Quantum Information Processing (2024) 23:89
https://doi.org/10.1007/s11128-024-04287-z

Routing in quantum communication networks using reinforcement machine learning

Jan Roik[1] · **Karol Bartkiewicz**[2] · **Antonín Černoch**[3] · **Karel Lemr**[1]

Received: 21 January 2023 / Accepted: 26 January 2024 / Published online: 4 March 2024

Abstract
This paper promotes reinforcement machine learning for route-finding tasks in quantum communication networks, where, due to the non-additivity of quantum errors, classical graph path or tree-finding algorithms cannot be used. We propose using a proximal policy optimization algorithm capable of finding routes in teleportation-based quantum networks. This algorithm is benchmarked against the Monte Carlo search. The topology of our network resembles the proposed 6G topology and analyzed that quantum errors correspond to typical errors in realistic quantum channels.

Keywords Routing in quantum networks · Reinforcement learning · Proximal policy optimization · Entanglement swapping

1 Introduction

Efficient communication has played a crucial role in the evolution of all civilizations since antiquity [1, 2]. As the evolution continued, our society began to globalize,

✉ Karel Lemr
k.lemr@upol.cz

Jan Roik
jan.roik@upol.cz

Karol Bartkiewicz
bark@amu.edu.pl

Antonín Černoch
acernoch@fzu.cz

1 Joint Laboratory of Optics of Palacký University and Institute of Physics AS CR, Faculty of Science, Palacký University in Olomouc, 17. listopadu 50A, 771 46 Olomouc, Czech Republic

2 Institute of Spintronics and Quantum Information, Adam Mickiewicz University, Uniwersytetu Poznańskiego 2, 61-614 Poznan, Poland

3 Joint Laboratory of Optics of PU and IP AS CR, Institute of Physics of the Czech Academy of Sciences, 17. listopadu 50A, 771 46 Olomouc, Czech Republic

and so have our communications needs, ultimately leading to the introduction of the Internet [3]. Nowadays, even these classical communication networks seem outdated, facing the development in the field of quantum communications [4]. The first proposed quantum communications protocols were designed for a one-to-one quantum key distribution (QKD) [5–7]. Subsequent strategies to encompass more parties have been proposed [7, 8]. One promising approach is the so-called teleportation-based quantum networks facilitated by standard or controlled teleportation [9, 10]. The idea is to concatenate many teleportation-based cells into a large global network, i.e., the quantum Internet [11]. Many studies have discussed this concept's potential [12–14], the network's possible topologies [15], platforms to realize it on [16], and fundamental problems to overcome [17].

Quantum networks, however, aim beyond mere QKD, which we must consider when designing quantum networks. The major problem that needs to be addressed is finding an efficient method for optimal dynamic routing in these large-scale quantum networks. It seems that teleportation (entanglement swapping) is, for now, the best method for establishing connections between distant parties in quantum networks [18]. Also, note that entanglement swapping is the core ingredient for quantum repeaters [19] and relays [20], allowing combating unfavorable scaling of losses. The unique features of quantum information prevent reliably employing classical tools such as shortest path and tree search algorithms [21]. This paper provides solutions to the routing problem in teleportation-based networks using reinforcement machine learning. The connection between two distant parties (Alice and Bob) in these networks is established by repeated use of entanglement swapping by several intermediate nodes resulting in an entangled state $\hat{\phi}$ shared by the above-mentioned parties, Alice and Bob (see Fig. 1) [22]. Once they share an entangled state, Alice and Bob are free to use it for secret key sharing [23], quantum state teleportation [24] or dense coding [25].

Consecutive entanglement swapping in practical quantum networks will necessarily result in entanglement decay. It is therefore imperative to employ methods that minimize such an adverse effect. A recipe is provided in this paper in the form of reinforcement machine learning that efficiently searches for the best entanglement-preserving route possible between two given nodes. We benchmark our method against a naive Monte Carlo search and show that reinforcement learning performs considerably better in terms of resulting entanglement quality as well as in searching speed. Moreover, we present two quantum-specific examples where intelligent routing allows restoring a partially decayed entanglement (case of amplitude damping and correlated phase noise).

There are many possibilities to quantify the quality of the repeated entanglement swapping and the quality of the shared entangled state between Alice and Bob [26, 27]. We chose the singlet fraction F as the figure of merit because for bipartite entangled states, F can be directly used to evaluate the usefulness of $\hat{\phi}$ for quantum teleportation [28]. Singlet fraction

$$F(\hat{\phi}) = \max_{|\psi\rangle} \langle\psi|\hat{\phi}|\psi\rangle, \tag{1}$$

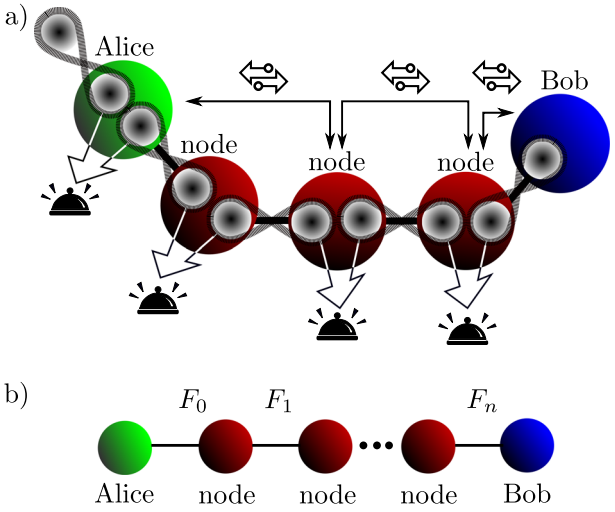

Fig. 1 Schemes represent entanglement swapping from the initial to the final point in the quantum communications network. **a** Each node possesses pair of particles belonging to two different entanglement states symbolized by the two black balls inside the nodes, black line between nodes depicts a quantum channel, green node symbolizes the initial point "Alice", red balls depict intermediate nodes used for entanglement swapping, and blue ball represents the final point "Bob". Bell icon mark where Bell measurement takes place and swap icon highlights where entanglement swapping is carried out. **b** Characterization of the road between the initial and final points where $F_0, \ldots, F_n$ are singlet fractions shared by two neighboring nodes (Color figure online)

is defined as the maximal overlap of the investigated state $\hat{\phi}$ with any maximally entangled state $|\psi\rangle$. Maximal achievable teleportation fidelity f of a qubit state is then calculated as:

$$f = \frac{2F + 1}{3}. \tag{2}$$

One can naively think that the singlet fraction of the final state shared by Alice to Bob F_{AB} is obtained as a product of singlet fractions of the entangled states introduced in the repeated n entanglement swappings (see Fig. 1)

$$F_{AB} = \prod_{i=0}^{n} F_i, \tag{3}$$

alternatively, one can establish an effective *distance* d between Alice and Bob using a logarithm of the singlet fraction

$$d = -\log F_{AB} = -\sum_i \log F_i. \tag{4}$$

However, this is not generally true. For example, in cases of amplitude damping [29] or correlated phase noise [30], errors can cancel each other out. If Eqs. (3) and (4)

were to hold, one should be able to assign a single quantifier to each quantum channel between nodes and use any graph path or tree-finding algorithm such as the Dijkstra algorithm and find the route minimizing the distance d [31]. As we show later in this paper, this yields suboptimal solutions. Note that even prominent dynamic algorithms such as Bellman–Ford [32] and A* [33] cannot handle these types of errors.

One possible solution capable of handling quantum effects is a brute force in the form of the Monte Carlo algorithm. The only downside is Monte Carlo's exponential scaling with the number of nodes. Such a scaling becomes a game stopper, especially in the case of an evolving network where it needs to be repeatedly executed. Hence, a smarter strategy needs to be adopted. In this paper, we propose using the proximal policy optimization (PPO), an artificial intelligence-based algorithm, developed to solve complex evolving problems [34]. This algorithm is commonly used in the gaming industry, where we found inspiration for how to approach the routing problem. We designed our network as a map in a game for the agent to play, intending to find the optimal path through the quantum network. We compare the performance of the PPO against the Monte Carlo and the Dijkstra algorithm demonstrating PPO's virtues.

2 Quantum network topology

We found the inspiration for our network topology in the low-density parity-check code structure, one of the possible topologies considered for designing the 6 G networks. For the details on the topology, see Fig. 2 [35]. This network simulates a real-world scenario where several local users form groups connected among themselves by central nodes. We chose this particular topology mainly due to its robustness against local connection problems, contributing to steady performance. In case of random malfunction in any specific node, this topology offers several possible reroutes to ensure stability. Each connection in the network structure represents a quantum channel using which two neighboring nodes share an entangled two-qubit state. For simplicity, we limit the network topology to a maximum of 4 connections per node. Moreover, each node can perform entanglement swapping, i.e., Bell measurement. All shared entangled states are fully characterized by their density matrices. This representation allows us to fully describe how noisy or damaged each connection is. We can easily simulate different sources of disturbance, such as white noise in the channel or amplitude damping. For an overview of the initialization part, see Pseudocode I. These essential characteristics enable us to simulate various scenarios in the communications network that we later present in the Results section.

Pseudocode I: Initialization of the quantum network

define nodes [N] // *vector of length N*
define connections [N;4] // *matrix: N nodes* $\times$ *max.* 4 *connections*
define shared_states [N;4;4;4] // *matrix: defines* 4×4 *(two-qubit) density matrix of shared state per connection; for density matrices, see Eqs.* (5–7)

The ultimate goal is to distribute entangled state between Alice and Bob. As mentioned in the previous section, the quality of this state is given in terms of singlet fraction F_{AB}, which we maximize. We cast this task as a "game" for the tested algorithms to play. The final reward is received in proportion to F_{AB}. Every connection can be used in each game only once because the used entangled pair is consumed in entanglement swapping. We choose the initial and final users' positions so that the agent can successfully connect them in a given number of actions. The actions count is further used to compare the agent performance because it represents the consumption of resources (i.e., computational time and entangled pairs). We made the routing in the network realistic by ensuring that even the unperturbed connections have a singlet fraction of the distributed state $F = 0.99$ by adding a corresponding amount of white noise, forcing the PPO algorithm toward the shortest path solutions. To represent white noise, we model the shared entangled states in the form of Werner states:

$$\hat{\rho}_w = p|\psi^-\rangle\langle\psi^-| + (1-p)\hat{1}/4. \tag{5}$$

$|\psi^-\rangle = (|01\rangle - |10\rangle)/\sqrt{2}$ represents singlet Bell state, $\hat{1}/4$ stands for the maximally mixed state, and p is the mixing parameter. Amplitude damping, on the other hand, is represented by generalizing the Bell states $|\psi^-\rangle$ to

$$|\psi_g^-(\theta)\rangle = \cos(\theta)|01\rangle - \sin(\theta)|10\rangle, \tag{6}$$

where $\theta \in [0; \frac{\pi}{2}]$ is the damping parameter. Lastly, an arbitrary phase shift can be described as:

$$|\psi_s^-(\phi)\rangle = (|01\rangle - e^{i\phi}|10\rangle)/\sqrt{2}, \tag{7}$$

where $\phi \in [0; \pi]$ is the phase shift parameter and, if uncompensated and random, renders the state shared between Alice and Bob effectively mixed.

3 Routing algorithms

We tested different algorithms capable to solve routing in quantum networks and compared their performance. Namely, we tested the PPO, Dijkstra algorithm, and Monte Carlo method on the quantum communications network (see Fig. 3). PPO is a policy gradient method for reinforcement learning, which uses multiple epochs of stochastic gradient ascent to perform each policy update. It is well known for the simplicity of implementation to various problems and overall performance compared to similar family algorithms. We use stable baseline 3 framework [36] and its implementation of the PPO in our work.

The PPO agents starts at Alice's node. It can choose from at most four actions corresponding to the maximum number of connections any node can have. If the agent chooses an invalid action (i.e., a non-existing connection), the game ends with a negative reward. If a valid link is selected, the agent moves to the node connected by the chosen connection (action). At this point, entanglement swapping is implemented,

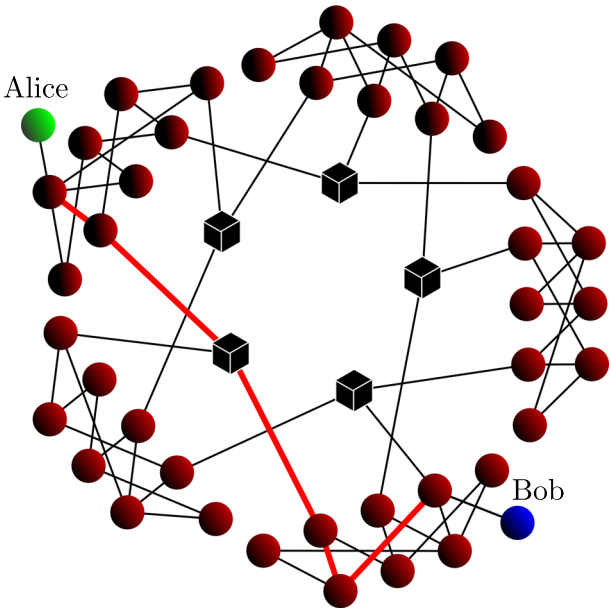

Fig. 2 This figure shows a visualization of the quantum communications network. The entangled photon pair marked "Alice" represents the initial position for our agents, and blue circle named "Bob" marks the ending point of the route. Full black lines set possible routes for entanglement swapping, red thick lines highlight the optimal solution under ideal conditions, and black cubes represent primary connection nodes between local clusters (Color figure online)

leading to a shared entangled state between Alice and the connected node. Selecting action and implementing entanglement swapping constitutes one action. A maximum of 15 actions limits the agent; if depleted, the game ends. The preliminary reward is calculated at the end of each action using the formula:

$$R_p = F_{A_i} - F_{A_{i-1}}. \tag{8}$$

where F_{A_i} stands for the singlet fraction of the newly established entangled state, while $F_{A_{i-1}}$ is the singlet fraction resulting from entanglement swapping in the preceding action ($F_{A_0} = 1$ in case of the first action). We tuned the n-steps hyperparameter of the PPO according to the complexity of the designed quantum network topology. Note that the n-steps hyperparameter determines the number of actions the agent takes before updating the parameters of its policy. We kept all other hyperparameters in default values because we did not notice significant changes when tuning them. It is the reward function structure that has the most noticeable influence on the agent's performance. We save the PPO's policy after every 100–5000 games based on the scenarios' complexity. If the agent reaches the final destination (Bob), it receives a final reward

$$R = 100 F_{AB}. \tag{9}$$

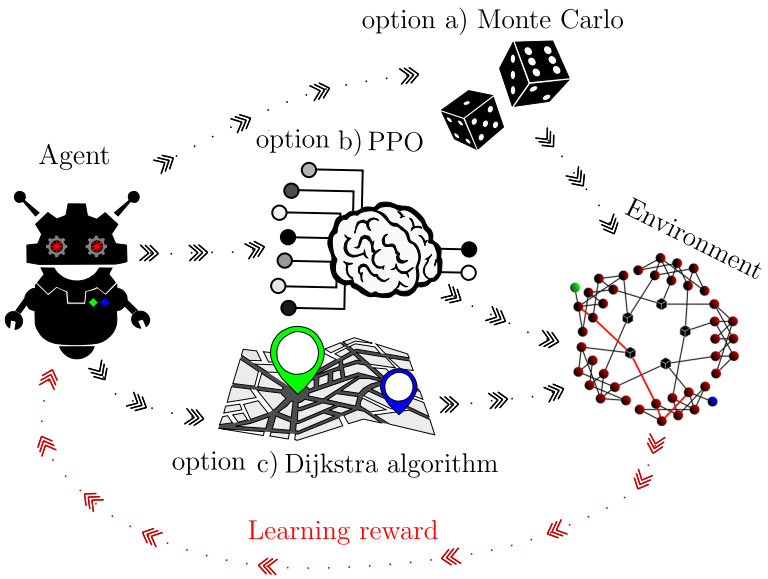

Fig. 3 The aim is to identify the optimal approach toward route finding in the quantum networks (i.e., environment). The agent can choose from three unique algorithms Monte Carlo, Dijkstra, and PPO. Ultimately, we can compare performance and various amounts of consumed resources and thus identify the most suitable candidate for solving a given task

In case of the Monte Carlo algorithm, we applied the same game rules as for the PPO. So, we can obtain a straightforward comparison. The only difference is that Monte Carlo chooses its actions randomly in each game with no intelligent policy. For the overview of the route-finding algorithm, see Pseudocode II. When using the PPO algorithm, the policy predicting neural network of the PPO algorithm is used to pick the connection in a given state (see **pick** connection in the Pseudocode II). Based on the calculated reward (see **calculate** preliminary reward in the Pseudocode II), the critic neural network of the PPO algorithm estimates the advantage and from it also the loss function. Subsequently, both the aforementioned neural networks of the PPO algorithm are updated using the back-propagation of the loss function gradient. Detailed working principle of the PPO algorithm itself is provided in "Appendix".

Dijkstra's algorithm, on the other hand, needs more information and the data structure of the task. Unlike the previous agents, it needs to know the exact topology of the communications network ahead as well as information about each connection. Therefore, the Dijkstra algorithm does not operate under the same conditions as the previously mentioned agents. At the expense of requiring all the information, it is very efficient at finding distance d from Alice to Bob. A brief description on the working principle of the PPO and Djikstra algorithms is presented in "Appendix", and the entire Python code is available as Digital Supplement.

Pseudocode II: Route-finding

```
set agent_node = Alice
set agent_state = singlet // initial state held by the agent
actions_used = 0
repeat:
    actions_used = actions_used + 1
    pick connection // from current node to a next one
        // PPO initially randomly, subsequently based on learned policy
        // Monte Carlo always randomly
    update agent_state // execute entanglement swapping on present state held by
        the agent using density matrix of chosen connection
    calculate singlet fraction of agent_state
    calculate preliminary reward // see Eq. (8)
    set agent_node = next node // depending on the connection chosen
until: agent_node = Bob
    or actions_used >15
    or no available connections exist at agent_node calculate final reward // see Eq. (9)
```

4 Results

Firstly, we investigate routing in a quantum network burdened solely by white noise. This scenario is close to the classical network because white noise is additive and cannot be compensated. In a quantum network, however, other types of errors can occur. As examples of such errors, we consider amplitude damping and correlated phase noise, which we investigate in the second and third subsections. Finally, a dynamically evolving network noise is considered in the last subsection.

4.1 Network affected by white noise

We start with a completely operational network (see the first topology in Fig. 9 in Appendix). A singlet fraction $F = 0.99$ characterizes all connections. Optimal routing through this network between Alice and Bob involves 6 intermediary nodes. Then, we started introducing damaged connections (i.e., connections with $F = 0.6$), thus increasing the number of the intermediary nodes (8,10,12,14,16) required for finding the optimal solution. The optimal routing paths, under those circumstances, are shown in Fig. 9 as well. The performances of the three agents (PPO, Monte Carlo, Dijkstra) are summarized in Table 1. The results show that the Monte Carlo method performs worst than PPO even in the case of the simplest scenario (fully operational network with 6 intermediary nodes to find a solution). The more complex scenarios become, the more prominent the PPO's performance gain is. More specifically, in the case of a network where at least 16 intermediary nodes are required to find a solution, PPO outperforms the Monte Carlo method by a factor of about 13000. For visualization, see Fig. 4. Given the additivity of white noise, the Dijkstra algorithm significantly outperforms both PPO and Monte Carlo in these almost classical scenarios if the complexity surpasses 8 intermediary nodes to complete the task. However, the situation significantly changes with the introduction of purely quantum noise.

Table 1 Results of the three agents applied to the networks of different complexities

Intermediary nodes (count)	6	8	10	12	14	16
	Number of actions					
PPO	780	1280	6880	7680	11,840	41K
Monte Carlo	1758	11,766	259K	683K	11M	546M
Dijkstra	2025					

This complexity is parametrized by the minimal number of intermediary nodes (first row of the table) that need to be visited in order to find a valid routing solution. The number of actions is the average number of actions required to find a solution for a particular topology. Here M = 10^6 and K = 10^3

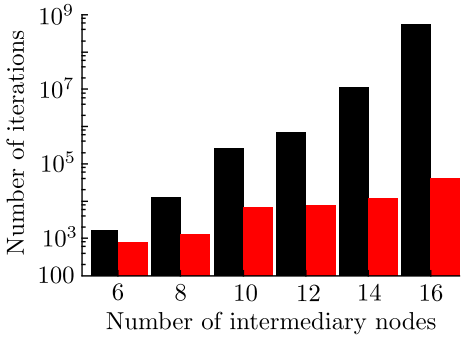

Fig. 4 The graph compares the performance of the PPO algorithm, represented by the red (right) columns, and the Monte Carlo algorithm, depicted by black (left) columns, on different scenarios requiring a given number of passes through intermediary nodes in the quantum networks (Color figure online)

4.2 Network affected by amplitude damping

Amplitude damping, as introduced in Eq. 6, skews the amplitude balance toward one of the two components ($|01\rangle$ or $|10\rangle$). As a result, the singlet fraction decreases. Two connections with mutual opposite component damping can rebalance the amplitudes increasing the singlet fraction (at the expense of overall losses). This feature is intractable by greedy or dynamic algorithms such as Dijkstra, Bellman–Ford, or A*. In order to use those algorithms, one needs to save all preliminary solutions and compare them, which would cause exponential scaling of the algorithm complexity. Ultimately, the agent needs to figure out that in order to complete the task, it needs to find such a route where individual amplitude damping cancels each other out as much as possible. We forced the agent to use this strategy by designing scenarios where the agent must choose at least one amplitude-damped connection to reach the final destination. Moreover, the resulting singlet fraction is maximized when a second (opposite) amplitude-damped connection is chosen by the agent. Similar to the previous subsection, we present the agent with scenarios ranging from 6 to 16 intermediary nodes (see Fig. 10).

The agents' performance is summarized in Table 2 and plotted in Fig. 5. One can notice that due to these complex initial conditions, both Monte Carlo and PPO algorithms require more actions to solve the initial (i.e., 6 intermediary nodes) scenario. However, one can observe a similar difference in scaling between PPO and Monte Carlo as in the previous scenario, i.e., Monte Carlo scales considerably less favorably. The PPO outperforms the Monte Carlo from the beginning, and in the case of the most

Table 2 Results of the two agents applied to the networks of different complexity, including effects such as amplitude damping

Intermediary nodes (count)	6	8	10	12	14	16
	Number of actions					
PPO	1740	13K	14K	15K	54K	60K
Monte Carlo	2730	38K	257K	5.5M	13.7M	546M

This complexity is parametrized by the minimal number of intermediary nodes (first row of the table) that need to be visited in order to find a valid routing solution. The number of actions is the average number of actions required to find a solution for a particular topology. Here M = 10^6 and K = 10^3

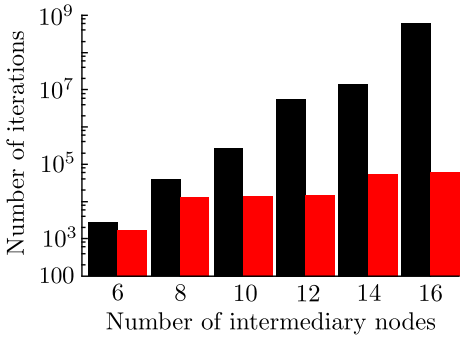

Fig. 5 The graph compares the performance of the PPO algorithm, represented by the red (right) columns, and the Monte Carlo algorithm, depicted by black (left) columns, on different scenarios in the quantum networks where we also introduced connections affected by amplitude damping (Color figure online)

complex scenario, i.e., 16 intermediary nodes, the PPO outperforms the Monte Carlo method by a factor of about 9000.

4.3 Network affected by correlated phase noise

This subsection demonstrates how agents handle another type of reversible damage caused by the correlated phase noise. These scenarios are motivated by one of the practical approaches toward quantum information distribution proposed by Xu et al. [30]. Testing Dijkstra algorithms is again pointless for the reasons we mentioned in the previous subsection. To demonstrate the versatility of the PPO agent, a brand new set of scenarios involving 6–16 intermediary nodes were generated. For more details, see Fig. 11. In the current scenario, the agent starts from the initial node Alice and in the first action, it can only choose from paths damaged by the correlated phase noise. The agent aims to search the network for a suitable path to reverse the initial correlated phase shift. If successful, it must then find the final node, Bob.

Results of this test are shown in Table 3 and plotted in Fig. 6. One can notice that the result once again supports PPO algorithm superiority.

4.4 Evolving quantum network

Ultimately, we test the agents on dynamically evolving scenarios in our quantum network. The agents' goal in this final test is to maximize the overall functionality of

Table 3 Results of the two agents applied to the networks of different complexity, including effects such as correlated phase noise

Intermediary nodes (count)	6	8	10	12	14	16
	Number of actions					
PPO	1320	2700	10K	20K	27K	39K
Monte Carlo	2129	13K	123K	1.4M	10.9M	60M

This complexity is parametrized by the minimal number of intermediary nodes (first row of the table) that need to be visited in order to find a valid routing solution. The number of actions is the average number of actions required to find a solution for a particular topology. Here M = 10^6 and K = 10^3

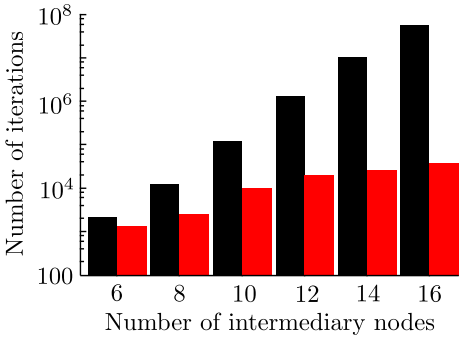

Fig. 6 The graph compares the performance of the PPO algorithm, represented by the red (right) columns, and the Monte Carlo algorithm, depicted by black (left) columns, on different scenarios, staged in the quantum networks where we also introduced connections causing correlated phase noise (Color figure online)

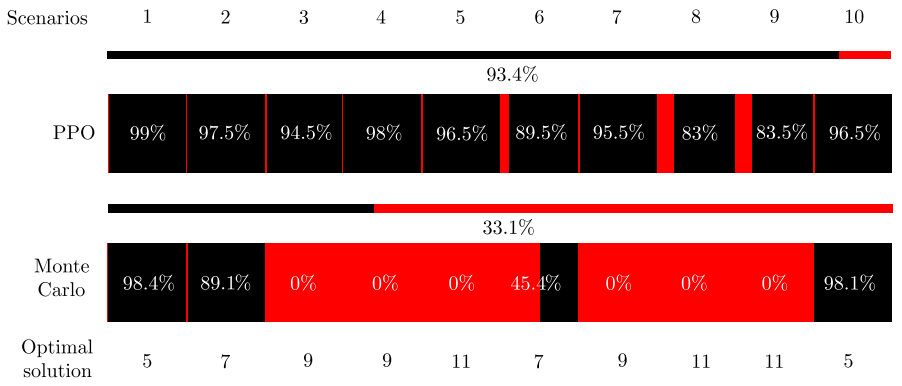

Fig. 7 Illustration of the Agents' performance on the evolving quantum communication network. Thin stripes show the overall functionality of the quantum network throughout its evolution, and the thick stripes show the functionality during each scenario of its evolution

the network throughout the evolutions. These scenarios reflect the realistic behavior of real-world quantum networks where various errors appear at random places and times. The entire routing task lasts for 10^6 actions, during which the quantum network undergoes ten scenarios (i.e., ten events when various connections become damaged or unperturbed). The evolution continues regardless of the agent's success. In this final test, we use all three types of errors discussed in previous chapters, namely white noise, amplitude dumping, and correlated phase noise. To make the interpretation of

the results clear, we set some ground rules. Suppose the agent finds a solution (i.e., a path between Alice and Bob with $F > 0.8$) to the current scenario. In that case, it will use this solution for as long as its singlet fraction remains $F > 0.8$ (i.e., until the scenario evolves). PPO agent at that point also saves its current policy. After the situation evolves, both agents search for a new solution. PPO starts searching from the last saved policy and Monte Carlo randomly from scratch. Each evolution introduces errors so that the previous solution is no longer valid ($F < 0.8$). Hence, agents need to find a new route. This condition does not mimic the natural network behavior, but it is the most extreme case where the PPO agent faces the most disadvantageous conditions. All evolutions of the quantum network are depicted in Fig. 12. Resulting success rates are shown in Fig. 7. From the obtained results, it is clear that if we let agents deal with an undamaged or slightly damaged network (scenarios 1,2,10), both agents can keep the network functional for more than 95% of the time. If the scenario becomes a bit more complex (scenario 6), the PPO agent noticeably outperforms the Monte Carlo agent. For even more complex scenarios, Monte Carlo could not find a solution in a given amount of actions. Due to these poor results, Monte Carlo kept the network functional for 33.1% of the overall time. On the other hand, the PPO found a solution in 10/10 scenarios and kept the network functional for 93.4% of the overall time.

5 Conclusions

This paper compares three different algorithms (PPO, Dijkstra, and Monte Carlo) for route-finding in quantum communication networks. We benchmark these algorithms on various scenarios in a realistic network topology using singlet fraction as the figure of merit. In these scenarios, we introduce additive white noise as well as purely quantum errors such as amplitude damping and correlated phase noise.

We explicitly show that the non-additivity of quantum errors prevents traditional graph path or tree-finding algorithms (Dijkstra) from finding the optimal solution. While the Monte Carlo search allows finding such optimal solutions, its exponential scaling makes its deployment prohibitive in large complex networks. We demonstrate that reinforcement machine learning in the form of the PPO algorithm circumvents the limitations of both aforementioned approaches. It can cope with purely quantum errors and, simultaneously, does not suffer from unfavorable scaling.

Our numerical model reveals that the PPO advantage over mere Monte Carlo search becomes significant when the number of intermediary nodes in the path increases (e.g., for 16 intermediary nodes, PPO outperforms Monte Carlo by a factor of several thousand). Moreover, in a dynamically evolving quantum network, the PPO could maintain an operational route for about 93% of the time, while Monte Carlo for less than 33%.

We believe that our research further promotes reinforcement learning as an invaluable method for improving quantum communications.

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/s11128-024-04287-z.

Acknowledgements The authors thank Jan Soubusta and Jaroslav Cimrman for a fruitful discussion on the topic. Authors thank Cesnet for providing data management services. Authors acknowledge financial support by the Czech Science Foundation under the project No. 20-17765S. AČ and KL acknowledge support by the project OP JAC CZ.02.01.01/00/22-008/0004596 of the Ministry of Education, Youth, and Sports of the Czech Republic and EU. KB also acknowledges the financial support of the Polish National Science Center under Grant No. DEC-2019/34/A/ST2/00081. JR acknowledges support from Palacky University internal Grant DSGC-2021-0026.

Funding Open access publishing supported by the National Technical Library in Prague.

Data availability The programming code (Jupyter notebook) needed to fully reproduce results presented in this paper is appended as Digital supplement.

Declarations

Conflict of interest The authors have no financial or other competing interests.

Appendix

Dijkstra algorithm

The Dijkstra algorithm is designed to find the shortest distance from one node to every other node in the network. The working principle of this algorithm is as follows. In the first step, we choose the initial node, i.e., Alice. Then, the algorithm starts filling a table storing all the information about the distances from the initial node to every other node. Distance is set to infinity if there is a nonexisting direct connection from the initial node. In the next step, it moves from the initial node to the closest node. The distance table is updated by replacing nonoptimal (longer) distances with optimal (shorter) distances from the initial node to every node, assuming the new path passes through the current node. This procedure continues until the algorithm visits all nodes and obtains the optimal path from Alice to all nodes. For an illustration, see Fig. 8 and Table 4.

PPO algorithm

Proximal policy optimization [34] is a type of deep reinforcement learning algorithm developed as a successor to Deep mind [37]. Unlike Deep mind, PPO uses online learning, which means that PPO does not use a replay buffer to store past experiences. Once the batch of experiences has been used to do gradient updates of the policy, these experiences are discarded. PPO also simplified the implementation of the trust region,

Table 4 Description of the evaluation of the Dijkstra algorithm on the illustrative communication network showcased in Fig. 8

Phase	1	2	3	4	5
Unvisited nodes	B,C,D,E	B,D,E	B,D	D	
Current node	A	C	E	D	B
Distance					
A → A	0	0	0	0	0
A → B	9	7	7	7	7
A → C	1	1	1	1	1
A → D	∞	7	4	4	4
A → E	10	2	2	2	2

Here, each phase contains five actions (corresponding to the number of nodes)

which significantly improved the efficiency of the reinforcement learning method. The primary motivation behind the trust region is to limit rapid changes in the policy, thus allowing the agent to train more efficiently. Unlike the original trust region policy optimization TRPO [38], which introduces a rather complex clip function, PPO simplified the implementation of this function, making it much more practical (Figs. 9, 10, 11, 12).

Let us define the PPO policy as

$$L_t^{\text{PPO}}(\theta) = \hat{E}_t \left[L_t^{\text{CLIP}}(\theta) - c_1 L_v^{VF}(\theta) + c_2 S[\pi_\theta](s_t) \right] \tag{10}$$

where L_t^{CLIP} represents the clipped version of the normal gradient objective, L_t^{VF} is squared-error loss responsible for updates of the baseline network, S stand for entropy bonus term, which ensures that the agent does enough exploration during training, and s_t represents the current state. Hyperparameters c_1 and c_2 set the contribution of L_t^{VF} and S to the final policy. The clipped version of the normal gradient objective

$$L_t^{\text{CLIP}}(\theta) = \hat{E}_t \left[\min \left(r_t(\theta) \hat{A}_t; \text{clip}\left(r_t(\theta), 1-\epsilon, 1+\epsilon\right) \right) \hat{A}_t \right] \tag{11}$$

represents the core of the whole algorithm. The expectation operator $\hat{E}_t$ is taken over the minimum of the two terms r_t and clip $(r_t, 1-\epsilon, 1+\epsilon)$ where ϵ ranges from $\{0, 1\}$.

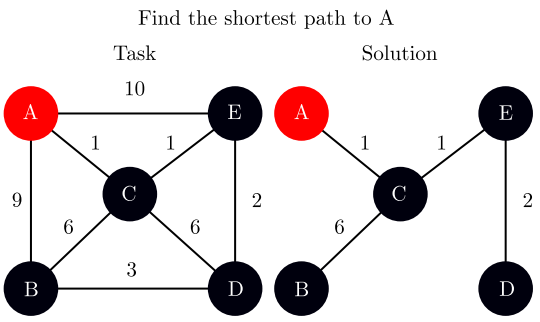

Fig. 8 Depiction of an illustrative communications network where the red circle A marks the initial position and the numbers beside the connections correlate to the distance (Color figure online)

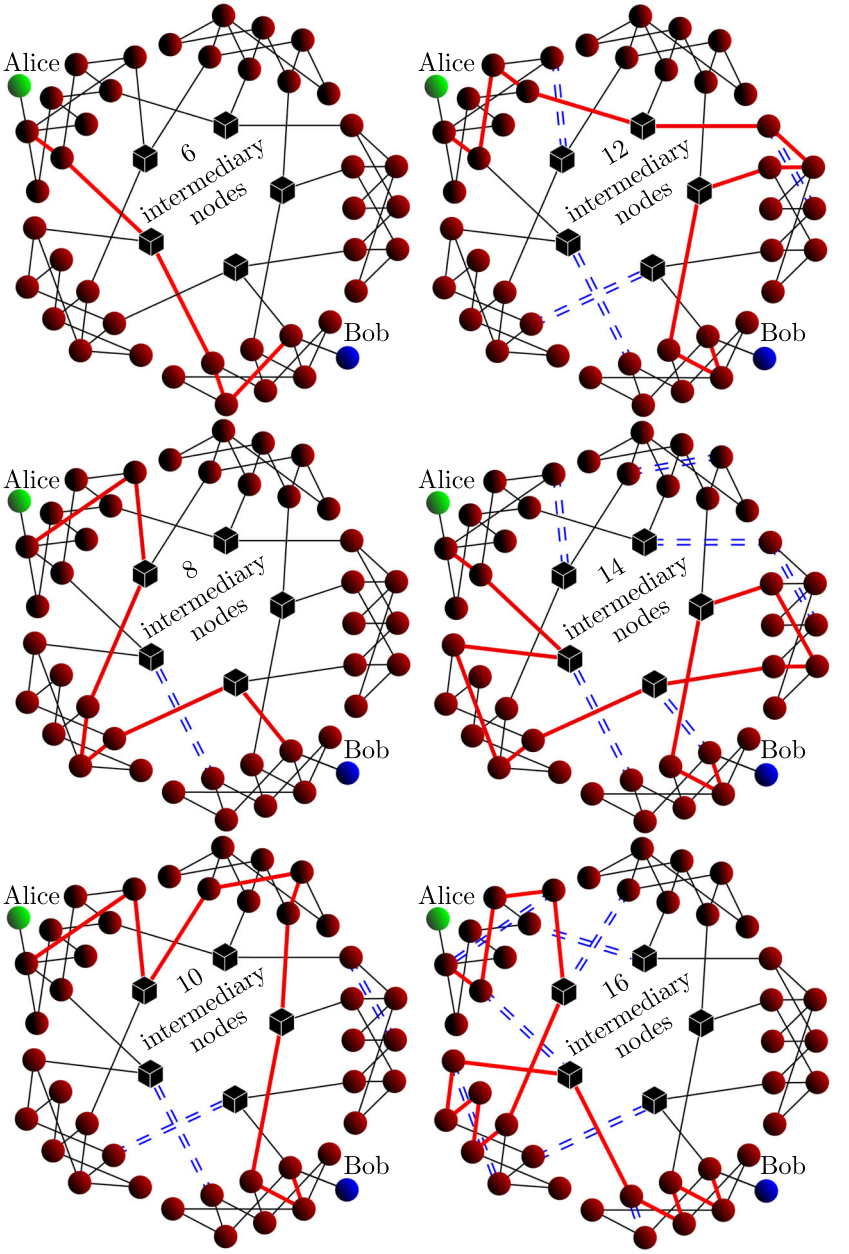

Fig. 9 The figure shows various quantum network scenarios where we consider connections affected by white noise only. We prepared scenarios ranging from simplest solvable using 6 intermediary nodes to complex requiring up to 16 intermediary nodes. The thick red line marks one of the optimal solutions to the presented scenario, and the double blue dashed lines represent irreversibly damaged connections (Color figure online)

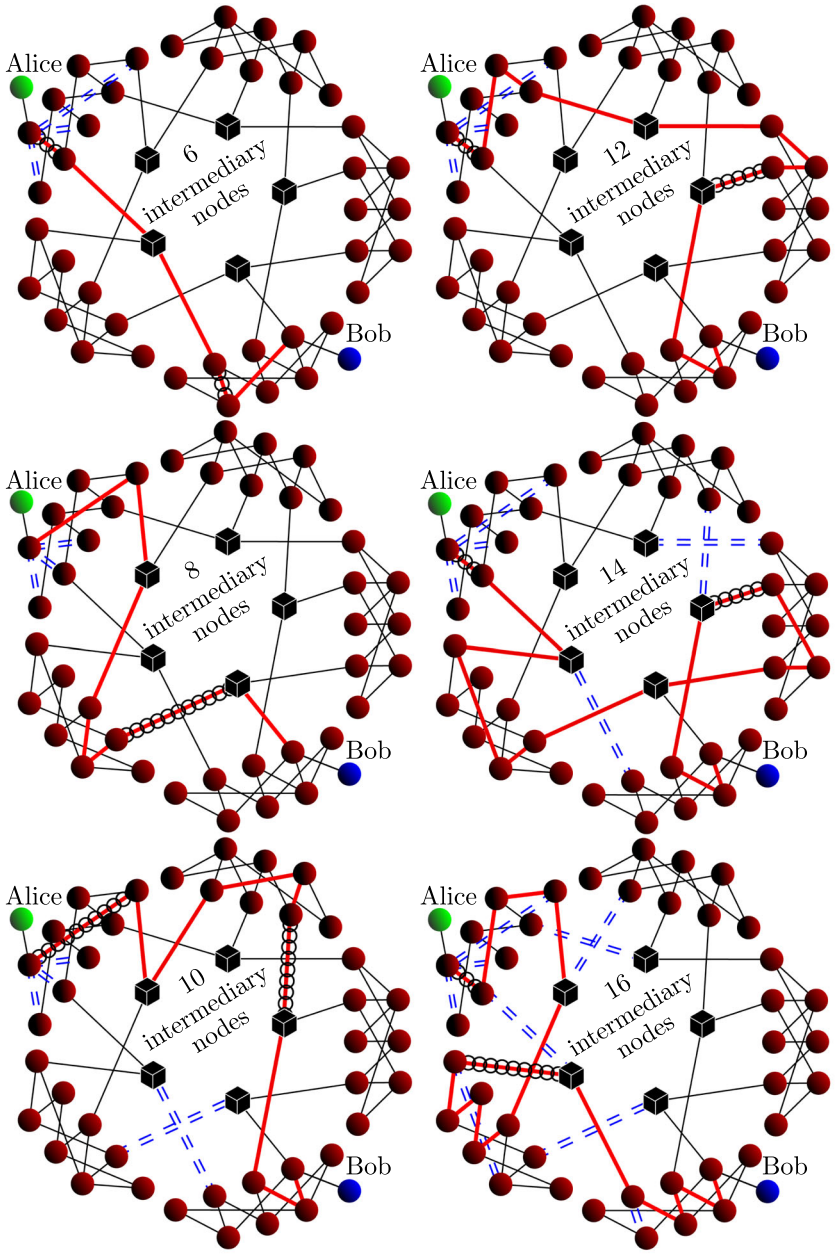

Fig. 10 The figure shows various quantum network scenarios where we consider connections affected by white noise and amplitude dumping. We prepared scenarios ranging from simplest solvable using 6 intermediary nodes to complex requiring up to 16 intermediary nodes. The thick red line marks one of the optimal solutions to the presented scenario, double blue dashed lines represent irreversibly damaged connections, and black chain marks connections cause amplitude damping (Color figure online)

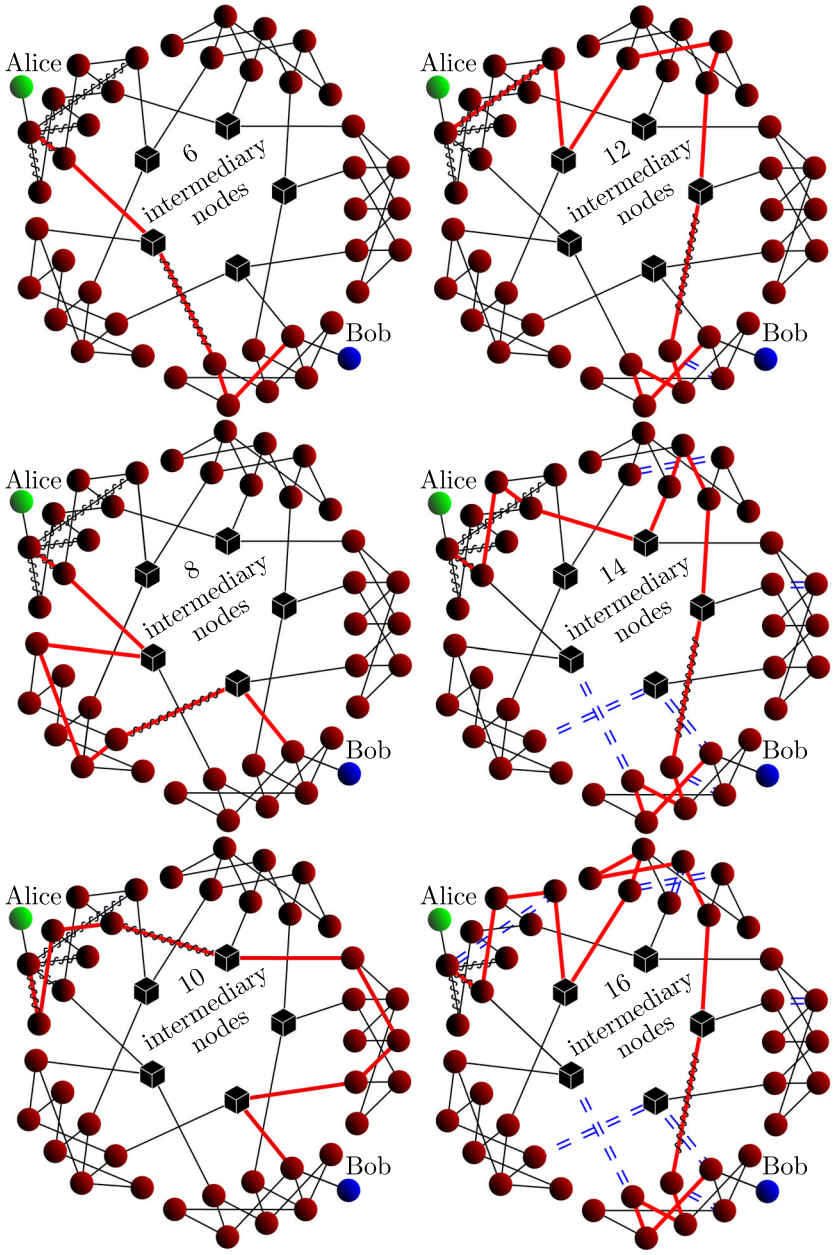

Fig. 11 The figure shows various quantum network scenarios where we consider connections affected by white noise and correlated phase noise. We prepared scenarios ranging from simplest solvable using 6 intermediary nodes to complex requiring up to 16 intermediary nodes. The thick red line marks one of the optimal solutions to the presented scenario, double blue dashed lines represent irreversibly damaged connections, and wrap-around lines mark connections causing correlated phase noise (Color figure online)

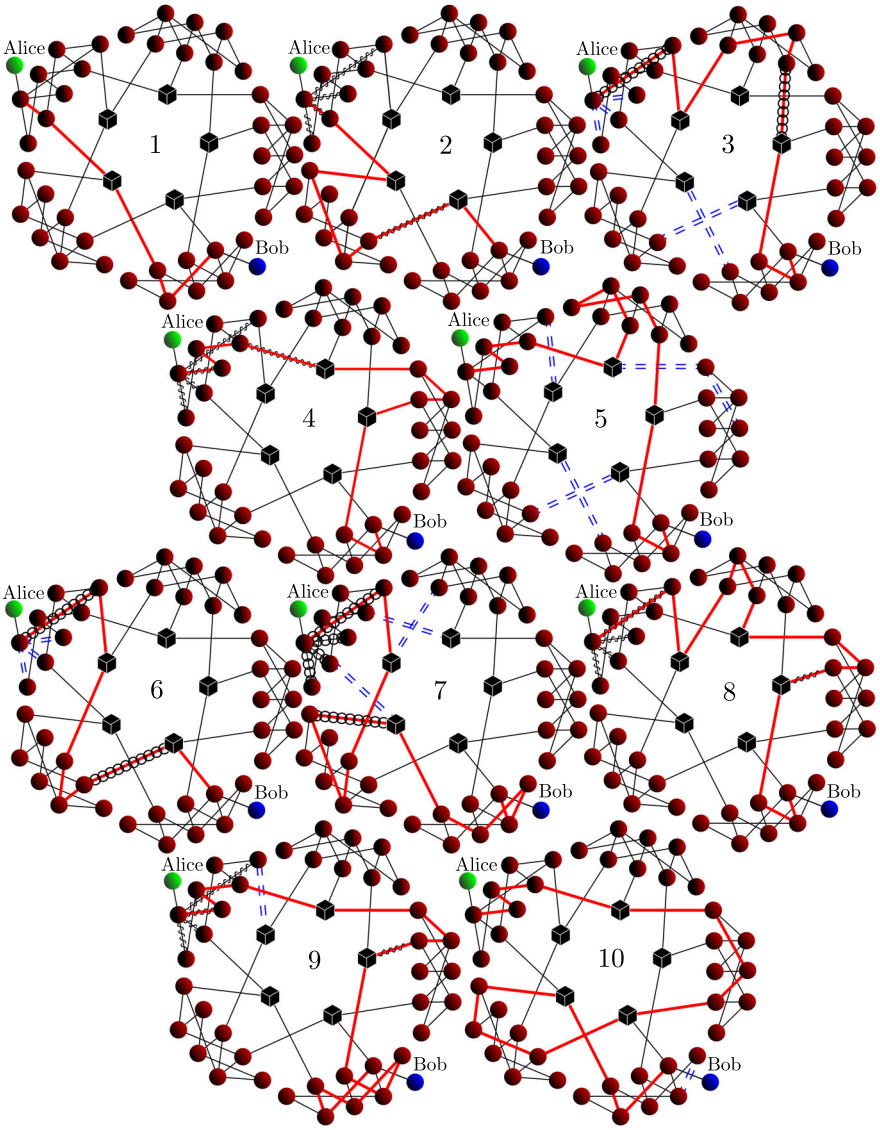

Fig. 12 The figure shows evolving quantum communications network and the response of the PPO agent to those changes. The thick red line marks the PPO's solution in the final iterations before the change in the current scenario. Double-blue dashed lines represent damaged connections, wrap-around lines mark connections causing correlated phase noise and black chain marks connections causing amplitude damping

r_t defines the objective for normal policy gradients, which pushes policy toward actions that yield a higher positive advantage over the baseline. Estimation of the advantage function $\hat{A}_t$ can be positive or negative, which dictates the effect of the min operator. The probability ratio r_t determines the relation between newly updated policy outputs and the previous old version of the network. If the r_t is > 1, the action becomes more likely than it was in the old policy version. On the other hand, if < 1, the action becomes less likely than it was before the last gradient step. Policy π_θ is represented by a neural network fed by observed states of the environment as input and gives suggestions of action as output. $\hat{A}_t$ has two contributors, discounted sum

$$D_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k} \tag{12}$$

and baseline estimation B_t. A discounted sum of rewards D_t is a weighted sum of all rewards r_{t+k} the agent receives during each time step k of the current episode. Parameter γ ranges from 0 to 1 and sets how much the agent values immediate reward r_t over future rewards. Baseline estimation B_t gives an estimate of the $\hat{A}_t$, trying to predict the final return of the episode from the current state. Since B_t is also implemented as a neural network, the result is a noisy value function estimation.

References

1. Miyagawa, S., Ojima, S., Berwick, R.C., Okanoya, K.: The integration hypothesis of human language evolution and the nature of contemporary languages. Front. Psychol. **5**, 564 (2014)
2. Hauser, M.D.: The Evolution of Communication. MIT Press, Cambridge (1996)
3. Leiner, B.M., Cerf, V.G., Clark, D.D., Kahn, R.E., Kleinrock, L., Lynch, D.C., Postel, J., Roberts, L.G., Wolff, S.: A brief history of the internet. ACM SIGCOMM Comput. Commun. Rev. **39**(5), 22–31 (2009)
4. Gisin, N., Thew, R.: Quantum communication. Nat. Photonics **1**(3), 165–171 (2007)
5. Bennett, C.H., Brassard, G.: Quantum cryptography: public key distribution and coin tossing (2020). arXiv:2003.06557
6. Ekert, A.K.: Quantum cryptography and Bell's theorem. In: Tombesi, P., Walls, D. F. (eds.) Quantum Measurements in Optics, pp. 413–418. Springer, Boston (1992)
7. Cao, Y., Zhao, Y., Wang, Q., Zhang, J., Ng, S.X., Hanzo, L.: The evolution of quantum key distribution networks: on the road to the qinternet. IEEE Commun. Surv. Tutor. **24**(2), 839–894 (2022)
8. Hillery, M., Bužek, V., Berthiaume, A.: Quantum secret sharing. Phys. Rev. A **59**(3), 1829 (1999)
9. Bennett, C.H., Brassard, G., Crépeau, C., Jozsa, R., Peres, A., Wootters, W.K.: Teleporting an unknown quantum state via dual classical and Einstein–Podolsky–Rosen channels. Phys. Rev. Lett. **70**(13), 1895 (1993)
10. Gottesman, D., Chuang, I.L.: Demonstrating the viability of universal quantum computation using teleportation and single-qubit operations. Nature **402**(6760), 390–393 (1999)
11. Kimble, H.J.: The quantum internet. Nature **453**(7198), 1023–1030 (2008)
12. Gyongyosi, L., Imre, S.: Advances in the quantum internet. Commun. ACM **65**(8), 52–63 (2022)
13. Wehner, S., Elkouss, D., Hanson, R.: Quantum internet: a vision for the road ahead. Science **362**(6412), 9288 (2018)
14. Singh, A., Dev, K., Siljak, H., Joshi, H.D., Magarini, M.: Quantum internet-applications, functionalities, enabling technologies, challenges, and research directions. IEEE Commun. Surv. Tutor. **23**(4), 2218–2247 (2021)
15. Gyongyosi, L., Imre, S.: Topology adaption for the quantum internet. Quantum Inf. Process. **17**(11), 1–12 (2018)

16. Goodenough, K., Elkouss, D., Wehner, S.: Optimizing repeater schemes for the quantum internet. Phys. Rev. A **103**(3), 032610 (2021)
17. Cacciapuoti, A.S., Caleffi, M., Tafuri, F., Cataliotti, F.S., Gherardini, S., Bianchi, G.: Quantum internet: networking challenges in distributed quantum computing. IEEE Netw. **34**(1), 137–143 (2019)
18. Munro, W.J., Luo, Y.-H., Chen, M.-C., Erhard, M., Zhong, H.-S., Wu, D., Tang, H.-Y., Zhao, Q., Wang, X.-L., Fujii, K., : Teleportation in quantum edge networks (conference presentation). In: Quantum Communications and Quantum Imaging XX, p. 122380. SPIE (2022)
19. Briegel, H.-J., Dür, W., Cirac, J.I., Zoller, P.: Quantum repeaters: the role of imperfect local operations in quantum communication. Phys. Rev. Lett. **81**(26), 5932 (1998)
20. Jacobs, B., Pittman, T., Franson, J.: Quantum relays and noise suppression using linear optics. Phys. Rev. A **66**(5), 052307 (2002)
21. Fu, L., Sun, D., Rilett, L.R.: Heuristic shortest path algorithms for transportation applications: state of the art. Comput. Oper. Res. **33**(11), 3324–3343 (2006)
22. Pan, J.-W., Bouwmeester, D., Weinfurter, H., Zeilinger, A.: Experimental entanglement swapping: entangling photons that never interacted. Phys. Rev. Lett. **80**(18), 3891 (1998)
23. Gisin, N., Ribordy, G., Tittel, W., Zbinden, H.: Quantum cryptography. Rev. Mod. Phys. **74**(1), 145 (2002)
24. Pirandola, S., Eisert, J., Weedbrook, C., Furusawa, A., Braunstein, S.L.: Advances in quantum teleportation. Nat. Photonics **9**(10), 641–652 (2015)
25. Mattle, K., Weinfurter, H., Kwiat, P.G., Zeilinger, A.: Dense coding in experimental quantum communication. Phys. Rev. Lett. **76**(25), 4656 (1996)
26. Horodecki, R., Horodecki, P., Horodecki, M., Horodecki, K.: Quantum entanglement. Rev. Mod. Phys. **81**(2), 865 (2009)
27. Gühne, O., Tóth, G.: Entanglement detection. Phys. Rep. **474**(1–6), 1–75 (2009)
28. Horodecki, M., Horodecki, P., Horodecki, R.: General teleportation channel, singlet fraction, and quasidistillation. Phys. Rev. A **60**(3), 1888 (1999)
29. Fortes, R., Rigolin, G.: Fighting noise with noise in realistic quantum teleportation. Phys. Rev. A **92**(1), 012338 (2015)
30. Xu, J.-S., Yung, M.-H., Xu, X.-Y., Tang, J.-S., Li, C.-F., Guo, G.-C.: Robust bidirectional links for photonic quantum networks. Sci. Adv. **2**(1), 1500672 (2016)
31. Mehlhorn, K., Sanders, P., Sanders, P.: Algorithms and Data Structures: The Basic Toolbox, vol. 55. Springer, Berlin (2008)
32. Bellman, R.: On a routing problem. Q. Appl. Math. **16**(1), 87–90 (1958)
33. Hart, P.E., Nilsson, N.J., Raphael, B.: A formal basis for the heuristic determination of minimum cost paths. IEEE Trans. Syst. Sci. Cybern. **4**(2), 100–107 (1968)
34. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal policy optimization algorithms (2017). arXiv:1707.06347
35. Yuan, Y., Zhao, Y., Zong, B., Parolari, S.: Potential key technologies for 6g mobile communications. Sci. China Inf. Sci. **63**(8), 1–19 (2020)
36. Raffin, A., Hill, A., Gleave, A., Kanervisto, A., Ernestus, M., Dormann, N.: Stable-baselines3: reliable reinforcement learning implementations. J. Mach. Learn. Res. **22**, 12348–12355 (2021)
37. Beattie, C., Leibo, J.Z., Teplyashin, D., Ward, T., Wainwright, M., Küttler, H., Lefrancq, A., Green, S., Valdés, V., Sadik, A., et al.: Deepmind lab (2016). arXiv:1612.03801
38. Schulman, J., Levine, S., Abbeel, P., Jordan, M., Moritz, P.: Trust region policy optimization. In: International Conference on Machine Learning, pp. 1889–1897. PMLR (2015)