2024 International Conference on Quantum Communications, Networking, and Computing (QCNC) | 979-8-3503-6677-8/24/$31.00 ©2024 IEEE | DOI: 10.1109/QCNC62729.2024.00030

# Reinforcement Learning Based Proactive Entanglement Swapping for Quantum Networks

Tasdiqul Islam
University of Texas at Arlington
txi7184@mavs.uta.edu

Md Arifuzzaman
Missouri University of Science and Technology
marifuzzaman@mst.edu

Engin Arslan
University of Texas at Arlington
engin.arslan@uta.edu

***Abstract*—Entanglement generation and swapping is a difficult process due to probabilistic nature of quantum mechanics. To overcome this issue, existing quantum routing algorithms try to create entanglement on multiple paths between source and destination. Although it is possible to save entangled qubits on unused links using quantum memories, the quantum routing algorithms discard them and try creating new entanglement in each time slot. In this work, we leverage the longevity of entanglement and introduce two enhancements to improve the performance of existing routing algorithms: (i) The generation and caching of entanglements across multiple time slots, and (ii) the proactively executing entanglement swapping for frequently utilized links, increasing the success rate of future requests. Since entanglement caching reduces available quantum memory on quantum repeaters, we apply reinforcement learning to identify the optimal set of segments that are likely to be used in the future. Through comprehensive simulations, we demonstrate that caching unused entanglements lead to $10-15\%$ improvement on the performance of quantum routing algorithms REPS and SEER. Complementing the caching with reinforcement learning based proactively entanglement swapping leads to further performance improvements. Specifically, using Q-Learning to decide on which segments to proactively swap leads to $52.55\%$ and $41.79\%$ improvement for REPS and SEER algorithms, respectively. Deep Q-Learning outperforms Q-Learning with $61.43\%$ and $48.27\%$ improvements rates.**

***Index Terms*—Quantum Network, Entanglement routing, End-to-end entanglement,**

## I. Introduction

Quantum networking has gained popularity due to its wide range of application areas such as quantum key distribution [1]–[3], distributed quantum computing [4]–[9], and clock synchronization [10]–[12]. Taking these benefits of quantum networking for long distance communication requires the use of quantum repeaters due to fidelity loss of qubits as they travel long distance [13], [14]. Routing algorithms in quantum communication play a key role for enhancing throughput of quantum networks by optimizing path selection based on several factors such as entanglement success rate and available capacity in quantum repeaters.

Figure 1 depicts a simple quantum network with several quantum repeaters and Entangled Photon Sources (EPS). EPSes that are positioned between nodes produce entangled qubit pairs then transfer them to nodes via quantum links. Each node is equipped with quantum memory to store these received qubit(s). The establishment of long-distance entanglement is facilitated by entanglement swapping performed

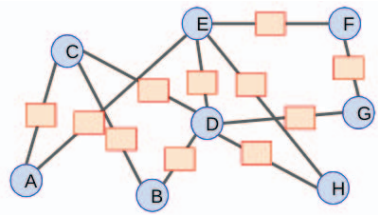



Fig. 1. A simple quantum network with physical links and entangled photon source (EPS) in the middle of every link.

by the intermediate nodes (i.e., quantum repeaters) that can perform Bell State Measurement (BSM). Despite the availability of resources, establishing quantum communication is not guaranteed as it is influenced by many factors including the success rate of generating entangles qubits, distance between nodes, and quality of quantum links, and success probability of BSM. Moreover, the capacity of EPSes (the number of entangled qubit producers), physical links (the number of qubit transmissions), and quantum repeaters (performing BSMs) is limited. Hence, routing algorithms play a crucial role in determining the quantity of entanglements to be generated between two nodes, considering uncertainties introduced by quantum noise during the entanglement swapping process. Numerous algorithms for quantum routing have been proposed [15]–[21], to enhance throughput and fairness in quantum networks.

Existing routing algorithms conduct four-phase process to establish a quantum connection in a time slot, as suggested by Shi et al. [18]. Phase 1 involves receiving requests and undertaking any necessary pre-processing. In the second phase, a set of links for entanglement generation is selected based on link and memory capacity considerations. The third phase attempts entanglement generation using EPSes for selected links. Finally, in the last phase, swapping process is executed on entangled links to finalize the end-to-end entanglement. These four phases collectively constitute a single time slot. A time slot is typically considered in the order of few hundred milliseconds. For example, authors of quantum routing algorithm, SEER, consider the time slot to be 200 milliseconds [21].

All entanglements are destroyed at the end of each time slot, thus all four phases need to be re-executed from scratch in the next time slot. However, recent studies show that entanglement can persist longer than few hundred milliseconds. As an exam-

979-8-3503-6677-8/24/$31.00 ©2024 IEEE
DOI 10.1109/QCNC62729.2024.00030

ple, researchers stored the state of qubits for over 100 seconds using carbon-13 atoms in a diamond [22].Similar performance is deemed possible for optically connected quantum networks [22]–[24]. In [25], authors kept the state of arbitrary single-qubit state for over 75 seconds and two-qubit entanglement for over 10 seconds. In [26], an optimization for the time of a quantum memory should hold the noisy bell states is shown. Thus, by taking advantage of the longevity of entangled qubits, it is possible to cache the unused entangled qubits and use them in the future when necessary. This will lead to an increase in the success rate of quantum connection establishment by minimizing the need for new entanglement generation.

In this paper, we introduce two strategies to take advantage of long (multi-time slot) lifespan of entanglement in long distance quantum communication. First, we proactively generate entanglements on quantum links and cache them. The rationale behind this approach is that since entanglement creation is a probabilistic process, trying it ahead of time is more likely to result in successful entanglement compared to just trying at the time of a request. Second, we propose to swap entanglement on commonly used network segments (i.e., a set of neighboring links) such that entanglement will not only be available on a link level but across multiple links, which will further increase the likelihood of success of future requests. It is however important to note that quantum repeaters have limited transmission and memory capacity, thus proactive entanglement swapping on multiple links must be executed carefully to efficiently utilize the available resources. We therefore introduce reinforcement learning methods to determine what percentage of available resources to allocate for proactive entanglement swapping.

We incorporated the proposed improvements into two state-of-the-art quantum routing algorithms (SEER [21] and REPS [19]) to demonstrate their effectiveness. We show that simply caching unused entangled qubits [1], for several time slots (e.g., $5-10$ time slots), the performance of quantum routing algorithms (SEER and REPS) increase by 20% in terms of request satisfaction rate. Taking it a step further by proactively entanglement swapping across multiple commonly used links with the help of reinforcement learning leads up to $61\%$ and $48\%$ improvement for the REPS and SEER, respectively.

## II. Related Work

Several routing algorithms have been proposed (Q-CAST [18], REPS [19], SEER [21], DQRA [27], and SEE [20]) for quantum communication with the objective of maximizing throughput and minimizing idle time. They mostly operate on a time-slot basis in which they gather all requests arrived in a time slot and execute them in the next time slot. Time slot is typically defined as several hundred milliseconds. In another work, requests are processed immediately after receiving them [15].

REPS [19] operates in discrete time slots and employs Integer Linear Programming to process requests and find optimal links to maximize throughput. SEE [20] combines all-optical switching of photonic qubits with entanglement swapping, allowing qubit transfers to the next node regardless of the hop distance, as long as decoherence is minimal. The algorithm utilizes Integer Linear Programming to identify optimal segments and links.

SEER [21] enhances source-destination entanglement reliability by dividing the main path into two. The qubit is initially sent to an intermediate trusted node, which then forwarded it to the final destination in the subsequent time slots. The intermediate node is chosen based on social relationships within the network. Dividing a long path into two segments reduces the path length, thereby increasing the success rate of entanglement establishment. On the other hand, it takes longer (i.e., at least two time slot) to complete a given request. SEER employs a greedy algorithm for path determination thus it executes faster than Linear Programming.

DQRA [27] employs a deep neural network to observe the current network states for scheduling requests, which are subsequently routed via a qubit-preserving shortest path algorithm.

Q-CAST [18] deviates from conventional routing approaches by determining paths not solely based on the shortest distance but also considering the probability of successful entanglement establishment. It prioritizes the establishment of entanglements between source-destination pairs, albeit lacking fairness among multiple requests.

While existing routing algorithms perform well in identifying to optimal routes for given requests, they operate in time slots with each time slot being in the order of few hundred milliseconds. In each time slot, they determine paths for given requests, try to generate entanglement, choose optimal path based on successfully entangled links, then finally perform entanglement swapping to extend the entanglement end-to-end. Due to probabilistic nature of quantum operations, they choose more links than necessary to ensure that at least one end-to-end path can be established. This results in many unused entanglements which are typically assumed to be destroyed at the end of time slot. Since recent studies show that entanglement can lasts for several seconds [22], [25], we examine the impact of caching the unused entangled links on the performance of existing algorithms.

## III. Proactive Entanglement Generation

The existing quantum routing algorithms generally adopt a four-phase process [18]. In the first phase, they select "optimal" path(s) consisting of set of links for a given request. Most algorithms generate multiple paths for one request since connection attempt on primary path may not be successful due to failed entanglement generation or swapping. In the second phase, entangled qubit pair is created for each link using Entanglement Photon Sources (EPS) whose success rate

[1]Most quantum routing algorithms attempt to create entanglement on multiple links/paths to increase the overall probability of creating at least one end-to-end connection. This in turn result in several unused entangled links/segments.

is contingent upon the inherent probability associated with each link. In other words, EPSes have different success rate of generating entangled qubits and transmitting them to quantum repeaters located at the end of link. As a result, some links may fail to create entanglement for the give time slot. From the pool of successfully entangled links, an optimal path is then selected for entanglement swapping. However, similar to the entanglement generation, entanglement swapping can fail due to possibility of failure in the Bell State Measurement (BSM) process. Ultimately, if at least one of the selected paths is successful in creating entanglement on all links and conducting BSMs, the request is deemed to be successful. If there are multiple successful paths, the routing algorithm chooses one of them and discards the remaining.

Discarding all unused entangled links and swapped segments require all four phases to be re-executed at every time slot [19], [21]. Since previous studies [25] showed that entanglement between two qubits can endure for up to ten seconds, we propose to cache unused entangled links for a few time slots (e.g., 10 time slots) to better utilize resources and improve the performance of routing algorithms. Specifically, we introduce two enhancements designed to augment the capabilities of existing routing algorithms.

### A. Entanglement Generation and Caching

Existing routing algorithms discard the unused entanglements which necessitates new entanglement generation in each time slot. Taking advantage of longevity of entanglement, we aim to maximize the success of connection establishment by caching unused entangled links. Moreover, we try to entangle all links even if they are not selected for current requests to be able to use them for future requests. Since entanglements cannot be cached forever, we mark the entanglement generation time for each unused link to keep track of their remaining lifespan. If a cached entanglement is not utilized within its lifetime, it is destroyed and the associated memory is freed.

Consider a scenario with four requests (A-F, A-E, B-E, B-F in Figure 2(a)) in a network where all links have three quantum entanglement capacity. Assume a quantum routing algorithm determined optimal routes as shortest path between source and destination. Figure 2(b) shows the successfully entangled links after the entanglement attempts. The routing algorithm then chooses paths based on successful entanglements extend the entanglement end-to-end via entanglement swapping. Due to capacity limitations and failure during entanglement swapping across links, assume A-E and B-F requests are successful and A-F and B-E requests are unsuccessful as depicted in Figure 2(c). Since two entanglements were created on links A-C and B-C while only one of them are used for A-E and B-F connections, the second ones can be cached for future requests.

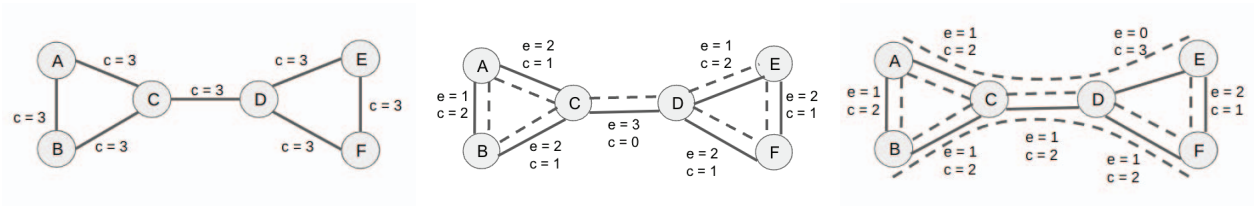
### B. Proactive Entanglement Swapping

As we cache unused entanglements in the link-level, it is also possible to conduct entanglement swapping across neighboring links to extend the entanglement from single link to multiple links (i.e, segments). These segments can then be used to meet future requests. Entanglement swapping over multiple links reduces the average path length, thereby increasing the probability of successful entanglement for forthcoming requests. Additionally, it creates more paths between source and destination, offering additional alternatives which enhances the success probability. On the other hand, the capacity of quantum repeaters are limited, thus, it is important to create entangled segments judiciously to avoid running out of quantum memory which would result in lack of resources to utilize in the following time slots. Moreover, since proactive entanglement swapping cannot guarantee predicting future perfectly, it is important to reserve some percentage of link capacity to meet the requests in the upcoming time slot. Therefore, we implemented heuristic, reinforcement learning (both Q-Learning, and Deep Q-Learning) methods to predict which segments to proactively establish entanglement such that the success rate of future requests can be improved.

**Heuristic Method:** Algorithm 1 outlines a simple heuristic solution that aims to determine which segments to select for proactive entanglement swapping. The algorithm maintains a record of frequently used links. Then, we identify pairs of links suitable for extending entanglement over. Based on past

**Algorithm 1:** Algorithm to determine segments for proactive entanglement swapping

```
Data: selected paths by the existing algorithms
Result: swapped links ahead of time
needLinkDict ← ∅;
for path in selected path do
    for node1, node2 in path do
        needLinkDict[(node1,node2)].append(timeslot)
    end
end
for node1 , node2 in needLinkDict do
    if needLinkDict[(node1, node2)] > needlink_timeslot then
        link1, link2 ← entangled links between node1,node2;
        if tryPreswap(link1, link2) is success then
            create a virtual link between node1 and node2
            remove link1 and link2
        else
        end
    else
    end
end
if virtual link not used then
    keep entangled qubits in node1 and node2;
    restore link1 and link2
else
    restore link1 and link2
end
return virtual links ;
```




(a) Initial configuration (b) Entanglement Generation (c) Final Configuration

Fig. 2. Illustration of how unused entangled links are cached. Solid lines show available capacity (i.e., c) whereas dashed lines show the number of entanglements (i.e., e). First, entanglement generation is attempted resulting in (b). Then, entanglement swapping is tried to satisfy A-E and B-F requests. Unlike existing models, we cache unused entanglements as shown in (c) to be able to utilize them in the future.

requests, we calculate how many times does each segment is used in last few time slots (lines 2 to 6). At the start of each time slot, the map is sorted based on usage count. If the usage count surpasses a specified threshold, entanglement swapping is attempted on that segment (lines 7 to 16). In our implementation, we add the successful entanglements created through our proactive entanglement swapping method to the network topology as virtual links. Available quantum memory capacity is reduced by one for the end nodes of the segments (i.e., quantum repeaters).

In our topology, a "quantum link" indicates the capacity of the entanglement photon source and the number of photons that can be transmitted through the "physical quantum link" in one time slot. The lifetime of the virtual link is set as the minimum lifespan among the swapped links. If the virtual link is not used withing its lifespan, the original links are restored (lines 17 to 22) since the entanglement photon source can generate photons in the next time slot. Only quantum memories at the end nodes of the entangled segment are allocated to create the virtual link. If virtual link is used or its lifespan is expired, the quantum memories of the nodes are released. As an example, if A-B and B-C links are used to create the entanglement between A-C through swapping, the quantum memories on node B is released after the BSM measurement as the qubits on A and C will be entangled.

While proactively creating entanglement on segments can be useful if they are used in the near future, it can hurt the performance as we are allocating scarce quantum memory on quantum repeaters. While the heuristic approach tries to find the most commonly requested links and segments, it can fail to extract complex patterns. For example, it will give higher priority to short segments over long segments as short segments (that are part of long segments) are used at least as much as long segments. Thus, we also explore reinforcement learning to find more complex patterns and offer better performance over the heuristic method.

**Q-Learning:** Q-learning [28] is a type of model-free reinforcement learning algorithm that iteratively learns based on actions and observed rewards. In Q-learning the impact of taking an action on a state is kept and the impact value is updated over time to determine optimal actions in a given condition. In this work, we first use table based Q-learning to decide on which links or segments to proactively entangle to increase the success rate of future requests. The Q-learning method engages in an iterative procedure where various components collaborate to facilitate model training. This iterative process involves the agent learning through environmental exploration and continuously updating the model as exploration progresses.

The Q-learning process begins with the initialization of the Q-table, which serves as a repository for tracking the progress of each action within every state. Next, the agent observes the current state of the environment before selecting an action to take. Once the action is executed, the model assesses its impact on the environment to determine its effectiveness. Subsequently, the Q-table is updated based on the outcome of the taken action. This sequence of act-observe-update process continues until model reaches to a termination state in which the objective of the model is satisfied. Here are the components of Q-learning for our problem.

- *Environment:* Previous studies proposed both linear programming (REPS [19]) and heuristic models (SEER [21]) for routing. We incorporated our proposed caching and proactive entanglement swapping approaches into these existing solutions. Hence, the routing algorithm (SEER or REPS) acts as the environment for the Q-learning model.
- *States:* We define all possible segments that are up to three hops long as states. Thus, state space is $s = \{(n_1, n_2) \mid 1 < dist(n_1, n_2) \leq 3\}$;
- *Actions:* We have two possible actions that can be taken for any given state (i.e., a segment). We can either choose it or not choose it to proactively entangle. So, the action space is $a = \{0, 1 \mid 0 : not\ select, 1 : select\}$
- *Rewards:* We define one reward and two penalty scenarios.
  1) We give a positive reward ($R(s, a) > 0$) if a proactively entangled segment is used to meet a request in the future.
  2) We give a negative reward (($R(s, a) < 0$) if a proactively entangled segment expires (i.e., lifespan ends) without being used. This penalty ensures that the model will try avoid choosing segment that are not highly likely to be used in the future.
  3) We give a penalty for missing opportunities to proactively entangle segments. In other words, if the routing algorithms choose a certain segment that the model did not proactively entangle, we assign a negative reward ($R(s, a) < 0$) to the model for

failing to choose that segment in the previous time slots.

- *Episodes:* We consider each time slot of routing algorithms (SEER and REPS) as an episode, so we update the Q-values in every time slot.
- *Q-values:* For every state, we maintain two Q-values (number of actions) in the Q-table. The Q-value is updated using Bellman equation as follows:

$$\underbrace{\text{New}Q(s,a)}_{\text{New Q-Value}} = Q(s,a) + \underbrace{\alpha}_{\text{Learning rate}} [\underbrace{R(s,a)}_{\text{Reward}} + \underbrace{\gamma}_{\text{Discount rate}} \overbrace{\max Q'(s',a')}^{\text{Maximum predicted reward}} - Q(s,a)] \quad (1)$$

Unlike the heuristic model that tries to proactively entangle all segments (as long as there is enough resource) that meet the usage threshold, Q-Learning choose them based on Q-values. However, selecting segments with largest Q-values may limit the models ability to explore the solution space by choosing the same segments all the time. Thus, we utilize exploration together with exploitation [29] to ensure that we can explore new solutions while taking advantage of discovered solutions. The value of $\gamma$ in Q-Learning indicates the probability of taking a random action (i.e., selecting a random segment to entangle). We set the initial value of $\gamma = 0.5$ and gradually lower over time. This exploration technique strikes a balance between taking advantage of discovered (optimal) actions and searching for more (optimal) actions.

**Deep Q-Learning:** In table based Q-learning, the states consists only of up to three hop long segments. While adding more information into the state such as network topology and available requests (both recent requests and previous requests that were not fulfilled) can be helpful for the agent to better observe the impact of the actions, this will significantly increase the complexity for table-based Q-learning due to increasing computational challenges. Using only 2-3 hop segments as states can also be difficult in Q-Learning if a network contains thousands of well connected nodes as it will generate billions of states [30]. Hence, we introduce a deep neural network to be able to represent more complex states.

- *States:* For Deep Q-learning the state is consist of three components:
  1) *Topology:* The connectivity of the network $G^{(t)}(V, E^{(t)})$ at time slot $t$ is calculated as a subset of the topology $G$. Only entangled links are considered as edges. Then, we extract the adjacency matrix $A^{(t)} = (a_{i,j})_{1 \le i,j \le |V|}$ of $G^{(t)}(V, E^{(t)})$ where $a_{i,j}$ is the number of entangled links between nodes $i$ and $j$. This is a $N \times N$ matrix where $N$ is the number of nodes.
  2) *Requests:* The new requests at time slot $t$ along with the remaining requests from previous time slots are represented a $N \times N$ matrix which is denoted as $R^{(t)} = (r_{s,d})_{1 \le s,d \le |V|}$ where $r_{s,d}$ is the number of requests between source and destination nodes.
  3) *Segments:* The $m^{th}$ pair $(n_1, n_2)$ that will be chosen to create proactive entanglement is represented by a binary vector $P_m^{(t)}$, where each element $v_i$ is either 1 (if $i$ belongs to $(n_1, n_2)$) or 0 otherwise.

  The above three components make up the input state for $m^{th}$ pair $S_m^{(t)} = [G^{(t)}; R^{(t)}; P_m^{(t)}]$
- *Output Actions:* The Deep Q neural network outputs a Q-value vector for each segment in each time slot $Q_m^{(t)} = [q_0^{(t)}, q_1^{(t)}]$ of length 2 and the action $a_m^{(t)} = argmax_{i \in 0,1}(q_i^{(t)})$ is applied on the $m^{th}$ pair. Figure 3 shows an example of the input and output.
- *Training Algorithm:* Two versions of Q networks are maintained in the Deep Q-learning method for model training. They are a prediction network and a target network. The Q-values are obtained from the prediction network, which undergoes training using the Stochastic Gradient Descent Algorithm. The target network's weights are updated periodically with those of the prediction network, typically at every few hundred time steps. The target network is exclusively utilized for forecasting future Q-values.

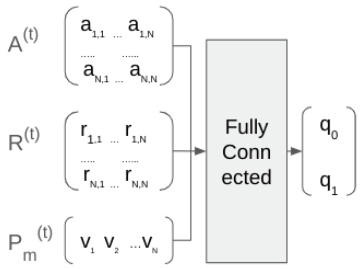



Fig. 3. An sample input and output of the Deep Q-Learning. It takes segments, network topology, and available requests as input and list of segments to proactively entangle as an output.

Deep Neural Network is evaluated for every segment to determine whether or not to swap the entanglement. Ideally, these decisions (i.e., actions) should be executed sequentially such that state changes caused by one action (e.g., reduction of available link capacity) can be reflected in the evaluation of another action. However, sequential execution introduces two challenges. First, evaluation order of actions will affect the performance since available capacity will decrease as more actions are selected, giving advantage to actions that are evaluated earlier. Second, it takes long time (up to 30 seconds) to evaluate DNN model sequentially for all possible actions. Hence, we evaluate actions in parallel, then choose the ones with the highest Q values as long as there are enough resources.

Successful attempts change the network state to $S^{(t+1)}$. Next, we calculate the reward similar to Q-learning at the end of each time slot. The pair of $(S^{(t)}, S^{(t+1)}, reward, a_m^{(t)})$ is kept in a buffer to train the model in batches.

Figure 4 illustrates an example of selected links with Q-learning. The model predicted to create entanglement between

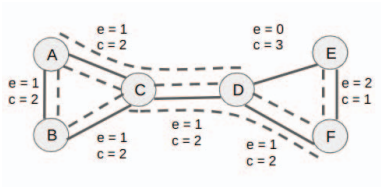



Fig. 4. Virtual links A-D and C-F are created after swapping entanglement between links A-C, C-D and C-D, D-F. Doing so reduced the average distance between nodes, which would increase the likelihood of success for future request.

A-D and C-F, so we conducted swapping operations (i.e., BSM) between links A-C and C-D and links C-D and D-F. Virtual links are created after successful swapping operations.

## IV. Evaluations

We integrated the proposed entanglements caching and proactive entanglement generation and swapping methods into two quantum routing algorithms algorithms, namely REPS [19] and SEER [21]. REPS uses linear programming to identify optimal links for each request in a given time slot. After entanglement is attempted on selected links, it tries entanglement swapping using successfully entangled links to create end-to-end entanglement for each request. On the other hand, SEER exploits the social relationship of the nodes and divides the requests (if possible) into two parts. Then, it tries to generate entanglement for each part separately. It then attempts entanglement swapping to combine the both parts. As a result, it takes several time slots to satisfy a given connection request.

### A. Evaluation Methodology

To construct the network, we employed the Waxman model [31] and adhered to the simulation methodology outlined in [18], [19], [21]. In accordance with these simulation parameters, we placed 50 quantum nodes within a rectangular area measuring 2000 km by 4000 km. Each link has a randomly generated (between 3-7) qubit transmission capacity and each node has randomly generated (between 10-14) quantum memory capacity. The probability for entanglement generation was formulated as $P(u,v) = e^{-\alpha \cdot l(u,v)}$, where $l(u,v)$ represents the Euclidean distance between two nodes. A typical value for the parameter $\alpha$ was set to $0.002$. With $\alpha = 0.002$ and $l(u,v) = 100km$ the value of $P(u,v) = e^{-\alpha \cdot l(u,v)}$ is $0.819$. The success probability for entanglement swapping was configured to $0.9$. Our simulations, on average, ran for 200,000 time slots, and the results are averaged over $10$ trials for each outcome. Given that the entanglement lifetime can last for $10$ seconds [25] and both SEER and REPS treat one time slot as less than one second, we establish the default entanglement lifetime as $10$ time slots. In Q-value updating phase of Q-learning, we use learning rate of $\beta = 0.1$ and discount factor of $\gamma = 0.95$.

We integrated the proposed solutions into REPS and SEER algorithms and evaluated the performance against original REPS and SEER algorithms. First, we compare them against simple entanglement caching solution (SEER-EC and REPS-EC) that only stores unused entanglements for up to $10$ time slots. We next compare them against various implementations of Proactive Entanglement Swapping (PES); heuristic (SEER-PES-Heuristic and REPS-PES-Heuristic), Q-learning (SEER-PES-QRL and REPS-PEG-QRL) and Deep Q-learning (SEER-PES-DQRL and REPS-PES-DQRL) models.

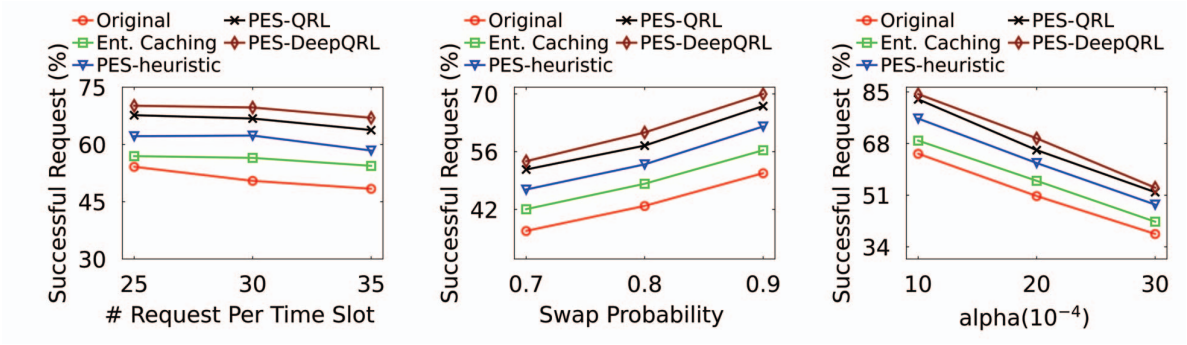
### B. Evaluation Results

*1) Request Count:* Figures 5(a) and 6(a) illustrate the impact of the number of new requests on SEER and REPS algorithms, respectively. As the number of requests increase, we observe an increase in the performance for both algorithms. While the original SEER implementation attains less than $50\%$, Deep Q Learning based Proactive Entanglement Swapping (PES-DeepQRL) can succeed more than 70% of requests. Caching unused entanglements and proactively entangling links (i.e., no swapping)(Ent. Caching) improves the performance of SEER by 22.34%. the performance improvement reached to 41.79% when entanglement caching is complemented with proactive entanglement swapping via Q-learning (PES-QRL).

For REPS, the performance improved by around 18.34% with entanglement caching alone (Figure 6(a)). Q-learning based proactive entanglement swapping increases the request success rate by 52.55%. Unlike SEER which finds only one path for each connection (one for source to intermediate and one for intermediate to destination), REPS identifies multiple paths, allowing proactive entanglement swapping to attain better performance.

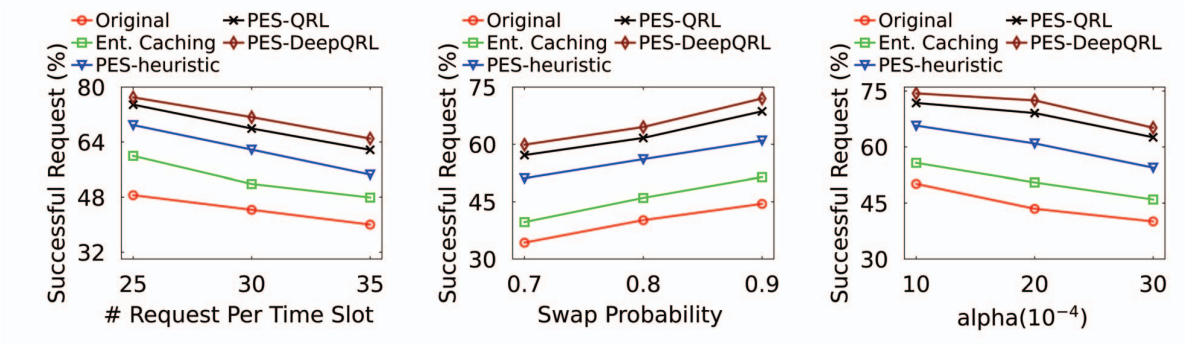
*2) Swap Probability:* The success of quantum entanglement routing algorithms is significantly influenced by the probability of successfully swapping entangled links. As depicted in Figure 5(b), an increase in the swap probability correlates with improved performance for SEER algorithm. Proactive entanglement leads up to to 55% improvement over the original SEER algorithm. Similarly, in Figure 6(b), we observe that higher swapping probabilities leads up to 54% improvement in REPS algorithm. We observe 18% improvement with entanglement caching alone. These findings emphasize the notable advantages of our proactive entanglement swapping strategy, particularly in scenarios with higher swap probabilities.

*3) Entanglement Generation Probability:* The probability of successful entanglement generation affects the performance of all algorithms as expected. Figure 5(c) and 6(c) show that as the success rate of entanglement reduces, the rate of successful requests also decreases. Caching entangled links gives more benefit when the entanglement success probability is low. The entanglement caching approach boosts REPS by 13.47% for low entanglement generation probability values and 9.74% for higher entanglement generation rates. For SEER, entanglement caching (Ent. Caching) with lower entanglement probability results in 10.65% improvement over baseline. On the other hand, the improvement reduces to 6.74% with high entanglement probability. Reduced improvement can be attributed to the fact that if entanglement generation and swapping is more




(a) Number of requests (b) Swap probability (c) Entanglement probability

Fig. 5. Performance analysis of entanglement caching and proactive entanglement generation on SEER quantum routing algorithm.




(a) Number of requests (b) Swap probability (c) Entanglement probability

Fig. 6. Performance analysis of entanglement caching and proactive entanglement generation on REPS quantum routing algorithm.

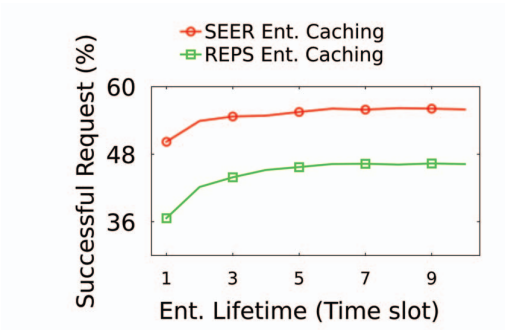



Fig. 7. Impact of entanglement lifetime on the performance of SEER and REPS algorithms with entanglement caching.

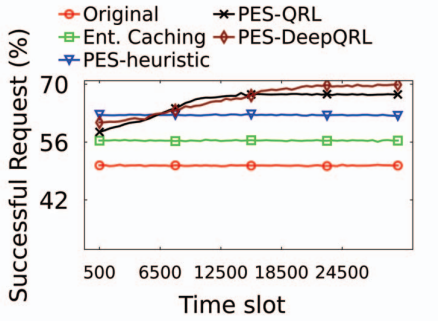



Fig. 8. Rate of successful request over time increases with Q reinforcement learning. It takes around 13,000 time slots for Q-learning and 20,000 time slots for Deep Q-learning to converge.

likely to be successful in the first attempt, then caching or proactive entanglement swapping would not be as useful.

*4) Entanglement Lifetime:* Our entanglement caching strategy's efficacy relies on the duration of entanglement. When entanglement lasts for only 1 time slot, no entangled link can be retained in the cache. However, when entanglement persists for more than 1 time slot, it significantly enhances the performance of both SEER and REPS algorithms. We illustrate how entanglement duration impacts the performance of both algorithms with entanglement caching strategy is applied in Figure 7. For SEER an entanglement duration of two time slots improves performance by 4%. The improvement reaches to 12% when entanglement lifetime is set to six time slots. For REPS, a duration of two time slots enhances the performance by 7%. The improvement rate increases to 19.21% when the entanglement lifetime is set to six or more time slots. The success rate reaches the maximum after six time slots both for SEER and REPS algorithms, indicating that they use all cached entangled links within six time slots.

### C. Performance Analysis of Q-Learning and Deep Q-Learning

Figures 5 and 6 show that Q Reinforcement Learning (QRL) outperforms the heuristic model as it attains 6-10% higher success rate for SEER and REPS algorithms. Moreover,

DeepQRL outperforms the heuristic solution by 14-18% and QRL by $8-10\%$. We attribute the performance gain of DeepQRL over QRL to its ability to capture more information in state representation. That it, QRL only considers segments as states whereas DeepQRL can incorporate network state and available requests.

Please note that both DeepQRL and QRL requires training of reinforcement learning agent. Figure 8 shows that it takes at around 13,000 time slots for QRL to converge to its optimal performance. DeepQRL, on the other hand, reaches to its peak performance at around 20,000 time slots. For DeepQRL, 400-500 training instances were generated in every time slot, thus it uses 8M - 10M data points to converge. This can be attributed to the fact that DQRL uses more complex state compared to QRL, thus it demands more training data to converge.

## V. Conclusion

In this paper, we introduced entanglement caching and proactive entanglement swapping methods to enhance the performance of quantum routing algorithms. Taking advantage of longevity of entanglement lifespan compared to connection intervals, we implemented entanglement caching to store unused entangled links. Additionally, we introduced proactive entanglement swapping to entangle commonly used segments to increase the connection success rates. Due to limited capacity and dynamic nature of requests, we implemented reinforcement learning algorithms to select the network segments. We incorporated the proposed methods into two quantum network routing algorithms (SEER and REPS) and showed that the proposed caching and proactive entanglement swapping (using Deep Q-Learning) contributions improves the successful connection rates in REPS and SEER algorithms by $61\%$ and $48\%$, respectively.

## References


[1] C. H. Bennett and G. Brassard, "Quantum cryptography: Public key distribution and coin tossing," *Theoretical computer science*, vol. 560, pp. 7–11, 2014.

[2] A. K. Ekert, "Quantum cryptography based on bell's theorem," *Physical review letters*, vol. 67, no. 6, p. 661, 1991.

[3] T. Islam and E. Arslan, "Quantum key distribution with minimal qubit transmission based on multiqubit greenberger horne zeilinger state," *arXiv preprint arXiv:2305.12725*, 2023.

[4] H. Buhrman and H. Röhrig, "Distributed quantum computing," in *International Symposium on Mathematical Foundations of Computer Science*. Springer, 2003, pp. 1–20.

[5] A. S. Cacciapuoti, M. Caleffi, F. Tafuri, F. S. Cataliotti, S. Gherardini, and G. Bianchi, "Quantum internet: Networking challenges in distributed quantum computing," *IEEE Network*, vol. 34, no. 1, pp. 137–143, 2019.

[6] D. Cuomo, M. Caleffi, and A. S. Cacciapuoti, "Towards a distributed quantum computing ecosystem," *IET Quantum Communication*, vol. 1, no. 1, pp. 3–8, 2020.

[7] R. Beals, S. Brierley, O. Gray, A. W. Harrow, S. Kutin, N. Linden, D. Shepherd, and M. Stather, "Efficient distributed quantum computing," *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 469, no. 2153, p. 20120686, 2013.

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

[9] A. Yimsiriwattana and S. J. Lomonaco Jr, "Distributed quantum computing: A distributed shor algorithm," in *Quantum Information and Computation II*, vol. 5436. SPIE, 2004, pp. 360–372.

[10] V. Giovannetti, S. Lloyd, and L. Maccone, "Quantum-enhanced positioning and clock synchronization," *Nature*, vol. 412, no. 6845, pp. 417–419, 2001.

[11] I. L. Chuang, "Quantum algorithm for distributed clock synchronization," *Physical review letters*, vol. 85, no. 9, p. 2006, 2000.

[12] M. de Burgh and S. D. Bartlett, "Quantum methods for clock synchronization: Beating the standard quantum limit without entanglement," *Physical Review A*, vol. 72, no. 4, p. 042301, 2005.

[13] J. Rabbie, K. Chakraborty, G. Avis, and S. Wehner, "Designing quantum networks using preexisting infrastructure," *npj Quantum Information*, vol. 8, no. 1, p. 5, 2022.

[14] T. Islam and E. Arslan, "A heuristic approach for scalable quantum repeater deployment modeling," in *2023 IEEE 48th Conference on Local Computer Networks (LCN)*. IEEE, 2023, pp. 1–9.

[15] L. Yang, Y. Zhao, H. Xu, and C. Qiao, "Online entanglement routing in quantum networks," in *2022 IEEE/ACM 30th International Symposium on Quality of Service (IWQoS)*. IEEE, 2022, pp. 1–10.

[16] H. Gu, Z. Li, R. Yu, X. Wang, F. Zhou, and J. Liu, "Fendi: High-fidelity entanglement distribution in the quantum internet," *arXiv preprint arXiv:2301.08269*, 2023.

[17] J. Li, M. Wang, K. Xue, R. Li, N. Yu, Q. Sun, and J. Lu, "Fidelity-guaranteed entanglement routing in quantum networks," *IEEE Transactions on Communications*, vol. 70, no. 10, pp. 6748–6763, 2022.

[18] S. Shi and C. Qian, "Concurrent entanglement routing for quantum networks: Model and designs," in *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication*, 2020, pp. 62–75.

[19] Y. Zhao and C. Qiao, "Redundant entanglement provisioning and selection for throughput maximization in quantum networks," in *IEEE INFOCOM 2021-IEEE Conference on Computer Communications*. IEEE, 2021, pp. 1–10.

[20] G. Zhao, J. Wang, Y. Zhao, H. Xu, and C. Qiao, "Segmented entanglement establishment for throughput maximization in quantum networks," in *2022 IEEE 42nd International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2022, pp. 45–55.

[21] S.-M. Huang, M.-H. Chien, C.-Y. Cheng, J.-J. Kuo, and L.-H. Yang, "Socially-aware concurrent entanglement routing with path decomposition in quantum networks," in *GLOBECOM 2022-2022 IEEE Global Communications Conference*. IEEE, 2022, pp. 3664–3669.

[22] H. Bartling, M. Abobeih, B. Pingault, M. Degen, S. Loenen, C. Bradley, J. Randall, M. Markham, D. Twitchen, and T. Taminiau, "Entanglement of spin-pair qubits with intrinsic dephasing times exceeding a minute," *Physical Review X*, vol. 12, no. 1, p. 011048, 2022.

[23] A. Reiserer, N. Kalb, M. S. Blok, K. J. van Bemmelen, T. H. Taminiau, R. Hanson, D. J. Twitchen, and M. Markham, "Robust quantum-network memory using decoherence-protected subspaces of nuclear spins," *Physical Review X*, vol. 6, no. 2, p. 021040, 2016.

[24] M. Pompili, S. L. Hermans, S. Baier, H. K. Beukers, P. C. Humphreys, R. N. Schouten, R. F. Vermeulen, M. J. Tiggelman, L. dos Santos Martins, B. Dirkse *et al.*, "Realization of a multinode quantum network of remote solid-state qubits," *Science*, vol. 372, no. 6539, pp. 259–264, 2021.

[25] C. E. Bradley, J. Randall, M. H. Abobeih, R. Berrevoets, M. Degen, M. A. Bakker, M. Markham, D. Twitchen, and T. H. Taminiau, "A ten-qubit solid-state spin register with quantum memory up to one minute," *Physical Review X*, vol. 9, no. 3, p. 031045, 2019.

[26] B. Li, T. Coopmans, and D. Elkouss, "Efficient optimization of cutoffs in quantum repeater chains," *IEEE Transactions on Quantum Engineering*, vol. 2, pp. 1–15, 2021.

[27] L. Le, T. N. Nguyen, A. Lee, and B. Dumba, "Entanglement routing for quantum networks: a deep reinforcement learning approach," in *ICC 2022-IEEE International Conference on Communications*, 2022.

[28] C. J. C. H. Watkins, "Learning from delayed rewards," 1989.

[29] A. K. Gupta, K. G. Smith, and C. E. Shalley, "The interplay between exploration and exploitation," *Academy of management journal*, vol. 49, no. 4, pp. 693–706, 2006.

[30] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[31] B. M. Waxman, "Routing of multipoint connections," *IEEE journal on selected areas in communications*, vol. 6, no. 9, pp. 1617–1622, 1988.