Adaptive Algorithm for Resource Generation in a Quantum Network

Using a Markov Decision Process Approach to Optimize the Quantum Resource Generation Algorithm

Ioana-Lisandra Draganescu[1]
Supervisors: 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: Ioana-Lisandra Draganescu
Final project course: CSE3000 Research Project
Thesis committee: Gayane Vardoyan, Bethany Davies, Rihan Hai

An electronic version of this thesis is available at http://repository.tudelft.nl/.

Abstract

Entangled links can be seen as a connection between two parties that persists remotely, but whose quality (fidelity) decreases over time. Many quantum applications rely on having a certain number of simultaneously active entangled links for their execution. Every application has a characteristic threshold fidelity - the minimum fidelity a link should have for it to be useful. There are several link generation protocols; a protocol i generates with probability P_i a link with initial fidelity F_i. Until now, research has focused on generating all links with the same protocol. Therefore, the aim of this research is to use multiple generation protocols in order to minimize the expected time needed to reach the desired number of links, and to analyze how this time changes with regard to the threshold fidelity of the target protocol. We present how to reach the minimum possible expected time by modelling the problem as a Markov decision process and determining the optimal sequence of protocols to be used. Then, we observe that our approach generally performs several orders of magnitude better than previous research in which all links were generated with the same protocol.

1 Introduction

An entangled link can be seen as a connection between two parties that persists even over large distances. Given that this connection does not depend on the distance between the parties, entangled links enable performing operations that cannot be done classically. Therefore, many quantum applications rely on them. In this paper, we use the terms *(quantum) application* and *(quantum) target protocol* interchangeably; they refer to the protocols that need the entangled links to function. Generally, quantum applications need a certain minimum number of simultaneously active entangled links to be able to be carried out [1]. The purpose of a quantum network is to generate these links, which are called, in this context, quantum resources.

One example of a protocol that relies on entangled links to function is a variation of Blind Quantum Computation (BQC). BQC is a name that broadly encompasses a set of protocols that aim to allow a client to perform a computation on one or more quantum servers without revealing the structure of the computation [2]. More specifically, BQC with multiple quantum servers requires entangled links shared between the participating servers [2].

The fidelity of an entangled link is a measure of how close it is to a maximally entangled state (for example, $|\phi^+\rangle$) [3]. Each target protocol is characterized by a minimum fidelity that a link can have for it to be useful to the protocol [1]. We will call this minimum value the threshold fidelity of a protocol. Unfortunately, the fidelity of a link decreases over time (see Figure 1) and, when it becomes lower than the threshold value, the link should be discarded, as it is no longer useful [1, 3]. Because the fidelity of a link decreases while it waits for subsequent links to be generated, producing a certain number of simultaneously active entangled links with a sufficiently high fidelity can be quite challenging.

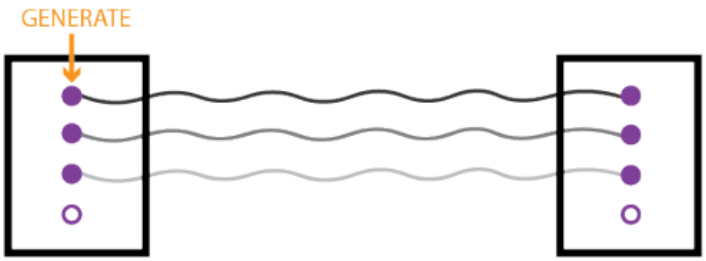

Figure 1: The fidelity of an entangled link between two parties decreasing over time. Adapted from B. Davies with permission.

Entangled links can be generated using different quantum resource generation protocols. Each of these protocols can produce with probability P a new entangled link of fidelity F, where F and P are values that characterize the protocol. Most of the time, there is a trade-off between F and P, meaning that there is no protocol that is ideal for all cases [1]. Thus, achieving the desired number of links is not a trivial problem.

Until now, research has focused on generating all of the entangled links needed by a quantum target protocol using resource generation protocols with identical parameters [1, 3]. This means that all links are generated with the same probability P and have the same initial fidelity F. In short, an optimal policy is defined as a function that, based on the current number and quality of links, specifies which resource generation protocol should be used next in order to reach the desired number of active links as soon as possible. A method of calculating the optimal policy is presented in [1] - the problem is modeled as a Markov decision process, and, then, value iteration and policy iteration are used to find the optimal policy.

In order to determine the optimal policy, we employed a method similar to the one used in [1]. To detail, we model the problem as a Markov decision process (MDP), and then we use a value iteration algorithm to calculate the optimal policy. The reason we chose this approach is that it provides an easy and certain method for computing the optimal policy.

The aim of this paper is to determine how the optimal resource generation protocol varies with the fidelity threshold of a given target protocol, when combining quantum resource generation protocols with different parameters is allowed. For example, one protocol could have a higher generation probability but a lower initial fidelity, while another could behave the exact opposite way. In this case, if two links are needed, it might be optimal to generate the first link with the second protocol, hoping that it would keep a high enough fidelity until the second link is generated, and the second link with the first protocol, in order to maximize the chance of a link to be generated. Thus, allowing the mixing of generation protocols can lead to even better resource generation policies.

Our main contributions are as follows:

- we model the problem as an MDP;
- we calculate the optimal policy for the case when generating the links with different protocols is allowed, by using a value iteration algorithm;
- we introduce a heuristic that, in some cases, reaches the desired number of links faster than generating all of the links with the same protocol.

In what follows, we present the background of the problem and some related work in Section 2. Then, we discuss the methodology in Section 3. Section 4 presents and analyzes the results, while Section 5 presents the conclusions and offers suggestions for potential future research directions. Lastly, in Section 6, we reflect on the ethics of the project and reproducibility of the results.

2 Background and Related Work

In this section, we will present the works that our research builds upon, highlighting the sections that are the most relevant for our approach. In short, [3] provides a model for the rate at which the fidelity of a link decreases, while [1] shows how to calculate the expected time to achieve two simultaneously active links and discusses a trade-off function between the initial fidelity F and generation probability p of a resource generation protocol.

Due to coupling with the environment, the fidelity of an entangled link decreases with time [4]. This process is called decoherence [3]. In [3], a model for the decoherence rate of an entangled link is presented:

$$F(t) = (F(t - \Delta t) - \frac{1}{4})e^{-\frac{\Delta t}{\tau}} + \frac{1}{4}, \tag{1}$$

where Δt is an arbitrary interval of time and τ characterizes the exponential decay in the fidelity of an entangled link. Thus, under the assumption that a time step is of length 1 time unit, we can model the decoherence of a link after one time step as follows:

$$F(t) = (F(t - 1) - \frac{1}{4})e^{-\frac{1}{\tau}} + \frac{1}{4}. \tag{2}$$

Then, from this, we can model the fidelity of a link after m time steps. If we initially have a link of fidelity F, its fidelity after m time steps in memory would be given by:

$$F \to (F - \frac{1}{4})e^{-m\Gamma} + \frac{1}{4}, \tag{3}$$

where Γ represents the rate of decoherence in memory. Therefore, by using this equation, we can calculate how many time steps we can keep a link before having to discard it, or, in other words, the time-to-live of a link. In Section 3.2 we will provide the details of this calculation.

A contribution that is particularly useful for our research is the calculation of the expected time to achieve two simultaneously active entangled links, in the case when these links are generated using the same protocol [1]. We extend this idea to calculate the expected time to generate two links when using protocols with different parameters is allowed. Then, we use the expected time to verify the validity of our approach (described in Subsection 4.1) in the case of generating two links.

Going back to the approach presented in [1], in order to determine the expected time to achieve two simultaneously active links, the idea of a time window w is introduced [1]. Then, the goal of the resource generation protocol is to generate two links within one time window [1]. The purpose of the window is to ensure that, by the time the second link is generated, the first one is still active and has a fidelity higher than the threshold. The window size is therefore directly related to the time-to-live of a link. Given that all links are generated with the same protocol, they all live for the same number of time steps; this number of time steps is then equal to the window size.

With these in mind, the following formula for the expected time to achieve two simultaneously active links generated with only one protocol is derived [1]:

$$\mathbb{E}[T] = \frac{1}{p} + \frac{1}{p(1-(1-p)^{w-1})}, \tag{4}$$

where p is the probability with which our resource generation protocol generates a link. A similar formula is derived for the case when we can use two different protocols. This derivation is shown in Section 4.1.

Lastly, [1] motivates the usage of a linear trade-off function between the probability P and fidelity F of a protocol. This trade-off is given by

$$F = 1 - \lambda P, \tag{5}$$

where λ is the trade-off factor. In our paper, we consider $\lambda = 1$, and, thus,

$$F = 1 - P. \tag{6}$$

3 Methodology and Problem Formulation

3.1 Methodology Background

Reinforcement learning problems are usually defined in terms of an agent and an environment. As stated in [5], the agent is an entity that makes decisions when prompted by the environment, with the goal of learning. Then, the environment is defined as everything outside of the agent. At each moment in time, the environment is in a specific state; all possible states form the state space. When presented with the current state, the agent selects an available action that takes the environment to the next state; all possible actions form the action space [5]. In order to facilitate the learning process, the agent is given a numerical reward based on the action it selected [5]. This way, the agent is encouraged to learn to make good choices - choices that maximize the reward. Therefore, we can define a reinforcement learning problem as a tuple (S, A, p, r), where S is the state space, A is the action space, $p : S \times A$ is the probability function, and $r : S \times A \times S$ is the reward function.

A Markov decision process (MDP) is a way of modelling a reinforcement learning problem that satisfies the Markov property [5]. The Markov property states that the transition to a new state is only determined by the current state, and not by a series of preceding states [6]. An example of an MDP can be seen in Figure 2: the state space is $S = \{S_0, S_1, S_2\}$, the action space is $A = \{a_0, a_1\}$, the probability function is $p : S \times A$ where $p((s, a))$ is given by the values on the black arrows, and the reward function is $r : S \times A \times S$ where $r((S_2, a_1, S_0)) = -1$, $r((S_1, a_0, S_0)) = +5$, and all other values are 0.

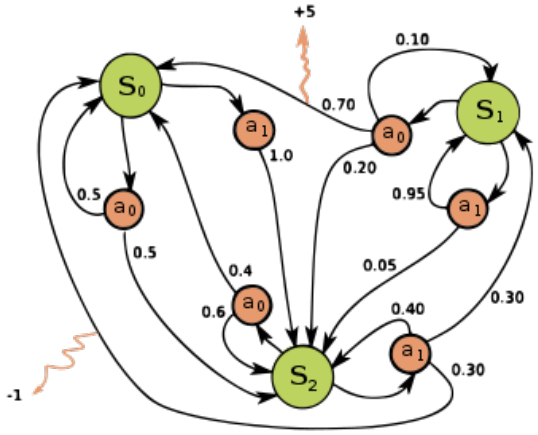

Figure 2: Example of an MDP [7].

A policy is a mapping from a state to a single action, in the case of deterministic policies, or to a set of actions, each with a specific probability, in the case of stochastic policies [5]. Given that in the case of an MDP with a finite and countable set of states there always exists an optimal deterministic policy, we will only consider deterministic policies in this paper [8, 1]. We define an optimal policy as a policy that maximizes the reward.

In order to determine an optimal policy for a given problem, we could solve the corresponding Bellman equation to determine the value of each state. The Bellman equation relates the value of the current state to the values of its successors [5]. Subsequently, by beginning with the start node and always following the successor with the highest value, we could reach a goal state while maximizing the reward. In the general case, the Bellman equation looks as follows:

$$v_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)[r + \gamma v_\pi(s')], \tag{7}$$

where π is the policy, S is the state space, A is the action space, R is the reward set, γ is the discount rate, $s, s' \in S$, $a \in A$, $r \in R$ [5]. According to [5], the discount rate γ, $0 \leq \gamma \leq 1$, determines how strongly the agent takes into account future rewards - at each step between the current state and a successor state, the reward given by the current state is multiplied by γ. Thus, the agent will consider only γ^{n-1} times the reward of a state that will be reached n steps in the future.

3.2 Modelling the Problem

In this subsection, we will model the problem as an MDP. This approach is particularly useful because it offers a tried-and-tested method of finding an optimal policy - solving the Bellman equation. In other words, it is possible to model the problem as an MDP with a finite and countable number of states, which already guarantees us a strategy that yields an optimal policy. Then, in order to analyze how the optimal policy changes with the threshold

fidelity of a target protocol, it would only be needed to change the value of the threshold fidelity in the model.

In order to define the state space, we first have to analyze the time-to-live of a link. As mentioned in Section 2, the time-to-live of a link is the number of time steps a link has a fidelity higher than the threshold. Assuming that the initial fidelity of a link generated with a given protocol i is F_i and denoting the number of time steps the link can live before it has to be discarded with m_i, we can calculate the value of m_i using Equation (3):

$$\begin{aligned}
(F_i - \frac{1}{4})e^{-m_i\Gamma} + \frac{1}{4} &\geq F_{thresh} \\
(F_i - \frac{1}{4})e^{-m_i\Gamma} &\geq F_{thresh} - \frac{1}{4} \\
-m_i\Gamma &\geq \ln \frac{F_{thresh} - \frac{1}{4}}{F_i - \frac{1}{4}} \\
m_i &\leq -\frac{1}{\Gamma} \ln \frac{F_{thresh} - \frac{1}{4}}{F_i - \frac{1}{4}}. \qquad (8)
\end{aligned}$$

From this, we can deduce that a link generated with a protocol i can live at most $m_i = \lfloor -\frac{1}{\Gamma} \ln \frac{F_{thresh} - \frac{1}{4}}{F_i - \frac{1}{4}} \rfloor$ time steps.

As a way of facilitating the modelling of the states, we introduce the concept of fidelity bins (see Figure 3). The purpose of the bins is to discretize the time-to-live, and, thus, the fidelity of the links in memory. With this in mind, we set the total number of bins

$$m = max_i(m_i), \qquad (9)$$

and we index the bins starting with 0. Then, if a link is generated with a protocol i, it would be initially placed in bin $m - m_i$ (or $m - m_i + 1$, if we start the indexation of the bins at 1, as shown in Figure 3). At each time step, a link is moved from bin k to bin $k + 1$; if $k + 1 \geq m$, then this means that the fidelity of the link is below the threshold and the link is thus discarded.

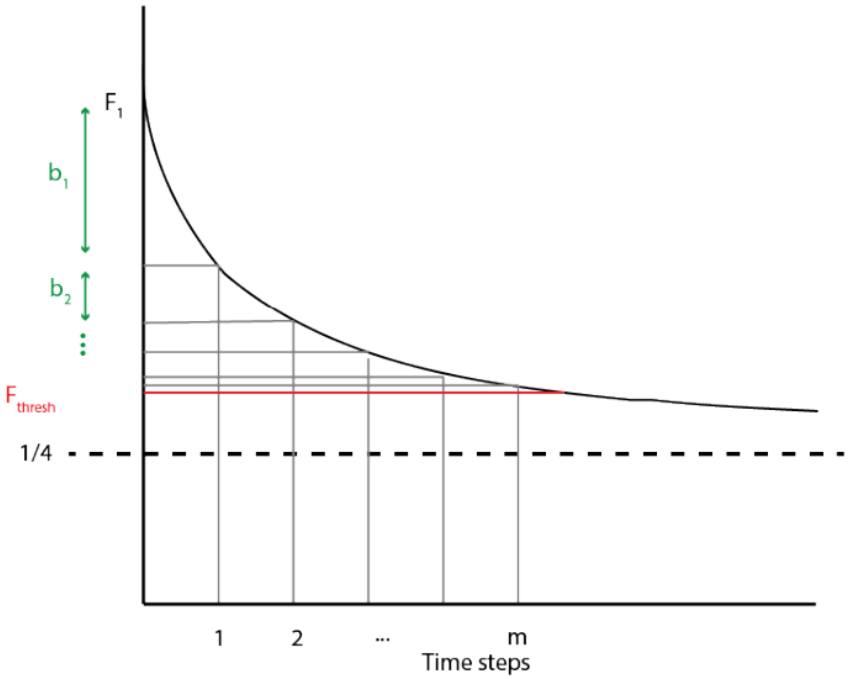

Figure 3: Fidelity bins. Adapted from B. Davies with permission.

Next, we define a state through the number of active links, the bins these links are in, and the number of links needed to reach a goal state. Denote L to be the desired number of links. Then, a state is a multiset of size L whose elements are the bins in which all the active links are, alongside $L - x$ appearances of -1 representing the links still needed to be generated, where x is the number of active links. The state space is then formally defined as

$$S = \{\{l_0, l_1, ..., l_{L-1}\} \,|\, l_0, l_1, ..., l_{L-1} \in \{-1, 0, ..., m-1\}\}. \tag{10}$$

In the context of our problem, we define an action as attempting to generate the next link with a specific protocol. Thus, if we have n generation protocols, the action space would be

$$A = \{a_0, a_1, ..., a_{n-1}\}, \tag{11}$$

where a_i means attempting to generate the next link with protocol i. Once an action a_i is chosen, it generates a link of fidelity F_i with probability p_i, or generates no link with probability $1 - p_i$. In both cases, all initially active links should first be moved one bin further, as described above.

Therefore, we can formally define the probability function the following way. First, for a clearer definition, we make use of an auxiliary function

$$f(l) = \begin{cases} -1, \text{ if } l = -1 \\ l + 1, \text{ if } l + 1 < m \\ -1, \text{ otherwise} \end{cases}, \tag{12}$$

where l is the bin in which a link is. The function f models the result of the passing of one time step on a link - it is either moved to the next bin, or is discarded. Then, we define the space of non-goal states as

$$S_{-1} = \{s \in S \mid -1 \in s\}. \tag{13}$$

Given that once we reach a goal state no subsequent transition is necessary, we can define the probability function on S_{-1}. With this in mind, the probability function $p : S \times S_{-1} \times A \to [0, 1]$ is defined as follows:

$$p(s' \mid s, a_i) = \begin{cases} p_i, \text{ if } s' = \{f(l) \mid l \in s\} \backslash \{-1\} \cup \{m - m_i\} \\ 1 - p_i, \text{ if } s' = \{f(l) \mid l \in s\} \\ 0, \text{ otherwise} \end{cases} \quad , \forall s' \in S,\ s \in S_{-1},\ a_i \in A. \tag{14}$$

In the case of our problem, an optimal policy is defined as a policy that minimizes the expected time required to reach a goal state. Therefore, we use an adapted Bellman equation derived by [3]:

$$T_\pi(s) = 1 + \sum_{s' \in S} p(s'|s, \pi(s)) T_\pi(s'),\ \forall s \in S, \tag{15}$$

where π is the policy we are following and T represents the expected time until reaching a goal state.

In order to solve the Bellman equation and thus determine the expected time to reach a goal state from each state, we used a slightly adapted version of the value iteration algorithm in [5, p. 115], as presented in Algorithm 1. This dynamic programming algorithm guarantees that the values for the times to reach a goal state will converge to the correct values in an infinite number of iterations [5, 3]. We thus stop the algorithm once the difference between iterations does not exceed a very small positive threshold θ, which we set to 10^{-7}.

Algorithm 1 Value Iteration [5]

Initialize array T arbitrarily
repeat
  $\Delta \leftarrow 0$
  for each $s \in S$ **do**
    $v \leftarrow T(s)$
    $T(s) \leftarrow min_a(1 + \sum_{s' \in S} p(s'|s, a) T(s'))$
    $\Delta \leftarrow max(\Delta, |v - T(s)|)$
  end for
until $\Delta < \theta$ (a small positive number)
Output a deterministic optimal policy π such that $\pi(s) = argmin_a(1 + \sum_{s' \in S} p(s'|s, a) T(s'))$

4 Experimental Setup and Results

4.1 Analytical Solution for the Two-Links, Two-Protocols Case

As mentioned in Section 2, we used an analytical solution for the two-links, two-protocols case as an additional check for the validity of our proposed solution. Assume we have two protocols, with $0 < P_0 < P_1 < 1$ and $1 > F_0 > F_1 > F_{thresh}$, and that our goal is to have

two simultaneously active links. Given that the last link to be generated does not have to wait for the generation of any subsequent links, its initial fidelity does not matter, as long as it is higher than the threshold. Therefore, the second link should always be generated with the protocol with the highest probability (in our case, P_1). In other words, when we have one link in memory, the optimal policy would always suggest using the second protocol. With these in mind, the expected time to reach a goal state is determined only by the action we choose when there are no active links in memory (the protocol with which we attempt to generate the first link).

We define E_i to be the expected time to reach a goal state if we attempt to generate the first link with protocol i. Then, we define

$$P_{fail} = (1 - P_2)^{m_i - 1} \tag{16}$$

to be the probability to fail generating the second link while the first one is still active. To detail, $1 - P_2$ is the probability that one attempt fails. If $m_i - 1$ consecutive attempts fail, then the first link would be discarded before the second link is generated, so the generation of the second link while the first one is still active fails. Then, the expected time to generate both links is defined as follows:

$$E_i = \frac{1}{P_i} + (1 - P_{fail})\frac{1}{P_2} + P_{fail}E_i. \tag{17}$$

The first term of this equation, $\frac{1}{P_i}$, represents the probability to generate the first link with protocol i. Then, the second term, $(1 - P_{fail})\frac{1}{P_2}$, represents the probability that the second link is generated while the first one is still active. Lastly, the third term, $P_{fail}E_i$, is a recursive term meaning that, if we failed to generate the second link while the first one is still active, we have to start again.

By solving Equation 17, we determine E_i:

$$E_i = \frac{1}{P_i} + (1 - P_{fail})\frac{1}{P_2} + P_{fail}E_i$$

$$(1 - P_{fail})E_i = \frac{1}{P_i} + (1 - P_{fail})\frac{1}{P_2}$$

$$E_i = \frac{1}{P_i(1 - P_{fail})} + \frac{1}{P_2}$$

$$E_i = \frac{1}{P_i(1 - (1 - P_2)^{m_i - 1})} + \frac{1}{P_2}. \tag{18}$$

As it can be observed, this equation is very similar to Equation 4. Then, we use the values of E_0 and E_1 to determine the protocol with which the optimal policy should choose to attempt the generation of the first link. Because we want to minimize the expected time to reach a goal state, if $E_0 > E_1$, we attempt to generate the first link with protocol 1; otherwise, we attempt to generate the first link with protocol 0.

4.2 Experimental Setup

As a means to better analyze the behaviour of the optimal policy, we use two other policies as a baseline. The first one is a single-protocol policy - at each step, we attempt generation with the protocol with the highest fidelity. The comparison between our policy and the

single-protocol policy is particularly interesting, as the single-protocol policy represents the current state of research on our topic. The second baseline policy makes use of a heuristic that we will describe in what follows.

An observation used in the derivation of the heuristic is that, as a rule of thumb, the first links should be generated with a fidelity that is high enough so that they are not discarded until all links are generated. Conversely, for the last links a high fidelity is not as important; instead, a high probability is more valued. Assuming we have n generation protocols, we first order them decreasingly according to the initial fidelity of a link generated with them. Then, if we have x active links and we want a total of L links, we sample from a normal distribution centered at $\frac{n}{L}x$ with a standard deviation $\sigma = \frac{n}{L}$. Finally, we choose the protocol whose index is the closest to the sampled number. Therefore, if there are fewer active links, it is more likely to generate the next link with a high fidelity, and if there are more active links, it is more likely to generate the next link with a high probability.

In order to perform the analysis, we run a Monte Carlo simulation on the optimal policy, as well as on the baseline policies. In other words, we simulate both policies 10000 times, we average the results, and we calculate the standard error. Given that this research focuses on the behaviour of the optimal policy with respect to the fidelity threshold of the target protocol, we run this simulation for different values of the threshold. In all cases, we begin with $F_{thresh} = 0.5$, and, at each iteration, we increase its value with 0.025. We stop when the number of fidelity bins determined by the threshold fidelity makes it impossible to generate all links.

4.3 Case 1: Three Links, Few Bins

In this case, we have seven protocols with probabilities 0.025, 0.05, 0.075, 0.1, 0.2, 0.3, 0.4. According to Equation 6, we have the following respective fidelities: 0.975, 0.95, 0.925, 0.9, 0.8, 0.7, 0.6. Our goal is to generate two simultaneously active links. In this case, we use the rate of decoherence in memory $\Gamma = 0.1$. This value of Γ provides us with relatively few fidelity bins. Therefore, once the threshold fidelity reaches 0.8, we do not have enough bins to possibly generate all links (the fidelity of the first link would be below the threshold until the last one is generated), and therefore we stop the simulation.

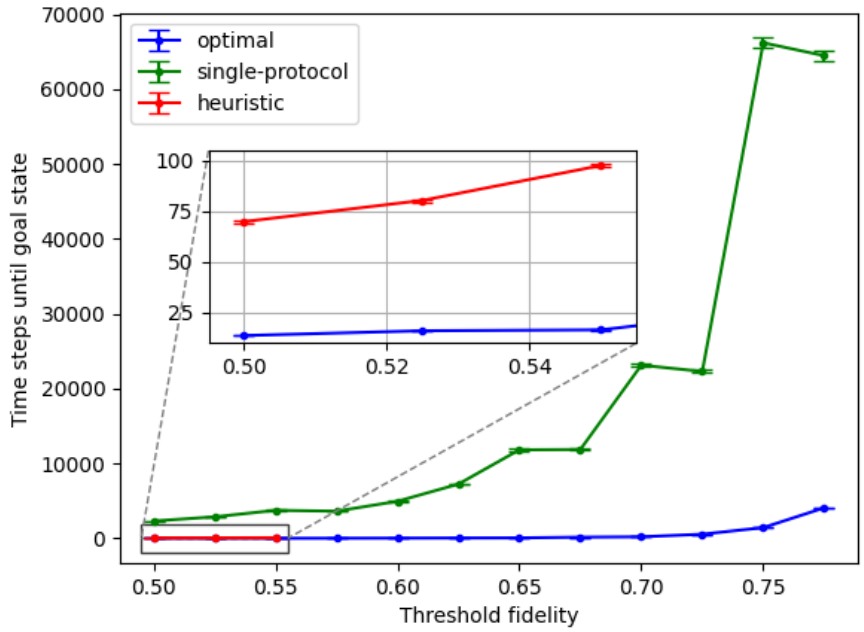

Figure 4: The behaviour of the optimal, single-protocol, and heuristic policies when our goal is three simultaneously active links, in the context of relatively few fidelity bins.

The plot in Figure 4 represents the average number of steps needed to reach the goal state over the 10000 simulations. Then, the error bars represent the standard error of the number of time steps needed to reach a goal state over the simulations. The zoomed-in section highlights the difference between the optimal and heuristic policies.

The first observation that can be made is that the graph of the heuristic policy stops after the threshold fidelity reaches 0.55. The reason behind this is that, for $F_{thresh} = 0.575$, the policy does not reach a goal state in the amount of time that the experiment was run (more than 6 hours). One possible explanation is that, as the threshold fidelity grows, the proposed heuristic repeatedly chooses to attempt to generate the next link with a protocol with an initial fidelity F that does not allow the link to live until all links are generated. In this case, the heuristic could be adjusted to prevent such situations. On the other hand, it is important to note that for the values of F_{thresh} for which the heuristic reaches a goal state, it does so in significantly less time than the single-protocol policy - tens of steps, instead of thousands.

One of our main contributions is the calculation of an optimal policy when using different generation protocols is allowed. When comparing it with the single-case policy, we observe that the difference in number of steps grows as the threshold fidelity grows. In particular, for $F_{thresh} = 0.775$, the average number of time steps needed to reach a goal state for the single-protocol policy is approximately 10 times larger than the one for the optimal policy. Even though the actual difference is larger for larger values of F_{thresh}, the ratio is larger for lower values of F_{thresh}. For example, for $F_{thresh} = 0.65$, the optimal policy performs on average 100 times better than the single-protocol policy. One possible explanation for the

ratio being larger for smaller values of F_{thresh} is that, at that point, more resource generation protocols produce viable links. Therefore, the optimal policy has more options to choose from, so a higher chance of lowering the expected time to reach a goal state. On the other hand, the single-protocol policy is forced to choose the same protocol, which significantly increases its average time to reach a goal state.

4.4 Case 2: Two Links, Many Bins

In this case, we use the same seven protocols as in Subsection 4.3. The difference is that, now, our goal is to generate two simultaneously active links, and the value we chose for the rate of decoherence in memory is $\Gamma = 0.01$. This value provides us with a relatively high number of fidelity bins, which allows us to calculate the policies for fidelity thresholds as high as 0.95. After the threshold is higher than this value, we do not have sufficient bins to be able to generate all links, and we thus stop the simulation.

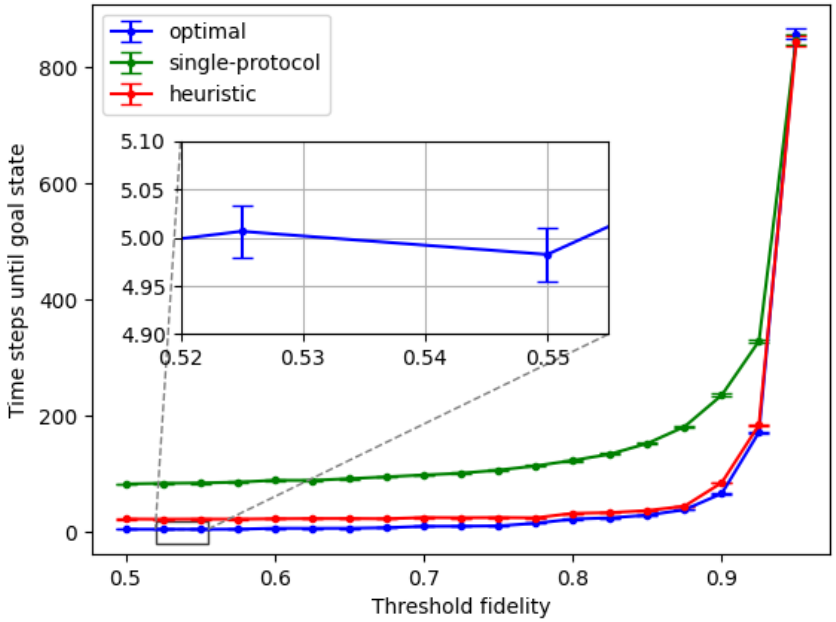

Figure 5: The behaviour of the optimal, single-protocol, and heuristic policies when our goal is two simultaneously active links, in the context of relatively many fidelity bins.

The plot in Figure 5 represents the average number of steps needed to reach the goal state over the 10000 simulations. Then, the error bars represent the standard error of the number of time steps needed to reach a goal state over the simulations. The zoomed-in section highlights the standard errors in the optimal policy for two values of F_{thresh}.

The first observation to be made is that, for values of the threshold smaller than or equal to 0.925, the optimal and heuristic policies perform comparably well on average, and both perform better than the single-protocol policy. Next, we can observe that, for $F_{thresh} = 0.95$,

all policies have comparable results. The reason behind this is that, at this point, only the protocol with $F = 0.975$ could be used. Therefore, all policies always choose the same action, and thus perform equally well.

A second observation is that, in general, the standard error is small enough to not be distinguishable in the figure. For this reason, we showed a close-up of the graph of the optimal policy, for $F_{thresh} = 0.525$ and $F_{thresh} = 0.55$. There, we can observe that the standard error is approximately 0.03. By looking at the graph and seeing that the two horizontal lines marking the standard error are usually almost overlapping, we can infer that, in this case, the error is sufficiently small for other values of F_{thresh}, too.

5 Conclusions and Future Work

In conclusion, we first determined an optimal policy for generating a number of links using multiple protocols. This policy specifies with which resource generation protocol to generate a link based on the current state of the memory. Then, we introduced a heuristic that we used as a baseline to compare our optimal policy against, alongside the single-protocol policy. By analyzing the results, we observed that, as the threshold decreases, the optimal policy usually performs much better than the single-protocol policy. A secondary observation we made is that the heuristic usually performs comparably well to the optimal policy; however, in some cases, it does not manage to reach a goal state within a very large time frame.

One limitation of our study is that it was not tested in practice. Therefore, we suggest that future research focuses on using the findings of this paper in practical applications. This way, any practical limitations that this study did not consider could be discovered. Alternatively, practical benefits that were not envisioned at the moment when this study was carried could be found. Additionally, the heuristic could be improved to avoid situations in which it does not reach a goal state.

6 Responsible Research

To begin with, this research does not rely on the usage of any data sets. Therefore, the privacy of the people who provided data or the validity of the data are not concerns in our case. Given this, in the context of responsible research, we are mainly concerned with the principles of honesty and reproducibility.

The meaning of honesty includes "reporting the research process accurately" and "refraining from presenting results more favourably or unfavourably than they actually are" [9, p. 13]. To ensure the honesty of our research, we aimed to report the process and the results as accurately as possible. First of all, we included error bars in both graphs in Section 4, which provide the necessary context for fully understanding the results. Second of all, we did not omit mentioning the cases in which the proposed heuristic performs poorly (see Subsection 4.3).

Transparency refers to, among other things, ensuring the clarity of "how results were achieved" [9, p. 13]. In this regard, we provide a link to the GitLab repository that contains the code used to produce the results in the *Data Availability* section. The code has proper and thorough documentation. Moreover, the high-level descriptions of the problem model and value iteration algorithm are presented in Section 3.

Lastly, it is important to mention that this research has a potential positive impact on the field of data privacy. As mentioned in Section 1, one application that uses entangled

links is a variation of Blind Quantum Computation [2]. This application allows a client to perform a computation on several quantum servers, without revealing the content or structure of the computation [2].

Data Availability

The GitLab repository that contains the code used to calculate the optimal policy and perform the experiments can be found at https://gitlab.ewi.tudelft.nl/cse3000/2023-2024-q4/Vardoyan_Davies/idraganescu-Adaptive-algorithm-for-resource-generation-in-a-q.

References

[1] 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*, 2024.

[2] J. F. Fitzsimons, "Private quantum computation: an introduction to blind quantum computing and related protocols," *npj Quantum Information*, vol. 3, no. 1, p. 23, 2017.

[3] Á. G. Iñesta, G. Vardoyan, L. Scavuzzo, and S. Wehner, "Optimal entanglement distribution policies in homogeneous repeater chains with cutoffs," *npj Quantum Information*, vol. 9, no. 1, p. 46, 2023.

[4] L. Chirolli and G. Burkard, "Decoherence in solid-state qubits," *Advances in Physics*, vol. 57, no. 3, pp. 225–285, 2008.

[5] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[6] M. L. Puterman, "Markov decision processes," *Handbooks in operations research and management science*, vol. 2, pp. 331–434, 1990.

[7] "Markov decision process." en.wikipedia.org. Accessed: May 24, 2024. [Online.] Available: https://commons.wikimedia.org/wiki/File:Markov_Decision_Process.svg.

[8] C. Szepesvári, *Algorithms for reinforcement learning*. Springer nature, 2022.

[9] "Netherlands Code of Conduct for Research Integrity." `https://doi.org/10.17026/dans-2cj-nvwu`, 2018.