad (\text{D.6})$$

$$E(P_{m'_x, n'_x}) = E_{m'_x}^{\text{ver}}[0, i_x] \cup E_{i_x}^{\text{hor}}[m'_x, n'_x] \cup E_{n'_x}^{\text{ver}}[0, i_x] \quad (\text{D.7})$$

This immediately implies that all P_{i_x, j_x} and $P_{m'_x, n'_x}$ are mutually edge-disjoint and moreover that no P_{i_x, j_x} nor any $P_{m'_x, n'_x}$ share edges with any of the mixed-pair paths $P_{k, l'}$. This last point can be seen by noting that an unprimed vertex v_i can either be part of a mixed pair or an unprimed pair but not both at the same time (with a similar argument for the primed vertices).

Hence we have constructed a set of edge-disjoint paths that connect all vertex pairs. However they do not yet contain all edges in $E(GG_v)$. It is however straightforward to extend the paths to include all remaining edges. To see this consider the grid graph gadget GG_v and remove all edges that are contained in one of the paths constructed above. What remains is a not necessarily connected graph G_{rem} of which the connected components, by construction, share a vertex with at least one of the constructed paths. Moreover, all these graphs will be Eulerian. Now choose for each graph G_{rem} a Eulerian tour U and insert it into exactly one of the paths that shares a vertex with G_{rem} . The resulting set of paths will still have mutually edge disjoint elements (since a Eulerian tour is edge-disjoint by definition) and furthermore the union of all the paths in the set contains all edges in the grid graph gadget GG_v . This completes the proof. \square

We will now make use of lemma D.1.2 to map instances of EDP to instances of 4-regular EDPDT. We will first construct a mapping from arbitrary demand graphs D to demand graphs of the form $K_2^{\times k}$. Then, to make the graph $G \cup D$ 4-regular, we replace any higher-degree vertices in G with the grid gadget above. Using lemma D.1.2, we can prove that this puts no restriction on the possible paths.

Theorem D.1.3. *Let G and D be graphs such that $V(D) \subseteq V(G)$ and $G \cup D$ is Eulerian. There exist graphs G'' and D' such that $D' = K_2^{\times k}$ where $k = |E(D)|$, $G'' \cup D'$ is 4-regular and (G, D) is a YES-instance of the EDP problem if and only if (G'', D') is.*

\diamond

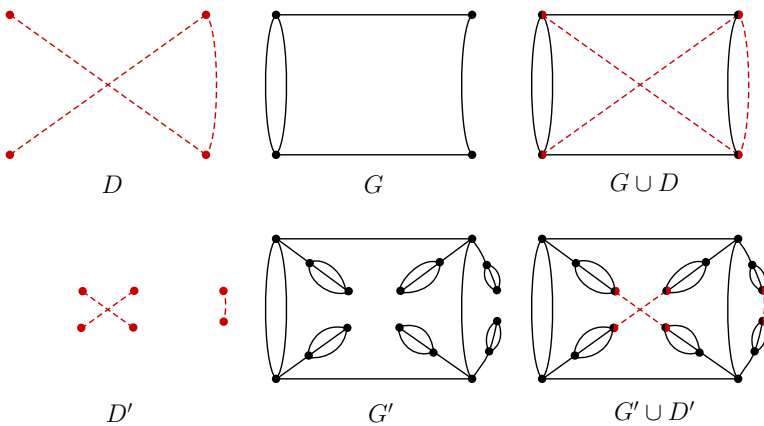
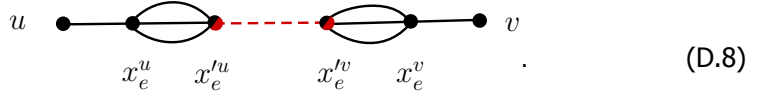


Figure D.3: An example showing the mapping from an instance (G, D) of EDP to an instance (G', D') of EDPDT. Black solid edges are edges from G or G' and red dashed edges are edges from D' .

Proof. To prove the theorem we will construct an explicit mapping from the graphs (G, D) to (G'', D') . We do this in two steps: (1) map (G, D) to (G', D') where $D' = K_2^{\times k}$ but $G' \cup D'$ is not necessarily 4-regular and (2) map (G', D') to (G'', D') such that $G'' \cup D'$ is 4-regular. The first mapping is visualized in fig. D.3 and formalized in eqs. (D.9) to (D.12). Informally, we replace each edge $e = (u, v) \in E(D)$ in the graph $G \cup D$ with the following gadget



where $x_e^u, x_e'^u, x_e'^v$ and x_e^v are all new vertices. Note that we label the vertex closer to the corresponding vertex u in the edge e without the prime and the further one with a prime. Formally, we define G' to be

$$V(G') = V(G) \cup \left(\bigcup_{e=(u,v) \in E(D)} \{x_e^u, x_e'^u, x_e'^v, x_e^v\} \right) \tag{D.9}$$

$$E(G') = E(G) \cup \left(\bigcup_{e=(u,v) \in E(D)} \left\{ (u, x_e^u), (x_e^u, x_e'^u), (x_e'^u, x_e'^v), (x_e'^v, x_e^v), (x_e^v, v), (x_e'^v, x_e'^u), (x_e'^u, x_e^u), (x_e^u, x_e^v) \right\} \right). \tag{D.10}$$

Note that we have added the edges $(x_e^u, x_e'^u)$ and $(x_e'^v, x_e^v)$ three times. We now define the new demand graph D' as

$$V(D') = \bigcup_{e=(u,v) \in E(D)} \{x_e'^u, x_e'^v\} \tag{D.11}$$

$$E(D') = \bigcup_{e=(u,v) \in E(D)} \{(x_e'^u, x_e'^v)\} \tag{D.12}$$

Note that $D' = K_2^{\times k}$. Note also that $G' \cup D'$ is still Eulerian. However it is in general not 4-regular.

Now let N be the set of vertices of $G' \cup D'$ of degree different than 4. Note that, by construction $N \cap V(D') = \emptyset$. That is, all vertices of degree other than 4 are exclusively vertices in G' . We will from G' construct a graph G'' such that $G'' \cup D'$ is 4-regular. For every vertex in N which has degree 2, we simply add a self-loop to the vertex, making the vertex have degree 4 without changing any connectivity. Furthermore, for every vertex $v \in N$ with degree larger than 4, we replace it by the grid graph gadget GG_v , as defined in definition D.1.1, attaching the edges incident on v to the vertices $v_1, \dots, v_n, v_1', \dots, v_n'$ (where $n = \text{deg}(v)/2$) in the grid graph gadget. Note the graph $G'' \cup D'$ obtained after this procedure is 4-regular.

To prove the theorem we now need to show that (G, D) is a YES-instance to EDP if and only if (G'', D') is a YES-instance. Again, we will do this in two steps: (1)

first show that (G, D) is a YES-instance if and only if (G', D') is and (2) show that (G', D') is a YES-instance if and only if (G'', D') is.

1. We now prove that (G', D') is a YES-instance of EDP if and only if (G, D) is. First assume (G, D) is a YES-instance of EDP. This means there exists for each $e = (u, v) \in E(D)$ a path P_e on G that begins and ends at the vertices u, v such that all paths P_e are mutually edge-disjoint. Now, for each $e \in E(D)$ define the path P'_e on G' as

$$P'_e = x_e'^u x_e^u x_e'^u x_e^u u P_e v x_e^v x_e'^v x_e^v x_e'^v \quad (\text{D.13})$$

where we have omitted writing out the edges that the path traverses for clarity. For a visual aid refer to eq. (D.8) where we instead of starting the path at u as in P_e we start at $x_e'^u$, traverse the edge $(x_e^u, x_e'^u)$ back and fourth three times and then move to u using the edge $(x_e^u, x_e'^u)$. From u the path P'_e is the same as P_e and when arriving at v we instead end at $x_e'^v$ similarly to how we started at $x_e'^u$. Thus P'_e is a path connecting the vertices $x_e'^u, x_e'^v$. Note that by definition $(x_e'^u, x_e'^v)$ is an edge in the demand graph D' (precisely corresponding to the edge $(u, v) \in D$). The paths P'_e are also mutually edge-disjoint, since the paths P_e are. Hence (G', D') is a YES-instance.

For the other direction, assume that (G', D') is a YES-instance of EDP and thus that there exists edge-disjoint paths P'_e connecting the vertices $x_e'^u, x_e'^v$ for all $e = (x_e'^u, x_e'^v) \in E(D')$. One can then see that P'_e is either of the form as in eq. (D.13) or as

$$P'_e = x_e'^u x_e^u u P_e v x_e^v x_e'^v. \quad (\text{D.14})$$

This means that the associated path P_e forms a path between u and v in G for all $e \in E(D')$. Furthermore, since all P'_e are pairwise edge-disjoint, so are the P_e . Hence (G, D) is also a YES-instance of EDP.

2. Next we argue that (G', D') is a YES-instance of EDP if and only if (G'', D') is. If (G'', D') is a YES-instance of EDP then so is (G', D') , since any edge-disjoint paths passing through a grid graph gadget GG_v can also be made into edge-disjoint paths passing through the vertex v . Hence assume that (G', D') is a YES-instance of EDP. Note that the only difference between (G', D') and (G'', D') is the replacement of vertices $v \in N$ with the grid graph gadget GG_v . Moreover recall that $N \cap V(D') = \emptyset$. Finally, by lemma D.1.2, we know that any $\deg(v)/2$ paths passing through a vertex v can be mapped to $\deg(v)/2$ paths passing through GG_v , and that these paths are mutually edge-disjoint and use all edges in GG_v . This implies that if (G', D') is a YES-instance of EDP, then so is (G'', D') . This proves the theorem.

□

We can now prove corollary 8.3.4.1.

Proof of corollary 8.3.4.1. Theorem D.1.3 states that there exists a many-one reduction from Eulerian EDP to 4-regular EDPDT . Furthermore, this reduction consists of constructing the graphs (G'', D') from the graphs (G, D) by the explicit rules in eqs. (D.9) to (D.12) and by replacing vertices of degree more than 4 with the grid graph gadget, which is clearly polynomial. This shows that 4-regular EDPDT is NP-Hard. Since any instance of 4-regular EDPDT is also an instance of the general EDP which is in NP, also 4-regular EDPDT is in NP and thus NP-Complete. \square

References

- [1] J. Vygen, *Np-completeness of some edge-disjoint paths problems*, Discrete Applied Mathematics **61**, 83 (1995).