Optimal Entanglement Distillation Policies for Bipartite Quantum Switches

by

Vivek Kumar

Bachelors in Technology, Manipal Institute of Technology, 2019

Masters of Science in Information Science, University of Pittsburgh, 2023

Submitted to the Graduate Faculty of the

School of Computing and Information in partial fulfillment

of the requirements for the degree of

Master of Science in Information Science

University of Pittsburgh

2023

UNIVERSITY OF PITTSBURGH

School of Computing and Information

This thesis was presented

by

Vivek Kumar

It was defended on

November 28, 2023

and approved by

Kaushik P Seshadreesan, Assistant Professor, Department of Information and Networked Systems

David Tipper, Professor, Department of Information and Networked Systems

Alan Andrew Scheller-Wolf, Professor, Tepper School of Business, Carnegie Mellon University

Thesis Advisor: Kaushik P Seshadreesan, Assistant Professor, Department of Information and Networked Systems

Optimal Entanglement Distillation Policies for Bipartite Quantum Switches

Vivek Kumar, MSIS

University of Pittsburgh, 2023

In an entanglement distribution network, the function of a quantum switch is to generate elementary entanglement with its clients followed by entanglement swapping to distribute end-to-end entanglement of sufficiently high fidelity between clients. The threshold on entanglement fidelity is any quality-of-service requirement specified by the clients as dictated by the application they run on the network.

We consider a discrete-time model for a quantum switch that attempts generation of fresh elementary entanglement with two clients in each time step in the form of maximally entangled qubit pairs, or Bell pairs, which succeed probabilistically; the successfully generated Bell pairs are stored in noisy quantum memories until they can be swapped. We focus on establishing the value of entanglement distillation of the stored Bell pairs prior to entanglement swapping in presence of their inevitable aging, i.e., decoherence: For a simple instance of a switch with two clients, exponential decay of entanglement fidelity, and a well-known probabilistic but heralded two-to-one distillation protocol, given a threshold end-to-end entanglement fidelity, we employ Markov Decision Process and Reinforcement Learning to find optimal policies.

With these combined methodologies, our goal is to pinpoint the optimal action policy—whether it's waiting, distilling, or swapping—that can effectively maximize throughput. We compare the switch's performance under the optimal distillation-enabled policy with that excluding distillation.

Whereas the MDP framework helps model the problem in a coarse-grained form (discrete observation state space), and solving the MDP provides exact optimal policies, the Reinforcement Learning-based approach allows us to consider a more fine-grained model (continuous observation state space) and provides approximate but improved optimal policies. Simulations of the two policies demonstrate the improvements that are possible in principle via optimal use of distillation with respect to average throughput, average fidelity, and jitter of end-to-end entanglement, as functions of fidelity threshold. Our work thus helps capture the role of entanglement distillation in mitigating the effects of decoherence in a quantum switch in an entanglement distribution network, adding to the growing literature on quantum switches.

Table of Contents

List of Tables

List of Figures

Preface

I am thankful for the guidance of my advisor Kaushik Seshadreesan and the support from Alan Scheller-Wolf and Sridhar Tayur in the completion of this research. I would also like to acknowledge my co-author Nitish Kumar Chandra for his contributions.

This research was partly supported by the University of Pittsburgh Center for Research Computing, RRID:SCR_022735, and the HTC cluster supported by NIH award number S10OD028483.

This research was funded by NSF Award 2134891 project QuanNeCQT under the Convergence Accelerator program (sub-award from University of Maryland).

Finally, my gratitude goes to my parents and family for their unwavering support.

1.0 Introduction

The prospective quantum internet [1] represents a global-scale network of interconnected quantum computers, sensors, and devices, marking a significant development in the ongoing second quantum revolution [2]. It will coexist with the current classical internet, primarily enhancing security through quantum cryptography [3] methods like quantum key distribution [4]. Additionally, it will support distributed quantum information processing [5], [6], and secure delegated quantum computation [7], [8] in the cloud, addressing limitations of the existing classical internet.

One of the main challenges in implementing entanglement-based quantum networks within the quantum internet is establishing quantum entanglement reliably across long distances at high speeds. This difficulty arises because quantum signals degrade exponentially with transmission distance [9] – [11]. Quantum repeaters [12] – [14] play a crucial role in addressing this challenge by amplifying quantum signals using quantum sources, memories, detectors, and logic, enabling the distribution of higher-quality entanglement at improved rates.

In addition to quantum repeaters, quantum switches play a vital role in the advancement of quantum networks. A quantum switch essentially functions as a quantum repeater, enhancing the rate of dependable quantum transmissions passing through it. Moreover, it possesses the capability to manage quantum communications among multiple users, allowing the redirection of quantum transmission flows within the network. This functionality enables the establishment of large-scale quantum networks with diverse topologies [15].

It is important to note that the measurements involved in entanglement swapping are generally probabilistic in nature. When successful, these measurements result in the establishment of end-to-end entanglement between the clients. The decisions made by the quantum switch are guided by a switching policy designed to optimize specific metrics, while also considering the quality-of-service requirements specified by its clients, such as a minimum threshold fidelity for end-to-end entanglement.

Various models and policies for quantum switches in entanglement distribution networks have been explored. These models include switches serving bipartite entanglement [16], [17] through Bell-state measurements, switches facilitating multipartite entanglement using Greenberger-Horne-Zeilinger (GHZ) basis measurements [18] on link-level entanglement [19], and hybrid switches that handle both scenarios [20]. Some studies consider the impact of decoherence on link-level entanglement, incorporating finite lifetime quantum memories [16], [17].

While certain models focus on static metrics like sum throughput, independent of client demands [5], [16], [17], [20], others take into account client requests for various types of end-to-end entanglement. They aim to optimize request service rates by prioritizing requests to prevent backlogs from growing indefinitely [21], [22]. Recent research also delves into a quantum switch model for continuous variable quantum encodings, considering request rates, queue stability, and the polarities of elementary entanglement to enhance end-to-end entanglement rates [23] – [25].

In a recent study, a switch model was examined where multiple noisy link-level Bell pairs are generated per client and last only one time step. It explored the use of nested two-to-one probabilistic but heralded entanglement distillation [26], [27] to surpass a fidelity threshold on end-to-end entanglement [28]. The study compared the performance of distilling elementary entanglement followed by entanglement swap with entanglement swap followed by end-to-end entanglement distillation.

This work focuses on a simplified switch model in an entanglement distribution network with two clients and bipartite qubit entanglement distribution. The primary objective is to evaluate the impact of distillation in mitigating decoherence effects. Link level entanglement with clients are modeled as "Bell pairs", which refer to two-qubit maximally entangled states that are retained over multiple time steps, potentially experiencing decoherence as they await pairing with link-level counterparts. During this waiting period, entanglement distillation between noisy Bell pairs across each link is considered to maintain link quality.

Two approaches have been explored, a Markov Decision Process (MDP) framework within a discrete-time model and a Reinforcement Learning based approach to account for continuous state spaces. Both techniques account for probabilistic creation of link-level entanglement, finite buffers for links that limits the number of Bell pairs that can be stored along a link, and the option for two-to-one entanglement distillation between Bell pairs on the same link to enhance fidelity. Since this operation carries a risk (both links can be lost with non-zero probability), the techniques determine the optimal action policy to balance benefits and risks. Numerical simulations are used to compare the performance of this switch with one that does not allow distillation, assessing average throughput and jitter of end-to-end entanglement generation concerning a target entanglement fidelity threshold.

All code has been written in python and the libraries used are mentioned in subsequent sections according to when they were utilized.

1.1 Thesis Outline

This document is organized as follows. To begin with, Chapter 2 delves into several key concepts that serve as the foundation for the later parts of the thesis. This includes concepts like entanglement, distillation, quantum repeaters, and quantum switches. We also provide a thorough examination of the fundamental principles underpinning quantum networking. This encompasses topics such as qubit state noise, fidelity, entanglement distillation, entanglement swapping, and quantum memory. Finally, we discuss the concepts that will allow us to formulate the switch model as a Markov Decision Process (MDP) and Reinforcement Learning (RL) problem.

The rest of the thesis describes finding and analyzing optimal policies that maximize the throughput of quantum switch involving two users and serving bipartite end-to-end entanglement by taking one of three available actions: wait, entanglement swap, entanglement distillation.

Chapter 3 defines the model of our switch for both the MDP and RL framework. It is essential to note that even though both frameworks solve the same switch model, the observation state spaces are different. Solving MDPs are a computationally intractable problem for large state spaces and hence we use it to solve states which are elementally discrete. RL lends itself to more conveniently solve continuous state spaces. This change in observation spaces makes it pertinent to modify the actions using different implementation strategies while programming.

In Chapter 4, we move forward with solving both our MDP and RL formulations. Our work used both approaches of solving MDPs, i.e., value iteration and policy iteration. The results showed that policy iteration proved to be a much more efficient method. The result is a csv file detailing the action that should be taken for each possible state. The inability of MDPs to solve large state spaces entails assumptions to discretize the states. This prevents the solution and framework to be representative of reality. We also explain our attempt to involve Reinforcement Learning (RL) so as to mitigate this issue. RL brings with itself certain problems for complex models that need to be addressed, namely presence of invalid actions and sparse rewards. These issues and their resolutions are discussed in length in this chapter.

The results and conclusions detailed in Chapter 5 and 6 provide evidence that including distillation as a prospective action for switches indeed helps in increasing the throughput for both the MDP and RL model. It's relation to jitter and average fidelity achieved over multiple fidelity thresholds is also discussed for the MDP model.

2.0 General Concepts

In this chapter, we will introduce some of the common concepts used throughout the thesis. These concepts will be quite general but will be put into a more specific context in the subsequent chapters. There will not be given an introduction to the fundamentals of quantum information theory and reinforcement learning and, if necessary, the reader is referred to books by Isaac Chuang and Michael Nielsen [29], Mark M. Wilde [30] and Richard S. Sutton [31].

2.1 Quantum States

Quantum states are the foundational elements in quantum mechanics, capturing the behavior of particles and systems at the quantum level. They introduce inherent uncertainty and superposition, giving rise to unique phenomena. Additionally, concepts like entanglement, noise, and fidelity play crucial roles in defining and understanding the properties and behavior of quantum states.

2.1.1 Qubits

The qubit, a quantum counterpart to the classical bit used for binary information representation, can be described mathematically as a two-dimensional complex vector space, where $|0\rangle$ and $|1\rangle$ vectors form the computational basis. These two states correspond to orthogonal states, such as "spin up" and "spin down".

In quantum computation, the foundation lies in applying quantum logic to transform the joint quantum state of multiple qubits. Universal sets of quantum logic gates, comprise single and two-qubit gates, which can be employed to construct any quantum computation [29]. A prime example of such a set includes single-qubit rotations and the CNOT gate. The CNOT gate, a two-qubit gate, facilitates the transformation.

$$\begin{aligned} &|0\rangle_a|0\rangle_b \rightarrow |0\rangle_a|0\rangle_b, \quad |0\rangle_a|1\rangle_b \rightarrow |0\rangle_a|1\rangle_b, \\ &|1\rangle_a|0\rangle_b \rightarrow |1\rangle_a|1\rangle_b, \quad |1\rangle_a|1\rangle_b \rightarrow |1\rangle_a|0\rangle_b, \end{aligned} \tag{2.1}$$

where the qubit with subscript a is referred to as the control qubit and the other is called the target qubit.

The key distinction between a classical bit and a qubit lies in the nature of the physical qubit, governed by the principles of quantum mechanics. Consequently, a qubit doesn't always follow a definite path of either being in state $|0\rangle$ or $|1\rangle$; instead, it can exist in a superposition of these states. A general qubit state, $|\psi\rangle$, can be expressed as:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle. \tag{2.2}$$

Here, $|\alpha|^2$ (or $|\beta|^2$) represents the probability of measuring the qubit in state $|0\rangle|0\rangle$ (or $|1\rangle|1\rangle$), and $|\alpha|^2 + |\beta|^2 = 1$. The capacity to exist as a superposition of two orthogonal states is a pivotal advantage of employing quantum systems for information processing.

2.1.2 Entanglement

Entanglement extends the superposition principle to encompass multiple qubits. When the joint state of two qubits exists in an equal superposition between two orthogonal states, it is termed a maximally entangled state. This state can be expressed as:

$$|\psi^{+}\rangle = \frac{1}{\sqrt{2}}(|0\rangle_a|1\rangle_b + |1\rangle_a|0\rangle_b), \tag{2.3}$$

where $|0\rangle_a$ ($|0\rangle_b$) refers to qubit a (b) being in state 0. Now, if we consider a continuous supply of these states and perform repeated measurements on the two qubits, we find an interesting behavior. Both qubits have an equal chance (50-50) of being in either state $|0\rangle$ or $|1\rangle$ after measurement. However, there is the strong anti-correlation between the states of the two qubits. That is, whenever qubit a is measured in state $|0\rangle$, qubit b is simultaneously found in state $|1\rangle$, and vice versa. Moreover, remarkably, the anti-correlation persists even if qubits a and b are measured in any basis, as long as they both are measured in same basis. It is important to emphasize that the anti - correlation remains consistent, regardless of the physical distance separating the two qubits. In a somewhat intriguing manner, the two qubits jointly decide to assume opposite states upon measurement, as their combined state exists in superposition before measurement. This type of correlation lacks a classical equivalent and is the foundation for phenomena like quantum teleportation [32]. The state described in Equation (2.3) is just one of the four possible maximally entangled two-qubit states, commonly referred to as such, with the other three states being

$$|\psi^{-}\rangle = \frac{1}{\sqrt{2}}(|0\rangle_a|1\rangle_b - |1\rangle_a|0\rangle_b), \tag{2.4}$$

$$|\phi^{\pm}\rangle = \frac{1}{\sqrt{2}}(|0\rangle_a|0\rangle_b \pm |1\rangle_a|1\rangle_b). \tag{2.5}$$

Note, that entanglement can in general exist for an arbitrary number of qubits, e.g., the GHZ states, $\frac{1}{\sqrt{2}}(|0\rangle|0\rangle \dots |0\rangle + |1\rangle|1\rangle \dots |1\rangle)$ are examples of multi-qubit, entangled states.

2.1.3 The Density Operator

The density operator, also referred to as the density matrix (when represented as a matrix) is a fundamental concept in quantum mechanics, particularly important in the contexts of statistical mixtures of states.

As we have seen in the previous sections, in quantum mechanics, the state of a system is typically described by a wave function, which is a solution to the Schrödinger equation. This wave function ψ is described in Equation (2.2) as a general qubit state and it gives complete information about the system's state. However, in many cases, especially in quantum statistical mechanics and quantum information theory, we might not have complete information about the system's state or the system might be a mixture of several states. In such cases, the density operator becomes a useful tool.

This leads us to define a pure state and a mixed state before explaining density operator in detail.

A pure state represents a quantum system in a specific, well-defined state, $|\psi\rangle$. This state provides complete information about the system's quantum state. An example of a pure state is an electron in a hydrogen atom occupying a specific energy level. The key characteristic of a pure state is that it exhibits coherent superposition as seen in Equation (2.2), meaning the state can be a combination of several basis states with specific probability amplitudes.

On the other hand, a mixed state represents a statistical ensemble of several different states, each with a certain probability. Unlike a pure state, a mixed state does not have a single wave function describing it. Instead, it is characterized by a set of states and their respective probabilities. A mixed state arises in situations where there is uncertainty or lack of information about which particular pure state the system is in. This can happen, for example, due to noise. E.G., decoherence, where a quantum system interacts with its environment, leading to a loss of coherent superposition.

With these concepts in mind, we can define the density operator (or density matrix) in quantum mechanics, which is a more general way to describe the state of a quantum system, encompassing both pure and mixed states. The density operator, denoted as ρ, is particularly useful because it can handle situations where a system is in a superposition of states (pure state) or when there is uncertainty about its state (mixed state).

For a pure state $|\psi\rangle$, the density operator is defined as:

$$\rho = |\psi\rangle|\langle\psi|, \tag{2.6}$$

which is the outer product of the state vector with its conjugate transpose. This representation captures the complete information of the pure state.

In the case of a mixed state, where the system is in a state $|\psi_i\rangle$ with probability p_i, the density operator is a weighted sum of the outer products of each state:

$$\rho = \sum_i p_i |\psi_i\rangle\langle\psi_i|, \tag{2.7}$$

with the condition that $\sum_i p_i = 1$, to ensure that the total probability is conserved. The density operator provides a unified framework to describe both pure and mixed states.

2.1.4 Fidelity

Fidelity is a crucial concept in quantum information theory, used to quantify how close or similar two quantum states are to each other. This measure is especially important in the context of quantum computing, quantum communication, and other quantum technologies, where maintaining the integrity of quantum states is paramount.

For pure states, fidelity between two states $\rho = |\psi\rangle\langle\psi|$ and $\sigma = |\phi\rangle\langle\phi|$ can be defined by the following equation:

$$F(\rho,\sigma) = |\langle\psi||\phi\rangle|^2. \tag{2.8}$$

For mixed states, the fidelity is defined in a more complex way, involving the square root of the density matrices:

$$F(\rho,\sigma) = \left(Tr\sqrt{\sqrt{\rho}\sigma\sqrt{\rho}}\right)^2. \tag{2.9}$$

If one of the states is pure and the other is mixed, the fidelity simplifies to:

$$F(\rho,\sigma) = \langle\psi|\sigma|\psi\rangle. \tag{2.10}$$

2.1.5 Werner States

Werner states are a specific class of quantum states in quantum mechanics, particularly relevant in the study of quantum entanglement. They were introduced by Reinhard Werner in 1989 to explore the distinctions between quantum entanglement and classical correlations. Werner states are important for understanding fundamental aspects of quantum mechanics, including nonlocality, separability, and quantum information theory.

Werner states are defined for bipartite systems, meaning systems that consist of two parts, typically referred to as part A and part B. These states are especially significant in the context of two-qubit systems, where each qubit is a two-level quantum system (akin to a quantum version of a bit).

The general form of a Werner state ρ_{AB} for a two-qubit system can be written as:

$$\rho_{AB} = F|\phi^+\rangle\langle\phi^+|_{AB} + \frac{1-F}{3}(|\phi^-\rangle\langle\phi^-|_{AB} + |\psi^+\rangle\langle\psi^+|_{AB} + |\psi^-\rangle\langle\psi^-|_{AB}), \tag{2.11}$$

where $F = Tr(\rho_{AB}|\phi^+\rangle\langle\phi^+|_{AB})$, $1 \geq F \geq 0$, captures the fidelity of the Werner state with the target Bell state $|\phi^+\rangle$. The states $|\phi^+\rangle, |\phi^-\rangle, |\psi^+\rangle$ and $|\psi^-\rangle$ states are given by Equations (2.3), (2.4) and (2.5).

2.1.6 Decoherence

Decoherence is the process by which a quantum system loses its quantum coherence. In quantum mechanics, coherence refers to the property of quantum superposition, where a quantum system can be in multiple states simultaneously. Decoherence is typically caused by the interaction of the quantum system with its environment.

In our work, we model decoherence by dropping the fidelity of entanglements exponentially with time, i.e.,

.

$$F(m) = e^{-\alpha m}. \tag{2.12}$$

Here α is a decoherence parameter, and governs the resultant fidelity due to decoherence after m discrete timesteps.

2.2 Quantum Repeaters

The concept of entanglement plays a pivotal role in quantum communication. For example, entanglement can be used to transfer quantum states from point A to B via the quantum teleportation protocol [29],[30]. Entanglement also enables quantum key distribution (QKD) protocols between distant entities, commonly referred to as Alice and Bob, as noted in references [33],[34]. Entanglement-based QKD involves repeatedly distributing halves of entangled pairs to Alice and Bob, resulting in each party amassing a large number of qubits. Through strategically measuring these qubits in various bases and communicating their measurement strategies, Alice and Bob can derive a confidential key for encryption and decryption. This technique's efficacy lies in the sensitivity of the entangled qubits' correlations to interference. Should an eavesdropper attempt to glean information by measuring the qubits in transit, this disturbance would be detectable. Alice and Bob can safeguard against such intrusions by analyzing a select portion of their qubits, thus gauging the extent of information potentially accessed by an eavesdropper. This approach underpins the creation of exceptionally secure communication channels, as highlighted in [34].

Extending entanglement over substantial distances is key to realizing quantum communication over large distances, however this presents considerable challenges due to quantum information's inherent vulnerability to noise. To establish entanglement between two remote parties, a quantum signal, such as a single photon in a specific quantum state, is typically transmitted. However, the likelihood of photon loss grows exponentially with the distance if no intervening signal processing mechanisms are in place [35]. This issue leads to a steep decrease in the rate of entanglement distribution as distance increases. In classical communication, repeater stations are employed to amplify and clarify signals for further transmission. Quantum communication adopts a similar architecture, but with a critical distinction: amplifying a quantum signal would entail interaction, introducing noise that deteriorates the quantum information [36], [37]. To overcome this, quantum repeaters utilize shorter distance-scale link-level entanglement generation and the phenomenon of entanglement swapping. In entanglement swapping, entanglement is teleported from one station to the next, effectively bypassing direct signal loss [35], [38].

2.2.1 Entanglement Swapping

To begin with, quantum repeaters break down the total distance into smaller segments known as elementary links. Within these links, entanglement is generated via the direct transmission of a quantum signal (as depicted in Figure 2-1). The process of establishing entanglement usually involves flying qubits and is often probabilistic in nature. To accommodate this, quantum memories are utilized to store the quantum information in stationary qubits until entanglement is successfully achieved in adjacent links.

Once entanglement is established in two neighboring links, it is then swapped by performing a Bell state measurement on one qubit from each pair (refer Fig. 2.1). A Bell state measurement projects a two-qubit state onto one of the four Bell states. This task requires some ingenuity, as a simple measurement of individual qubit states is insufficient and will not differentiate between $|\psi^{\pm}\rangle$ and $|\phi^{\pm}\rangle$. A CNOT gate followed by single qubit measurements can be used for this purpose.

To understand this mathematically, consider two adjacent links containing Bell states $|\psi^{+}\rangle_{ab}$ and $|\psi^{-}\rangle_{ab}$, with qubits a and b in the first link, and c and d in the second. The combined state of both links can be expressed as:

$$|\psi^{+}\rangle_{ab}|\psi^{+}\rangle_{cd} = \frac{1}{2}(|\psi^{+}\rangle_{bc}|\psi^{+}\rangle_{ad} - |\psi^{-}\rangle_{bc}|\psi^{-}\rangle_{ad} + |\phi^{+}\rangle_{bc}|\phi^{+}\rangle_{ad} - |\phi^{-}\rangle_{bc}|\phi^{-}\rangle_{ad}). \quad (2.13)$$

Therefore, by conducting a Bell state measurement on qubits b and c and relaying the outcome to the locations of qubits a and d, we can implement specific single qubit rotations to achieve the targeted Bell state between qubits a and d. Essentially, entanglement is transferred to qubits a and d through a local Bell measurement on b and c, the transmission of a classical signal (the measurement result), and local single qubit rotations. This process eliminates the need for a quantum signal to be sent between qubits a and d. This method is repeated until entanglement is successfully established across the chain of repeaters that span the entire distance.

For Werner States described in Section 2.1.5, with fidelities F_1 and F_2 with respect to $|\phi^2\rangle$, the resultant fidelity upon entanglement swapping is given by:

$$F_{12} = \frac{\frac{(4F_1 - 1)(4F_2 - 1)}{3} + 1}{4}. \tag{2.14}$$

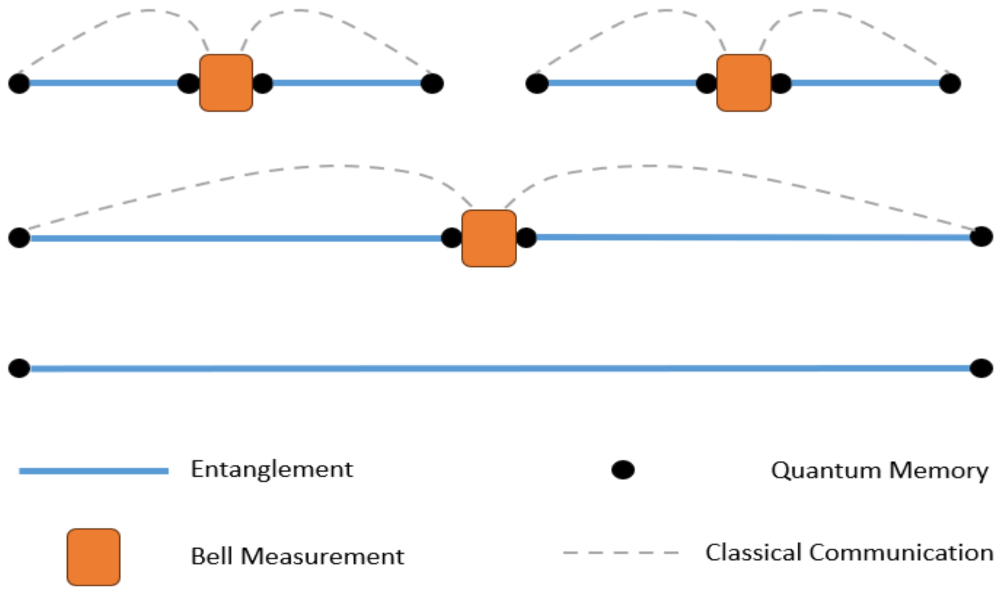


Figure 2-1: Entanglement Swap. The entire distance is broken down into smaller sections. In these sections, entanglement is achieved via the direct transfer of a quantum signal. By executing Bell measurements, this entanglement is transferred over increased distances. The measurements' results are then sent to the final stations, where they adjust with specific single qubit rotations. It's crucial to acknowledge that this repeater system relies on quantum memories, which are used to store and handle these signals.

2.3 Quantum Switch

A quantum switch is a device in quantum networks that efficiently routes quantum signals and establishes end-to-end entanglement among users as per their requirements. This capability enables more precise and user-specific quantum communication pathways within the network. In our work, the quantum switch also incorporates the process of entanglement distillation, enhancing its functionality.

Specifically, we employ the Bennett, two-to-one entanglement distillation protocol within the switch. This protocol involves using two pairs of less entangled qubits to produce a single pair with higher entanglement. The process is akin to filtering out quantum noise, improving the quality of entanglement at the expense of reducing the number of entangled pairs. The integration of this process makes the quantum switch not just a router of quantum information but also an enhancer of its quality.

2.3.1 Entanglement Distillation: Bennett Protocol

Entanglement distillation is a technique used to convert a number of low-quality, noisy entangled states into a smaller number of entangled states with higher quality. Specifically, for a pair of Werner states, ρ_{A1B1} and ρ_{A2B2}, each characterized by fidelities F_1 and F_2. Bennett et al. [39] proposed a notable protocol for entanglement distillation. This method is recognized for its probabilistic nature and the clear indication (heralding) of its success or failure.

The protocol operates by executing CNOT operations, refer Figure 2-2. A1 and B1 act as control qubits, and A2 and B2 serve as target qubits. Following this, A2 and B2 are measured in the computational basis. The process is deemed successful if the measurements of A2 and B2 are the same, and it fails if they differ. The likelihood of the protocol's success is calculated by the formula:

$$p_{succ} = \frac{8}{9}F_1F_2 - \frac{2}{9}(F_1 + F_2) + \frac{5}{9}. \tag{2.15}$$

If the protocol is successful, it yields a Werner state with an improved fidelity described by:

$$F_{12} = \frac{1}{p_{succ}}\left(\frac{10}{9}F_1F_2 - \frac{1}{9}(F_1 + F_2) + \frac{1}{9}\right). \quad (2.16)$$

Through entanglement distillation, it is possible to achieve entanglement fidelities in a quantum network that surpass specific quality of service benchmarks, albeit at the cost of utilizing numerous link-level entanglements of typically lower fidelity. Additionally, the probabilistic nature of such protocols introduces an element of unpredictability, moving away from deterministic approaches.

Figure 2-3 shows the resultant fidelity values on success of distillation using Bennett protocol for individual fidelities between 0.7 and 1.

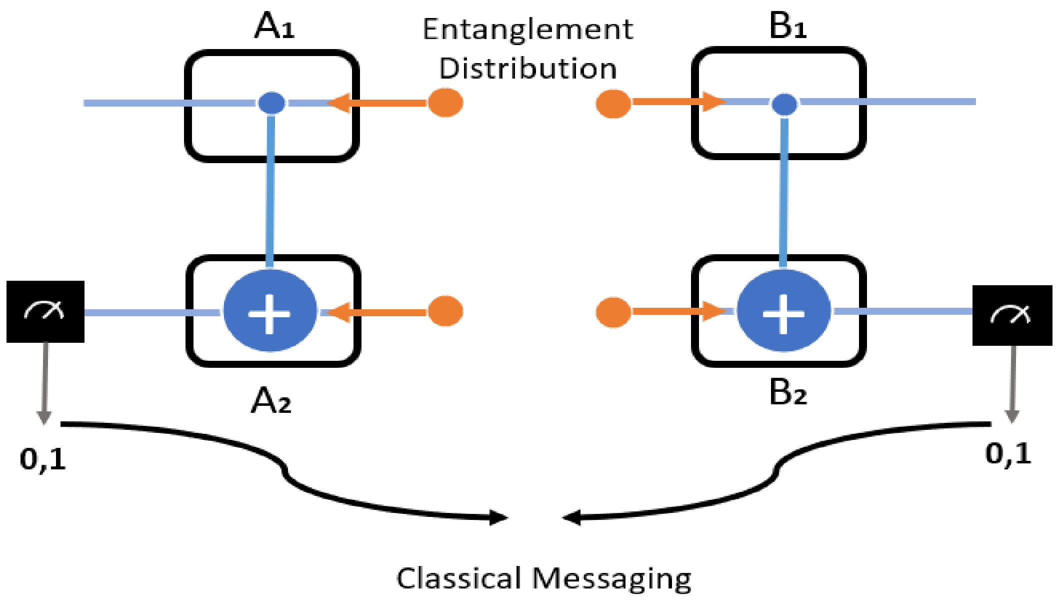
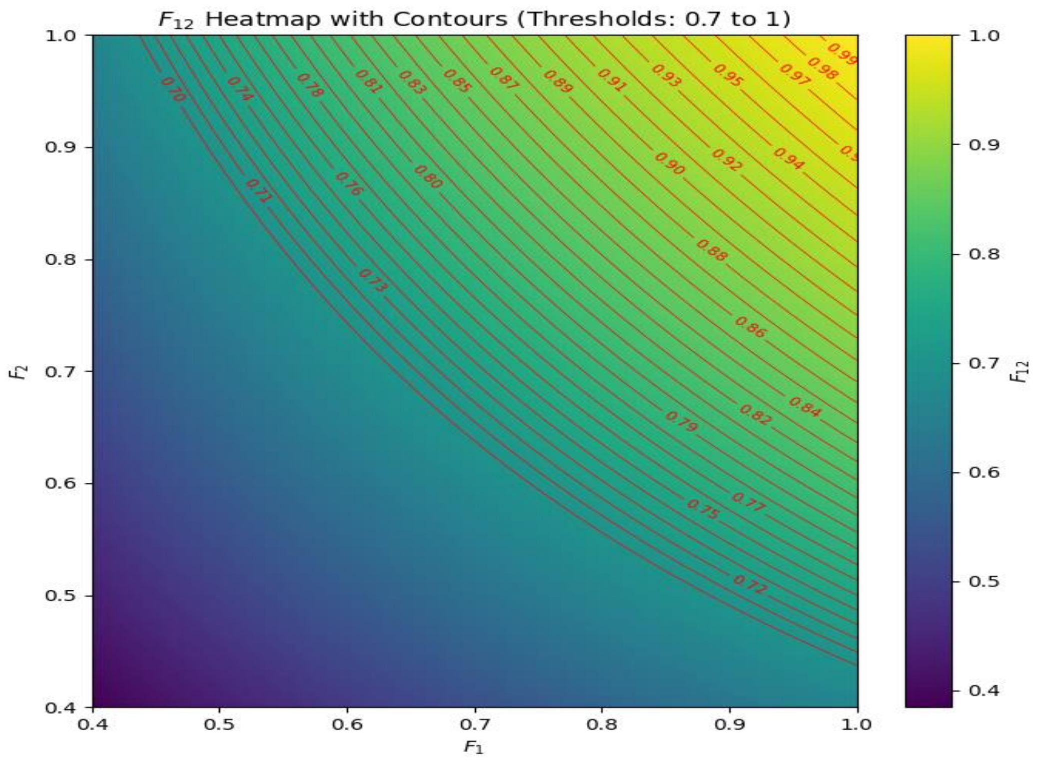


Figure 2-2: Bennett Protocol For Distillation.



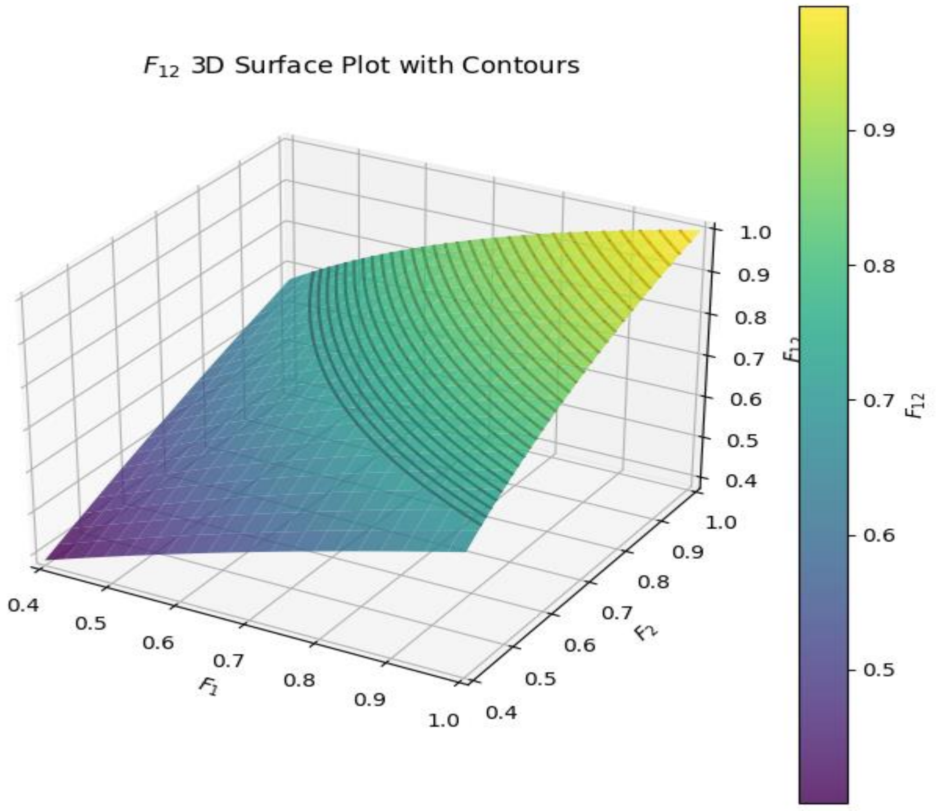


Figure 2-3: Graphs showing resultant $\boldsymbol{F_{12}}$ after distillation

2.4 Markov Decision Process

Markov Decision Processes (MDPs) offer a structured approach for modeling decision-making in environments where outcomes are influenced by both random factors and the actions of a decision-maker. It is an iterative process where an agent observes an environment in state s_t, takes an action a_t, and receives a reward r_{t+1}, refer Figure 2-4. The action changes the state from s_t to s_{t+1}. Central to the concept of MDPs are several key components: states, actions, transition functions, reward functions, discount factors, and policies. In the context of near-term quantum networking, an MDP was recently employed to determine policies for optimal elementary entanglement generation [40]

States in an MDP represent the various conditions or configurations in which the system can exist, encapsulating all relevant information for decision-making. Actions are the choices or maneuvers that can be taken in each state, leading to transitions between states. The transition function, typically denoted as $T(s, a, s')$, quantifies the probability of moving from one state to another given a specific action, capturing the stochastic nature of the environment.

The reward function, expressed as $R(s, a, s')$, assigns a numerical value to each transition, providing immediate feedback on the desirability of an action taken in a particular state. This function is fundamental, as it directs the decision-making process towards favorable outcomes by quantifying the immediate success or failure of actions.

The discount factor, represented by γ, is a value between 0 and 1 that balances the importance of immediate versus future rewards. It is a crucial parameter in computing the expected return, ensuring the convergence of the sum of future rewards. A higher γ places greater emphasis on future rewards, influencing the long-term strategy of the decision-making process.

Policies in an MDP, denoted as $\pi(a|s)$, dictate the behavior of an agent by specifying the probability of choosing an action in a given state. The aim is to identify an optimal policy that maximizes the cumulative reward.

The Value Function, $V^{\pi}(s)$, is a key concept that reflects the expected total reward that an agent can accumulate over time, starting from a state s and following a policy π. It is defined as:

$$V^{\pi}(s) = \mathbb{E}_{\pi}[\textstyle\sum_{k=0}^{\infty} \gamma^{k} R_{t+k+1} \mid S_t = s]. \tag{2.17}$$

Here the expectation in case of stochastic processes is calculated by using the transition probability function wherein now,

$$R_{t+k+1} = \textstyle\sum_{s'} T(s, \pi(s), s') \times R_{t+k+1}(s, \pi(s), s').$$

Building on this, the Q-value Function, $Q^{\pi}(s, a)$, considers both the state and the action, representing the expected return of taking an action a in state s and then following the policy π. It is formulated as:

$$Q^{\pi}(s, a) = \mathbb{E}_{\pi}[\textstyle\sum_{k=0}^{\infty} \gamma^{k} R_{t+k+1} \mid S_t = s,\ A_t = a]. \tag{2.18}$$

Lastly, the Advantage Function, $A^{\pi}(s, a)$, measures the relative benefit of a specific action over the average action at a state under the given policy. It is defined as:

$$A^{\pi}(s, a) = Q^{\pi}(s, a) - V^{\pi}(s). \tag{2.19}$$

These elements collectively establish the framework of MDPs, enabling the analysis and formulation of optimal decision-making strategies in uncertain and dynamic environments.

We can use Value Iteration or Policy Iteration to solve an MDP formulation to obtain a final optimal policy. In our work we have chosen policy iteration to solve our model since it gave the best and fastest results after numerous trial and error iterations.

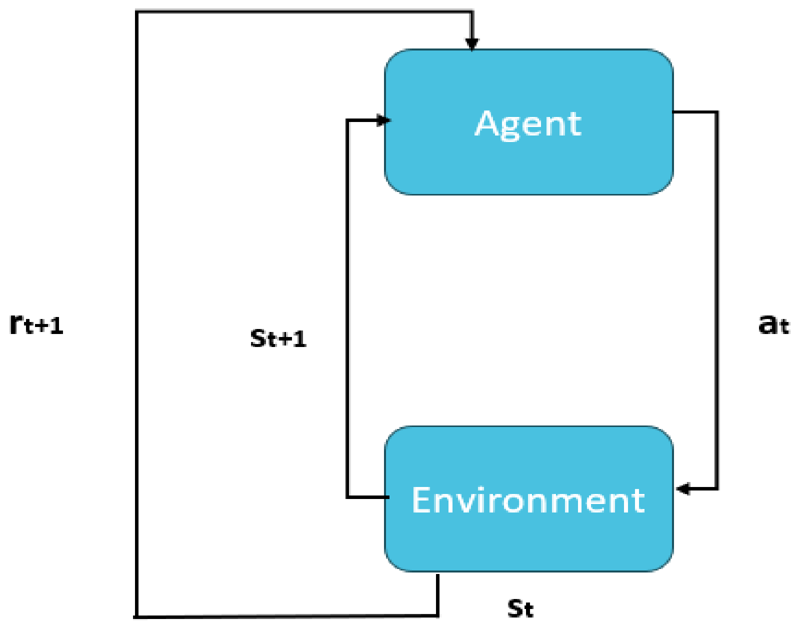


Figure 2-4: Iterative process defining Markov Decision Processes.

2.4.1 Value Iteration

Value iteration is one of the two primary algorithms used in Markov Decision Processes (MDPs) to find an optimal policy. It leverages the framework of states, actions, transition probabilities, reward functions, and the discount factor mentioned in the previous section to iteratively improve the estimation of the optimal policy.

The core idea of value iteration is to repeatedly update the value of each state under the current policy until these values converge to a stable set, indicative of an optimal policy. The algorithm proceeds using following steps:

Step 1

Initialization: Start by initializing the value of all states, typically to zero. This is the initial value function, $V_0(s)$ for all states s.

Step 2

Iteration: At each iteration k, update the value of each state based on the expected return of taking each possible action from that state and then following the current policy. This is done using the Bellman optimality equation, which provides a recursive way to update the values:

$$V_{k+1}(s) = \max_{a \epsilon A} \sum_{s' \epsilon S} T(s,a,s')[R(s,a,s') + \gamma V_k(s')]. \tag{2.20}$$

Here, V_{k+1} is the updated value of state s at iteration $k+1$, $\max\limits_{a \epsilon A}$ is the maximum value over all possible actions a in the action set A.

Step 3

Convergence Check: The process is repeated until the change in value function between iterations is below a predetermined threshold, indicating convergence.

Step 4

Policy Extraction: Once the value function has converged, the optimal policy $\pi^*(a|s)$ can be extracted. For each state s, the optimal action a^* is chosen as the one that maximizes the expected return:

$$a^* = \arg\max_{a \in A} \sum_{s' \epsilon S} T(s,a,s')[R(s,a,s') + \gamma V^*(s')]. \tag{2.21}$$

If a = a* for all states then, $\pi^*(a|s)$ is called the optimal policy. Here, $V^*(s')$ is the final converged value of state s'.

Value iteration thus systematically improves the estimation of the value function, eventually converging to the optimal values that reflect the maximum expected return from each state. This forms the basis for determining the optimal policy in an MDP, effectively solving the decision-making problem.

2.4.2 Policy Iteration

Policy iteration is another method used in Markov Decision Processes (MDPs) for finding an optimal policy. Unlike value iteration, which focuses on the values of states, policy iteration alternates between policy evaluation and policy improvement steps to converge on an optimal policy.

The algorithm involves the following steps:

Step 1

Initialization: Begin with an arbitrary policy π_0, which is a mapping from states to actions.

Step 2

Policy Evaluation (Critic): For the current policy π_k, calculate the value function $V^{\pi_k}(s)$ for each state s. This is done by solving the set of linear equations given by the Bellman expectation equation for each state:

$$V^{\pi_k}(s) = \sum_{s' \epsilon S} T(s, \pi_k(s), s')[R(s, \pi_k(s), s') + \gamma V^{\pi_k}(s')]. \tag{2.22}$$

Here, $T(s, \pi_k(s), s')$ is the probability of transitioning from state s to state s' under action $\pi_k(s)$, $R(s, \pi_k(s), s')$ is the reward for this transition.

Step 3

Policy Improvement (Actor): Once the value function for the current policy is known, improve the policy by choosing actions that maximize the expected utility for each state. The new policy π_{k+1} is obtained by:

$$\pi_{k+1}(s) = \underset{a \epsilon A}{\arg\max} \sum_{s' \epsilon S} T(s, a, s')[R(s, a, s') + \gamma V^{\pi_k}(s')]. \tag{2.23}$$

This step updates the policy by making it greedy with respect to the current value function.

Step 4

Convergence Check: The algorithm repeats the policy evaluation and improvement steps until the policy remains unchanged after an improvement step, indicating convergence to the optimal policy.

Policy iteration is characterized by its two-phase approach: first evaluating how good the current policy is (policy evaluation), and then using this information to find a better policy (policy improvement). The process ensures that the policy is continually updated to be more optimal, converging eventually to an optimal policy that maximizes the expected return from each state in the MDP. This method is particularly useful when the policy space is smaller or more structured, making the process of policy improvement more straightforward compared to the value space in value iteration. Figure 2-5 provides a pictorial explanation of this iterative process.

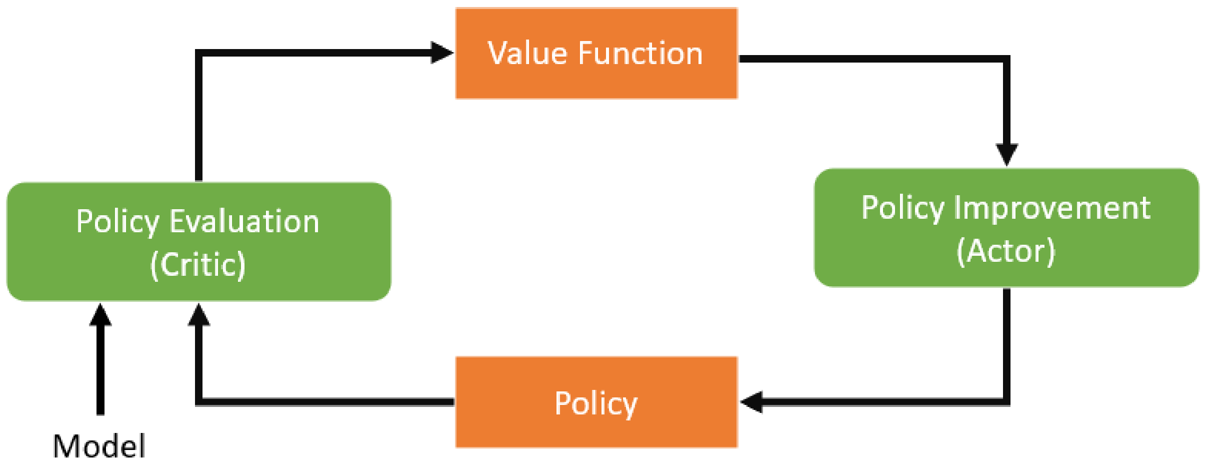


Figure 2-5: Flowchart describing Policy Iteration Algorithm

2.5 Reinforcement Learning

Reinforcement Learning (RL) is a pivotal area in the field of machine learning and artificial intelligence, concerned with the development of algorithms and methodologies that enable an agent to learn optimal behaviors through interactions with its environment. Characterized by a trial-and-error approach, RL focuses on the agent's ability to make sequential decisions, aiming to maximize some notion of cumulative reward over time.

Central to RL is the concept of the agent-environment interaction. The agent, situated within an environment, is tasked with learning to perform actions that yield the most favorable outcomes, typically quantified in terms of rewards. The modelling of the problem is still formalized within the framework of Markov Decision Processes (MDPs), providing a structured representation of the decision-making problem. RL applies techniques to solve so that it is tractable for large state spaces. The agent's objective is to discover a policy—a mapping from states to actions—that maximizes the expected sum of discounted rewards, known as the return.

RL algorithms are generally categorized into three primary approaches: value-based, policy-based, and model-based methods. Value-based methods, such as Q-learning and Deep Q-Networks (DQNs), focus on estimating the value of actions in each state, guiding the agent towards high-reward strategies. Policy-based methods, exemplified by Policy Gradients and Proximal Policy Optimization (PPO), directly learn the policy that dictates the agent's actions. Model-based methods attempt to model the dynamics of the environment, using this model to plan and make decisions.

The learning process in RL can be either model-free, where the agent learns solely from the rewards received without any knowledge of the environment's dynamics, or model-based, where the agent constructs a model of the environment to guide its decision-making process. Model-free methods are typically simpler and more widely applicable, as they do not require explicit modeling of the environment, which can be complex or infeasible in many real-world scenarios.

Exploration and exploitation are two fundamental aspects of RL. Exploration involves the agent trying new actions to discover their effects, essential for acquiring new knowledge. Exploitation involves using the knowledge the agent has already acquired to make the best decisions. Balancing exploration and exploitation are a critical challenge in RL, as excessive exploration can lead to suboptimal immediate performance, while excessive exploitation can prevent the discovery of more rewarding actions.

2.5.1 Deep Reinforcement Learning

Deep Reinforcement Learning (DRL) extends traditional reinforcement learning by employing deep neural networks to approximate critical functions. These approximations are central to enabling DRL algorithms to handle complex, high-dimensional environments.

In DRL, the functions, denoted as $f(s, a, \theta)$ and parameterized by neural network weights θ, estimates the expected return of taking action a in state s. The goal is to learn a function that can predict the quality of each action given the current state. We will next look at some of the critical functions that can be approximated using DRL.

2.5.1.1 Value Function Approximation

The Value Function, denoted as $V(s; \theta)$, where θ represents the parameters of a neural network, is an estimate of the expected return from a given state s. In DRL, this function is approximated using deep learning techniques to handle complex state spaces that traditional methods may struggle with.

The learning process involves adjusting the parameters θ to minimize the difference between the predicted value $V(s;\theta)$ and the actual return. The return G_t, which is the cumulative discounted reward from state s, is a crucial component in this process. The loss function for value function approximation is typically formulated as a mean squared error loss:

$$L(\theta) = \mathbb{E}_{s\sim S}[\,(V(s;\theta) - G_t)^2\,]. \tag{2.24}$$

Here G_t is computed as:

$$G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}. \tag{2.25}$$

The return G_t is a key concept in reinforcement learning, representing the total discounted reward that an agent can expect to accumulate, starting from a time step t. It's calculated as the sum of rewards that the agent receives in the future, discounted by the factor γ at each time step.

2.5.1.2 Action-Value Function Approximation

In DRL, the action-value function, denoted as $Q(s,a;\theta)$ and parameterized by neural network weights θ, estimates the expected return of taking action a in state s. The goal is to learn a function that can predict the quality of each action given the current state.

An example of this approach is found in Deep Q-Networks (DQNs). DQNs aim to minimize a loss function that measures the difference between the current estimated action values and the 'target' action values. The following steps are followed to achieve this:

Step 1

Approximation with Neural Networks: A deep neural network, parameterized by weights θ, is used to approximate the Q-function. This network, $Q(s, a; \theta)$, takes a state s and action a as inputs and outputs an estimate of the Q-value for that state-action pair.

Step 2

Training the Network: The training of this neural network involves adjusting the parameters θ to minimize the difference between the network's Q-value predictions and the 'target' Q-values, which represent more accurate estimates of the true Q-values. The loss function used for this purpose in DQN is often the mean squared error between the predicted Q-values and the target Q-values. For a batch of experiences s, a, r' *and* s' sampled from the replay buffer, the loss function is:

$$L(\theta) = \mathbb{E}_{s,a,r',s' \sim Experience}\left[\left(y - Q(s, a; \theta)\right)^2\right]. \tag{2.26}$$

Where y is the target Q-value, given by:

$$y = r + \gamma \max_{a'} Q(s', a'; \theta^-), \tag{2.27}$$

where θ^- are the parameters of a separate target network, and s' is the subsequent state.

In DQN, a technique called 'fixed Q-targets' is employed, where the target Q-values are calculated using a separate network with parameters θ^-. This target network is periodically updated with the weights of the main Q-network. This approach stabilizes the training by providing consistent targets during iterative updates.

Step 3

Policy Derivation: The policy is derived using Q-function. In DQN, an ϵ-greedy policy is often used, where with probability ϵ a random action is chosen (exploration), and with probability $1 - \epsilon$, the action with the highest estimated Q-value is chosen (exploitation).

Action-value function approximation using deep neural networks thus enables handling of environments with high-dimensional state and action spaces, facilitating the learning of effective policies in complex and diverse scenarios.

2.5.1.3 Policy Function Approximation

In DRLs the policy $\pi(a \mid s; \theta)$, is a mapping using neural networks which are parameterized by θ, from s to a. In policy gradient methods, $\pi(a \mid s; \theta)$, itself is directly approximated. The objective is to maximize the expected return by adjusting the policy parameters θ, often employing gradient ascent on the expected return.

The objective function in policy gradient methods, denoted as $J(\theta)$, represents the expected return which is the expectation of $G(t)$ given by Equation (2.22). The gradient of this objective with respect to the policy parameters θ is given by:

$$\nabla_\theta J(\theta) = \mathbb{E}_{s\sim S, a\sim\pi}[log\pi(a|s;\theta)G_t]. \tag{2.28}$$

This equation indicates that the policy is adjusted in the direction that increases the probability of actions that lead to higher returns.

Policy gradient methods are particularly advantageous in scenarios where the action space is high-dimensional or continuous, as they can directly model complex policy structures and provide more nuanced control over actions compared to value-based methods. We use a variant of Policy gradient method called Proximal Policy Optimization (PPO) for modelling our work as an RL problem.

2.5.2 Invalid Action and Invalid Action Masking

In Reinforcement Learning (RL), the problem of invalid actions arises when the agent selects an action that is not feasible or permissible in the current state of the environment. This issue is particularly prevalent in environments where the set of valid actions varies depending on the state. For instance, in a board game like chess, the valid moves depend heavily on the current configuration of the board and the rules of the game. Choosing an invalid move (like moving a piece in a way not allowed by the game's rules) would not be a meaningful action for the agent to take. It can significantly impede the learning process if not properly managed. We observed this issue in our work when utilizing RL since possible actions are heavily dependent upon the current state of the switch.

To address this issue, a technique called invalid action masking is employed. This technique involves modifying the learning algorithm to ensure that the agent only considers valid actions at each decision point. There are several ways to implement this:

1. Modifying the Action Space: This method involves dynamically adjusting the action space based on the current state. While this approach is conceptually straightforward, it can be complex to implement, as it requires the action space to be redefined at every time step.

2. Masking Invalid Actions: A more common approach is to apply a mask to the action probabilities generated by the agent. The agent computes the probability of taking each possible action, but before making a final decision, the probabilities corresponding to invalid actions are set to zero. The remaining probabilities are then normalized to sum to one. This procedure ensures that the agent only selects from the set of valid actions.

3. Penalizing Invalid Actions: Alternatively, the agent is allowed to choose invalid actions but receives a negative reward or penalty when it does so. This approach is simpler in terms of implementation but can slow down the learning process as the agent might need to experiment with actions to understand their validity.

4. Rule-based Preprocessing: A rule-based system can preprocess the state information to determine valid actions. This filtered set of actions is then passed to the RL agent for decision-making.

Invalid action masking is crucial in environments where the action space is large or complex and the validity of actions is highly state-dependent. By focusing the agent's learning on valid actions, the training becomes more efficient, leading to improved performance and quicker convergence.

2.5.3 Proximal Policy Optimization Algorithm

Proximal Policy Optimization (PPO) is an advanced policy gradient method in Reinforcement Learning (RL), widely recognized for its effectiveness and simplicity in implementation. Developed by Schulman et al. [41], PPO addresses some of the key challenges in policy optimization, particularly those related to the stability and efficiency of training.

Traditional policy gradient methods update the policy by taking steps proportional to the gradient of the expected return. However, large updates can lead to unstable training and poor performance. PPO introduces a novel approach to control the size of policy updates, aiming for a balance between making significant progress and maintaining training stability.

The central idea of PPO is to limit the change in the policy at each update, ensuring that the new policy is not too far from the old one. This is achieved by modifying the objective function that the algorithm maximizes. PPO uses a clipped surrogate objective function, which penalizes changes to the policy that move the probability ratio away from 1 by more than a specified threshold, denoted as ε (epsilon).

The objective function $J(\theta)$ is replaced in PPO by:

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t \left[\min\left(r_t(\theta)\hat{A}_t, clip(r_t(\theta), 1 - \epsilon, 1 + \epsilon)\hat{A}_t\right], \quad (2.29)$$

$$r_t(\theta) \text{ is the probability ratio } \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}, \quad (2.30)$$

where $\pi_{\theta_{old}}$ and π_θ are the old and new policies, respectively. $\hat{A}_t$ is an estimator of the advantage function (refer Equation 2.19), at time t, $clip(r_t(\theta), 1 - \epsilon, 1 + \epsilon)$ ensures that the ratio between the new and old policies remains within the interval $[1 - \epsilon, 1 + \epsilon]$.

$\hat{\mathbb{E}}_t$ is used to estimate the average value of the clipped objective across a batch of experiences. This approach allows the algorithm to work with real-world samples, enabling it to learn and optimize policies from actual interactions with the environment.

The use of the clipping mechanism in the objective function prevents the policy from changing too drastically, which helps in maintaining training stability. The PPO algorithm alternates between sampling data through interaction with the environment using the current policy and optimizing the clipped surrogate objective function using stochastic gradient ascent.

One of the vital reasons we use PPO in our work other than its performance benefits for our model is the fact that, as of current status (December, 2023), PPO can be implemented using Invalid Action Masking technique using stablebaselines3-contrib python package. This is important to our work, as involving invalid action masking improves results on our switch model.

2.5.4 Sparse Reward Issue During Application of Reinforcement Learning

The sparse reward issue is a significant challenge in the application of Reinforcement Learning (RL). In many RL problems, especially in complex environments, an agent may receive feedback (rewards) infrequently or only in specific situations. This sparsity of rewards can greatly hinder the learning process, as the agent struggles to understand which actions lead to successful outcomes.

We can see in Figure 2-6 that if we apply RL to a simple grid world problem [15], for the first figure we will face the issue of sparse rewards, as we are getting only one high reward if reach the goal. In contrast the model will have no issues learning an optimal strategy in the second figure, as highly intermittent rewards act as feedback to guide its learning process.

One of the primary challenges associated with sparse rewards is the problem of delayed feedback. In environments with sparse rewards, an agent might need to take a series of actions before receiving any meaningful reward. This delay complicates the task of associating specific actions with their consequences, leading to a complex credit assignment problem where it becomes challenging to pinpoint which actions were crucial for achieving success. Additionally, sparse rewards can hinder effective exploration. Since rewards are few and far between, the agent may not have sufficient motivation to explore different strategies, potentially leading to a suboptimal policy.

Sparse rewards also lead to sample inefficiency, requiring a larger number of interactions with the environment for the agent to learn effectively. This can be particularly problematic in real-world scenarios where each interaction can be costly or time-consuming.

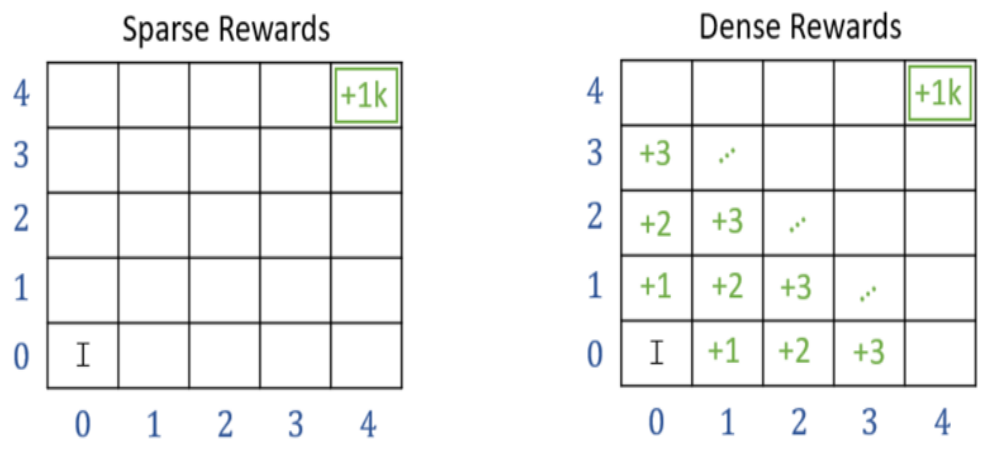


Figure 2-6: Sparse Rewards Grid World vs Dense Rewards Grid World

There are multiple methods of mitigating this issue. Some of the major ones and which we are using or tried for our work are the following:

1. Reward Shaping: This involves modifying the reward structure of the environment to make it more informative. Supplementary rewards are added to guide the agent's learning, though this requires careful design to avoid introducing biases.

2. Intrinsic Motivation: Here, agents are provided with internal rewards that encourage exploration, such as rewards for discovering new states or significant changes in the environment.

3. Curriculum Learning: This technique involves gradually increasing the complexity of the learning task, starting from simpler scenarios with more frequent rewards and progressing to more challenging ones.

4. Random Network Distillation: It addresses the challenge of sparse rewards and encourages efficient exploration. RND provides a mechanism for intrinsic motivation, incentivizing an agent to explore its environment more thoroughly using a pair of target – predictor neural networks for incentivizing exploration.

2.5.5 Random Network Distillation

Random Network Distillation (RND) [42] is an approach in reinforcement learning for encouraging exploration, especially in environments where rewards are sparse. The method revolves around two neural networks: a fixed, randomly initialized target network f and a trainable predictor network $\hat{f}$. Both networks take the state (or observation) s as input and output a vector. The core idea is to use the prediction error of $\hat{f}$ in estimating f as an intrinsic reward for the agent.

The target network f is defined as $f(s;\theta_{target})$, where θ_{target} are the parameters (weights) of the network, initialized randomly and kept constant throughout the training process.

The predictor network $\hat{f}$ is defined as $\hat{f}(s;\theta_{predictor})$, where $\theta_{predictor}$ are the trainable parameters of the network.

The intrinsic reward $r_{instrinsic}$ at each state s is computed based on the mean squared error between the outputs of these two networks:

$$r_{intrinsic}(s) = \left|\left| f(s;\theta_{target}) - \hat{f}(s;\theta_{predictor})\right|\right|^2. \tag{2.31}$$

During training, the agent receives this intrinsic reward in addition to any extrinsic rewards from the environment. The intrinsic reward is high in states where the predictor network's output significantly differs from the target network's output, indicating novel or less explored states. As the predictor network learns to approximate the target network over repeated visits to the same states, the intrinsic reward diminishes, encouraging the agent to explore new areas.

The training objective for the predictor network is to minimize the prediction error, which is typically formulated as a loss function L:

$$L(\theta_{predictor}) = \mathbb{E}_{s\sim Environment}[\, r_{intrinsic}]. \tag{2.32}$$

By continuously updating $\theta_{predictor}$ to minimize this loss using batch optimization, the predictor network becomes better at mirroring the target network's output, thereby reducing the intrinsic reward for states it has already explored and promoting exploration of new states. Refer Figure 2-7.

Optimizing $L(\theta_{predictor})$ using batch optimization use following steps:

Step 1

Data Collection: In RND, as the agent interacts with its environment, it collects data comprising states, actions, rewards, and possibly other relevant information. This data is typically stored in an experience replay buffer or a similar data structure.

Step 2

Batch Formation: An optimization batch is formed by sampling a subset of this data. The batch size is a predefined parameter that determines how many data points (experiences) are included in each batch. The choice of batch size can affect both the efficiency and effectiveness of the learning process.

Step 3

Training the Predictor Network: The primary use of the optimization batch in RND is to train the predictor network. The network learns to predict the output of the fixed, randomly initialized target network based on the input states. The optimization batch provides the necessary input-output pairs (states and corresponding target network outputs) for this training.

Step 4

Calculating the Loss: The predictor network's parameters are updated by minimizing a loss function. In RND, this is typically the mean squared error between the predictor network's outputs and the target network's outputs for the given states in the batch:

$$L(\theta_{predictor}) = \frac{1}{batch\ size}\left[\sum_{batch} r_{intrinsic}\right]. \tag{2.33}$$

Step 5:

Gradient Descent: The loss calculated from the optimization batch is used to perform a gradient descent step, adjusting the predictor network's parameters $\theta_{predictor}$ to reduce the prediction error.

$||.||^2$ represents the squared Euclidean norm of a vector. The Euclidean norm, often simply called the norm, is a measure of the length (or magnitude) of a vector in Euclidean space. In the RND framework, when we calculate the intrinsic reward $r_{intrinsic}$. We are essentially measuring the squared distance between the outputs of the target network f and the predictor network $\hat{f}$. This squared distance acts as an intrinsic reward signal for the agent, indicating how different the current state is from those it has previously encountered. The squaring operation here ensures that the reward is always non-negative and emphasizes larger differences more than smaller ones.

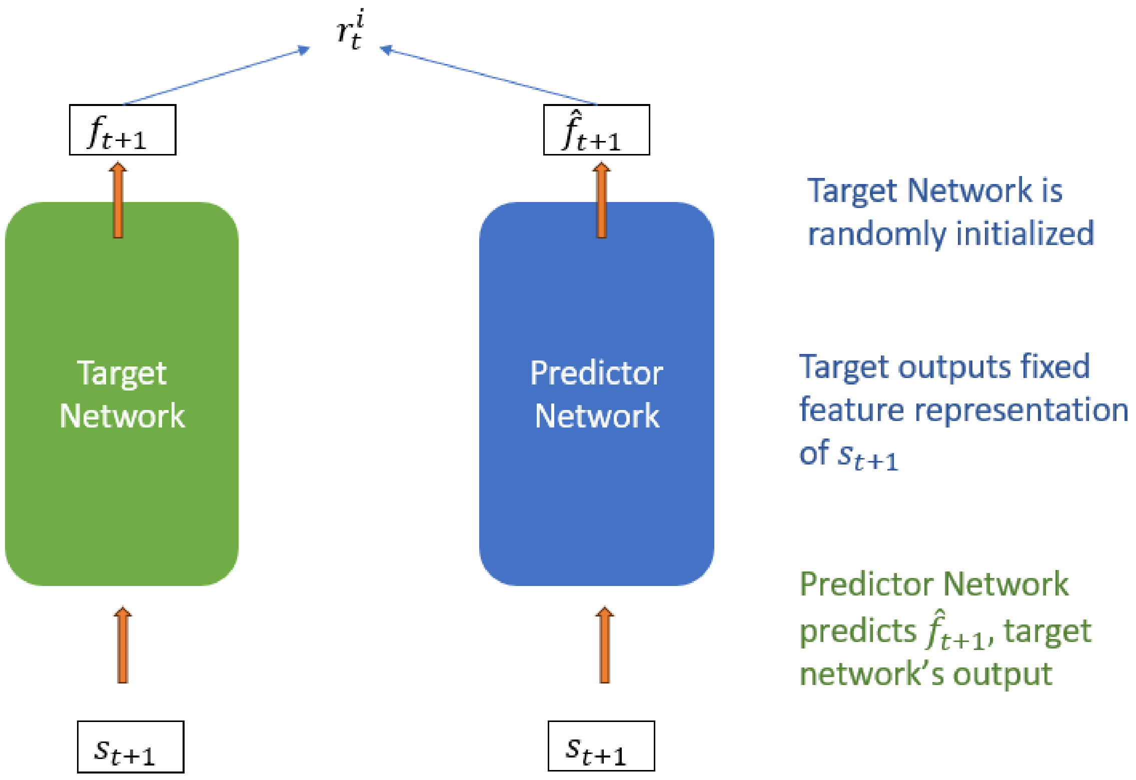

r_t^i
f_{t+1}
$\hat{f}_{t+1}$
Target Network
Predictor Network
s_{t+1}
s_{t+1}
Target Network is randomly initialized
Target outputs fixed feature representation of s_{t+1}
Predictor Network predicts $\hat{f}_{t+1}$, target network's output

Figure 2-7: Schematic of RND algorithm

This method of using prediction error as an intrinsic reward has been shown to significantly improve exploration in reinforcement learning, particularly in environments where extrinsic rewards are rare or misleading.

3.0 Bipartite Switch Model

Building upon the foundational concepts outlined in the previous sections, this segment is dedicated to elucidating the formulation of our model, specifically designed for determining optimal policies for our quantum switch model. This part of the discussion will focus exclusively on the model's formulation.

Our model is rooted in the framework of Markov Decision Processes (MDP), which provides a systematic approach to modeling decision-making in environments where outcomes are partly stochastic and partly under the control of a decision-maker, in this case, the switch. The MDP framework is characterized by its sets of states, actions, and rewards, along with transition probabilities that dictate the dynamics of the system. It will be later extended to account for solution using Reinforcement Learning Methods.

The formulation of our model does not yet delve into the solution methods, such as specific reinforcement learning algorithms or optimization techniques. Instead, it sets the stage by clearly defining the problem within the MDP framework, ensuring that the switch's decision-making process is structured in a way that aligns with the goal of maximizing throughput of end-to-end entanglement.

3.1 Base Model

We consider the simplest case of a quantum switch with just two clients, as illustrated in the schematic in Figure 3-1. The clients periodically attempt to establish a single link-level entanglement in the form of Bell pairs with the switch at a pre-defined rate $R_{clock} = \frac{1}{T_{clock}}$ per unit time step of the switch, where T_{clock} is the duration of a time step (that can be set to 1 without loss of generality).The attempts succeed with probabilities $\lambda 1, \lambda 2$, respectively.

The function of the switch is to connect successfully generated link-level Bell pairs with the two clients. However, it can do so only when:

1. They are available with both clients.
2. The resulting state after entanglement swapping at the switch can yield a Bell pair of entanglement fidelity higher than a target threshold F_{th}.

Until then, Bell pairs that are successfully generated with any one of the two clients alone are stored in quantum memory.

The link-level Bell pairs are modeled as Werner states. When stored in quantum memory, the Werner state fidelity is modeled to decay exponentially with time (starting from $F = 1$ at time $t = 0$).

Additionally, the link-level Bell pairs are discarded as unusable after a certain, finite number of time steps, m^* , i.e., time $t = m^*$ (when the fidelity reaches a cutoff fidelity F^*). In other words, for a link-level Bell pair of age $m \epsilon$ (0, 1, 2...., m^*) the resultant fidelity is given by Equation (2.12)

Since fidelity $F < ½$, would imply a separable state, a possible value for the fidelity cutoff is $F^* = ½$. However, more generally, for any cutoff fidelity F^* of a link-level Bell pair, and any given positive integer m* the value of the decoherence parameter is:

$$\alpha = \frac{-\ln(F^*)}{m^*}. \tag{2.34}$$

The switch functions as follows: Given one or many successfully generated link-level Bell pairs among the two clients, if there are pairs on both sides, the switch can either connect any pair from one client directly with any one of the Bell pairs of the other client that are stored and active, or when equipped with the option of entanglement distillation, perform probabilistic entanglement distillation between any pair of stored link-level Bell pairs. We assume that at the start of each timestep the new pairs arrive (if successful) and after this up to one action (swap/distill) can be performed in the same timestep.

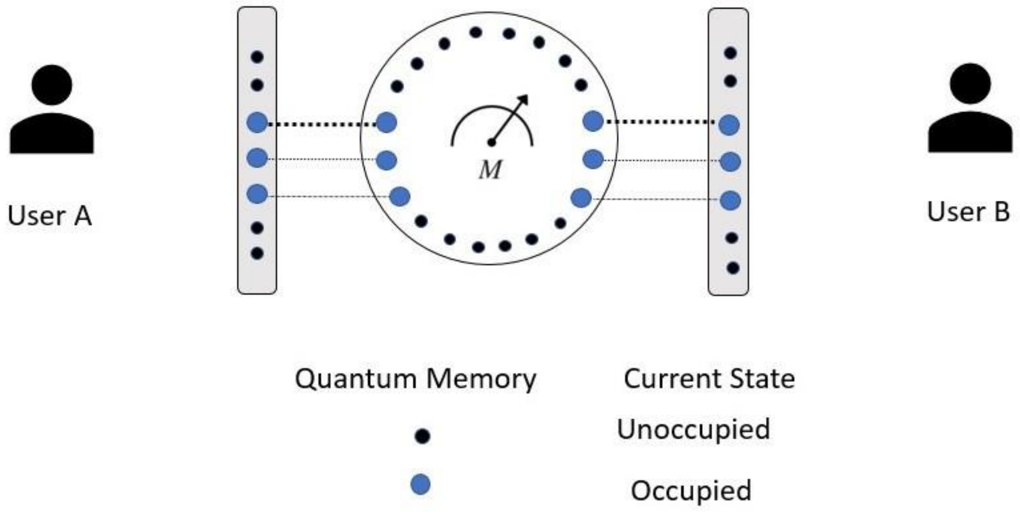


Figure 3-1: A schematic of quantum switch with two users (A, B) is depicted above. The thicker dotted line represents a newer elementary link, exhibiting higher fidelity compared to the thinner line, which has lower fidelity.

The length of the LLE buffer size is represented by q^i_{length}, where i represents the user to which the buffer belongs. E.g., referring to Figure 3-1, user A has $q^1_{length} = 7$ and user B has $q^2_{length} = 7$. We take the liberty to define only q_{length} if it is same for all users.

The model is simulated in Python and in summary, the simulation follows the steps as represented in Figure 3-2.

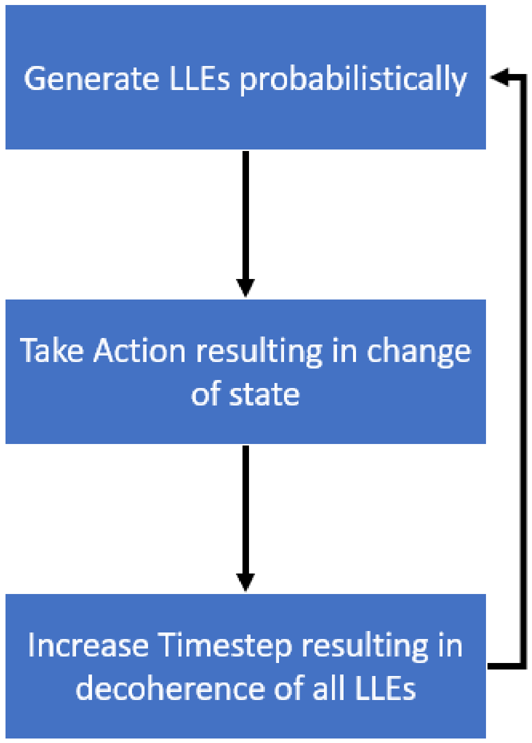

Generate LLEs probabilistically
Take Action resulting in change of state
Increase Timestep resulting in decoherence of all LLEs

Figure 3-2: Schematic for switch model simulation

3.2 Bipartite Switch Model for MDP Framework

In this work, we use the MDP formulation to identify optimal policies for distributing end to end Bell pairs by connecting randomly generated elementary Bell pairs in a quantum switch having two clients, with and without the option of entanglement distillation. In particular, we derive policies that optimize the end-to-end throughput of Bell pairs that exceed any given entanglement fidelity threshold, F_{thresh}.

In developing an optimal policy within the framework of a Markov Decision Process (MDP) for a quantum switch, our approach necessitates the adoption of several foundational assumptions. These assumptions are integral to the conceptualization and computational feasibility of the MDP model.

1. Probability of Entanglement Swap Success:

- We posit that the probability of successful entanglement swap operations, denoted as q, is unity. This assumption is valid in scenarios involving Bell measurements conducted on matter-based quantum memories, such as trapped ion qubits [43] and NV centers in diamond qubits [44], [45]. These scenarios demonstrate the possibility of achieving entanglement swaps with a success probability approaching one.

2. Memory Capacity of the Quantum Switch:

- As explained in previous section, the memory capacity is constrained to store a maximum of q^{i}_{length} elementary Bell pairs for each client.

3. Temporal Constraints on Actions and Link Generation:

- It is assumed that both the actions executed by the switch and the generation of probabilistic links are completed within discrete unit time steps.

4. State Space Discretization:

- To maintain computational tractability, the state space within the MDP is discretized. It comprises a finite series of states, each corresponding to the entanglement fidelities linked with the ages of elementary Bell pairs, expressed as {0, 1, 2, ..., m*}.
- Incorporating the full spectrum of potential fidelities, particularly those emerging from entanglement distillation actions, would lead to an untenable expansion of the state space. Therefore, the fidelity of each newly distilled Bell pair is mapped to the nearest corresponding age in the MDP state space.

Given these constraints, the model is approximate. Solving the MDP provides a policy that is optimal for this approximate model. To empirically validate the model and its policy implications, simulations employing actual fidelity values of Bell pairs are conducted. These simulations are crucial for evaluating the practical efficacy of the proposed policy, particularly in metrics such as network throughput, average fidelity, and jitter.

3.2.1 States

The state space of our MDP formulation of the quantum switch (not to be confused with quantum states) most generally consists of combinations (tensor products) of N – dimensional age vectors associated with $N \epsilon Z^{\geq}$ elementary Bell pairs generated by client A with the quantum switch and M – dimensional age vectors associated with $M \epsilon Z^{\geq}$ elementary Bell pairs of client B, where these vectors are of the form $[m_1, m_2, m_3 \dots \dots]$, with $m_i \epsilon [0,1,2 \dots \dots m^*]$ being the age of the i_{th} elementary Bell pair. We assume that all elementary Bell pairs have same cut-off age, i.e., $m_i^* = m^*$.

We denote the state vector at any time t as $S_t = [m_1, m_2, m_3 \dots \dots, m_N]_A$, $[m_1, m_2, m_3 \dots \dots, m_M]_B$, and the set of MDP states by S.

The state of the switch is represented by list of queues, one queue reserved for each user. A possible state of the switch/environment with $q_{length} = 3$ and at time t for discrete time can be:

$s(t)$ = [[2,1,1], [2,0]].

Here we have a switch whose user A has a full queue where one LLE has an age of 2 timesteps and the other two have the same age of 1. User B only has two LLEs in its queue and hence is partially filled. The ages are 2 and 0, i.e., one LLE was created 2 timesteps ago while the other is freshly made and did not have time to decohere, hence having a fidelity of 1.

3.2.2 Actions

Given an MDP state, the switch performs actions based on a policy. The list of possible actions that could feature in a policy of a quantum switch enabled with entanglement distillation are $\mathcal{A} = \{a_1, a_2, a_3\}$:

1. $a1$ is the action in which the switch decides to wait out the time-step in order for more link-level Bell pairs to be generated. This is equivalent to no action being performed.

2. $a2$ is the action in which entanglement swap operation of elementary Bell pairs of the two clients are performed. This action if successful, generates end-to-end entanglement between clients A and B. If multiple swaps are possible this action also entails choosing which pairs to attempt to swap.

3. $a3$ describes the action of entanglement distillation of two elementary Bell pairs. If and when successful, this action leads to creation of an elementary Bell pair of improved fidelity. This action is performed to counter the effect of decoherence at the cost of sacrificing some elementary Bell pairs. If multiple distillations are possible this action also entails choosing which pairs to attempt to distill.

Actions possible to a switch for both discrete and continuous state spaces are the following:

1. Wait: The switch refrains from taking any action in this timestep. All LLEs present decohere by 1 timestep and their resultant fidelity is given by Equation (2.31).

2. Distill: The switch decides to choose from available LLEs of queues belonging to same users. They are distilled and generate LLE probabilistically with new fidelity given by the Equations (2.14) and (2.15). In our work, the switch performs 2-1 distillation according to Bennett Protocol as explained in Section 2.3.1, but can distill 2 links from each user in the same time step. If the distillation fails, all participating LLEs are lost.

3. Swap: The switch chooses 1 LLE from each user and swaps them. The resultant fidelity is given by Equation (2.14)

The actions are also represented as a list of numbers for each queue, representing choice of links.

E.G. An action for distilling for a state $s(t)$ = [[2,1,1], [2,0]] can be:

$$a(t) = [[2,1],[\,]].$$

This means the switch is choosing to distill LLE of age 2 and one of the other LLE of age 1 from user A, and deciding to do nothing for user B, thus decohering it's links.

Similarly, an action of swap can be represented by:

$$a(t) = [[2], [0]].$$

Here the switch decides to swap the LLE of age 2 from user A with the freshly generated LLE of User B.

3.2.3 Transition Probability

P is a state transition probability matrix, where $P_{ss'}^{a} = P[S_{t+1} = s' | S_t = s, A_t = a]$ is the probability of transitioning from state s to s 0 when an action a ∈ A is applied on the initial state and probabilistic link level entanglement is attempted: In a unit time step, first probabilistic link-level entanglement with each client occurs; either or both may succeed or fail. Then the switch action is taken.

For each state, s, action, a and resultant state final state after increasing timestep, s', and both sources providing LLEs successfully, the transition probability is given by:

$$P_{ss'}^{a} = \begin{cases} \lambda_1 \times \lambda_2 \,, & for\ wait \\ \lambda_1 \times \lambda_2 \times p_{succ}, & for\ distill \\ \lambda_1 \times \lambda_2 \times q & for\ swap \end{cases} . \tag{2.35}$$

In case if one or both sources fail to provide LLEs, $P_{ss'}^{a}$ in case of failures will be calculated by replacing λ_1 by $1 - \lambda_1$, and λ_2 by $1 - \lambda_2$.

3.2.4 Rewards

In our quantum switch model, formulated within a Markov Decision Process (MDP), the reward function is a crucial aspect that drives the decision-making process. This function, defined as $r: S \times A \rightarrow \mathbb{R}$, maps the transition from a current state s to a subsequent state s' with an action a from the set A. The immediate reward for being in state s and executing action a is denoted as $r_a(s)$. It's important to note that in our model, the generation of probabilistic links is a continuous process unaffected by the switch's actions. Additionally, in scenarios involving probabilistic rewards, $r_a(s)$ may be representative of an expected value.

For the action $a1$, which implies 'no action', the reward function $r_{a1}(s)$ is zero. This aligns with situations where the switch decides to passively wait, not engaging in any active operations.

The reward function for $a2$, $r_{a2}(s)$ is zero as our metric is the creation of end-to-end entanglement links using entanglement swaps. While distillation alone does not create these links, it can move to a state which is more likely to perform entanglement swaps in future. This preparation is evaluated through the MDP.

The reward function for $a3$, i.e., entanglement swap between two elementary Bell pairs, say between the i^{th} Bell pair with client A denoted by A_i, and the k^{th} Bell pair with client B denoted by B_k, of fidelities F_i and F_k, respectively, resulting in end-to-end entanglement is:

$$r_{a3}(s) = q \times F_{ik} = \begin{cases} 0, & F_{ik} < F_{th} \\ 1, & otherwise \end{cases}, \tag{2.36}$$

where, q is the success probability of the entanglement swap operation, and F_{ik} is given by Equation (2.14) with $F_1 = F_i$, $F_2 = F_k$ and F_{th} is the fidelity threshold for the end-to-end entanglement link.

3.3 Bipartite Switch Model for Reinforcement Learning Framework

In the course of developing our Markov Decision Process (MDP) model for the quantum switch, we confronted the challenge inherent in solving MDPs with extensive state spaces. Given the computational complexity and intractability of solving MDPs with large state spaces, we were compelled to introduce certain assumptions to maintain a finite and manageable state space.

These assumptions constrain the model's capacity to accurately represent the complexities of a real-world quantum switch. The simplifications made to keep the state space finite led to a more abstracted portrayal of the switch's behavior and operational dynamics.

Introducing RL to solve the problem is beneficial, because it allows us to transition from a discrete to a continuous state space. This is because RL algorithms, especially those using function approximation techniques, can generalize across similar states. This property is invaluable in continuous spaces where it is impractical to learn a value or policy for every possible state explicitly. Instead, the RL agent learns to infer values or policies for unseen states based on its learning from experienced states.

This change in the working framework brings with itself certain implementational changes to the state, action and reward formulation. The basic model remains the same as that of the MDP formulation, with all parameters remaining unchanged. The ability to work with continuous state spaces allows us to relax the State Space Discretization Assumption.

For creating the model, so as to be able to train RL algorithms on it, we used Gymnasium. It is an open-source Python library which is specifically designed for developing and comparing reinforcement learning (RL) algorithms. We used it to create a custom environment having all the functionalities of the switch described in Section 3.1.

3.3.1 States

Reinforcement Learning gives us the flexibility to work with continuous state spaces and hence now the states are not represented by age of the LLEs, but rather their actual fidelities. This leads us to be able to relax the State Space Discretization assumption and letting us use the actual fidelity of a resultant LLE after distillation.

Now the state space of our formulation of the quantum switch consists of combinations of N – dimensional fidelity vectors associated with $N \epsilon \mathbb{R}^{\geq}$ elementary Bell pairs generated by client A with the quantum switch and M – dimensional fidelity vectors associated with $M \epsilon \mathbb{R}^{\geq}$ elementary Bell pairs of client B, where these vectors are of the form $[f_1, f_2, f_3 \dots\dots]$, with $f_i \epsilon (F^*, 1]$ being the fidelity of the i_{th} elementary Bell pair. F^* is the fidelity threshold.

We denote the state vector at any time t as $S_t = [f_1, f_2, f_3 \dots\dots, f_N]_A$, $[f_1, f_2, f_3 \dots\dots, f_M]_B$, and the set of states by S.

From an implementation point of view, we model the states using Box datatype in Gymnasium. It aligns perfectly with our definition of states as combinations of fidelity vectors.

3.3.2 Actions

There is fundamentally no change in the representation of actions when compared to our MDP formulation, but since MDP was formulated using python and RL framework is formulated using Gymnasium, we have to look for proper data structures to represent our actions.

For the algorithms we are using for solving the RL environment (PPO augmented with RND), there was no appropriate data structure available in Gymnasium. As a workaround we used `discrete` space in Gymnasium to model our actions. This is done by externally generating a dictionary and mapping discrete values to all available action choices represented in Section 3.2.2 as key – value pairs, and then solving the RL to choose actions from that mapping. These actions are then used to simulate the switch.

3.3.3 Rewards

The rewards are modelled differently than mentioned in Section 3.2.4. This is since we need to use various techniques to mitigate for sparse reward issue faced by our RL model.

As explained in Section 2.5.4, sparse rewards are a hindrance to RL training. In our case, it is caused because we only give feedback to the agent with a reward when Swap is done and resultant fidelity is above F_{thresh}. This kind of swap is a novel action for our switch model and hence causes low feedback from the environment. If not mitigated, the agent learns to wait for a freshly generated LLE on both sides and swap them as soon as they are available. The low feedback dissuades the agent to explore new states and actions and hence it outputs a trivial policy. To solve this, we use a mixture of RND, (refer Section 2.5.5) and Curriculum Learning (refer Section 2.5.4).

The resultant reward is same as the MDP for Wait (a_1) and Swap (a_3), but differs considerably for Distillation (a_2). The new reward for doing distillation at state s is

$$r_{a2}(s) = \mathrm{r}_{\mathrm{intrinsic}}(s) + F_{12}\left(\overbrace{\frac{1}{1+e^{-5(1-x)}}}^{\kappa}\right). \tag{2.37}$$

Let's look at Equation (2.37) carefully. $r_{intrinsic}(s)$ is the intrinsic reward we are getting for exploring new states using RND, refer Section 2.5.5 and Equation (2.31). F_{12} is the resultant fidelity after distillation succeeds, given by Equation (2.16). If distillation fails F_{12} is 0, thus we get only reward for exploring a new state. The term labelled as κ is used as a tunable parameter to enable curriculum learning.

If we plot the κ vs x, we get the following graph:

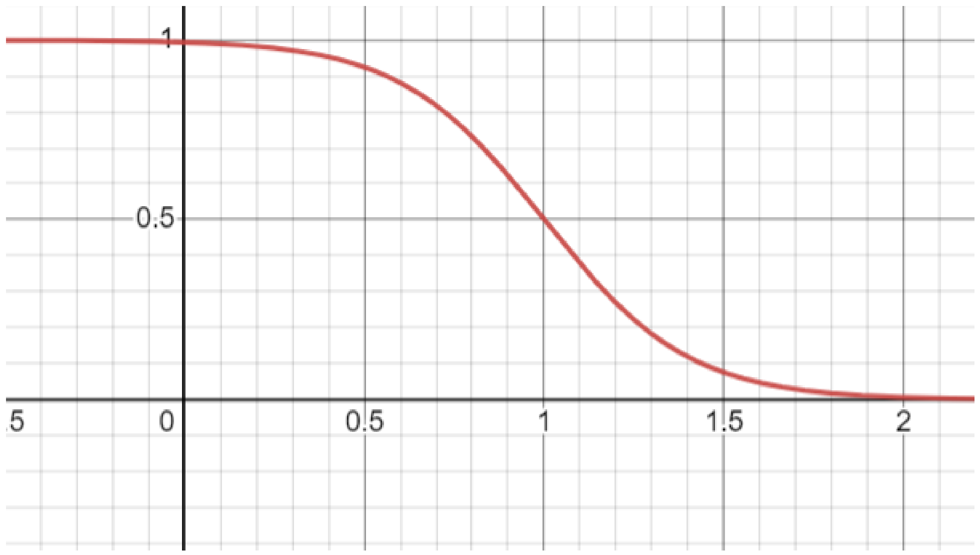


Figure 3-3: Plot of $\boldsymbol{\kappa}$

Looking at Figure 3-3, we can see that as we increase x, the value of κ follows an S shape. This is important for curriculum learning as now we can train the RL algorithm and increase value of x as our current iteration number increases. This will allow us to initially give high importance to F_{12} but as we see the model performing well, we can increase the value of x to stop adding F_{12} to our reward.

We need to incrementally reduce the involvement of F_{12} by increasing x because our final motivation is to increase the throughput of the switch and that is mainly dependent upon SWAP that generates high fidelity end-to-end entanglements. Giving too much importance on F_{12} will make the agent start doing distillation indiscriminately, thus veering us away from our end goal

4.0 Bipartite Switch Model Solution

After framing our model according to both MDP and Reinforcement Learning formulation we will now solve it to find the optimal policy and their characteristics.

4.1 Using MDP

We use the Policy Iteration method described in Section 2.4.2 to generate the final optimal policy using our MDP switch formulation. But first we need to describe our Transition Function for all (s, a, s') tuples, so as to be able to calculate the value function for each state using Equation (2.17).

A sample Transition Function table after calculating probability of all (s, a, s') tuples using Equation (2.35) is given in Table 1:

Table 1: Sample Transition Function Table

Initial States	Action	Choice of Links	Link Gen	Final State	Transition Probability
[[2,1], [1,0]]	Swap	[1, 2]	All Fail	[[ ], [1]]	0.09
[[2,1,0], [1,0]]	Wait	None	All Fail	[[2,1], [2,1]]	0.09
[[2,1], [1,0]]	Swap	[2, 1]	All Fail	[[2], [2]]	0.09
[[2,1], [1,0]]	Distill Succeeds	[[1, 2], [1, 2]]	All Succeed	[[0], [0]]	0.480

After the Transition Function table is generated, we use transition probability values and the rewards to calculate $V(s)$ of each state. apply Policy iteration and generate the optimal policy.

To solve the MDP formulation, we run 40,000 iterations of Policy Iteration Algorithm for Fid_{thresh} intervals of 0.05, from 0.7 to 0.95. The model that we solved had the following values of the parameters:

$\lambda1 = \lambda2 = 0.7$; $m^* = 3$; $F^* = 0.85$; $q_{length} = 3$.

4.2 Using Reinforcement Learning

To solve the RL model, we use an RND augmented PPO algorithm. The PPO algorithm uses an estimator of the Advantage Function in Equation (2.29). As explained in Section 2.4 and Equation 2.19, the advantage function uses the expected discounted reward in its formulation through the use of value function and Q-value function.

We use the reward defined in Equation (2.37) which also consists of the RND formulated $r_{intrinsic}$ to calculate the advantage function, which further down the line is used by PPO algorithm in its loss function in Equation (2.29).

stablebaselines3-contrib is used to run an algorithm called *maskedPPO*. This algorithm is the same as PPO but implicitly applies Invalid Action Masking described in Section 2.5.2 using python functions that provide the algorithm with a list of invalid states at each time step of the iteration.

We use the default neural network parameters as the policy parameters for $\pi(a \mid s; \theta)$ when using PPO. The default neural network in Stable Baselines' PPO is a

Multi-Layer Perceptron (MLP) with two hidden layers. Each hidden layer has 64 neurons and uses rectified linear unit (ReLU) activation functions. This default neural network can be changed.

We have kept the predictor and target neural networks the same as that of the default PPO neural network, except for the target, the parameters are assigned randomly.

We run 80,000 iterations for each fidelity threshold from 0.7 to 1.0 at intervals of 0.025. The value of x in Equation (2.37) is increased as the KL-Divergence between two batches of iterations goes below 0.05. This directs the agent towards slowly getting 0 rewards for distillation so that it learns to distill solely on the basis of throughput achieved.

5.0 Results

After solving the MDP and RL models, we simulated the switch's operation using the corresponding optimal policies for 10,000 timesteps. We have analyzed the optimal policies generated for MDP framework and the RL framework. We have also modelled the case when distillation is not allowed, and compared the optimal polices generated in both cases.

The results for MDP are plotted for three key metrics: average throughput, average fidelity, and jitter against fidelity which varies from 0.7 to 0.95. Average throughput represents the number of end-to-end entanglement links above the fidelity threshold; average fidelity represents the average quality of these end-to-end entanglement links. Finally, jitter is the standard deviation of the time between successive end-to-end entanglements across all time steps.

The same plot is generated for RL framework too but only for average throughput.

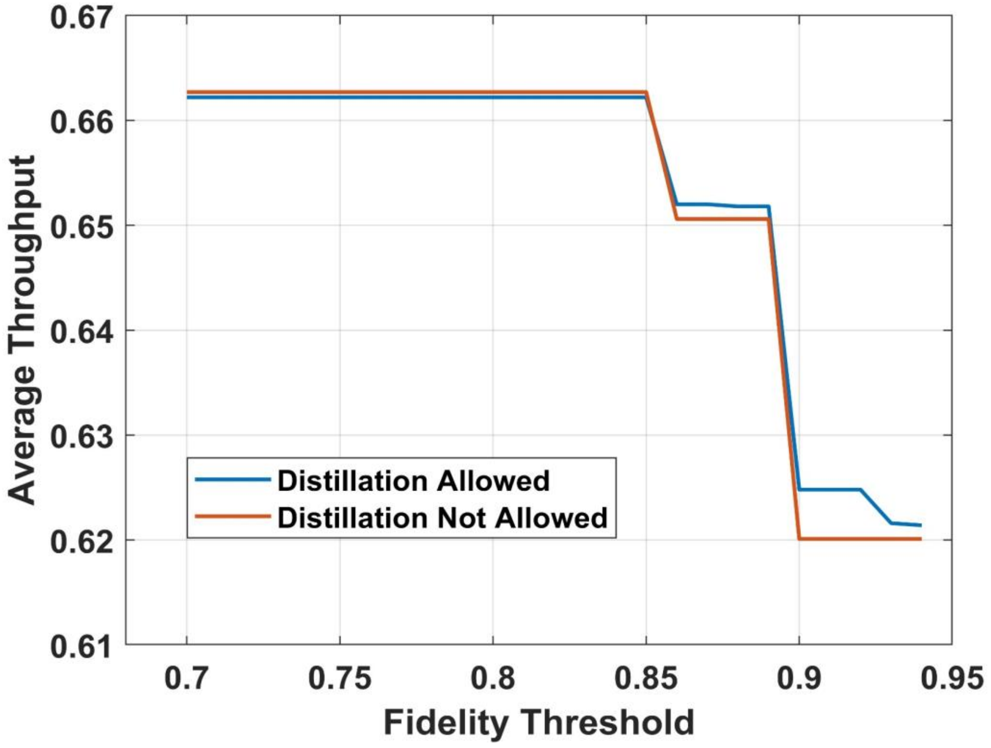


Figure 5-1: Average Throughput vs Fidelity Threshold for cutoff age $\boldsymbol{m}^* = 3$, link generation probabilities $\boldsymbol{\lambda_1} = \boldsymbol{\lambda_2} = \mathbf{0.7}$, cutoff fidelity $\boldsymbol{F}^* = \mathbf{0.85}$, storage capacity of quantum memory, $\boldsymbol{q_{length}} = 3$ for policies with and without entanglement distillation that optimize the end-to-end entanglement throughput.

Figure 5-1, shows the average throughput versus the fidelity threshold for the model with distillation (blue curve) and without (orange curve).

At lower fidelities, the policy obtained through MDP that allows for distillation vs the one that does not, i.e., only based on entanglement swaps, yield the same average throughput. This is because entanglement swaps of low fidelity (decohered) elementary links can still generate an end-to-end link that meets the low required fidelity threshold.

As the fidelity threshold increases, the optimal policy with distillation outperforms the policy that doesn't allow distillation, as not every pair of swaps can produce an end-to-end entanglement exceeding the threshold. The reason behind this is that only entanglement swaps between the established highest fidelity (i.e., age 0) links can lead to end-to-end entanglements surpassing the fidelity threshold, thereby providing no benefit for distillation. Thus, at intermediate target thresholds distillation, while it does not yield any end-to-end entanglement links itself, it enables the system to preserve fidelity of aging links, which yields greater throughput in the future.

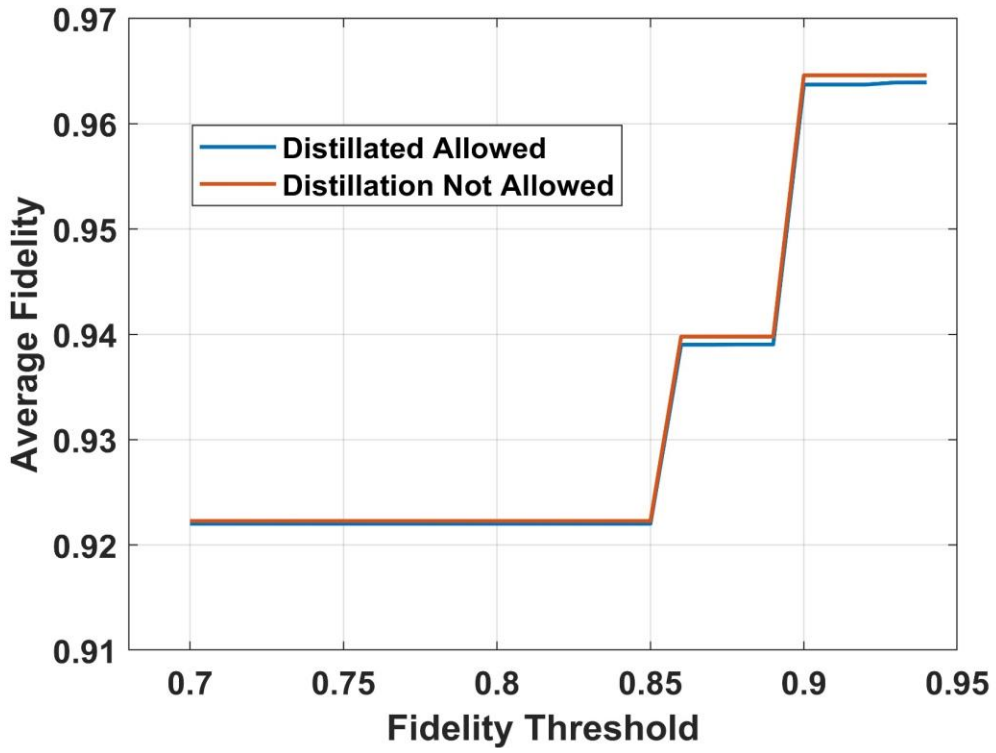


Figure 5-2: Average Fidelity vs Fidelity Threshold for cutoff age $\boldsymbol{m}^{*} = 3$, link generation probabilities $\boldsymbol{\lambda_1} = \boldsymbol{\lambda_2} = \mathbf{0.7}$, cutoff fidelity $\boldsymbol{F}^{*} = \mathbf{0.85}$, storage capacity of quantum memory, $\boldsymbol{q_{length}} = 3$ for policies with and without entanglement distillation that optimize the end-to-end entanglement throughput.

The graph in Figure 5-2 shows the relationship between average fidelity and fidelity threshold for two different policies. The policy that allows distillation is optimized to achieve higher throughput rather than fidelity.

As a result, it tends to produce more end-to-end entanglements, as shown in Figure 5-1. However, this policy does not take into account the quality of the end-to-end links, as long as they meet the fidelity threshold. On the other hand, the policy that does not allow distillation always performs entanglement swaps to ensure high-quality links. Consequently, we observe that the average fidelity of the policy that does not distill is higher than the one where distillation is allowed.

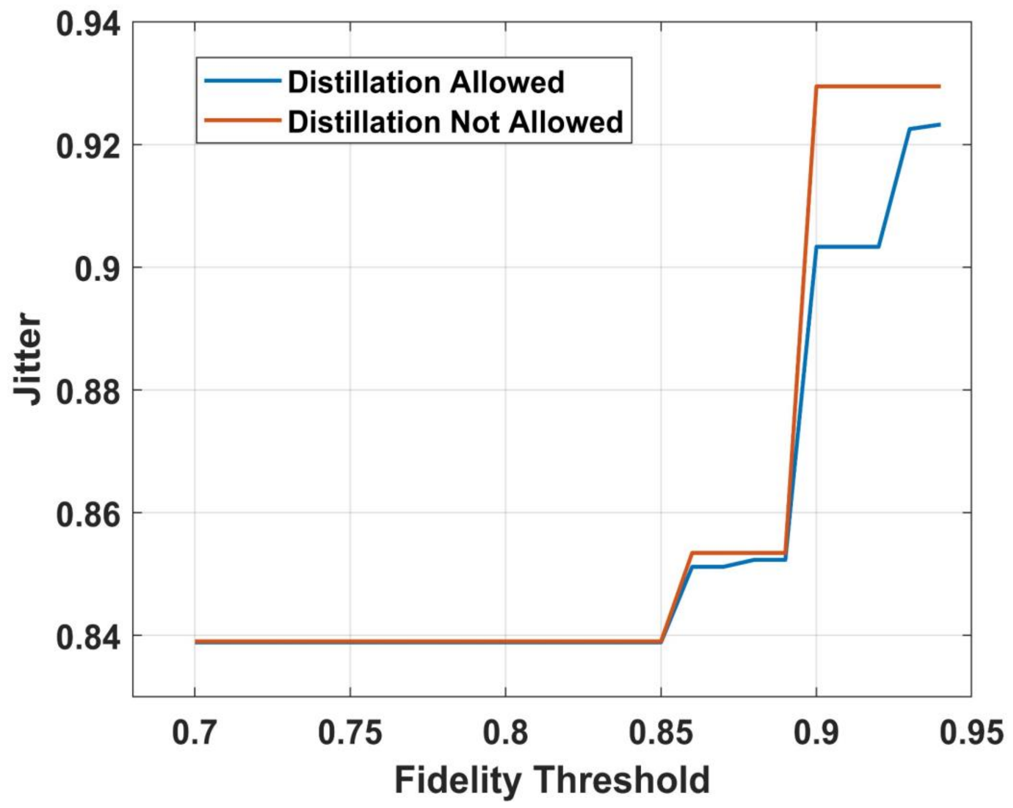


Figure 5-3: Jitter vs Fidelity Threshold for cutoff age $\boldsymbol{m}^{*} = 3$, link generation probabilities $\boldsymbol{\lambda_1} = \boldsymbol{\lambda_2} = \mathbf{0.7}$, cutoff fidelity $\boldsymbol{F}^{*} = \mathbf{0.85}$, storage capacity of quantum memory, $\boldsymbol{q_{length}} = 3$ for policies with and without entanglement distillation that optimize the end-to-end entanglement throughput.

The plotted graph in Figure 5-3 shows the relationship between jitter and fidelity threshold. It reinforces the message that following an optimal policy which allows for distillation is superior to the one without distillation as it leads to a slightly lower interarrival time. Therefore, it can be inferred that the use of distillation in the optimal policy results in improved performance compared to policies that do not allow for distillation.

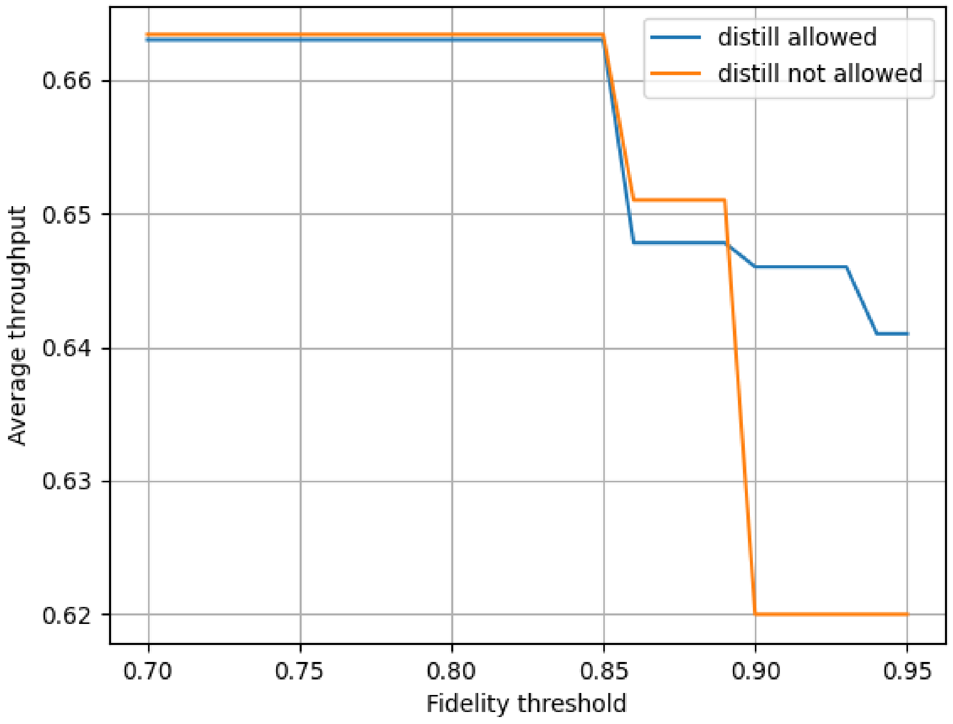


Figure 5-4: Average Throughput vs Fidelity Threshold for RL Model for cutoff age $\boldsymbol{m}^* = 3$, link generation probabilities $\boldsymbol{\lambda_1} = \boldsymbol{\lambda_2} = \mathbf{0.7}$, cutoff fidelity $\boldsymbol{F}^* = \mathbf{0.85}$, storage capacity of quantum memory, $\boldsymbol{q_{length}} = 3$ for policies with and without entanglement distillation that optimize the end-to-end entanglement throughput.

In Figure 5-4, we can see that allowing distillation makes the switch perform at much higher throughput for fidelity thresholds between 0.90 – 0.95.

But there is still a glaring concern regarding the policy between fidelity thresholds 0.86 - 0.90. We can see that the orange curve for the simulation where distillation is not allowed is considerably higher than the blue for these thresholds. This is undesirable since not allowing distillation should never realistically increase the throughput more than when allowing distillation.

This is because the action space, when we allow distillation completely contains the action space when distillation is not allowed. So, in worst case scenario they should perform equally. This is simply means that our RL algorithm could not handle the increased state and action space when distillation is allowed for thresholds 0.86 – 0.90, and could not get trained properly.

Hence, we require some more work in hyperparameter tuning and modelling of the RL model. Nevertheless, the switches performance between 0.90 – 0.95 thresholds is encouraging.

6.0 Conclusions

In conclusion, our work provides a framework to generate and analyze optimal switching policies for a quantum network using MDP and RL frameworks. It shows that including distillation as an option for a quantum switch in a network can increase the throughput of end-to-end entanglement distribution. The plots of the three metrics indicate that the policy with quantum distillation performs better as compared to the policy with no distillation over a certain range of fidelity thresholds. However, beyond a threshold of 0.95, both policies have same values across all three metrics. This is because the policy with quantum distillation provides no advantage at such high fidelity thresholds. The RL model needs involved hyperparameter tuning to get the most optimal policy at all fidelity thresholds.

Future work can include more complex protocols could be considered, where a higher n number of elementary Bell pairs are transformed into a higher fidelity but fewer k ($k < n$) number of elementary Bell pairs [36]– [39]. Also, within two-to-one distillation protocols, nested distillation [28], [40], i.e., multi-stage protocols aimed at enhancing the fidelity of entangled links, could be considered.

Our model has so far considered scenarios with only two clients. Future work could investigate scenarios where the switch must generate end to-end bipartite entanglement for three or more users as well as multipartite entanglement.

These results were presented at IEEE QCE 2023 (to appear in proceedings) [46], [47]

All code and documentation will be available at https://github.com/vivek-kumar9696/OptimalEntanglementDistillationPoliciesQuantumSwitches-

Bibliography

[1] S. Wehner, D. Elkouss, and R. Hanson, “Quantum internet: A vision for the road ahead,” Science, vol. 362, no. 6412, Oct. 2018

[2] J. P. Dowling and G. J. Milburn, “Quantum technology: the second quantum revolution,” Philos. Trans. A Math. Phys. Eng. Sci., vol. 361, no. 1809, pp. 1655–1674, Aug. 2003.

[3] S. Pirandola, U. L. Andersen, L. Banchi, and others, “Advances in quantum cryptography,” Advances in optics, 2020.

[4] M. Mehic, M. Niemiec, S. Rass, J. Ma, M. Peev, A. Aguado, V. Martin, S. Schauer, A. Poppe, C. Pacher, and M. Voznak, “Quantum key distribution: A networking perspective,” ACM Comput. Surv., vol. 53, no. 5, pp. 1–41, Sep. 2020.

[5] G. Vardoyan, M. Skrzypczyk, and S. Wehner, “On the quantum performance evaluation of two distributed quantum architectures,” SIGMETRICS Perform. Eval. Rev., vol. 49, no. 3, pp. 30–31, Mar. 2022.

[6] R. Van Meter and S. J. Devitt, “The path to scalable distributed quantum computing,” Computer, vol. 49, no. 9, pp. 31–42, Sep. 2016

[7] E. Kashefi and A. Pappa, “Multiparty delegated quantum computing,” Cryptogr. Commun., vol. 1, no. 2, p. 12, Jul. 2017.

[8] J. F. Fitzsimons, “Private quantum computation: an introduction to blind quantum computing and related protocols,” npj Quantum Information, vol. 3, no. 1, pp. 1–11, Jun. 2017.

[9] S. Pirandola, R. Laurenza, C. Ottaviani, and L. Banchi, “Fundamental limits of repeaterless quantum communications,” Nat. Commun., vol. 8, p. 15043, Apr. 2017.

[10] K. Azuma, A. Mizutani, and H.-K. Lo, “Fundamental rate-loss trade-off for the quantum internet,” Nat. Commun., vol. 7, p. 13523, Nov. 2016.

[11] M. Takeoka, S. Guha, and M. M. Wilde, “Fundamental rate-loss tradeoff for optical quantum key distribution,” Nat. Commun., vol. 5, p. 5235, Oct. 2014

[12] S. Muralidharan, L. Li, J. Kim, N. Lutkenhaus, M. D. Lukin, and ¨ L. Jiang, “Optimal architectures for long distance quantum communication,” Sci. Rep., vol. 6, p. 20463, Feb. 2016.

[13] W. J. Munro, K. Azuma, K. Tamaki, and K. Nemoto, “Inside quantum repeaters,” IEEE J. Sel. Top. Quantum Electron., vol. 21, no. 3, pp. 78–90, May 2015.

[14] 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,” Phys. Rev. A, vol. 92, no. 2, p. 022357, Aug. 2015.

[15] R. Van Meter, Quantum Networking. John Wiley & Sons, May 2014.

[16] G. Vardoyan, S. Guha, P. Nain, and D. Towsley, “On the exact analysis of an idealized quantum switch,” SIGMETRICS Perform. Eval. Rev., vol. 48, no. 3, pp. 79–80, Mar. 2021.

[17] ——, “On the stochastic analysis of a quantum entanglement switch,” ACM SIGMETRICS Performance Evaluation Review, vol. 47, no. 2, pp. 27–29, 2019.

[18] M. A. Nielsen, I. Chuang, and L. K. Grover, “quantum computation and quantum information,” Am. J. Phys., vol. 70, no. 5, pp. 558–559, May 2002.

[19] P. Nain, G. Vardoyan, S. Guha, and D. Towsley, “On the analysis of a multipartite entanglement distribution switch,” Proc. ACM Meas. Anal. Comput. Syst., vol. 4, no. 2, pp. 1–39, Jun. 2020.

[20] G. Vardoyan, P. Nain, S. Guha, and D. Towsley, “On the capacity region of bipartite and tripartite entanglement switching,” ACM Transactions on Modeling and Performance Evaluation of Computing Systems, vol. 8, no. 1-2, pp. 1–18, 2023.

[21] W. Dai, A. Rinaldi, and D. Towsley, “The capacity region of entanglement switching: Stability and zero latency,” in 2022 IEEE International Conference on Quantum Computing and Engineering (QCE). IEEE, 2022, pp. 389–399.

[22] T. Vasantam and D. Towsley, “Stability analysis of a quantum network with Max-Weight scheduling,” Jun. 2021.

[23] I. Tillman, T. Vasantam, and K. P. Seshadreesan, “A continuous variable quantum switch,” in 2022 IEEE International Conference on Quantum Computing and Engineering (QCE). IEEE, 2022, pp. 365–371.

[24] I. J. Tillman, A. Rubenok, S. Guha, and K. P. Seshadreesan, “Supporting multiple entanglement flows through a continuous-variable quantum repeater,” Mar. 2022.

[25] K. P. Seshadreesan, H. Krovi, and S. Guha, “Continuous-variable quantum repeater based on quantum scissors and mode multiplexing,” Phys. Rev. Research, vol. 2, no. 1, p. 013310, Mar. 2020.

[26] C. H. Bennett, G. Brassard, S. Popescu, B. Schumacher, J. A. Smolin, and W. K. Wootters, “Purification of noisy entanglement and faithful teleportation via noisy channels,” Phys. Rev. Lett., vol. 76, pp. 722–725, Jan 1996. [Online]. Available: https://link.aps.org/doi/10. 1103/PhysRevLett.76.722

[27] W. Dur, H.-J. Briegel, J. I. Cirac, and P. Zoller, “Quantum repeaters ¨ based on entanglement purification,” Phys. Rev. A, vol. 59, pp. 169–181, Jan 1999. [Online]. Available: https://link.aps.org/doi/10. 1103/PhysRevA.59.169

[28] N. K. Panigrahy, T. Vasantam, D. Towsley, and L. Tassiulas, “On the capacity region of a quantum switch with entanglement purification,” arXiv preprint arXiv:2212.01463, 2022.

[29] Nielsen, Michael A., and Isaac L. Chuang. *Quantum computation and quantum information*. Cambridge university press, 2010.

[30] Wilde, Mark M. *Quantum information theory*. Cambridge university press, 2013.

[31] Sutton, Richard S., and Andrew G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[32] Charles H. Bennett, Gilles Brassard, Claude Cr´epeau, Richard Jozsa, Asher Peres, and William K. Wootters. Teleporting an unknown quantum state via dual classical and einstein-podolsky-rosen channels. Phys. Rev. Lett., 70:1895–1899, 1993. doi: 10.1103/PhysRevLett.70.1895.

[33] Artur K. Ekert. Quantum cryptography based on bell’s theorem. Phys. Rev. Lett., 67:661–663, 1991. doi: 10.1103/PhysRevLett.67.661.

[34] Valerio Scarani, Helle Bechmann-Pasquinucci, Nicolas J. Cerf, Miloslav Duˇsek, Norbert L¨utkenhaus, and Momtchil Peev. The security of practical quantum key distribution. Rev. Mod. Phys., 81:1301–1350, 2009. doi: 10.1103/RevModPhys. 81.1301.

[35] H.-J. Briegel, W. D¨ur, J. I. Cirac, and P. Zoller. Quantum repeaters: The role of imperfect local operations in quantum communication. Phys. Rev. Lett., 81: 5932–5935, 1998. doi: 10.1103/PhysRevLett.81.5932.

[36] D. Dieks. Communication by epr devices. Physics Letters A, 92:271 – 272, 1982. doi: DOI:10.1016/0375-9601(82)90084-6.

[37] W. K. Wootters and W. H. Zurek. A single quantum cannot be cloned. Nature, 299:802, 1982. URL http://dx.doi.org/10.1038/299802a0

[38] L.-M. Duan, M. D. Lukin, J. I. Cirac, and P. Zoller. Long-distance quantum communication with atomic ensembles and linear optics. Nature, 414:413–418, 2001. doi: 10.1038/35106500.

[39] C. H. Bennett, G. Brassard, S. Popescu, B. Schumacher, J. A. Smolin, and W. K. Wootters, "Purification of noisy entanglement and faithful teleportation via noisy channels," Phys. Rev. Lett., vol. 76, pp. 722–725, Jan 1996. [Online]. Available: https://link.aps.org/doi/10. 1103/PhysRevLett.76.722

[40] S. Khatri, "On the design and analysis of near-term quantum network protocols using markov decision processes," AVS Quantum Science, vol. 4, no. 3, p. 030501, 2022.

[41] Schulman, John, et al. "Proximal policy optimization algorithms." *arXiv preprint arXiv:1707.06347* (2017).

[42] Burda, Yuri, et al. "Exploration by random network distillation." *arXiv preprint arXiv:1810.12894* (2018).

[43] L.-M. Duan and C. Monroe, "Colloquium: Quantum networks with trapped ions," Rev. Mod. Phys., vol. 82, no. 2, pp. 1209–1224, Apr. 2010.

[44] H. Yan, Y. Zhong, H.-S. Chang, A. Bienfait, M.-H. Chou, C. R. Conner, E. Dumur, J. Grebel, R. G. Povey, and A. N. Cleland, "Entanglement ´ purification and protection in a superconducting quantum network," Phys. Rev. Lett., vol. 128, no. 8, p. 080504, Feb. 2022.

[45] Z.-Q. Yin, W. L. Yang, L. Sun, and L. M. Duan, "Quantum network of superconducting qubits through an optomechanical interface," Phys. Rev. A, vol. 91, no. 1, p. 012333, Jan. 2015.

[46] Kumar, V., Chandra, N. K., Seshadreesan, K. P., Scheller-Wolf, A., & Tayur, S. (2023). Optimal Entanglement Distillation Policies for Quantum Switches. *arXiv preprint arXiv:2305.06804*.

[47] Kumar, V., Chandra, N. K., Scheller-Wolf, A., Tayur, S., & Seshadreesan, K. P. (2023). "Optimal Entanglement Distillation Policies for Quantum Switches using Markov Decision Process and Reinforcement Learning" (in preparation).