Optimising Adaptive Resource Generation in Near-Term Quantum Networks

A Markov Decision Process Model to Produce an Optimal Resource Generation Policy

Boris Goranov[1]

Supervisor(s): Gayane Vardoyan[1]**, Bethany Davies**[1]

[1]**EEMCS, Delft University of Technology, The Netherlands**

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
June 23, 2024

Name of the student: Boris Goranov
Final project course: CSE3000 Research Project
Thesis committee: Gayane Vardoyan, Bethany Davies, Rihan Hai

Abstract

A quantum network allows us to connect quantum information processors to achieve capabilities that are not possible using classical computation. Quantum network protocols typically require several entangled states available simultaneously. Previously, an entanglement generation process was analysed where, at each time step, we generate an entangled state with success probability p. Here, we consider adaptive entangled state generation with more flexibility. At each time step, our process chooses a protocol (p_i, F_i) from a discrete number of entanglement generation protocols. An entangled state is generated successfully with probability p_i, and its fidelity F_i defines how close the entangled state is to an ideal Bell state. The new state is subject to depolarising noise in the quantum memory. Because of the memory noise, states are discarded after a certain number of time steps t_i when they are no longer useful to our application. We model our process as a Markov decision process and derive a policy π to generate n entangled states with minimal expected time $\mathbb{E}_\pi[\tau]$. We analyse the offered improvement of the optimal policy of our adaptive entanglement generation process over the previously studied static process. We conclude that this improvement becomes more significant as the required number of links in memory increases.

1 Introduction

A quantum network (QN) enables quantum communication between the participating remote parties. It allows quantum information processors to connect and achieve capabilities not possible with classical computation [1]. An important QN application is secure computation on a quantum server. It can be achieved using a Blind Quantum Computation (BQC) protocol [2] that models the communication between a client and the server as a black box with interfaces both parties can use. That way, the client can utilise the quantum server's potential while their computations remain secret to the server.

BQC is only one example of what QNs would enable us to achieve – there are other protocols which offer an improvement over classical networks [3]. However, most QN protocols require multiple entangled states, or entanglement links between the QN nodes, to be available simultaneously [4]. Therefore, we refer to an entanglement link as a quantum resource and the protocols for generating entanglement between nodes as resource generation protocols. Here, we focus on near-term QNs. A QN in the near term may cover short distances while using powerful end nodes capable of implementing a large set of protocols [1].

We can define a resource generation protocol as a tuple (p, F) where p is the probability of successful resource generation and F is the fidelity of the generated link. The link fidelity models how close the generated entangled state is to an ideal Bell state [5]. Research to determine the expected time of a process to produce a fixed number of links and characterise the distribution of the fidelity of the generated links already exists [4]. The previously conducted research in this direction has focused on using a single resource generation protocol for all links in the network [4], [6]. We define the setting for our process similarly. The generated links reside in quantum memory, and we model their fidelity using an exponential decay defined as follows:

$$F \to \left(F - \frac{1}{4}\right) e^{-w\Gamma} + \frac{1}{4}. \tag{1}$$

This model simulates a noisy environment in which links lose quality over time, as illustrated in Figure 1. The variables used are defined as follows.

- F is the link fidelity w.r.t. an ideal Bell state;
- Γ is the quantum memory parameter that defines how fast the link's fidelity decays;
- w is the number of time steps since the link has been generated.

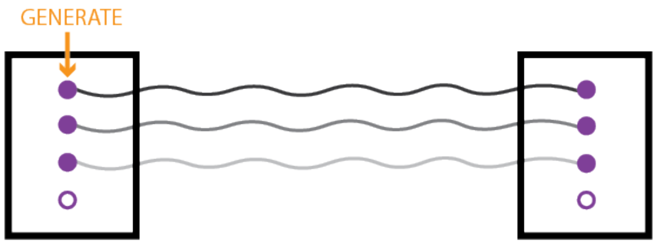

Figure 1: Entanglement links with exponentially decaying fidelity between two nodes in a quantum network. Adapted from B. Davies with permission.

After a link's fidelity drops below a predefined threshold fidelity F_{thresh}, the link is discarded from memory. We refer to this event as a cutoff [6]. An optimal policy is defined as a set of decisions specifying what protocol to use for link generation, aiming to minimize the expected delivery time of n entanglement links. Here, we consider a process in which the nodes in the network have information about the links already generated. However, the previously considered process that uses a single protocol for generating each link makes no use of the available information about existing links in memory.

Here, we study an adaptive generation process where, at each time step, we can choose from multiple generation protocols (p_i, F_i), each having different probability p_i and fidelity F_i. This makes the process more flexible than a static process that consistently selects the same protocol from the range of protocols available to the adaptive process. An adaptive process utilises the available information about the number of links in memory and their fidelity. Because the adaptive process is more flexible, the optimal policy can vary depending on the parameters chosen, as some protocols will be more favoured than others. We observe the changes in the optimal policy as the number of required links in memory for the network protocol varies. We analyse those changes by looking into the generation protocols the adaptive process uses at each time step, the time to completion of the optimal policy, and its variance. In addition, we compare the runtime of the optimal policy to a heuristic policy that always picks the highest probability of successful link generation that still allows for n links to be available at the end.

Our main contributions are as follows:

- We introduce a finite Markov decision process (MDP) model for a near-term QN with memory cutoffs that selects from a collection of protocols to generate n entanglement links;
- We find optimal policies for minimising the expected delivery of n entanglement links by solving the MDP via policy iteration;
- Our optimal policies take advantage of global knowledge about the expected time to generate n links for each quantum memory state and act as a lower bound to the expected delivery time of policies based on heuristics that use only local state information.

Our main findings are as follows:

- The optimal policies gradually shift from actions with lower probability to actions with higher probability as the number of links in memory increases;
- The actions taken for states that mostly contain links with low fidelity tend to have lower probability;
- As the number of required links increases, the gap between the optimal policy and our heuristic policy increases because the heuristic fails to capture the complexity of the process.

The structure of this paper is as follows. First, we describe how we model the problem of finding an optimal policy as an MDP in Section 2. This section also includes how we simulate the optimal policy using a Monte Carlo simulator [7]. We use those methods to derive the results we interpret in Section 3. Then, we discuss the integrity of our research and the reproducibility of the listed results in Section 4. Finally, we conclude this paper by describing how this work might be extended to analyse larger search spaces in Section 5.

2 Methods

In this section, we include detailed information about what methods we have used to derive the optimal policy for an adaptive link generation process. We model the problem of finding the optimal policy to generate a required number of links n as an MDP. We define what an MDP is and how we derive the optimal policy in Section 2.1. Section 2.2 outlines the model used for the probabilistic decision process. In Section 2.3, we explain how the optimal policy is verified and simulated to analyse its properties later.

2.1 Background

An MDP represents the domain of a problem via a set of states, with actions that introduce stochastic transitions from one state to another [8]. An agent starts from the starting state and has to reach a target state using the set of actions. Each action has a certain probability of successfully transitioning to a new state. We associate a reward to each state-action pair to guide the agent to a goal state in the search problem. The rewards and the action probabilities define the values of every action taken from a given state. Usually, the rewards are derived from the metric we wish to optimise by the actions we are taking in the MDP.

We can use dynamic programming (DP) algorithms to compute optimal policies from a perfect model of the environment as an MDP. There are two main algorithms used in practice – policy and value iteration [9]. Both produce a provably optimal policy by iteratively computing the values for each state-action pair until convergence [9]. For each state s, there will be one or more actions at which the maximum value is obtained in the Bellman optimality equation [10]. We can define an optimal policy as any policy that assigns nonzero probability only to these actions [8].

An MDP has already been used for deriving optimal entanglement generation policies in repeater chains [6]. In our adaptive entanglement generation process, we can model the environment as a set of states representing the quantum memories at a particular time step and a set of actions defined by the protocols (p_i, F_i) we can choose from. For each memory state, we can either successfully generate a link with probability p_i or fail with probability $1 - p_i$. The transition probabilities are defined by the chosen protocol. In addition, the fidelity of the links in memory decreases as per the model introduced in (1). At each time step, we wish to pick an action that minimises the expected time to reach n simultaneously existing links in memory.

We use a policy iteration algorithm to solve the MDP and find an optimal link generation policy. Our algorithm starts with a policy which assigns each action equal probability. At each iteration k, it evaluates the current policy π_k to define a value function V_k. Then, we use the value function to select the actions that maximise the values for each state, and a new policy π_{k+1} is constructed. The process is repeated until the policy has become stable, i.e. it has not changed between two iterations. Because the MDP that we define only has a finite number of policies, this process converges to an optimal policy after a finite number of iterations [9]. We define the pseudocode for the policy iteration algorithm in Appendix A.

2.2 Model

Here, we provide a formal definition of the MDP that models our problem to generate n entangled states. Mathematically, an MDP can be defined as a 4-tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R})$ [11], where:

- $\mathcal{S}$ is the state space;
- $\mathcal{A}$ is the set of actions which we can choose from;

- $\mathcal{P} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the set of transition probabilities for each state-action pair;
- $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the set of rewards.

For our purpose, the action space $\mathcal{A}$ is the set of all generation protocols that the process can choose from at each time step. We assume a heralded entanglement generation process [12] with probability of successful generation p_i. The fidelity F_i with respect to the maximally entangled Bell state is defined as follows [4]:

$$F_i = 1 - \lambda p_i. \tag{2}$$

In this paper, we assume $\lambda = 1$. The linear relationship between p_i and F_i for a protocol i given a parameter λ allows us to model our action space as simply the set of success probabilities p_i for all s number of protocols that we can choose from:

$$\mathcal{A} = \{p_i \mid p_i \in (0, 1)\}. \tag{3}$$

We define the state space of the system $\mathcal{S}$ as the set of all possible configurations of the quantum memory containing up to the required number of entanglement links n:

$$\mathcal{S} = \{\{t_1, t_2, ..., t_m\} \mid t_j \in (0, \max t_i], m \in [0, n]\}. \tag{4}$$

The memory states contain the available links in memory. Each link is represented by the number of time steps until its fidelity falls below a given threshold F_{thresh} and the link is discarded. The threshold fidelity F_{thresh} is determined by the application of the entanglement generation process.

Following the exponential decay model in (1) and the probability-fidelity relationship in (2), we can express the number of steps t_i a newly generated link lives in memory using the action probability. We call t_i the time-to-live (TTL) of a newly generated link with action probability $p_i \in \mathcal{A}$. We discard a link from memory when its TTL reaches zero. There is a trade-off between higher success probability and smaller link TTL, which is defined as follows:

$$t_i = \left\lfloor -\ln \frac{1 - \lambda p_i - 1/4}{F_{\text{thresh}} - 1/4} \times \Gamma^{-1} \right\rfloor. \tag{5}$$

In our model, we assume that the probabilities in the action space do not result in links that expire immediately, i.e. $t_i \geq 1$. Figure 2 presents an example transition from a given state using one generation protocol (p_i, F_i). We denote the maximum number of steps a link can live in memory given a selection of generation protocols as $\max t_i$.

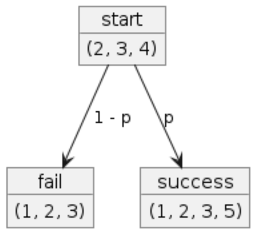

Figure 2: Transition from memory state $(2, 3, 4)$ using generation protocol with probability p and fidelity F of the newly generated link resulting in TTL $t = 5$.

The starting state is the empty memory set $\emptyset$. The final states are all sets $L_i, |L_i| = n$, where n is the number of links we want to generate. The transition probabilities p_i are the success probabilities of the chosen protocols. Appendix B outlines the procedure we use to derive the state space size formula. The size of the state space grows with the number of required links n and $\max t_i$ [13], [14]:

$$|\mathcal{S}| = \binom{\max t_i + n}{\max t_i}. \tag{6}$$

To reach an optimal policy, we maximise the negative expected time until n links have been successfully generated. The Bellman optimality equation we use in our policy iteration algorithm is the following:

$$V_\pi(s) = -1 + \sum_{s' \in S} P(s'|s, \pi) \times V_\pi(s'), \forall s \in S. \tag{7}$$

2.3 Analysis

We can solve the most basic scenario analytically. We consider a situation where we want to generate $n = 2$ entanglement links using two protocols $(p_1, F_1), (p_2, F_2)$ with $F_1 > F_2$ and $p_1 < p_2$. Let $\mathbb{E}_{ij}[\tau]$ be the expected time to generate two links using protocol i for the first link and j for the second link. The generation process is modelled as a sequence of independent Bernoulli trials with the protocols' probability p_i [15]. We use the equation for the expected time derived in [4], which we generalise for two actions as:

$$\mathbb{E}_{ij}[\tau] = \frac{1}{p_j} + \frac{1}{p_i} \frac{1}{(1 - (1 - p_j)^{t_i - 1})}. \tag{8}$$

We calculate the TTL of the first link using (5). As soon as the second entanglement link is generated, both links are used for the particular network protocol application. Therefore, we would always use the higher probability p_2 for the second link because we do not care how long it lives in memory. For the first link, we have to consider both protocols. For any set of parameters, we can calculate the expected times and determine which generation protocol to choose at each time step to achieve the shortest expected time. We use this analytical method to verify our policy iteration algorithm for deriving the optimal policy in the most basic case. A more detailed solution for the expected time in this scenario can be found in Appendix C.

For larger policies, we associate an action $a \in \mathcal{A}$ to each memory state $s \in \mathcal{S}$. We simulate a policy π using Monte-Carlo simulation. Starting from the empty memory state $\emptyset$, we make random throws at each time step to determine whether a link has been successfully generated with probability p_i for the associated action i. We use the same simulation method for the baseline policies we compare to the derived optimal policy. Repeating this simulation N times allows us to calculate the following estimates for the true expected runtime $\mathbb{E}_\pi[\tau]$ and the standard deviation of the runtime $\sigma_\pi(\tau)$ using the sample runtimes τ_i [16]:

$$\hat{\mathbb{E}}_\pi[\tau] = \frac{1}{N} \times \sum_{i=0}^{N-1} \tau_i; \tag{9}$$

$$\hat{\sigma}_\pi(\tau) = \sqrt{\frac{1}{N-1} \times \left(\tau_i - \hat{\mathbb{E}}[\tau]\right)^2}. \tag{10}$$

Finally, we use the standard deviation estimator to calculate the standard error in our experiments [17]:

$$SE_\pi(\tau) = \frac{1}{\sqrt{N}} \times \hat{\sigma}_\pi(\tau). \tag{11}$$

When we make comparisons with our chosen baselines, we plot the ratio between the optimal policy runtime and the chosen baseline. To support any conclusions we make from our comparisons, we also plot the standard error of this ratio. We model the ratio between the optimal policy runtime and the chosen baseline as a ratio between two independent random variables. Calculating the standard error of a ratio of two independent random variables is more complex than the single-variable case defined in (11).

For any two random variables X, Y, the standard error of the ratio is a function of the standard errors of both variables SE_X, SE_Y. Propagation of uncertainty [18] is the effect of the variables' uncertainties on the uncertainty of a function based on them. Appendix D includes the complete derivation of the ratio standard error using propagation of uncertainty. We define the formula for this ratio, which we use for all comparisons using the variable means μ_X, μ_Y as follows:

$$SE_{\frac{X}{Y}} \approx \left|\frac{\hat{\mu}_X}{\hat{\mu}_Y}\right| \sqrt{\left(\frac{SE_X}{\hat{\mu}_X}\right)^2 + \left(\frac{SE_Y}{\hat{\mu}_Y}\right)^2}. \tag{12}$$

3 Results

In this section, we present and interpret the results from experimenting with the optimal policy for adaptive resource generation. First, we list the different problem parameters that we pass to the policy iteration algorithm in Section 3.1. The following sections analyse the optimal policy properties for a fixed collection of problem parameter values. Section 3.2 takes a look at the actions that are taken by the optimal policy and identifies a clear pattern in the action sequence. In Section 3.3, we analyse the runtime distribution of the optimal policy given the Monte-Carlo process simulator. Following, Section 3.4 and Section 3.5 compare the optimal policy runtime against our chosen baselines and heuristics, respectively.

3.1 Experimental Setup

The optimal policy that the algorithm described in Section 2.2 outputs depends on the required number of links in memory n and the selection of protocols $\mathcal{A}$ that we can use at each time step. In addition, the link generation process is influenced by a number of parameters. Below we provide a list of definitions for those parameters.

- F_{thresh} is the threshold fidelity where links with fidelity smaller than this value are discarded from memory;
- λ defines the linear relationship between the probability of successful generation and the fidelity of the generated link in the heralded generation process that we assume;
- Γ is used in the depolarising noise model assumed in (1).

The default values that we select for the experiments with the optimal policy are defined in Table 1. The only parameter we vary within an experiment is the required number of links in memory n.

Table 1: Parameter selection for the policy iteration algorithm.

Parameter	Values
F_{thresh}	0.5
λ	1.0
Γ	0.1
$\mathcal{A}$ (actions)	$\{0.1, 0.2, 0.3, 0.4\}$
n (required links)	$\{2, 3, 4, 5, 6, 7\}$

We pick a value of $F_{\text{thresh}} \geq 0.5$ because in practice, any QN application would require states which have fidelity at least 0.5 compared to an ideal Bell state [19]. For λ, we select a value of 1.0 to keep the assumed probability-fidelity relationship in [12]. The small value of $\Gamma = 0.1$ allows us to explore a larger state space because links are kept in memory for more time steps. We also provide the algorithm a diverse selection of generation protocols that it can pick depending on the number of links in memory. The maximum TTL for a link generated using these parameter values is $\max t_i = 9$. We increase the required number of links n as much as it is computationally feasible. This allows us to make general conclusions from the patterns that we observe for small numbers of n.

3.2 Optimal Policy Structure

We first take a look at the probabilities that the optimal link generation process chooses as the number of links in memory increases. For this experiment, we fix the required number of links in memory to be $n = 7$. The optimal policy is complex enough so we can see more general patterns in its structure. The results are plotted on Figure 3.

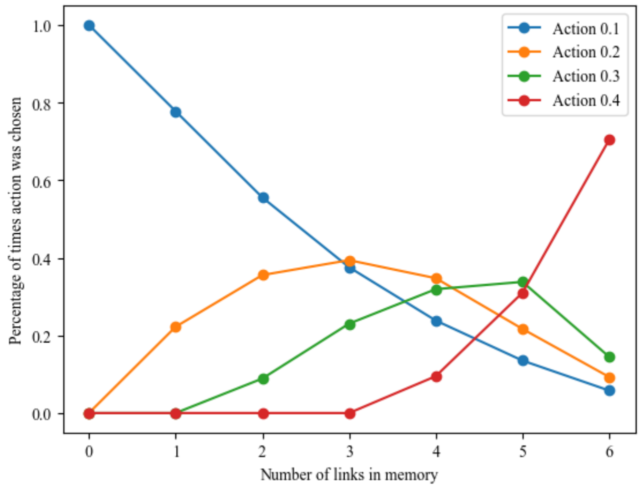

Figure 3: Actions picked by the optimal policy for all possible link counts in memory when $n = 7$. Because there are different memory states with the same link count, different actions are chosen for the same link count.

As the number of links in memory increases, there is a gradual trend toward actions with higher probability of successful generation. This follows from the trade-off between high probability and low link TTL described in Section 2. The optimal policy tends to pick the action with the highest probability that still generates a link which can survive until all of the required links have been generated. We use this intuition to derive a heuristic we use as a baseline for the optimal policy in Section 3.5.

Taking this further, we plot the exact actions that the policy takes for each memory state. That way, we can observe for which particular states the optimal policy picks a different probability action and when the transition towards actions with higher probability happens. Figure 4 shows a plot of the complete policy structure for $n = 3$ required links in memory. For this experiment, we fix $\Gamma = 0.2$ to get a smaller state space with $\max t_i = 4$ that is easier to interpret.

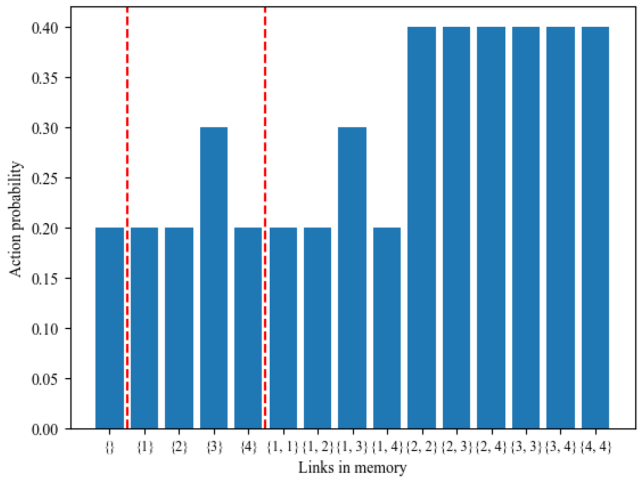

Figure 4: Probabilities picked by the optimal policy for all possible quantum memory configurations when $n = 3$. The red lines separate states with a different number of links in memory.

The chart on Figure 4 deviates from our assumption of a global upward probability trend. For the state $\{1, 1\}$ with two links in memory, the optimal policy picks a probability lower than the highest probability 0.3 chosen for one link in memory. However, we can see a pattern in the chosen probability. If we divide the plot where the number of links in memory increases, we can see that a jump occurs at the same spot for $m = 1$ and $m = 2$ links in memory.

The first states when $m = 2$ are equivalent to the first states when $m = 1$. For $m = 2$, some of the links will expire after exactly one step and the memory would transition to an equivalent state when $m = 1$. Because the process will not reach the required $m = 3$ links after one step, those states are interchangeable. If we exclude the equivalent states, we only observe a gradual trend toward higher probabilities as the number of links in memory increases. This is shown on the plot of the trimmed policy on Figure 5. Note that we are only examining the probability differences between sections with different numbers of links, not between individual states.

3.3 Optimal Policy Runtime Distribution

In this experiment, we simulate the optimal policy to observe the distribution of its runtime. Studying the properties of our simulation allows us to perform meaningful comparisons with baseline policies later on. Figure 6 plots the distribution of the process runtime until four links are generated in memory. The runtime is measured for $N = 100\,000$ number of policy runs. The expected time is estimated using the Monte Carlo method described in Section 2.3. More policy runs are unnecessary because the runtime variance does not change.

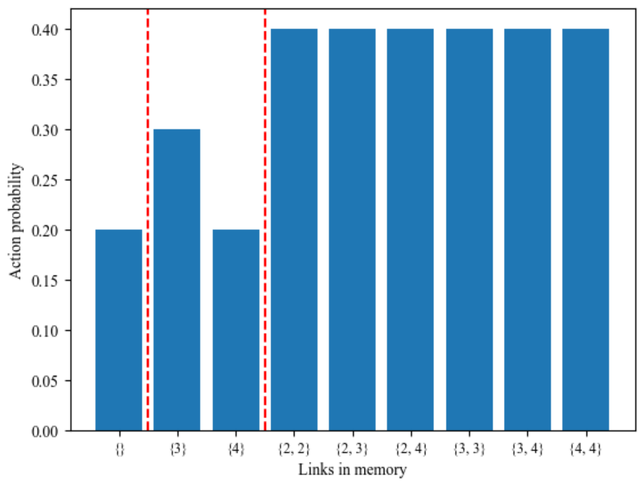

Figure 5: Probabilities picked by the optimal policy for a reduced set of memory configurations when $n = 3$. The red lines separate states with a different number of links in memory.

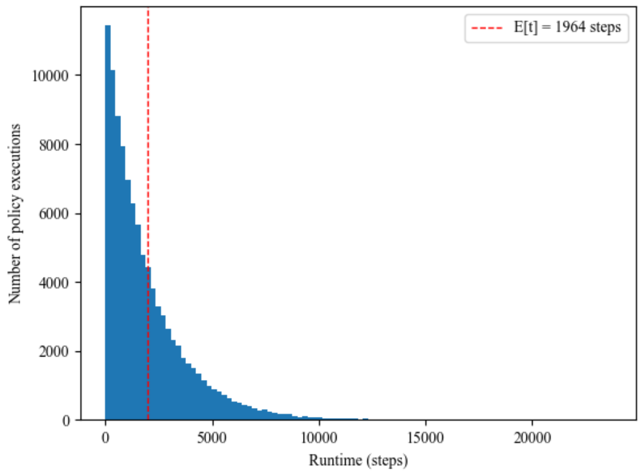

Figure 6: Distribution of the runtime of an optimal policy simulated $N = 100\,000$ times using our Monte Carlo simulator for $n = 7$.

The plot of the optimal policy runtime closely resembles the density of a geometric distribution. If we ignore the cutoff, our process can be defined as a negative binomial distribution since we model the number of failures until we have successfully generated n links [4]. The introduction of the cutoff policy as described in Section 1 alters the probability structure of generating links and makes the distribution inherently more complex. We look at a precise definition of the simple case when $n = 2$ and we have $s = 2$ possible actions in Appendix E.

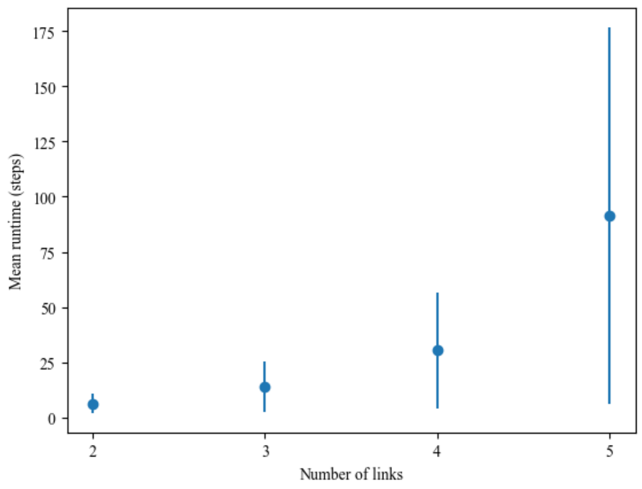

Figure 7: Estimated mean runtime $\hat{\mathbb{E}}_\pi[\tau]$ and standard deviation $\hat{\sigma}_\pi(\tau)$ of the optimal policy π for different required numbers of links in memory.

In Figure 7, we plot how the standard deviation of a simulated policy runtime changes as the number of required links in memory n increases. In this experiment, we analyse up to $n = 5$ required links since after that the values of the mean runtime become very large. For the examples observed, the standard deviation of our simulator increases proportionally to the mean time to generate n links in memory. This makes it hard to perform comparisons on large policies.

3.4 Runtime Against Baseline

First, we compare the derived optimal policy with one that picks a random action at each time step. Figure 8 plots the ratio $\mathbb{E}_{\text{opt}}[\tau]/\mathbb{E}_{\text{random}}[\tau]$. The standard error of the plotted ratio is calculated as defined by (12) in Section 2.3. As expected, we see a notable decrease in the ratio and convergence to zero as the number of links increases. This is because the random policy does not use any available information about the quantum memory and fails to capture the complexity of the optimal policy.

In addition, we compare the optimal adaptive policy with the best static policy, which selects only one action from the set of actions available to the adaptive algorithm. The adaptive optimal policy will never choose an action that makes the expected runtime longer than any static policy. Otherwise, it will no longer be optimal. However, it may select a different action that outperforms the static policy's choice. Therefore, the runtime of the adaptive optimal policy serves as a lower bound for the runtime of any static policy that uses the same set of possible actions. A comparison with the best static policy allows us to quantify the improvement offered by the adaptive optimal policy.

To select the best static policy, we simulate all policies that select only one action from the ones available to our adaptive process. The optimal static policy is the one with the smallest mean runtime. In our case, this is the policy that

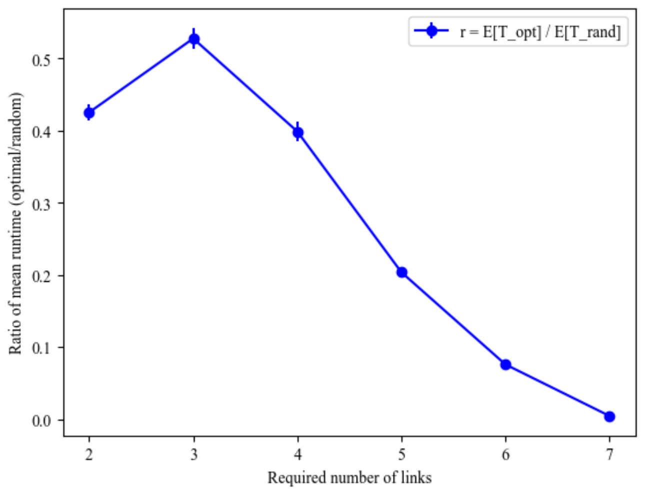

Figure 8: Ratio between the mean runtime of the optimal policy for generating n links and a policy that picks a random action at each time step. The vertical lines plot the standard error of the ratio. We sample $N = 1000$ policy runs for each policy.

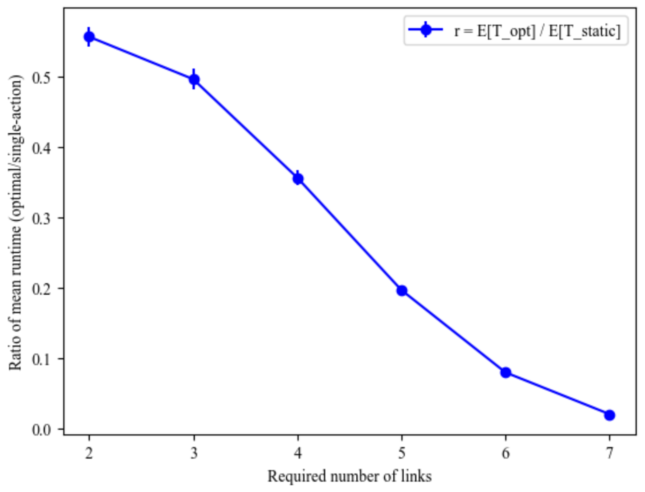

Figure 9: Mean runtime of the optimal policy π for generating n links against the best static policy that picks $p = 0.2$ at each time step. The vertical lines plot the standard error of the ratio. We sample $N = 1000$ policy runs for each policy.

picks the action $p = 0.2$ at each time step. Figure 9 shows that the best static policy is a much better baseline for the adaptive policy than the random one. The change in the ratio $\mathbb{E}_{\text{opt}}[\tau]/\mathbb{E}_{\text{static}}[\tau]$ is more gradual. However, we still observe the expected downward trend and convergence to zero as the number of links increases and the adaptive optimal policy becomes more complex.

3.5 Maximum-Probability Heuristic

Finally, we define a heuristic to compare to the optimal policy. The heuristic is defined as follows. For each memory state, we pick the maximum probability from the provided selection that would still generate a link which would live long enough for the process to complete. Algorithm 1 formally defines how our chosen heuristic selects an action based on a given state.

Algorithm 1 Heuristic picking the maximum probability that still allows the generation of n links.

```
function GetAction(state)
    actions ← all possible actions $a_i \in \mathcal{A}$
    r ← remaining links to generate
    min_diff ← Inf
    best_action ← 0
    for action in actions do
        ttl ← TTL of new link generated by action
        if ttl $\geq$ r and ttl - r $<$ min_diff then
            min_diff ← ttl - r
            best_action ← action
    return best_action
```

This heuristic utilises the available information about our quantum memory. However, it does not completely capture the complexity of the optimal policy. That is why there is still a considerable difference between the runtimes of the optimal policy and the heuristic policy.

Figure 10 shows that the mean ratio for the chosen heuristic decreases slower than the one against a random policy. The plotted ratio also stays higher than the best static policy for the chosen values of n. It is still decreasing because sometimes the optimal policy picks a lower probability to generate a link with a higher TTL that outperforms the heuristic's action. Still, we have derived an intuitive heuristic from the properties of the optimal policy that serves as a sensible baseline.

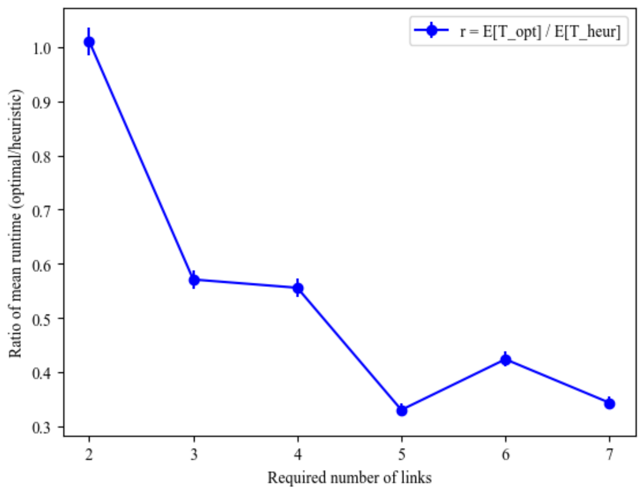

Figure 10: Ratio between the mean runtime of the optimal policy for generating n links in memory and a policy that picks the maximum allowable probability at each time step. The vertical lines plot the standard error of the ratio. We sample $N = 1000$ policy runs for each policy.

4 Responsible Research

This research work was conducted by Boris Goranov, a student at Technische Universiteit (TU) Delft, under the supervision of Bethany Davies, MSc., and Prof. Gayane Vardoyan. The development of our analysis has closely followed the principles introduced in the Netherlands Code of Conduct for Research Integrity [20]. Namely: honesty, scrupulousness, transparency, independence, and responsibility.

Honesty. The code for the Policy Iteration algorithm used to derive an optimal entanglement link generation policy is publicly available in a GitHub repository as specified in Section 6. Therefore, all the findings here can be verified. We adhere to the FAIR principles [21] and thus our research has the Apache License 2.0 so that others can reuse our findings and modify our code with ease.

Scrupulousness and transparency. The code for our Policy Iteration algorithm is meticulously developed and validated through peer review. All experiments were conducted through simulations distributed in the GitHub repository as Jupyter notebooks. This ensures that our experiments are fully transparent, facilitating open access to our methodologies and results.

Independence. Our research is conducted independently without any external influences that could bias our results or conclusions. The project has received no funding from external organizations, and there are no conflicts of interest that could compromise the objectivity of the work. Our conclusions are based purely on the empirical evidence obtained through simulations.

Responsibility. When showcasing improvements, we plot the standard error of our experimental results, which is calculated using (12) defined in Section 2.3. This allows us to quantify the variability within our results and the consistency of our findings. Ultimately, this helps us determine whether our conclusions are reliable and generalisable. The source code has been carefully documented to allow researchers to obtain and validate our findings. In this way we address the existing reproducibility crisis in science [22].

5 Conclusions and Next Steps

Our work presents how we can derive an optimal policy to generate a required n number of entanglement links in a near-term quantum network. We can use the generated links for quantum network protocols such as BQC and Quantum Key Distribution (QKD) that achieve capabilities impossible with classical computing. We have shown that the optimal policy outperforms our chosen baselines. In addition, we have identified a clear pattern in the optimal policy structure that allows us to reduce the search space of our policy iteration algorithm.

In the process, we have found a suitable heuristic for selecting an action for a given memory state. This heuristic can be used as a baseline for our optimal policy. In addition, we can use it to efficiently generate sub-optimal policies with lower expected runtime than a static policy.

In this work, we have assumed a heralded entanglement generation protocol where we can immediately verify that an entanglement link has been generated [12]. In addition, we have not considered any differences in the process time steps. For example, a successfully-generated link may need to get processed before the next attempt can commence [23]. Hence, our optimal policies may become sub-optimal when a successful link generation is slower than a failed one.

Also note that we have restricted our analysis to quantum network protocols that require up to seven links. This is due to the exponentially large computational cost of solving the MDP for larger search spaces. In addition, it becomes unfeasible to analyse the generated policies by hand and make any conclusions about their structure. Having more required links in memory would limit the achievable fidelity of the generation process. Therefore, we consider the analysis of smaller near-term networks more relevant.

An interesting addition to this work would be to extend the definition of the identified heuristic that picks the maximum allowable probability. We believe this heuristic can be further tuned to generate sub-optimal policies closer to the provably optimal policy. For example, it can provide a good benchmark if we use other methods like Machine Learning to approximate optimal policies for large search spaces.

As a final remark, our algorithm assumes the probabilities the generation process can select from are fixed. In the problem setup, we could also consider a more realistic model of a quantum network where the success probability decreases over time because of increased noise. However, this makes the model inherently more complex and solving the MDP analytically might be unfeasible.

6 Code Availability

Our code can be found in the following GitHub repository: https://github.com/Bggoranoff/optimal-qn.git.

7 Funding Sources

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

A Policy Iteration Algorithm

Figure 11 presents the pseudo code for a generic Policy Iteration algorithm for solving an MDP. First, we initialise a random policy. In our case, we use a policy that assigns equal probability to each action. We continue by repeatedly performing the following steps until the policy has not changed.

1. We calculate the value for each state $s \in \mathcal{S}$ using (7) in the policy evaluation step until the values have converged within some predefined tolerance.
2. For each state, we pick the actions that have maximal values and define this as the optimal policy.

```
1. Initialization
   $V(s) \in \mathbb{R}$ and $\pi(s) \in \mathcal{A}(s)$ arbitrarily for all $s \in \mathcal{S}$

2. Policy Evaluation
   Repeat
       $\Delta \leftarrow 0$
       For each $s \in \mathcal{S}$:
           $v \leftarrow V(s)$
           $V(s) \leftarrow \sum_{s',r} p(s',r|s,\pi(s))\big[r + \gamma V(s')\big]$
           $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
   until $\Delta < \theta$ (a small positive number)

3. Policy Improvement
   $policy\text{-}stable \leftarrow true$
   For each $s \in \mathcal{S}$:
       $a \leftarrow \pi(s)$
       $\pi(s) \leftarrow \arg\max_a \sum_{s',r} p(s',r|s,a)\big[r + \gamma V(s')\big]$
       If $a \neq \pi(s)$, then $policy\text{-}stable \leftarrow false$
   If $policy\text{-}stable$, then stop and return $V$ and $\pi$; else go to 2
```

Figure 11: Policy iteration algorithm for solving the Markov decision process. Couresy of [9].

B State Space Size

In this section, we derive the size of the state space for the MDP that we define in Section 2.2. Each memory state has $m \in [0, n]$ number of links where n is the required number of links for the network application. We can group the memory states by size, so $\mathcal{S}_m$ is a subset of $\mathcal{S}$ that contains only the memory states of size m like so:

$$\mathcal{S}_m = \{L_i = \{t_1, t_2, ..., t_m\} \mid t_i \in (0, \max t_i]\}. \tag{13}$$

The number of states in $\mathcal{S}_m$ is $\max t_i$-choose-m with replacement because we can have multiple links with the same TTL. The size of $\mathcal{S}_m$ is defined as [13]:

$$|\mathcal{S}_m| = \left(\!\!\binom{\max t_i}{m}\!\!\right) = \binom{\max t_i + m - 1}{m}. \tag{14}$$

Therefore, the size of $\mathcal{S}$ is a sum of sizes of all $\mathcal{S}_m$:

$$|\mathcal{S}| = \sum_{m=0}^{n} |\mathcal{S}_m| = \sum_{m=0}^{n} \binom{\max t_i + m - 1}{m}. \tag{15}$$

We can calculate a sum of binomial coefficients using the Hockeystick identity [14] defined as:

$$\sum_{i=r}^{n} \binom{i}{r} = \binom{n+1}{r+1}. \tag{16}$$

Finally, we rewrite (15) to make it compatible with the definition of the Hockeystick identity in (16). We then use this identity to remove the summation and get a simplified formula for $|S|$ as follows:

$$|S| = \sum_{m=0}^{n} \binom{\max t_i + m - 1}{m} = \\ = \sum_{i=\max t_i - 1}^{\max t_i + n - 1} \binom{i}{\max t_i - 1} = \\ = \binom{\max t_i + n}{\max t_i}. \tag{17}$$

C Optimal Policy Analytical Solution

Here, we analyse the case where we have $s = 2$ available generation protocols to generate $n = 2$ required links in memory. The protocols we can choose from are (p_1, F_1) and (p_2, F_2) where $F_1 > F_2$ and $p_1 < p_2$. Generating a link is a stochastic process which can be modelled as independent Bernoulli trials with success probability p_i [15]. The expected time to generate a link with success probability p is inversely proportional to p:

$$\mathbb{E}[\tau] = \frac{1}{p}. \tag{18}$$

Let us call p_f the probability of failing to generate the second link after the first within the TTL of the first link and $p_s = 1 - p_f$ the success probability of this event. Therefore, for a protocol selecting first action $i \in \{1, 2\}$ and then action $j \in \{1, 2\}$ we have a recursive definition for $\mathbb{E}_{ij}[\tau]$:

$$\mathbb{E}_{ij}[\tau] = \frac{1}{p_i} + (1 - p_f)\frac{1}{p_j} + p_f \mathbb{E}_{ij}[\tau]. \tag{19}$$

Rearranging (19) to solve for $\mathbb{E}_{ij}[\tau]$, we get the following analytical solution for the expected time to completion for some policy ij:

$$\mathbb{E}_{ij}[\tau] = \frac{1}{p_j} + \frac{1}{p_i}\frac{1}{(1-p_f)} =$$
$$= \frac{1}{p_j} + \frac{1}{p_i}\frac{1}{(1-(1-p_j)^{t_i-1})}. \qquad (20)$$

Note that the TTL of the first link is t_i which we calculate using p_i (5). We want to maximise the probability of generating the second link. Therefore, we would always use the higher probability p_2 for that link. So the optimal expected time is either $\mathbb{E}_{12}[\tau]$ or $\mathbb{E}_{22}[\tau]$, depending on the success probability values. We use a program to determine the values of the expected times and verify the optimal policy generated by the MDP solver for the basic case.

D Ratio Standard Error

In this section, we calculate the standard error of the ratio of two independent random variables X, Y using propagation of uncertainty as described in [18]. We use the standard errors of both variables SE_X, SE_Y and their sample means $\hat{\mu}_X, \hat{\mu}_Y$. The ratio of two random variables is a function of both random variables defined as follows:

$$Z = f(X,Y) = \frac{X}{Y}. \qquad (21)$$

The propagation of uncertainty formula in [18] can be used to describe how the standard error of the two random variables influence the standard error of the function $f(X,Y)$:

$$SE_Z \approx \sqrt{\left(\frac{\partial Z}{\partial \hat{\mu}_X} SE_X\right)^2 + \left(\frac{\partial Z}{\partial \hat{\mu}_Y} SE_Y\right)^2}. \qquad (22)$$

For $Z = \frac{\hat{\mu}_X}{\hat{\mu}_Y}$ the partial derivatives w.r.t. the sample means $\hat{\mu}_X, \hat{\mu}_Y$ are defined as follows:

$$\frac{\partial Z}{\partial \hat{\mu}_X} = \frac{1}{\hat{\mu}_Y}, \frac{\partial Z}{\partial \hat{\mu}_Y} = -\frac{\hat{\mu}_X}{\hat{\mu}_Y^2}. \qquad (23)$$

We combine (22) and (23) to obtain the closed form of the standard error of the ratio Z we use in Section 2.3:

$$SE_Z \approx \sqrt{\left(\frac{1}{\hat{\mu}_Y} SE_X\right)^2 + \left(-\frac{\hat{\mu}_X}{\hat{\mu}_Y^2} SE_Y\right)^2}$$
$$= \left|\frac{1}{\hat{\mu}_Y}\right| \sqrt{(SE_X)^2 + \left(\frac{\hat{\mu}_X}{\hat{\mu}_Y} SE_Y\right)^2}$$
$$= \left|\frac{\hat{\mu}_X}{\hat{\mu}_Y}\right| \sqrt{\left(\frac{SE_X}{\hat{\mu}_X}\right)^2 + \left(\frac{SE_Y}{\hat{\mu}_Y}\right)^2}. \qquad (24)$$

E Optimal Policy Runtime Distribution

In this section, we provide a closed form of the optimal policy runtime. We look at the simple case when $n = 2$ and we have $s = 2$ actions. Suppose that we have an ordered policy $\pi = (p_1, p_2)$ where it is possible that $p_1 \neq p_2$. Then, we can use a similar model for the process runtime τ as the one described in [4]:

$$\tau(t_1, n=2) = \sum_{j=1}^{M} T_j + (M-1)(t_1-1) + L. \qquad (25)$$

Below we explain the role of the different parameters in (25).

- $T_j \sim \text{Geom}(p_1)$ is the number of attempts to generate the first link with probability p_1;
- $M \sim \text{Geom}(1-(1-p_1)^{t_1-1})$ is the number of times the first link needs to be generated with probability p_1;
- $t_1 \in \mathbb{N}$ is the time until the first link is discarded from memory;
- $L \sim \text{Geom}(p_2) \mid M$ is the number of attempts to generate a second link with probability p_2 after the first one has been generated.

Therefore, the process runtime is a sum of geometric distributions with different probabilities. Geometric distribution is a special case of Negative Binomial distribution (NB) [24]. The distribution of a sum of independent and ideally distributed (i.i.d.) NB random variables is a mixture Negative Binomial distribution [25]. Appendix B of [4] provides a formal proof that an upper bound for this sum is a Geometrically distributed random variable.

References

[1] S. Wehner, D. Elkouss, and R. Hanson, "Quantum internet: A vision for the road ahead," *Science*, vol. 362, p. eaam9288, Oct. 2018.

[2] A. Broadbent, J. Fitzsimons, and E. Kashefi, "Universal blind quantum computation," in *2009 50th Annual IEEE Symposium on Foundations of Computer Science*, pp. 517–526, Oct. 2009. arXiv:0807.4154 [quant-ph].

[3] V. Scarani, H. Bechmann-Pasquinucci, N. J. Cerf, M. Dusek, N. Lutkenhaus, and M. Peev, "The Security of Practical Quantum Key Distribution," *Reviews of Modern Physics*, vol. 81, pp. 1301–1350, Sept. 2009. arXiv:0802.4155 [quant-ph].

[4] B. Davies, T. Beauchamp, G. Vardoyan, and S. Wehner, "Tools for the analysis of quantum protocols requiring state generation within a time window," *IEEE Transactions on Quantum Engineering*, pp. 1–20, 2024. Conference Name: IEEE Transactions on Quantum Engineering.

[5] R. Jozsa, "Fidelity for mixed quantum states," *Journal of Modern Optics*, vol. 41, no. 12, pp. 2315–2323, 1994.

[6] G. Iñesta, G. Vardoyan, L. Scavuzzo, and S. Wehner, "Optimal entanglement distribution policies in homogeneous repeater chains with cutoffs," *npj Quantum Information*, vol. 9, pp. 1–7, May 2023. Publisher: Nature Publishing Group.

[7] S. Raychaudhuri, "Introduction to Monte Carlo simulation," in *2008 Winter Simulation Conference*, pp. 91–100, Dec. 2008. ISSN: 1558-4305.

[8] R. S. Sutton and A. G. Barto, "Reinforcement Learning: An Introduction," pp. 53–80.

[9] R. S. Sutton and A. G. Barto, "Reinforcement Learning: An Introduction," pp. 89–107.

[10] R. Bellman, "A Markovian Decision Process," *Indiana University Mathematics Journal*, vol. 6, no. 4, pp. 679–684, 1957.

[11] Y. Wang, "Markov chains and markov decision processes,"

[12] H. Bernien, B. Hensen, W. Pfaff, G. Koolstra, M. S. Blok, L. Robledo, T. H. Taminiau, M. Markham, D. J. Twitchen, L. Childress, and R. Hanson, "Heralded entanglement between solid-state qubits separated by three metres," *Nature*, vol. 497, pp. 86–90, May 2013.

[13] C. Liu, "Introduction to combinatorial mathematics," Computer science series, pp. 14–15, McGraw-Hill, 1968. ISSN: 6802-2763.

[14] C. H. Jones, "Generalized hockey stick identities and iv-dimensional blockwalking," 1994.

[15] "Chapter 2 - estimation of probability densities," in *Stochastic Processes* (K. Najim, E. Ikonen, and A.-K. Daoud, eds.), pp. 93–166, Oxford: Kogan Page Science, 2004.

[16] F. M. Dekking, C. Kraaikamp, H. P. Lopuhaä, and L. E. Meester, "Unbiased estimators," in *A Modern Introduction to Probability and Statistics: Understanding Why and How* (F. M. Dekking, C. Kraaikamp, H. P. Lopuhaä, and L. E. Meester, eds.), pp. 285–297, London: Springer, 2005.

[17] R. R. Wilcox, "Chapter 3 - estimating measures of location and scale," in *Introduction to Robust Estimation and Hypothesis Testing (Fifth Edition)* (R. R. Wilcox, ed.), pp. 45–106, Academic Press, fifth edition ed., 2022.

[18] J. Taylor, "An introduction to error analysis: The study of uncertainties in physical measurements," ASMSU/Spartans.4.Spartans Textbook, pp. 45–79, University Science Books, 1997.

[19] R. Van Meter, "Quantum Networking," pp. 37–38, Newark, UNITED STATES: John Wiley & Sons, Incorporated, 2014.

[20] "Netherlands Code of Conduct for Research Integrity | NWO."

[21] D. M. A. I. e. a. Wilkinson, M., "The FAIR Guiding Principles for scientific data management and stewardship," *Scientific Data*, vol. 3, p. 160018, Mar. 2016. Publisher: Nature Publishing Group.

[22] M. Baker, "1,500 scientists lift the lid on reproducibility," *Nature*, vol. 533, pp. 452–454, May 2016. Publisher: Nature Publishing Group.

[23] 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*, vol. 356, pp. 928–932, June 2017. Publisher: American Association for the Advancement of Science.

[24] O. C. Ibe, "1 - basic concepts in probability," in *Markov Processes for Stochastic Modeling (Second Edition)* (O. C. Ibe, ed.), pp. 1–27, Oxford: Elsevier, second edition ed., 2013.

[25] E. Furman, "On the convolution of the negative binomial random variables," *Statistics & Probability Letters*, vol. 77, pp. 169–172, Jan. 2007.