Enhancing Routing in SD-EONs through Reinforcement Learning: A Comparative Analysis

Ryan McCann[†]
ryan_mccann@student.uml.edu

Arash Rezaee[†]
arash_rezaee@student.uml.edu

Vinod M. Vokkarane[†]
vinod_vokkarane@uml.edu

[†]*Electrical and Computer Engineering Department, University of Massachusetts Lowell, United States*

2024 IEEE Latin-American Conference on Communications (LATINCOM) | 979-8-3315-2111-0/24/$31.00 ©2024 IEEE | DOI: 10.1109/LATINCOM62985.2024.10770647

***Abstract*—This paper presents an optimization framework for routing in software-defined elastic optical networks using reinforcement learning algorithms. We specifically implement and compare the epsilon-greedy bandit, upper confidence bound (UCB) bandit, and Q-learning algorithms to traditional methods such as K-Shortest Paths with First-Fit core and spectrum assignment (KSP-FF) and Shortest Path with First-Fit (SPF-FF) algorithms. Our results show that Q-learning significantly outperforms traditional methods, achieving a reduction in blocking probability (BP) of up to 58.8% over KSP-FF, and 81.9% over SPF-FF under lower traffic volumes. For higher traffic volumes, Q-learning maintains superior performance with BP reductions of 41.9% over KSP-FF and 70.1% over SPF-FF. These findings demonstrate the efficacy of reinforcement learning in enhancing network performance and resource utilization in dynamic and complex environments.**

***Index Terms*—Software Defined Elastic Optical Networks (SD-EONs), Reinforcement Learning, Artificial Intelligence (AI)**

I. Introduction

The rapid advancement of technology and the exponential growth in data-intensive applications necessitate an in-depth understanding of the evolution and optimization of communication networks. The emergence of Elastic Optical Networks (EONs) and Software Defined Networking (SDN) represents a significant shift in this landscape, offering novel solutions to address the increasing demands for network flexibility and efficiency [1], [2].

Optical networks, particularly those based on Wavelength Division Multiplexing (WDM), have traditionally played a pivotal role in global communication. However, the fixed channel spacings and limited adaptability of WDM systems are increasingly insufficient to meet the dynamic requirements of modern network traffic. In contrast, EONs, with their ability to provide dynamic bandwidth allocation and more granular channel spacings, offer a more efficient utilization of the optical spectrum and enhanced network performance [3].

The integration of Software Defined Networking in EONs (SD-EONs) has further advanced network management capabilities. SD-EONs' separation of control and data planes allows for a more flexible and efficient network configuration, which is crucial for managing the dynamic characteristics of EONs. This integration has led to improved network agility and an enhanced ability to respond to real-time changes in network conditions [4].

Amidst these advancements, traditional routing and spectrum allocation methods in optical networks, such as the first-fit and shortest-path algorithms, have shown limitations in dealing with rapidly changing network scenarios due to their static nature and lack of adaptability to real-time conditions [5]; to address these challenges, this paper explores three reinforcement learning algorithms—epsilon-greedy bandit, Upper Confidence Bound (UCB) bandit, and Q-learning—which span a spectrum of complexity from the straightforward epsilon-greedy bandit, balancing exploration and exploitation through random decisions based on past experiences, to the UCB bandit, using statistical confidence bounds to systematically explore and exploit network configurations, and finally to Q-learning, which incorporates considerations of future potential rewards into its iterative learning process, optimizing routing policies based on both immediate and long-term cumulative rewards in dynamic and stochastic environments [6], [7].

The primary aim of this study is to evaluate whether the increased complexity of the UCB bandit and Q-learning algorithms translates into enhanced performance in optimizing routing in SD-EONs compared to the epsilon-greedy bandit and traditional methods. By assessing their efficiency, adaptability, and scalability against each other and established baselines, this research seeks to provide insights into the trade-offs between algorithmic complexity and performance in modern network traffic scenarios.

The paper is structured as follows: Section II reviews relevant literature on reinforcement learning in optical networking, Section III explains the background and functioning of the epsilon-greedy bandit, UCB bandit, and Q-learning algorithms, Section IV describes the proposed algorithms and their implementation for routing optimization, Section V outlines the system architecture and assumptions, and Section VI presents the experimental results and comparative analysis. Finally, Section VII concludes the paper.

II. Literature Review

EONs, employing technologies like Orthogonal Frequency Division Multiplexing (OFDM) for its spectral efficiency and flexibility, and Flexgrid for future network architectures, dynamically allocate spectrum, advancing high-speed network design beyond traditional static management. However, their reliance on conventional network design principles, rather than artificial intelligence, highlights a growing need for more adaptive, AI-driven approaches to meet the increasing complexity and dynamic demands of modern EONs [8].

Recent advancements in EONs have explored the use of Reinforcement Learning (RL), particularly Q-learning, for network optimization, marking a shift towards AI-based so-

This paper was partially supported by NSF project award #2008530.

lutions. Unlike traditional network management techniques such as integer linear programming and heuristic approaches, RL offers efficient real-time responses to changing network conditions, paving the way for more adaptive and scalable network management solutions. The exploration of these networks extends to experimental demonstrations and assessing their practical viability, highlighting both achievements and challenges in elastic optical networking [7].

Further advancements in SD-EONs have also explored the integration of Q-learning and hybrid approaches to improve Routing, Modulation, and Spectrum Assignment (RMSA). Bryant et al. discuss the application of off-policy Q-learning to enhance routing efficiency in optical networks, highlighting its effectiveness with a large number of k-paths [9]. Ríos-Villalba et al. present a hybrid approach combining Q-learning with k-shortest paths to address the RSA problem, demonstrating improved resource utilization and reduced latency [10]. Other studies explore deep reinforcement learning and multi-band networks, which are beyond the scope of this research [11]–[13].

Despite these advancements, several gaps remain. Bryant et al.'s study did not explore multiple reward policies or compare Q-learning with the traditional KSP algorithm. The computational feasibility of calculating numerous paths *(10-20)* for every request remains questionable, and agent convergence and hyperparameter settings lack demonstration. Similarly, Ríos-Villalba et al. do not show conclusive results with a fully deployed Q-learning model or specify the number of paths considered.

No study has deployed a fully autonomous Q-learning model significantly outperforming traditional baselines with a moderate selection of paths, nor compared it to simpler bandit algorithms to justify Q-learning's complexity. Additionally, the impact of hyperparameter tuning and reward function shaping on reinforcement learning model performance remains underexplored.

III. Reinforcement Learning Algorithms

A. Introduction to Reinforcement Learning

Reinforcement Learning (RL) is a type of machine learning where an agent learns to make decisions by interacting with an environment. The agent takes actions in the environment, receives feedback in the form of rewards, and updates its knowledge to improve future decision-making. The goal is to learn a policy that maximizes the cumulative reward over time. Key concepts in RL include the agent, which is the entity making decisions, and the environment, the system with which the agent interacts. The state (s) represents the current situation of the environment, and the action (a) is a decision made by the agent that affects the environment. The reward (r) is the feedback from the environment, indicating the immediate benefit of an action. The policy $\pi(s,a)$ is a strategy used by the agent to determine actions based on the current state. The value function estimates the expected cumulative reward from a given state-action pair (Action Value Function $Q(s,a)$).

B. Epsilon-Greedy Bandit

The epsilon-greedy bandit algorithm is a simple RL method used in multi-armed bandit problems. It balances exploration (trying new actions to discover their rewards) and exploitation (choosing the best-known action to maximize reward). At each step, with probability ϵ, the agent selects a random action (exploration). With probability $1-\epsilon$, it selects the action with the highest estimated reward (exploitation). The value function in this context is the estimated reward for each action, typically maintained in a table $Q(s,a)$. After taking action a in state s and receiving reward r, the value estimate $Q(s,a)$ is updated incrementally as shown by:

$$Q(s,a) \leftarrow Q(s,a) + \frac{1}{N(s,a)}(r - Q(s,a)) \tag{1}$$

where $N(s,a)$ is the number of times action a has been taken in state s. This update rule ensures that each new reward slightly adjusts the estimated value of the action, weighted by the inverse of the number of times the action has been selected, allowing the estimate to converge over time as more rewards are observed.

C. Upper Confidence Bound (UCB) Bandit

The UCB bandit algorithm is designed to address the exploration-exploitation trade-off by selecting actions based on both their estimated reward and the uncertainty (or confidence) in these estimates. At each step, the agent selects the action a that maximizes the following:

$$Q(s,a) + c\sqrt{\frac{\ln t}{N(s,a)}} \tag{2}$$

where $Q(s,a)$ is the estimated value for action a, t is the total number of steps taken, s is the current state, $N(s,a)$ is the number of times action a has been taken in state s, and c is a parameter controlling the degree of exploration. Similar to the epsilon-greedy algorithm, $Q(s,a)$ represents the estimated reward for each action in the current state. The value estimates are also updated by Equation (1) after action selection.

D. Q-Learning

Q-learning is a model-free RL algorithm that aims to learn the optimal policy by iteratively improving the estimates of the action-value function $Q(s,a)$. At each step, the agent takes action a in state s, observes the reward r and the next state s', and updates the action-value function $Q(s,a)$ using the following rule:

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left(r + \gamma \max_{a'} Q(s',a') - Q(s,a) \right) \tag{3}$$

where α is the learning rate and γ is the discount factor, representing the importance of max potential future rewards $Q(s',a')$, where s' is the next state and a' is the optimal action in s'. $Q(s,a)$ represents the expected cumulative reward of taking action a in state s and following the optimal policy thereafter. The policy $\pi(s,a)$ is derived from the action-value function. Typically, the agent selects the action with the highest Q-value in each state. But, the Q-learning algorithm also depends on ϵ in order to act with some degree of randomness.

IV. Proposed Methodology

This section outlines the methodology for optimizing SD-EON routing using the described RL algorithms.

A. Epsilon-Greedy and UCB Bandit Algorithms

Both the epsilon-greedy and UCB bandit algorithms are designed to optimize routing by selecting the best path from a set of pre-computed paths for each source-destination pair in the network topology. The pre-computed paths are generated using the KSP algorithm, providing k potential paths for each source-destination pair. The Q-table is constructed statically based on these pre-computed paths, but the paths are learned and selected dynamically by the reinforcement learning agent. After selecting a path, the agent sends it to the controller, which then uses the first-fit algorithm for core and spectrum assignment. If the request is blocked, a negative reward is returned to the agent. If the request is routed, a positive or non-negative reward is assigned.

Algorithm 1 outlines the pseudocode for the epsilon-greedy and UCB bandit algorithms as implemented in this research. The algorithm starts by initializing the Q-values for all state-action pairs and initializing the count of actions $N(s,a)$ for all actions. The exploration rate ϵ and the exploration parameter c are then set, the latter being specific to the UCB bandit.

Each episode begins with iterating over each request within the episode (Lines 1-2). For every request, a source-destination pair is chosen randomly and uniformly (Line 3). An action a is then chosen using the ϵ-greedy policy: if a random number is less than ϵ (Line 4), a random path a is selected (Line 5); otherwise, for the UCB algorithm, the path that maximizes the sum of the Q-value and the confidence bound is selected (Lines 7-8). If not using UCB, the path with the highest Q-value is selected (Lines 9-10).

The selected path a is then sent to the controller for assignment (Line 13). If the request is blocked, a reward $R(s,a) =$ *negative* is received (Lines 14-15); otherwise, a reward $R(s,a) =$ *non-negative* is received (Lines 16-17). The Q-value $Q(s,a)$ is updated using the received reward (Line 19). Finally, the count $N(s,a)$ for the action is incremented (Line 20).

The time complexity for the overall process includes components done once and those done for each request. Path computation using the k-shortest path algorithm for each source-destination pair, done once, is $O(S \cdot k \cdot (E + V \cdot \log V))$, where S is the number of source-destination pairs, k is the number of paths, V is the number of vertices, and E is the number of edges. Q-table initialization, also done once, has a complexity of $O(S \cdot k)$. For every request, the process includes looking up the paths for the source and destination with a complexity of $O(1)$ and evaluating and selecting a path using the epsilon-greedy or UCB bandit algorithms with a complexity of $O(k)$. The UCB bandit algorithm is slightly more memory intensive due to the additional calculation of the confidence bound.

Algorithm 1 Epsilon-Greedy and UCB Bandit Algorithms

```
Require: Initialize $Q(s,a)$ for all state-action pairs
Require: Initialize $N(s,a)$ for all actions
Require: Set exploration rate $\epsilon$ and exploration parameter $c$
 1: for each episode do
 2:     for each request do
 3:         Choose a source-destination pair randomly and uniformly (state $s$)
 4:         if random number $< \epsilon$ then
 5:             Select a random path $a$
 6:         else
 7:             if UCB then
 8:                 Select $a = \arg\max \left( Q(s,a) + c\sqrt{\frac{\ln t}{N(s,a)}} \right)$
 9:             else
10:                 Select $a = \arg\max Q(s,a)$
11:             end if
12:         end if
13:         Send path $a$ to controller for assignment
14:         if request is blocked then
15:             Receive reward $R(s,a) =$ negative
16:         else
17:             Receive reward $R(s,a) =$ non-negative
18:         end if
19:         Update $Q(s,a) \leftarrow Q(s,a) + \frac{1}{N(s,a)}(r - Q(s,a))$
20:         Increment $N(s,a)$
21:     end for
22: end for=0
```

B. Q-Learning Algorithm

The Q-learning algorithm is designed to optimize routing by learning from the congestion levels in the network. For each source and destination node pair in the network topology, there are k pre-computed paths the agent may select from, a very similar setup to the previously mentioned bandit algorithms. Unlike the simpler bandit algorithms, Q-learning also takes into account the congestion of a path before and after allocation, allowing the agent to understand and adapt to network conditions over time.

The state s is defined based on the congestion level of the network before the allocation of a new request. Congestion is calculated as the average congestion along all the cores and links of a path, defined as the number of spectral slots occupied divided by the total number of free spectral slots. The congestion state is categorized into two levels: Level 1 for congestion below 0.3 (30%) and Level 2 for congestion above or equal to 0.3 (30%). Each congestion level accesses the same set of k pre-computed paths, but the Q-values associated with these paths differ for each congestion level. This is because a path's performance can vary depending on the congestion level, making it better or worse at different levels of congestion.

The action a corresponds to selecting one of the k pre-computed paths for a given source-destination pair. After selecting a path and observing the result (routed or blocked), the congestion level is re-measured. This post-allocation congestion state is used to update the Q-value. Specifically, the Q-value $Q(s,a)$ is updated using the maximum future Q-value from the new state s', which reflects the congestion level after allocation. The update rule for Q-learning is later used, which is shown in Equation (3).

Algorithm 2 outlines the pseudocode for the Q-learning process as implemented in this paper. Each episode begins with iterating over each request within the episode (Lines 1-2). For every request, the initial state s (pre-allocation congestion level for the current source-destination pair) is observed (Line 3). The state includes the current source and destination pair for the request, as well as the congestion levels for each of the k pre-computed paths with respect to their current congestion. An action a is chosen from these paths using the ϵ-greedy policy based on their Q-values in the current congestive state (Lines 4-8). The chosen action a is executed by sending the request to the controller (Line 9). If the request is routed, a reward $R(s,a) =$ *non-negative* is received (Lines 10-11); otherwise, a reward $R(s,a) =$ *negative* is received (Lines 12-13). The new state s' (post-allocation congestion level for the selected path) is observed (Line 15). The Q-value $Q(s,a)$ is updated using Equation (3) (Lines 16-17). This process continues for each request in the current episode until all requests are processed.

Algorithm 2 Q-Learning Algorithm

```
Require: Initialize $Q(s,a)$ to zero for all state-action pairs
Require: Set learn rate $\alpha$, disc. factor $\gamma$, and exploration $\epsilon$
 1: for each episode do
 2:     for each request do
 3:         Observe initial state $s$ (pre-allocation congestion level
            for current source-destination pair)
 4:         if  random number $< \epsilon$ then
 5:             Choose action $a$ randomly
 6:         else
 7:             Choose action $a = \arg\max Q(s,a)$
 8:         end if
 9:         Execute action $a$ (send path to controller)
10:         if request is routed then
11:             Receive reward $R(s,a) =$ non-negative
12:         else
13:             Receive reward $R(s,a) =$ negative
14:         end if
15:         Observe new state $s'$ (post-allocation congestion level
            for the selected path)
16:         Update $Q(s,a)$ using:
17:         $Q(s,a) \leftarrow Q(s,a) + \alpha\big(R(s,a) + \gamma \max_{a'} Q(s',a')$
                      $- Q(s,a)\big)$
18:     end for
19: end for=0
```

Mod Format	Bit Rate (Gbps)	Slots	Distance (KM)
QPSK	25	1	22,160
	50	2	11,080
	100	4	5,540
16-QAM	25	1	9,500
	50	1	4,750
	100	2	2,375
64-QAM	25	1	3,664
	50	1	1,832
	100	2	916

TABLE I: Modulation Formats with Bit Rates and Slots

The complexity of the Q-learning algorithm involves first considering the source-destination pair for the current request. For each source-destination pair, the algorithm considers the k pre-computed paths. The congestion state, which has two possible levels, is computed based on the current congestion along these paths. The action involves selecting one of these paths based on their Q-values in the current state. Thus, the overall time complexities are: path computation $O(S \cdot k \cdot (E + V \cdot \log(V)))$ and Q-table initialization $O(S \cdot k \cdot C)$ done once, and source-destination lookup $O(1)$ and path evaluation and selection $O(C \times k)$ for Q-learning for each request, where C is the number of congestion levels.

V. System Architecture and Assumptions

The foundation of this research is a dynamic simulation model of a SD-EON, constructed in Python [14]. Based on concepts from a doctoral dissertation [15], this model utilizes the NSFNet topology with 14 nodes and 22 bi-directional links, as depicted in Figure 1.

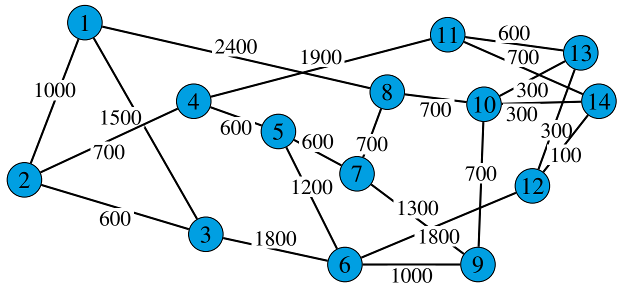

Fig. 1: NSFNet Topology.

Each link in the simulation comprises of four cores, with 128 spectral slots per core. Network conditions are simulated using requests with bit rates of 25, 50, and 100 Gbps, distributed in a 3:5:2 ratio. While each lightpath requires a guard band, these are not included in Table I, which details specific bit rates, spectral slot requirements, and supported distances for each modulation format. It should be noted that we adopt bandwidth-tunable transceivers with a slot spectral width of 12.5 GHz.

Request arrival and holding times follow exponential and Poisson random distributions, respectively, with source-destination pairs determined uniformly. The arrival rate is normalized by the number of cores per link, and each simulation episode is defined as 2,000 requests. A constant mean holding time of 5 time units is used for each request generated.

VI. Results and Discussion

This section presents the experimental results of the proposed reinforcement learning algorithms—epsilon-greedy bandit, UCB bandit, and Q-learning—compared to traditional K-Shortest Paths with First-Fit (KSP-FF) $k = 3$, Shortest Path with First-Fit (SPF-FF), and KSP-FF with $k = inf$ (KSP-inf) algorithms. The experiments were conducted using Erlang values of 500, 750, and 1000 to evaluate the reduction in blocking probability (BP) achieved by the proposed methods.

The KSP-FF and SPF-FF algorithms select paths based on their link lengths, favoring shorter paths to minimize resource utilization. Regarding time complexity, the KSP-FF algorithm has a time complexity of $O(k \cdot (E + V \cdot \log V))$, where E is the number of edges and V is the number of vertices in the network. The SPF-FF algorithm has a time complexity of $O(E + V \cdot \log V)$, as it involves a single shortest path computation.

A. Results

For Erlang 500, the RL algorithms exhibited improvements in reducing BP over the baselines, shown in Fig. 2. The epsilon-greedy bandit algorithm, configured with an epsilon of 1%, a positive reward of 1, and a negative reward of 100, led to a reduction in BP of 98% over KSP-FF, 99% over SPF-FF, and 84.6% over KSP-inf. The UCB bandit algorithm, configured with an epsilon of 20%, a confidence interval value of 2, a positive reward of 1, and a negative reward of 10, resulted in a reduction in BP of 30% over KSP-FF and 73% over SPF-FF, but showed an increase in BP of 81% over KSP-inf. The Q-learning algorithm, with an epsilon linearly decaying from 10% to 5%, a positive reward of 1, and a negative reward of 100, showed a reduction in BP of 92% over KSP-FF, 97% over SPF-FF, and 38.5% over KSP-inf. The learning rate was set to 0.05, and the discount factor was 0.01.

The results for Erlang 750, as displayed in Fig. 3, demonstrate improvements by the RL algorithms. The epsilon-greedy bandit algorithm, with an epsilon of 6%, a positive reward of 10, and a negative reward of 100, led to a reduction in BP of 36.9% over KSP-FF, 72.3% over SPF-FF, and an increase of 28.2% over KSP-inf. The UCB bandit algorithm, configured with an epsilon of 10%, a confidence interval value of 2, a positive reward of 10, and a negative reward of 10, resulted in a reduction in BP of 35.4% over KSP-FF, 71.6% over SPF-FF, and an increase of 25.6% over KSP-inf. The Q-learning algorithm, with an epsilon linearly decaying from 20% to 5%, a positive reward of 10, and a negative reward of 100, showed a reduction in BP of 58.8% over KSP-FF, 81.9% over SPF-FF, and 15% over KSP-inf. The learning rate was 0.01, and the discount factor was 0.95.

For Erlang 1000, the RL algorithms also outperformed the traditional baselines, though the degree of improvement varied, as shown in Fig. 4. The epsilon-greedy bandit algorithm, configured with an epsilon of 1%, a non-negative reward of 0, and a negative reward of 10, showed a reduction in BP of 27% over KSP-FF, 62.4% over SPF-FF, and an increase of 19.7% over KSP-inf. The UCB bandit algorithm, with an epsilon of 20%, a positive reward of 10, and a negative reward of 10, led to a reduction in BP of 34% over KSP-FF, 66.1% over SPF-FF, and an increase of 4.5% over KSP-inf. The Q-learning algorithm, with a constant epsilon of 5%, a positive reward of 1, and a negative reward of 10, resulted in a reduction in BP of 41.9% over KSP-FF, 70.1% over SPF-FF, and 5.3% over KSP-inf. The learning rate was set to 0.05, and the discount factor was 0.01.

B. Discussion

The results demonstrate that reinforcement learning algorithms, particularly Q-learning at higher traffic volumes and the Epsilon-greedy bandit at lower ones, significantly reduce BP in SD-EONs compared to traditional methods. For Erlang 500, as shown in Fig. 2, Q-learning's superior performance is due to its ability to adapt to network conditions over time, considering both immediate and future rewards, with a decaying ϵ strategy enhancing exploration. The epsilon-greedy bandit algorithm showed the most significant BP reductions, likely due to its simplicity being optimal at low traffic volumes, though Q-learning came very close.

Typically, the bandit algorithms converge faster than Q-learning, but over time, Q-learning usually converges to a lower blocking probability. However, Q-learning does not reach a lower blocking probability compared to the Epsilon-greedy bandit at a lower traffic volume, signifying that as the problem becomes simpler (with fewer requests), a less complex bandit algorithm proves its importance.

At Erlang 750, Q-learning began to outperform every algorithm, benefiting from its dynamic learning approach, reducing BP by 15% over KSP-inf. For Erlang 1000, Q-learning remained robust, maintaining significant BP reductions, outperforming KSP-inf by 5.3%. The UCB bandit algorithm, while less effective than Q-learning, showed consistent improvements due to systematic exploration, and the epsilon-greedy algorithm also achieved considerable BP reductions.

Hyperparameters were tuned with 150-200 configurations for each algorithm at each traffic volume, ensuring optimal performance across a total of 450-600 configurations. Reward shaping was also crucial for effective learning. Despite the increased complexity of Q-learning, its significant BP reduction justifies its use in dynamic and complex network environments with higher traffic.

VII. Conclusion

In this study, we explored the optimization of routing in SD-EONs using reinforcement learning algorithms. Among the methods investigated, Q-learning emerged as the best-performing algorithm as Erlang values increased, significantly outperforming simpler bandit algorithms such as epsilon-greedy and UCB bandits. Q-learning's ability to consider both immediate and future rewards enabled it to adapt more effectively to dynamic network conditions, resulting in superior

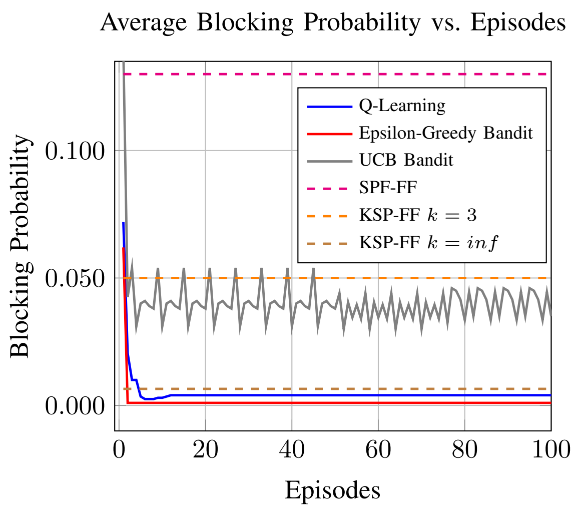

Fig. 2: BP vs. Episodes $Erlang = 500$

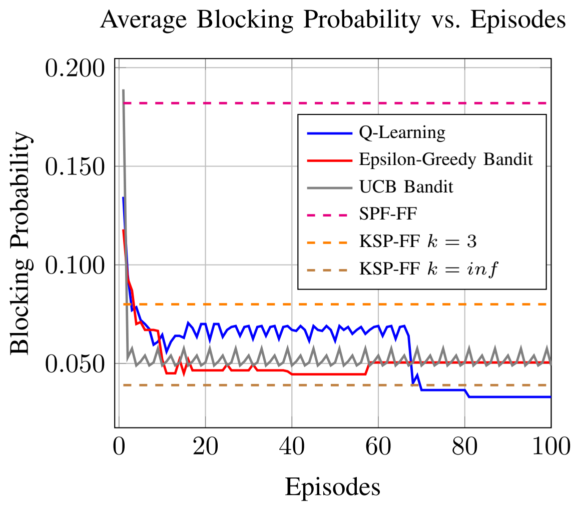

Fig. 3: BP vs. Episodes $Erlang = 750$

performance in reducing blocking probability. Notably, Q-learning performed better than KSP-inf, despite KSP-inf potentially considering every possible path. Future work will extend this research to include core and spectrum assignment, further enhancing network optimization. Additionally, we plan to implement deep learning techniques to potentially improve the efficiency and scalability of RL approaches in SD-EONs.

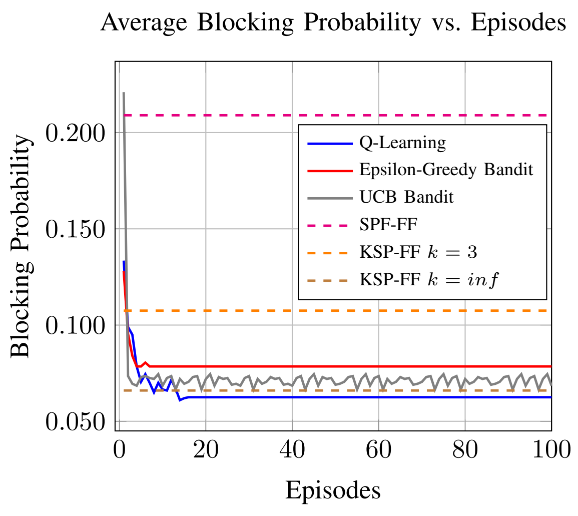

Fig. 4: BP vs. Episodes $Erlang = 1000$

References

[1] R. Gu, Z. Yang, and Y. Ji, "Machine learning for intelligent optical networks: A comprehensive survey," *Journal of Network and Computer Applications*, vol. 157, p. 102576, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1084804520300503

[2] Cisco, "Cisco annual internet report (2018–2023) white paper," https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html, 2020, [Accessed: insert-access-date-here].

[3] O. Gerstel, M. Jinno, A. Lord, and S. B. Yoo, "Elastic optical networking: a new dawn for the optical layer?" *IEEE Communications Magazine*, vol. 50, no. 2, pp. s12–s20, 2012.

[4] K. Nisar, E. R. Jimson, M. H. A. Hijazi, I. Welch, R. Hassan, A. H. M. Aman, A. H. Sodhro, S. Pirbhulal, and S. Khan, "A survey on the architecture, application, and security of software defined networking: Challenges and open issues," *Internet of Things*, vol. 12, p. 100289, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2542660520301219

[5] B. C. Chatterjee, S. Ba, and E. Oki, "Fragmentation problems and management approaches in elastic optical networks: A survey," *IEEE Communications Surveys Tutorials*, vol. 20, pp. 183–210, 2018.

[6] C. Watkins, "Learning from delayed rewards: A foundation of reinforcement learning," 01 1989.

[7] N. C. Luong, D. T. Hoang, S. Gong, D. Niyato, P. Wang, Y.-C. Liang, and D. I. Kim, "Applications of deep reinforcement learning in communications and networking: A survey," *IEEE Communications Surveys Tutorials*, vol. 21, pp. 3133–3174, 2019.

[8] B. C. Chatterjee, N. Sarma, and E. Oki, "Routing and spectrum allocation in elastic optical networks: A tutorial," *IEEE Communications Surveys Tutorials*, vol. 17, no. 3, pp. 1776–1800, 2015.

[9] N. B. Bryant, K. K. Chung, J. Feng, S. Harris, K. N. Umeh, and M. Aibin, "Q-learning based routing in optical networks," in *2022 IEEE Canadian Conference on Electrical and Computer Engineering (CCECE)*, 2022, pp. 419–423.

[10] I. I. Ríos-Villalba, S. Arce, L. A. Albertini, and D. P. Pinto-Roa, "Routing and spectrum assignment in elastic optical networks through a hybrid approach based on k-shortest paths and q-learning," in *2023 IEEE Latin American Conference on Computational Intelligence (LA-CCI)*, 2023, pp. 1–6.

[11] X. Chen, B. Li, R. Proietti, H. Lu, Z. Zhu, and S. J. B. Yoo, "Deeprmsa: A deep reinforcement learning framework for routing, modulation and spectrum assignment in elastic optical networks," *Journal of Lightwave Technology*, vol. 37, no. 16, pp. 4155–4163, 2019.

[12] N. E. D. E. Sheikh, E. Paz, J. Pinto, and A. Beghelli, "Multi-band provisioning in dynamic elastic optical networks: a comparative study of a heuristic and a deep reinforcement learning approach," in *2021 International Conference on Optical Network Design and Modeling (ONDM)*, 2021, pp. 1–3.

[13] J. Errea, D. Djon, H. Q. Tran, D. Verchere, and A. Ksentini, "Deep reinforcement learning-aided fragmentation-aware rmsa path computation engine for open disaggregated transport networks," in *2023 International Conference on Optical Network Design and Modeling (ONDM)*, 2023, pp. 1–3.

[14] A. Rezaee, R. McCann, K. Bempah, K. Tice, and V. Vokkarane, "SDON_simulator," https://github.com/SDNNetSim/SDON_simulator, 2022.

[15] Y. Wang, "Dynamic traffic scheduling frameworks with spectral and spatial flexibility in sdm-eons," Ph.D. dissertation, University of Massachusetts Lowell, 2022.