





Self-learning Routing for Optical Networks

Yue-Cai Huang¹(✉) , Jie Zhang², and Siyuan Yu^{3,4}(✉) 

¹ School of Physics and Telecommunication Engineering,
South China Normal University, Guangzhou, China
huangyuecai@scnu.edu.cn

² XenLink Co. Ltd., Guangzhou, China

³ State Key Laboratory of Optoelectronic Materials and Technologies,
School of Electronics and Information Technology, Sun Yat-sen University,
Guangzhou, China

⁴ School of Computer Science, Electronic and Electrical Engineering and Engineering
Mathematics, University of Bristol, Bristol BS8 1UB, UK
s.yu@bristol.ac.uk

Abstract. It is generally very difficult to optimize the routing policies in optical networks with dynamic traffic. Most widely-used routing policies, e.g., shortest path routing and least congested path (LCP) routing, are heuristic policies. Although the LCP is often regarded as the best-performing adaptive routing policy, we are often eager to know whether there exist better routing policies that surpass these heuristics in performance. In this paper, we propose a framework of reinforcement learning (RL) based routing scheme, that learns routing decisions during the interactions with the environment. With a proposed self-learning method, the RL agent can improve its routing policy continuously. Simulations on a ring-topology metro optical network demonstrate that, the proposed scheme outperforms the LCP routing policy.

Keywords: Optical networks · Routing · Self-learning

1 Introduction

Routing of optical networks has been under research for more than two decades [4, 9, 17]. There are basically three approaches: fixed routing, fixed-alternative routing [6, 14], and adaptive routing [3, 11]. For the fixed routing, one pre-determined route is always chosen for a given source-destination pair. If the route is unavailable, the call request is blocked. For the fixed-alternative routing, there is an ordered list of fixed candidate routes for each source-destination pair. Upon call request, each route is tried following the order until a route is found available; if no route is available, the request is blocked. For the adaptive routing, the path is chosen dynamically depending on the network status upon the request arrival. Among all routing policies, adaptive routing generally gives the best performance [17].

© IFIP International Federation for Information Processing 2020

Published by Springer Nature Switzerland AG 2020

A. Tzanakaki et al. (Eds.): ONDM 2019, LNCS 11616, pp. 467–478, 2020.

https://doi.org/10.1007/978-3-030-38085-4_40

Most well-known adaptive routing policies are heuristic policies. For example, the adaptive shortest-cost-path routing tries to minimize the resources used for each connection upon its arrival, aiming to hold more simultaneous connections. The least congest path (LCP) routing balances the load over the network, so as to avoid bottlenecks. Among the adaptive routing policies, the LCP routing is generally regarded to have the best performance [11]. These heuristic adaptive routing policies can be regarded as to capture some features of the network status and then intuitively exploit the features to achieve good performance. For instance, the LCP policy considers the feature of load on each available path and select the least congested path. As known, besides the load, there are many other features that may affect the network performance, e.g., the number of hops, the link availability, the traffic in service, the topology, etc. From this perspective, A good adaptive policy needs to include enough features and properly exploit them, while on the other hand, in a realistic situation, the variations of network status are often too many to be summarized. These variations come from the change of spectrum availability of each link, the change of remaining holding time of each call, etc. Even if we only consider a simplified version of network status with only the spectrum availability and source-destination pairs, a huge number of network status still exist. Consider the following example. For a five-node bi-directional ring-topology network, with five wavelengths on each link, and only four source-destination pairs (1-2, 2-1, 1-3, 3-1), there are over four billion simplified network status. Therefore, making good adaptive routing policies is challenging, which is a major reason that heuristic algorithms were popular over the past decades.

Recently, Google successfully applied reinforcement learning in playing Atari games and achieved above-human-level performance [12]. The RL-based agent can derive efficient representations of the environment, and use these to generalize past experience to new situations. This gives us a possible way to overcome the difficulty of feature extraction for adaptive routing. In [2, 8, 10, 13], the authors apply reinforcement learning to the routing of optical networks. These methods allow the RL-based agent to learn a routing policy during the interaction with the network environment. In a more recent work [5], a Deep Q-Network (DQN) algorithm is used to capture the features inspired by [12] and obtain some performance improvement compared to the shortest path routing policy. While we still have the following open question: whether there are better routing policies than the existing ones, and how can we find the best routing policies. This paper tries to answer these questions.

In this paper, we propose a RL-based self-learning method. It is based on the following observation: suppose we already have one routing policy, if we can change this policy with just one better action under one specific circumstance, we should have a better policy. With our proposed method, the RL agent changes its current policy according to the competition result of the current policy and a reference policy. In this way, the RL agent should be able to learn for a policy that is no worse than the reference policy. Then, the reference policy is periodically updated with the learnt policy. By repeating the above learning process iteratively, the RL agent can improve its policy continuously.

2 RL for Optical Network Routing

In this section, we first model the optical network routing problem into an RL problem with self-learning (Subjects. 2.1 and 2.2). Then, we introduce how to apply DQN algorithm to the routing problem in Subject. 2.3. Due to space limitation, the basic knowledge of RL will not be covered in this paper, and readers are strongly recommended to refer [16] for better understanding of this paper.

2.1 Mathematical Representation of State and Action

We consider the RL state only at the time when a new connection request arrives. That means at time $t, t + 1, t + 2, \dots$, there is one and only one connection request arriving, and there is no connection request arrivals in the time periods between.

The state at time t , denoted by S_t is composed of two parts: the network state S_t^{net} and the arrival traffic state S_t^{tra} . $S_t = [S_t^{\text{net}}, S_t^{\text{tra}}]$.

The network state S_t^{net} , representing the resource occupation state at time t , is defined as,

$$S_t^{\text{net}} = \begin{bmatrix} b_{11}(t), \dots, b_{1w}(t), \dots, b_{1W}(t) \\ \dots \\ b_{l1}(t), \dots, b_{lw}(t), \dots, b_{lW}(t) \\ \dots \\ b_{L1}(t), \dots, b_{Lw}(t), \dots, b_{LW}(t) \end{bmatrix}, \quad (1)$$

where,

$$b_{lw}(t) = \begin{cases} 1, & \text{if wavelength } w \text{ of link } l \text{ is} \\ & \text{available at time } t \\ -1, & \text{if wavelength } w \text{ of link } l \text{ is} \\ & \text{unavailable at time } t. \end{cases} \quad (2)$$

From Eqs. (1) and (2), network state S_t^{net} is an $L \times W$ matrix, with element $b_{lw}(t)$ denoting the availability of wavelength w on link l . W is the total number of wavelengths and L is the total number of links.

The arrival traffic state at time t (i.e., upon new traffic arrival) is S_t^{tra} , which is defined as Eq. (3). K is the total number of paths for the newly arrived traffic under study.

$$S_t^{\text{tra}} = \begin{bmatrix} b'_{11}(t), \dots, b'_{1k}(t), \dots, b'_{1K}(t) \\ \dots \\ b'_{l1}(t), \dots, b'_{lk}(t), \dots, b'_{lK}(t) \\ \dots \\ b'_{L1}(t), \dots, b'_{Lk}(t), \dots, b'_{LK}(t) \end{bmatrix}, \quad (3)$$

where,

$$b'_{lk}(t) = \begin{cases} 1, & \text{if link } l \text{ is not included by path } k \\ -1, & \text{if link } l \text{ is included by path } k. \end{cases} \quad (4)$$

S_t^{net} is an $L \times W$ matrix, and S_t^{tra} is an $L \times K$ matrix. S_t is to connect the two matrix along the rows, and therefore an $L \times (W + K)$ matrix.

The action of the routing problem is to choose one path from the K paths under study. Therefore, action $A_t \in \{1, 2, \dots, K\}$.

2.2 Reward

The choice of reward is essential for RL, since the objective of RL is to maximize the expected cumulative reward. Well-designed reward setting should be in consistent with the objective of the routing, i.e., to minimize the call blocking probabilities. As for the routing problem, the evaluation of a routing policy is not determined by one routing decision, but by a sequence of routing decisions. Therefore, the reward should reflect the performance of an action within a sequence. Although we have feed-back from future rewards by discounting, this kind of reward cannot fit well with the objective to minimize the blocking probabilities.

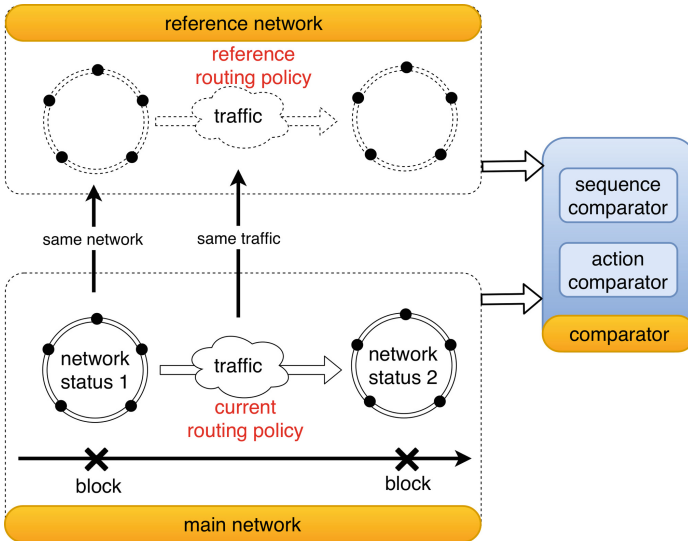


Fig. 1. The illustration of self learning.

In this paper, the reward is set based on the concept of self-learning, which is inspired by the method of “self-play” used by Alpha Zero [15]. The reward is set by a comparison of the network running a learning routing policy with a reference

network running a reference routing policy, as illustrated in Fig. 1. It includes three parts: the main network, the reference network and the comparator.

The main network is the optical network adopting the learning routing policies, with ϵ -greedy approach used to balance the exploration and exploitation. When an arrival is blocked, the network state is recorded and the traffic till the next block is also recorded.

The reference network is a virtual optical network for comparison. It is configured following the recorded network state from the main network, and the recorded traffic is injected to this network. A reference routing policy is adopted. The performance of the saved network can be evaluated by simulations from the initial network state and traffic injected with the given policy.

The comparator compares the two networks with its two components: sequence comparator and action comparator:

1. *Sequence comparator* compares the sequence of actions, by considering which sequence incurs a blocking state earlier. There are three kinds of comparison outputs: “better”, “same”, and “worse”. If the main network incurs blocking earlier than the reference network, we regard the sequence of actions from the learning routing policy performs worse than the reference, then the comparator outputs “worse”. Similarly, if the main network incurs blocking at the same time with the reference network, the comparator outputs “same”. Finally, if the main network incurs blocking later than the reference network, the comparator outputs “better”.
2. *Action comparator* compares each action in the sequence. The sequence comparator gives a general impression of the performance of the actions within a sequence, while not every action is better than (worse than/same with) the actions in the reference policy. Therefore, we should compare them one by one. The action comparator considers each state-action pairs from the reference network. It compares the same state but with action from the learning policy. It outputs two results: same action or different action.

Finally, based on the comparison results, the reward can be set, as given by Table 1. An intuitive explanation on this kind of setting is as follow. (1) If action comparator outputs “same” or the sequence comparator outputs “same”, the actions of the main network and that of the reference network tend to have little difference in affecting the performance, therefore, a relatively small and positive reward 0.1 is given. (2) If actions of the two networks are different, and the sequence in the main network performs better, the action of the main network is potentially a key action to lead to a better sequence, therefore, a relatively big and positive reward 1.0 is given. (3) If actions of the two networks are different, and the sequence in the main network performs worse, this action is potentially a key action to lead to a worse sequence, therefore, a relatively big but negative reward -1.0 is given. (4) If blocking occurs, give a very big negative reward -10 . Since the rewards are set by comparisons, policies better than the reference policy can be obtained, when maximizing the cumulative reward.

Table 1. The setting of reward

	Sequence comparison	Action comparison	Reward
Accept	Better than ref	=	1.0
		≠	0.1
	Same with ref	=	0.1
		≠	0.1
	Worse than ref	=	0.1
		≠	-1.0
Block	NA	NA	-10.0

= means same actions, ≠ means different actions.

Another question is how to choose the reference policy. It can be a well-known policy, e.g., shorted path routing or least congest path routing, or it can be the learning policy as well. When the reference policy is some known policy, the learner is learning a policy no worse than the known policy. If the reference policy is the learning policy itself, then it becomes “self-learning”. In this case, the reference policy is the learnt policy with greedy action, and in the main network, the ϵ -greedy policy can be used to explore the state space. After some time, the reference policy can be updated with the newly learnt policy. In this way, the RL-agent improves itself continuously.

2.3 Learning with Deep Q-Network

Due to the curse of dimensionality, we use a neural network (NN) shown by Fig. 2 as the Q-value function approximator. The input of the neural network is the state at each time step, S_t , and the output is the predicted action-value (Q-value) for each individual action for the input state. In this way, the Q-values of all actions is computed with a single forward pass at the same time for a given state.

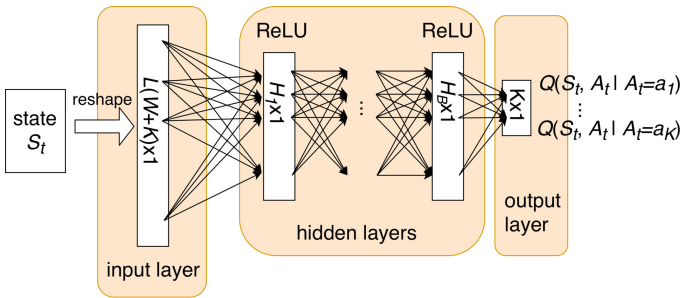


Fig. 2. The NN for Q-value approximation (dropout ignored).

The neural network with weights Θ is used as the Q-network. Denote $Q(s, a; \Theta)$ as the parameterized Q-value for state s and action a . Denote $Q(s', a'; \Theta)$ as the parameterized Q-value for the next state s' and the next action a' . The NN can be trained by adjusting Θ_i at iteration i to reduce the mean-squared error in the Bellman equation, where the optimal target value is given by:

$$\begin{aligned} \text{Target: } y &= r + \gamma \max_{a'} Q(s', a'; \Theta), \\ r &\text{ is the instantaneous reward, } \gamma \text{ is the discount factor, } \gamma < 1. \end{aligned} \quad (5)$$

Then, the loss function at each iteration i is given by the mean-squared error of between the Q-network and the Q-learning targets,

$$\begin{aligned} L_i(\Theta_i) &= \mathbb{E}_{s, a, r, s'} \left[(y_i - Q(s, a; \Theta_i))^2 \right] \\ &= \mathbb{E}_{s, a, r, s'} \left[\left(r + \gamma \max_{a'} Q(s', a'; \Theta_i^-) - Q(s, a; \Theta_i) \right)^2 \right]. \end{aligned} \quad (6)$$

A gradient descent can be applied to minimize the loss function. Besides, experience replay [12] is adopted, to randomly sample previous transitions, so as to smooth the training sample distribution over past behaviors. Moreover, several techniques are also used to facilitate the learning, including mini batch gradient descent, dropout in NN, and Adam optimizer [7]. The DQN algorithm is shown by Algorithm 1. The experience replay technique is shown. For simplicity, the reward setting part is ignored. To get a full picture, Algorithm 1 and Fig. 1 need to be combined.

In the Algorithm 1, two steps are taken recursively: the *sampling* and the *training*. During the sampling, samples of transitions are stored to the replay memory. During the training, samples are randomly chosen from the replay memory to train the neural network. Since the samples are chosen with reward given by self-comparing, the samples are generally better than the current routing policy. Therefore, the routing policies becomes better and better. The learnt policy is embedded in the trained Q-networks, $\arg \max_a Q(s, a; \Theta)$. The policy in the reference network is updated periodically.

3 Simulations

A discrete event simulator written in Python is used to simulate the traffic and the resource provision of the optical network. The DQN reinforcement learning part is written based on TensorFlow [1].

3.1 Parameter Settings

Consider a five-node bidirectional-ring-topology metro network. The key notations are listed in Table 2 for convenience, and they are also explained in the context. The total number of wavelength is five. The traffic pattern is non-uniform

Algorithm 1. Reinforcement learning algorithm

```

1 #INITIALIZATION
2 Create and initialize the optical network
3 Create and initialize the neural network
4 Initialize replay memory  $D$ 
5 while not stopping criterion do
6   while not policy update criterion do
7     #SAMPLING
8     while not sampling stopping criterion do
9       Upon traffic coming, store state to  $S_t$ 
10      With probability  $\epsilon$  select a random action  $A_t$ 
11      Otherwise select  $A_t = \arg \max_A Q(S_t, A; \Theta)$ 
12      Execute action  $A_t$  and wait until new traffic coming, get new state
13       $S_{t+1}$ 
14      Get reward  $r_{t+1}$ 
15      Store transition  $(S_t, A_t, r_{t+1}, S_{t+1})$  in  $D$ 
16    end while
17    #TRAINING
18    while not training stopping criterion do
19      Sample a random minibatch of transitions  $(s, a, r, s')$  from  $D$ 
20      Set  $y = r + \gamma \max_{a'} Q(s', a'; \Theta)$ 
21      Perform a gradient step on  $(y - Q(s, a; \Theta))^2$  with respect to the
22      neural network parameters  $\Theta$ 
23    end while
24  end while
25  Policy is  $\arg \max_a Q(s, a; \Theta)$ 
26  Update policy in the reference network
27 end while
28 Policy is  $\arg \max_a Q(s, a; \Theta)$ 

```

with traffic matrix given as follows:

$$T = \lambda \begin{bmatrix} 0, 1, 1, 0, 0 \\ 1, 0, 0, 0, 0 \\ 1, 0, 0, 0, 0 \\ 0, 0, 0, 0, 0 \\ 0, 0, 0, 0, 0 \end{bmatrix}. \quad (7)$$

This means, traffic only exists for source-destination pair (1, 2), (1, 3), (2, 1), and (3, 1). The arrival processes are independent Poisson processes, with arrival rate λ , and the service time distributions follow exponential distributions with average service time μ . In this bidirectional ring topology, routing can be done clockwise or anti-clockwise. Therefore, for each accepted request, only two actions can be chosen: routing clockwise or anti-clockwise. For wavelength assignment, the first-fit strategy is used.

Table 2. Key parameters

	Notation	Meaning	Value
Optical network	N	Number of nodes	5
	L	Number of links	$L = 2N = 10$
	W	Number of wavelengths	5
	K	Number of routing paths	2
Traffic	λ	Arrival rate	0.4
	μ	Average service time	1.0
RL agent	γ	Discount rate	0.99
	α	Learning rate	1×10^{-5}
	ϵ	Explore rate	0.1
	B	Number of hidden layers of the Q-network	3
	H	Number of neurons for each hidden layer same number for each hidden layer	{32, 256, 1024}
	P	Dropout probability for NN	0.5

A five-layer forward feed fully-connected neural network is chosen as the function approximator for the state-action value function. The neural network is illustrated in Fig. 2. Recall that the state of the RL algorithm is a $L \times (W + K)$ matrix. With total number of link $L = 10$, total number of wavelength $W = 5$, and total number of paths $K = 2$, the state is a 10×7 matrix. This matrix is reshaped to a 70×1 array and forms the input layer of the neural network. Then it is followed by three hidden layers with number of nodes H_1, H_2, H_3 . The ReLU activation function is chosen for all hidden layers. Besides, not shown in Fig. 2, the dropout technique is introduced in the neural network to avoid overfitting, with the dropout probability 0.5. Besides, the discount rate γ is 0.99, the explore rate ϵ is 0.1, and the learning rate α is 1×10^{-5} .

3.2 Learning for a Reference Policy

With the aim of outperforming the existing routing policies, we should check first whether the RL-routing can learn to be a given policy, i.e., policy-fitting. Therefore, we run a set of simulations to fit the RL-routing to be the best-performed LCP routing. We compare the actions given by the Q-network and those given by the LCP policy. If the actions are the same, then give reward 1, otherwise, give reward -1 . The simulation results are shown in Figs. 3 and 4. In the two figures, the x-axis is the number of arrivals batches (1 arrival batch includes 1000 arrivals), i.e., the results are collected every arrival batches (every 1000 arrivals). Figure 3 shows the blocking probabilities of LCP and three RL-based policies with different number of nodes in the hidden layer of the Q-networks. The accuracy shown by Fig. 4 representing the portion that the actions

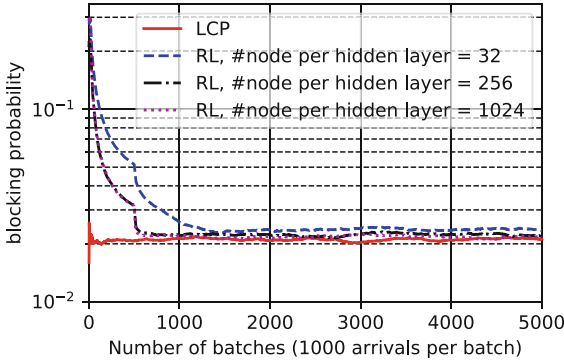


Fig. 3. Performance of RL-routing with LCP fitting.

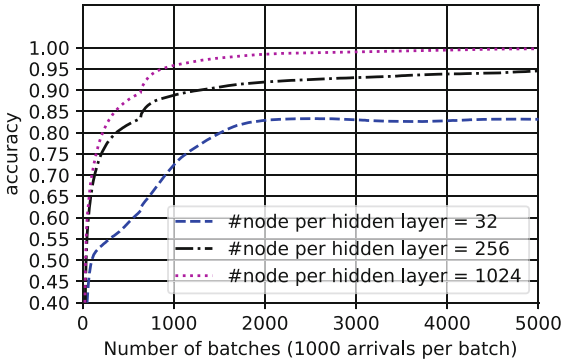


Fig. 4. The accuracy of fitting LCP by RL routing.

from RL routing are the same with those from LCP policy. From Fig. 3, we can see that the blocking probabilities become similar with LCP soon (after 2000 bathes) for all Q-networks. While from Fig. 4, it is revealed that only the case with number of hidden layer nodes 1024 fits the LCP after 4000 batches. We can make two points here: (1) To fit even a simple policy such as LCP, we need a large enough neural network. (2) There are many different policies that give similar performance with the LCP.

3.3 Self learning

Taking the RL-routing policy that is already fit to the LCP, we apply the self-learning process to pursue better policies. The simulation results are shown in Fig. 5. It is demonstrated by Fig. 5 that the RL-routing, incurs some degradation at the beginning of the training, gradually it outperforms LCP and maintains the advantage over LCP during the training. This demonstrate that, by the self-learning method, indeed better routing policies can be learnt. We should also notice that, the performance of the learnt policies incurs some variations.

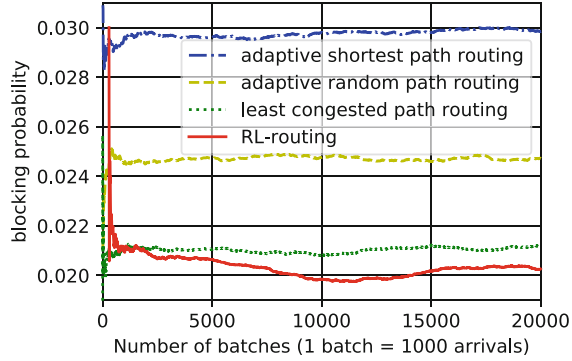


Fig. 5. Performance of RL-routing with self-learning.

These variations may come from the exploration and the variations of the reference policy to be compared.

4 Discussion

This is an early work for applying reinforcement learning to the optical network resource allocation, there are some limitations that need to be investigated for future research. One challenge is the scalability. For the simulated 5-node-5-wavelength ring metro network, a five layer fully-connected neural network with 1024 nodes per hidden layer is already required. Some efforts need to be paid to reduce the complexity of the neural networks as the size of the optical network scales up. One potential approach is to use convolutional neural networks, leveraging the locality of the network status. There are also plenty of open problems for future research, including the impact of traffic variations and network topologies, and the spatial/spectral resource assignment.

5 Conclusion

This paper provides a new direction to optimize the routing of optical networks. The proposed reinforcement learning algorithm is able to continuously improve its routing policy with self learning method. Simulation on a ring-topology metro network demonstrate that the learnt policy outperforms the least congest path routing policy which was regarded the best routing policy. With the computation power increasing rapidly, this RL method can be a scalable approach for dynamic resource allocation and control for optical networks. It can also be combined with software defined network to achieve an agile control plane.

Acknowledgment. This work is supported by: National Natural Science Foundation of China (61490715, U1701661); Local Innovative and Research Teams Project of Guangdong Pearl River Talents Program (2017BT01X121); Fundamental Research Funds for the Central Universities of China (SYSU:17lgpy51).

References

1. Abadi, M., et al.: TensorFlow: large-scale machine learning on heterogeneous systems (2015). <https://www.tensorflow.org/>. software available from tensorflow.org
2. Alyatama, A.: Dynamic routing and wavelength assignment using learning automata technique [all optical networks]. In: Proceedings of IEEE GLOBECOM, vol. 3, pp. 1912–1917 (2004)
3. Chan, K.M., Yum, T.S.P.: Analysis of least congested path routing in WDM light-wave networks. In: Proceedings of INFOCOM, pp. 962–969 (1994)
4. Chatterjee, B.C., Sarma, N., Oki, E.: Routing and spectrum allocation in elastic optical networks: a tutorial. *IEEE Commun. Surv. Tutorials* **17**(3), 1776–1800 (2015)
5. Chen, X., Guo, J., Zhu, Z., Proietti, R., Castro, A., Yoo, S.: Deep-RMSA: a deep-reinforcement-learning routing, modulation and spectrum assignment agent for elastic optical networks. In: Proceedings of Optical Fiber Communication Conference, pp. W4F-2 (2018)
6. Harai, H., Murata, M., Miyahara, H.: Performance of alternate routing methods in all-optical switching networks. In: Proceedings of INFOCOM, vol. 2, pp. 516–524 (1997)
7. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. arXiv preprint [arXiv:1412.6980](https://arxiv.org/abs/1412.6980) (2014)
8. Kiran, Y., Venkatesh, T., Murthy, C.S.R.: A reinforcement learning framework for path selection and wavelength selection in optical burst switched networks. *IEEE J. Sel. Areas Commun.* **25**(9), 18–26 (2007)
9. Klinkowski, M., Lechowicz, P., Walkowiak, K.: Survey of resource allocation schemes and algorithms in spectrally-spatially flexible optical networking. *Opt. Switching Netw.* **27**, 58–78 (2018)
10. Koyanagi, I., Tachibana, T., Sugimoto, K.: A reinforcement learning-based light-path establishment for service differentiation in all-optical WDM networks. In: Proceedings of IEEE GLOBECOM, pp. 1–6 (2009)
11. Li, L., Somani, A.K.: Dynamic wavelength routing using congestion and neighborhood information. *IEEE/ACM Trans. Netw. (TON)* **7**(5), 779–786 (1999)
12. Mnih, V., et al.: Playing Atari with deep reinforcement learning. [arXiv:1312.5602](https://arxiv.org/abs/1312.5602) [cs.LG] (2013)
13. Pointurier, Y., Heidari, F.: Reinforcement learning based routing in all-optical networks. In: Proceedings of IEEE Fourth International Conference on Broadband Communications, Networks and Systems, pp. 919–921 (2007)
14. Ramamurthy, R., Mukherjee, B.: Fixed-alternate routing and wavelength conversion in wavelength-routed optical networks. *IEEE/ACM Trans. Netw.* **10**(3), 351–367 (2002)
15. Silver, D., et al.: Mastering the game of go without human knowledge. *Nature* **550**(7676), 354 (2017)
16. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction (2018). <http://incompleteideas.net/>
17. Zang, H., Jue, J.P., Mukherjee, B.: A review of routing and wavelength assignment approaches for wavelength-routed optical WDM networks. *Opt. Netw. Mag.* **1**, 47–60 (2000)