Submitted to *INFORMS Journal on Optimization*
manuscript IJO-2022-07-020.R1

Dynamic Routing and Wavelength Assignment with Reinforcement Learning

Peyman Kafaei
Department of Mathematics and Industrial Engineering, Polytechnique Montréal, Montreal, Canada,
peyman.kafaei@polymtl.ca

Quentin Cappart
Computer Engineering and Software Engineering Department, Polytechnique Montréal, Montreal, Canada,
quentin.cappart@polymtl.ca

Nicolas Chapados
Department of Mathematics and Industrial Engineering, Polytechnique Montréal, Montreal, Canada,
nicolas.chapados@servicenow.com

Hamed Pouya
Department of Mechanical and Industrial Engineering, University of Toronto, Toronto, Canada,
h.pouya@utoronto.ca

Louis-Martin Rousseau
Department of Mathematics and Industrial Engineering, Polytechnique Montréal, Montreal, Canada,
louis-martin.rousseau@polymtl.net

With the rapid developments in communication systems, and considering their dynamic nature, all-optical networks are becoming increasingly complex. This study proposes a novel method based on deep reinforcement learning for the routing and wavelength assignment problem in all-optical wavelength-decision-multiplexing networks. We consider dynamic incoming requests, in which their arrival and holding times are not known in advance. The objective is to devise a strategy that minimizes the number of rejected packages due to the lack of resources in the long term. We employ graph neural networks to capture crucial latent information from the graph-structured input to develop the optimal strategy. The proposed deep reinforcement learning algorithm selects a route and a wavelength simultaneously for each incoming traffic connection as they arrive. The results demonstrate that the learned agent outperforms the methods used in practice and can be generalized on network topologies that did not participate in training.

Key words: Routing and wavelength assignment; Graph neural network; Reinforcement learning

1. Introduction

All-optical-networks play a critical role in wide-area backbone networks (Pinto 2002). In this paper, we investigate the performance of a learning-based algorithm on the *Routing and Wavelength*

Assignment (RWA) problem (Ramaswami and Sivarajan 1995) in the context of *wavelength-division multiplexing* (WDM) networks. Solving RWA corresponds to selecting a route between two endpoints for all incoming requests and assigning a wavelength to transmit the data across them. In WDM networks, packages that use the same fiber links cannot use the same wavelengths at the same time. Due to the limited number of wavelengths, incoming traffic requests may be blocked if there are no available wavelengths across the links of the selected route.

RWA problems are conventionally either static or dynamic. The static version is based on the assumption that the complete information of the incoming requests, such as arrival time or the duration (in case of releasing the requests after a finite time) is known. Several *lightpaths* (i.e., a combination of a route and a wavelength) are set up all at once, and they remain in the network. However, in realistic applications, these assumptions do not hold. A lightpath needs to be generated for each incoming request as it arrives and the admitted traffic connection request may be released after a finite amount of time.

The dynamic arrival and departure of the traffic requests and the uncertainty of future ones substantially destabilize the problem and reduce the accuracy and efficiency of the conventional optimization methods.

In general, the lightpaths already assigned cannot be re-routed to accommodate the new requests that arrive. However, aggressive reconfiguration (i.e. lightpath defragmentation) has recently been proposed to modify the current lightpaths to reduce the blockage probabilities at the expense of high computational and operational costs (Chen et al. 2019, Pouya 2018, Daryalal and Bodur 2020).

In the settings of this work, wavelengths are assigned to each of the links comprising a route for an incoming request. If all of these links use the same wavelength along the route of the request, the information can be successfully transferred from source to destination. Otherwise, the incoming request is blocked. This is known as the *wavelength continuity constraint*, which makes the wavelength-routed networks different from the traditional circuit-switched telephone networks.

Recently, researchers have started applying machine learning approaches in the field of RWA problems, but only routing has been under investigation. The wavelength assignment and routing are substantially intertwined and selecting one affects the other. In this paper, an algorithm based on deep reinforcement learning is presented for the dynamic RWA problem. This algorithm selects the route assign a wavelength to it simultaneously. The algorithm is entirely online and is trained via simulation, and thereby does not require a large amount of historical data. We assume that accurate network state information is always available. The specific contributions are as follows:

1. We propose a *deep reinforcement learning* (DRL) algorithm that finds the routes and assigns wavelengths to incoming traffic requests simultaneously. Experimental results obtained show that

this approach can obtain a high-quality solution against standard baselines used to solve this problem for four standard network topologies (NSFNET, GEANT, EON, and USA).

2. We exploit the topology of the underlying network as graph-structured data and apply graph neural networks, and in particular, *graph attention networks* (GATs), to encode latent information. This enables us to capture more features and leads to a better performance of the proposed algorithm.

The rest of the paper is organized as follows. Section 2 examines the existing literature on the use of graph neural networks and deep reinforcement learning in routing and wavelength assignment as well as the previous algorithms to solve the RWA problem. Section 3 discusses the deep reinforcement learning algorithm to solve this problem. The related environment and the proposed neural networks architecture are introduced in this section. Experiments and discussions of the results are presented in Section 4.

2. Background

In this section, we study previous methods developed to solve the RWA problem.

2.1. Routing and Wavelength Assignment

Routing in telecommunication networks is known to be NP-hard when a set of traffic requests is considered (Di Ianni 1998). As the size and complexity of the network increase, opting for conventional *shortest path first routing* algorithms may lead to a high number of blocked requests. In addition to routing, assigning wavelengths to routes results in a more complex problem. Then, the routing and wavelength assignment problem is NP-hard in its general settings (Gu et al. 2020). It can be formulated as an integer linear program (ILP) and solved with dedicated solvers. This solving process however gets solutions only at the expense of complex, expensive, and time-consuming computations even for medium-sized instances (Martin et al. 2019). The majority of the literature on RWA has focused on solving processes based on metaheuristics, such as genetic algorithm and simulated annealing (Zhou et al. 2012). These methods are faster but attain sub-optimal solutions (Daryalal and Bodur 2020, Martin et al. 2019). Most of the exact algorithms are proposed to find only a single route, while the heuristics are capable of finding several routes (Rodriguez et al. 2016). A taxonomy of the RWA problem and its variants is provided by Zang et al. (2000). Jaumard and Daryalal (2017) propose methods to enable solving RWA instances with realistic sizes. To this end, they represent an exact method based on column generation decomposition as well as two heuristics for the RWA problem. The authors outlined that the algorithm may fail for large instances in a reasonable solution time; however, it provides ϵ-solutions with high accuracy. Daryalal and Bodur (2020) propose to formulate the RWA problem as a stochastic optimization problem where

uncertainty exists for the future traffic. In their study, the distributions of the random variables are known to the decision-maker.

The wavelength assignment problem has been also considered in related works. Randhawa and Sohal (2010) show that by increasing the load of the network, the blocking probability increases. Zhou and Yuan (2002) compared different wavelength assignment strategies (*random-fit*, *first-fit*, and *most-used*) and analyzed their performance with imprecise network state information. In this study, they opted for a shortest path algorithm for routing the incoming requests.

While several researchers have studied dynamic RWA problems, the majority of them focus on finding the optimal routing strategy while the requests arrive in real-time, and only opted for a heuristic to assign the wavelengths (Zang et al. 2001, Huimin et al. 2017, Duhovnikov et al. 2006). Mohan and Kaushal (2020) introduced a dynamic conversion sensing algorithm to reduce the blocking probability. Besides, Chu et al. (2003) developed a dynamic algorithm that considers the length of the k-shortest-paths and that frees wavelengths. This is carried out by combining a weighted least-congestion routing and a first-fit wavelength assignment strategy. It has been shown that plugging wavelength converters over the network does not highly contribute to the reduction of the blockage rate as time goes on (Jaumard et al. 2005, 2006).

Recently, different machine learning techniques have been applied to solve this complex problem. Martin et al. (2019) presented supervised learning approaches based on logistic regressions and deep neural networks. They formulated the RWA problem as a classification problem. Several optimal RWA configurations are computed prior to training the model using algorithms from the literature. Their method learns relations between features of a problem, such as the topology of the network, capacity, and wavelengths with the characteristics of corresponding good (or optimal) configurations. Although this approach reaches solutions quickly, it only considers a few possible configurations for all diverse network configuration inputs and is then sub-optimal. Several researchers leverage reinforcement learning to solve the RWA problem. Kiran et al. (2007) formulates the path and wavelength selection in *optical burst switched networks* as a multi-armed bandit problem and solves it with a tabular Q-learning method. They provide separate algorithms for path and wavelength selection. The corresponding Q-value of path selection and wavelength selection are updated based on a successful burst in the network. The selections of the route and wavelength are not directly related in the method they propose. Applying DRL to network optimization problems has received high interest in recent years. Chen et al. (2019) propose a DRL framework for routing, modulation, and spectrum assignment in *elastical optical networks*. They deploy advantage actor-critic methods to parameterize the action selection policy and showed a reduction in request blockage following their proposed algorithm. Xu et al. (2018) leverage experience replay to enhance the performance of a proposed Q-learning method to solve the traffic

engineering problem in communication networks. Pointurier and Heidari (2007) solve the routing problem in optical networks with physical impairment by developing a linear reward ϵ-penalty method. They disaggregate the wavelength assignment from the routing and opt for a first-fitted heuristic. The experimental results show a lower blocking rate in comparison with the shortest path and uniform path selection strategies. By proposing a novel elaborate representation of the states in a DRL algorithm, Suárez-Varela et al. (2019) show an increase in the performance of applying DRL to the routing problem in optical transport networks which outperforms traditional heuristics. A few studies use *graph neural networks* (GNN) as a part of the DRL approach to solve variants of the RWA problem. Swaminathan et al. (2021) developed a GNN-driven RL algorithm, referred to as *GraphNet*, for optimal routing of the incoming requests on *software-defined networks* (SDN) to minimize the package delay. Unlike WDM, they do not consider the existence of multiple wavelengths over the fiber links of the network. In order to predict the distribution of request delays in SDNs, Rusek et al. (2020) proposed a novel network based on message-passing GNNs using recurrent neural networks as aggregation functions. A recent comprehensive survey on machine learning methods for optical networks and SDNs is presented in (Gu et al. 2020) and (Amin et al. 2021).

Our closest related work is the approach proposed by Almasan et al. (2022). We both restricted the set of actions to the best shortest paths and defined the reward function similarly. We also used similar features, such as the *betweenness centrality* measure. However, unlike Almasan et al. (2022) who only focus on the *routing* part, our work focuses on finding the optimal *lightpath* (route and wavelengths at the same time), which raises additional challenges. In the problem they consider, each edge has a fixed capacity that cannot be exceeded. In our case, the demands correspond to a request for a specific wavelength, which introduces symmetry issues between all the combinations of lightpaths. Another important difference lies in the fact that the traffic requests in our work are *dynamic* and will be dropped after a certain amount of time. Almasan et al. (2022) is *static*, and consider the problem where once the requests are provisioned, they will remain in the network. This dynamic departure of the incoming traffic requests increase the difficult of the problem, as the information on future requests is completely unknown. On the methodological point of view, we propose to use an architecture based on a graph attention network (Veličković et al. 2018), instead of a standard graph neural network (Gilmer et al. 2017). The promise of graph attention networks is to capture more latent information among graph elements to take better actions.

3. Methodology

We propose an approach based on *graph neural networks* (GNNs) and *deep reinforcement learning* (DRL) to learn appropriate routing and assignment strategies from a given situation on a network

topology. We use GNNs to model the current state of the network at each time so that a DRL agent is able to learn how to leverage it to achieve its goal. An overview of the proposed framework is presented in Figure 1.

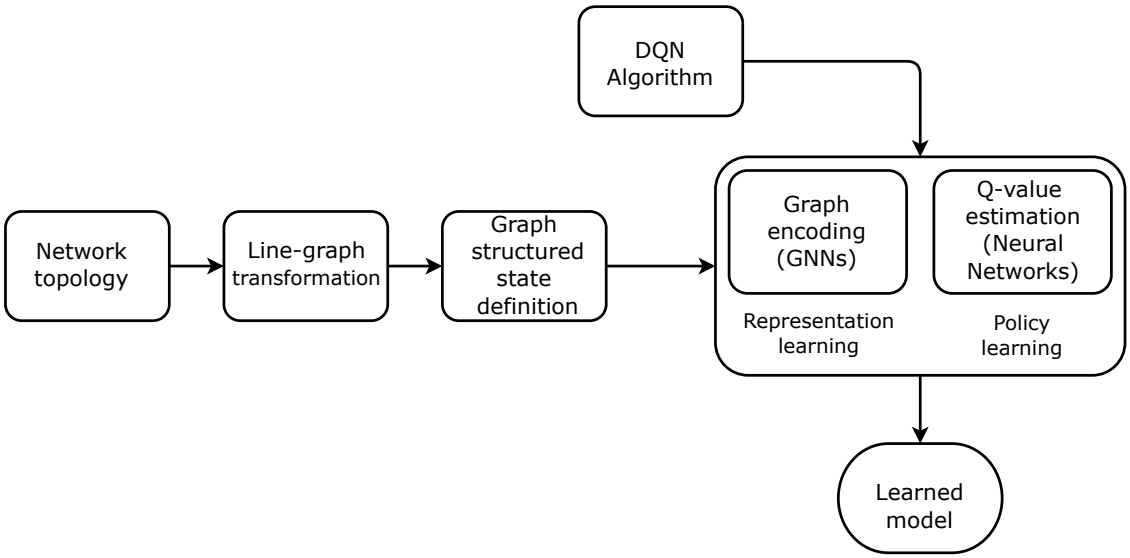


Figure 1 Overview of the proposed framework.

3.1. Topology and Dynamics of the Network

In our context, a *wavelength-division multiplexing* (WDM) optical network topology is represented by a graph $G = (V, E)$ where V are the nodes, and E are the edges, i.e., the fiber links of the physical network. For each fiber link, a set of possible wavelengths Ω indexed by ω exist. The number of possible wavelengths is fixed and denoted by $\mathcal{W} = |\Omega|$. All wavelengths are assumed to have identical capacities. At particular stochastic times τ, traffic demands are requested from the network. A traffic request is represented by a source node n^1, a destination node n^2, and a service time duration δ. We represent a lightpath request for an incoming traffic demand as $\ell^{\kappa}_{\omega}(u, v)$ where κ denotes the path between the source $u \in V$ and the destination $v \in V$ (among the k-shortest paths), and where ω is the wavelength assigned to it. Feasible lightpaths are generated such that: (1) the same wavelength cannot be assigned to two paths sharing a link, (2) all the links throughout a path must use the same wavelength. The latter is commonly referred to as the *wavelength continuity constraint*. Figure 2 depicts a few possible lightpaths for a simple topology with $|\Omega| = 2$ and an arriving request.

In the static RWA problem, a set of predetermined, incoming traffic connection requests is given (Jaumard and Daryalal 2017). The objective of this problem is generally to minimize the number of used resources i.e. the number of wavelengths used (Zang et al. 2000).

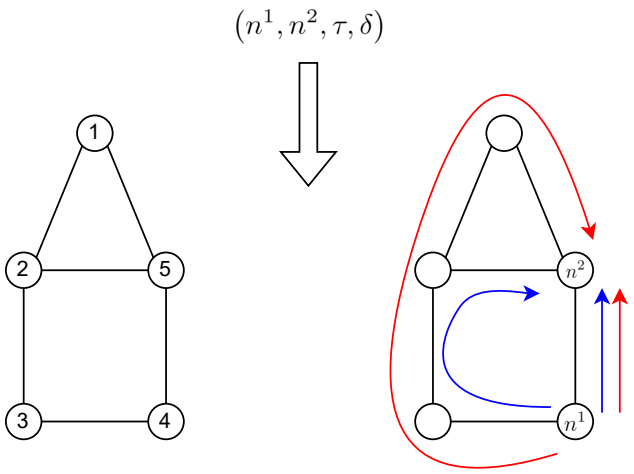


Figure 2 **Examples of possible lightpaths for a sample topology with $|\Omega| = 2$ and an arriving request between nodes n^1 to n^2 at arrival time τ and for duration δ.**

The dynamic RWA problem is defined as follows. For a WDM network represented by a graph G, the problem finds a lightpath for an incoming traffic request at the time of its arrival. We assume that no wavelength conversion exists over the network; therefore, the wavelengths assigned must follow the wavelength continuity constraint. The traffic adds and drops over time by assigning new lightpaths and releasing the wavelength by the end of their corresponding durations. The objective of the problem is to reduce the blockage rate of traffic requests over time.

Requests are generated by uniformly selecting a source destination from the set of nodes. Each request has an arrival time and a duration. Arrival times follow a Poisson process and the durations are generated by an exponential distribution. These values represent the load of the network at each time of the simulation. Load is measured in *erlang* (a standard unit in telecommunication), and is calculated by multiplying the arrival rate and the average duration of the requests. It is therefore the average number of connections measured at any time in the network (Zang et al. 2001). No information regarding these distributions and the future is available to the agent at each decision point. This results in a challenging optimization problem for the agent to devise an optimal strategy.

3.2. Graph Neural Network

As a WDM network has an intrinsic graph structure, it seems natural to design a method able to leverage it. Graph neural networks (Scarselli et al. 2009) are novel neural networks that operate over graph-structured inputs. They generate a vectorial representation of an input graph. This is done by an iterative process that associates the latent features of the neighboring elements (nodes and edges) of a graph. Through this iterative message-passing algorithm, the features of these elements are updated to generate and to learn an output vector. Different architectures have been

introduced in the literature and we based our framework on a *graph attention network* (Veličković et al. 2018).

The final learned representation of the graph, referred to as an *embedding*, encodes hidden information of the graph that can be used to solve complex combinatorial optimization problems (Cappart et al. 2021). Since network topologies can be directly translated to a graph, using GNNs offers more advantages than applying other neural network architectures, such as fully-connected or grid-convolutional neural networks. This potential performance of GNNs has been identified in recent related works (Rusek et al. 2019, Almasan et al. 2022).

3.3. Deep Reinforcement Learning

In deep reinforcement learning, an agent interacts with the environment by performing actions and observing information about the environment (Sutton and Barto 2018). The information contains the state of the environment and a reward, evaluating the immediate quality of the performed action. The agent has no prior knowledge of the environment and aims to maximize the sum of the collected rewards. To do so, it explores the environment using the reward as signal. At each state $s \in S$, an action $a \in A$ is performed. The state is thereby transitioned to a new state (s') and the environment sends a reward $r \in \mathbb{R}$ to the agent. The goal is to design a strategy to maximize the cumulative reward collected by the agent when the episode is finished. This problem can be modeled as a *Markov decision process* (MDP). There exists in the literature a plethora of reinforcement learning algorithms for solving this problem. We refer to the textbook of Sutton and Barto (2018) for an explanation of the main algorithms.

Among them, Q-learning (Watkins and Dayan 1992) is a reinforcement learning algorithm that learns Q-values, i.e., an estimation of how good an action is. Provided that the Q-values are optimal, the optimal decision for the agent is to always take the action with the highest Q-value. However, computing exact Q-values is intractable in practice. Indeed, in the standard Q-learning algorithm, Q-values are stored in a table, with a specific entry for each pair of state and action. This table is substantially large for most practical problems. *Deep Q-learning* (DQN) (Mnih et al. 2013) is a variant that uses a deep neural network to estimate Q-values and to get rid of the table. The high generalization capabilities of deep neural networks enable the agent to approximate the Q-value for states never seen before during the training. For such a reason, our approach is based on DQN.

3.4. Reinforcement Learning Environment

This section describes the environment we built to model the dynamic RWA problem. At the arrival of a new incoming traffic request, the agent receives the state of the environment as a graph. The request contains the source, the destination, the current time, and the duration of the package.

The requests do not have any preference over individual routes or wavelengths. By observing the state (formalized latter), the agent decides on the lightpath to assign to the current traffic request.

The role of the GNN is to receive the state as the graph, process it, and output a new representation. To this end, the GNN aggregates the information of the neighboring nodes and edges of the graph and encodes them with a new representation. This new encoded vectorized representation of the state graph is then forwarded to a fully-connected neural network that approximates the Q-values $Q(s,a)$ for all the possible actions $a \in A$ (i.e., the set of lightpaths) to be performed in state s.

The most important information about the network is related to the physical links, i.e. their capacity and the available wavelengths already assigned to them. However, standard GNN architectures mainly operate on graphs with features attached to nodes, even if architecture dedicated to edge features exists (Kamiński et al. 2022). To reflect this in our implementation, we resort to a *line-graph transformation* of the original network graph G. In such a transformation, the new graph G' has a node for each edge in G, and an edge joins those nodes if the two edges in G share a common node. The goal is to facilitate the message-passing steps in standard GNN architectures. Figure 3 illustrates a sample line-graph transformation. Defining a reinforcement learning environment (i.e., a MDP) requires to define, adequately, the set of states, the set of actions, the transition function, and the reward function. We formalized it as follows.

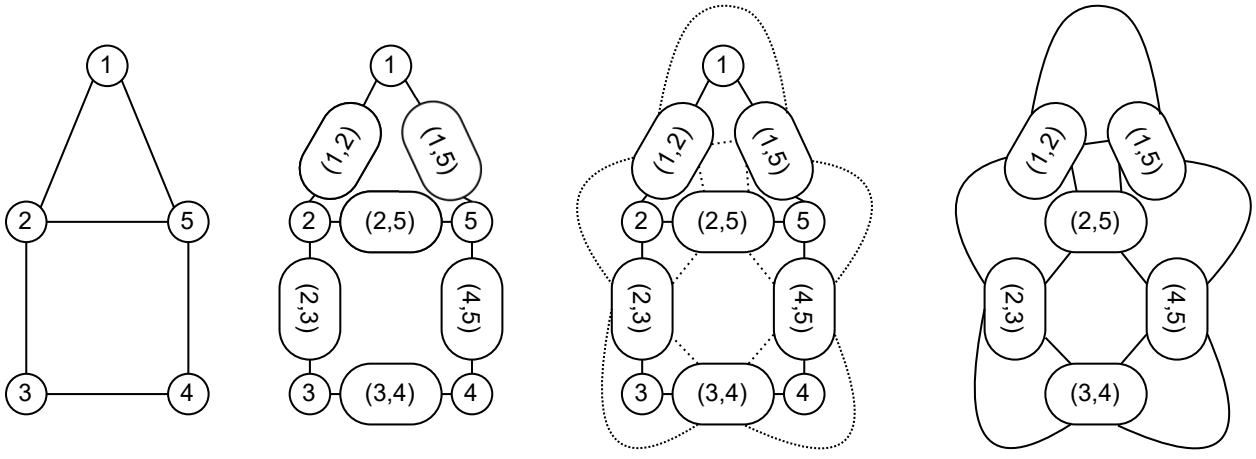


Figure 3 Line-graph transformation example.

State Upon receiving a traffic demand (n^1, n^2, τ, δ) at time τ (i.e., a lightpath request), the agent observes a graph-structured representation of the environment (i.e., the line-graph transformation of the initial network). The GNN module retrieves the state features as well as the graph structure of the state s. Each node l in the graph (i.e., a link in the telecommunication network) is decorated with the following features:

1. The wavelength $\omega \in \Omega$ assigned to the link l (ω_l).

2. The remaining time of the requests currently reserving the link l. Let $\tau(r)$ and $\delta(r)$ denote the arrival time and the duration of a request r, respectively. The feature is computed as follows for each request r: $\tau(r) - \delta(r)$.

3. The betweenness centrality of link l as defined by Freeman (1977). It is computed as follows: $\sum_{s,t \in E}(\sigma_{s,t}(l)/\sigma_{s,t})$, where $\sigma_{s,t}$ is the total number of shortest paths from link s to link t, and $\sigma_{s,t}(l)$ is the total number of shortest paths from link s to link t passing through link l.

4. The relative popularity of the wavelengths at link l. It is calculated as the number of times the current wavelength was selected before at link l since the beginning of the episode (ω_l^*) divided by the total number of wavelengths that was already assigned at this link ($\sum_{\omega \in \Omega} \omega_l$). The mathematical formalization is as follows: $(\omega_l^*)/(\sum_{\omega \in \Omega} \omega_l)$. Intuitively, it shows whether a particular wavelength is used much more than the other ones.

5. The number of available wavelengths at time τ, computed as follows: $|\Omega| - |\Omega_\tau^{used}|$, where Ω_τ^{used} is the number of wavelengths already used at time τ.

This set of features is summarized in Table 1.

Table 1 Features of each link in the graph-structured state.

Feature	Description	Type	Mathematical Definition
f_1	Wavelength assigned to the link	One-hot encoding vector	ω_l
f_2	Remaining service time of traffic request	Vector	$\tau(r) - \delta(r)$
f_3	Betweenness centrality	Scalar	$\sum_{s,t \in E}(\sigma_{s,t}(l)/\sigma_{s,t})$
f_4	Popularity of the wavelengths	Vector	$(\omega_l^*)/(\sum_{\omega \in \Omega} \omega_l)$
f_5	Number of available wavelengths	Scalar	$\lvert\Omega\rvert - \lvert\Omega_\tau^{used}\rvert$

Action Considering all possible paths between any selection of nodes of the underlying network graph results in large action space, even for small networks. There are numerous combinations of links to create a path between every two nodes in a network. For instance, the average action space size considering all routes for the topologies used in this work is around 4500. This reduces the capabilities of the proposed algorithm as it needs to approximate Q-values for every possible path and every wavelength. To overcome this challenge, we restrict the action space to a limited number of paths. For every source-destination node pair in the graph, the first k shortest paths are generated. Then, the agent is restricted to select a path only from this subset. Upon the arrival of incoming traffic demand, the agent receives the new representation of the state and the information on the current lightpath request (source, destination, and duration). Besides the

path to choose (among the k options), the agent needs to select a wavelength to assign. Finally, the agent has also to reject the request. Therefore, the number of possible actions is $\mathcal{W}k+1$. It should be pointed out that the action state is different based on incoming traffic requests since the source and destination nodes of requests are different. In summary, the agent can, at each timestep, either assign a lightpath (ℓ_ω^κ) to the current request arriving at that time, or (2) reject the request ($a_\emptyset$). Assuming a request from the source u to the destination v, the set of possible actions is $a \in \left\{\ell_1^1(u,v),\ldots,\ell_{\mathcal{W}}^k(u,v)\right\} \cup \{a_\emptyset\}$. In this paper, we considered 4 paths ($k=4$) and 10 wavelengths ($\mathcal{W}=10$). This gives a reasonable action space of 41 actions.

Transition Three operations are realized during a transition to the next step: (1) updating the network and the features with the information of the new request, (2) updating the current timestep, (3) releasing links if the duration of the related requests is reached. We emphasize that the transition function is deterministic.

Reward The goal of the dynamic RWA is to minimize the blocked traffic connections, or similarly, maximize admitted ones. To this end, a positive reward is returned each time the agent is able to fulfill a request. This corresponds to any action excepting the rejection ($a_\emptyset$). It is formalized in Equation (1).

$$r = \begin{cases} 1 & \text{if } a \neq a_\emptyset \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

Finally, we empirically observed that a large number of transitions are driven by the action corresponding to the rejection of a request. This is mainly due to the fact that when there is no lightpath available, the decision is automatically a rejection. The realization of such an action is trivial and is not interesting to learn. We define a *non-trivial state* as a state where for an incoming request, there is at least one lightpath available. We propose to use only non-trivial states for the training. Besides, the agent cannot reject a request if there is at least one lightpath available.

3.5. Neural Network Architecture

The neural network we designed is composed of two main modules, namely a GNN and a fully-connected neural network that outputs the predicted Q-values. It is illustrated in Figure 4. The goal of the GNN is to embed the line-graph given in input as a vectorized representation. This process is first carried out with an operation referred to as *message passing*, and the way it is performed depends on the specific architecture used. With a *graph attention network* (i.e., the architecture we selected), the vectorized representation of a node is computed by taking and aggregating the information of the neighboring nodes, while weighting the importance of each neighbor. We refer this combination of message-passing and aggregation to as a *graph filtering* step. In our case, this operation is repeated three times. At the end of this process, an *embedding* (i.e., a vectorized representation) is obtained for each node. In order to obtain a representation on the *graph-level*,

a subsequent step is required to aggregate the information of each node. This is referred to as *graph pooling* operation. We use the standard *max-pooling* on this purpose (i.e., taking only the maximum value among each piece of information from the nodes). This graph embedding is finally used as input of a standard fully-connected neural network. The output is a Q-value $Q(s, a)$, giving the score to execute a specific action a from the current state s. The fully-connected neural network is configured with 32 hidden layers, each with 128 hidden units, and rectified linear units (ReLU) (Glorot et al. 2011) are used as activation function.

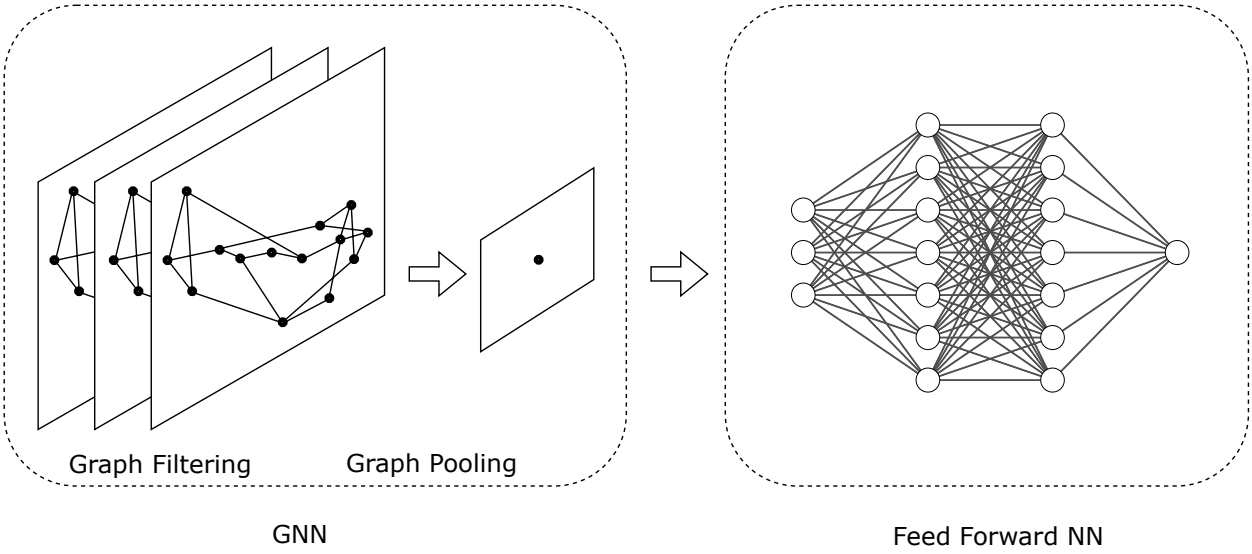


Figure 4 Illustration of the architecture: a graph neural network and fully-connected neural network.

3.6. Learning Algorithm

The next step is to train the neural network to give it the ability to estimate accurately the Q-value associated to a state-action pair. The core algorithm is *deep Q-learning* Mnih et al. (2013), improved with several mechanisms. The pseudo-code is illustrated in Algorithm 1.

First, a memory, referred to as *replay memory buffer* in the literature, is initialized (Line 8). Its goal is to store the history of the actions performed by the agent and use them to drive the learning. Then, the line-graph is built from the network topology as explained in Section 3.4 (Line 9). The training is carried out for a fixed number of episodes (Lines 10 to 21). An episode corresponds to a simulation of the network usage with incoming requests. At each episode, the environment is first initialized (Line 11). Each episode is executed until a fixed number of requests have been issued (Lines 12 to 21). Without loss of generality, we set this threshold to 100. Each request asks for a path from a source node n^1 to a destination node n^2. Each time a request is issued (Line 13), the agent has to perform an action. As explained previously, the agent can assign a lightpath to any of the K shortest paths from n^1 to n^2 (if the lightpath is available), or reject the request.

Algorithm 1: Pseudo-code of the DQN algorithm for training the model.

1 ▷ **Pre:** G is the topology of the network considered.
2 ▷ **Pre:** E is the number of episodes.
3 ▷ **Pre:** Θ is the size of an episode (i.e., the number of requests).
4 ▷ **Pre:** B is the batch size.
5 ▷ **Pre:** K is the number of shortest paths to compute between each pair of nodes.
6 ▷ **Pre:** $\mathcal{W}$ is the number of wavelengths available.
7 ▷ **Pre:** $\mathbf{w}$ are all the weights used to configure the neural network (randomly initialized).
8 $M := \emptyset$ ▷ *Initializing a memory dedicated to store the transitions performed*
9 $G' := \mathsf{lineGraphTransformation}(G)$ ▷ *Building the line-graph transformation of the network*
10 **for** e **from** 1 **to** E **do**
11 $\quad s_1 := \mathsf{InitializeEnvironment}(G')$ ▷ *Initializing the environment and getting the initial state*
12 $\quad$ **for** t **from** 1 **to** Θ **do**
13 $\quad\quad (n^1, n^2, \tau, \delta) := \mathsf{getRequest}(s_t)$ ▷ *Getting the request associated to the current state*
14 $\quad\quad A := \left\{ \ell_\omega^k(n^1, n^2) \;\middle|\; k \in \{0, \dots, K\} \wedge \omega \in \{0, \dots, \mathcal{W}\} \right\} \cup a_\emptyset$ ▷ *Computing the action space*
15 $\quad\quad a_t := \mathsf{epsilonGreedySelection}(A, s_t, \mathbf{w})$ ▷ *Selecting an action with an ε-greedy policy*
16 $\quad\quad s_{t+1} := \mathsf{transition}(s_t, a_t)$ ▷ *Performing the transition to the next state*
17 $\quad\quad$ **if** $a_t = a_\emptyset$ **then**
18 $\quad\quad\quad \mathsf{goToNextRequest}()$ ▷ *Skipping the current request for the training*
19 $\quad\quad r_t := \mathsf{getReward}(s_t, a_t)$ ▷ *Computing the reward associated to the action*
20 $\quad\quad M := M \cup (s_t, a_t, r_t)$
21 $\quad\quad \mathbf{w} := \mathsf{updatingWeights}(\mathbf{w}, B, M)$ ▷ *Updating the model with samples from the memory*
22 **return** $\mathbf{w}$

We highlight that all the K shortest paths are precomputed before the training for each pair of nodes. Among this set of actions, the agent has to select one of them. This is carried out thanks to a *ε-greedy policy* (Line 15). Briefly, the agent will perform, most of the time, the available action with the highest Q-value. As described in Section 3.5, these Q-values are obtained by the neural network parametrized by the weights $\mathbf{w}$. On a few occasions, the agent will not follow this greedy strategy and will take a random action instead. This mechanism is used to increase the exploration ability of the agent. Once the action is selected, it is executed and the next state is generated (Line 16). When the action consists in rejecting the request (i.e., there is no lightpath available), it is not considered for the training (Lines 17 to 18). Otherwise, the reward associated to the action is collected (Line 19), and the memory is updated with this information (Line 20). The memory has a bounded size of 500,000 samples. When the limit is reached, the oldest sample is removed. The

weights are then updated through backpropagation in the neural network using B samples taken randomly from the memory (Line 21). This value corresponds to the *batch size*, and we set it to 32. This procedure is executed each time the agent has to perform an action and for each episode. The parameters $\mathbf{w}$ are finally returned (Line 22).

Concerning the learning, it is carried out by backpropagation using Adam optimizer (Kingma and Ba 2015). The loss function to minimize corresponds to the squared difference between the estimated Q-value and the real one, as given by Q-learning update function (Watkins and Dayan 1992). Once the loss is calculated during the forward pass, the weights of all the neural networks are updated during the backpropagation. This includes the weights of the dense layers, and the weights of the graph neural network as well. The weights are initialized randomly using Xavier scheme (Glorot and Bengio 2010). The learning rate has been experimentally set to 7×10^{-7}. Larger values resulted in an unstable learning, and smaller values resulted in slow convergence. The memory buffer has been implemented as a *prioritized experience replay* (Schaul et al. 2016). Briefly, each sample has a priority corresponding to its estimated impact on the training. When the memory is exceeded, samples with the lowest priority are replaced first. The memory is initialized with samples obtained by an agent acting randomly. This is a common strategy which has the benefit to easily increase the exploration ability of the agent at the beginning of the training. Besides, the ε-greedy policy is improved with a dynamic exploration rate. At the initialization, the agent has a probability of 100% to take a random action. This probability decreases with a rate of 0.995 starting at episode 200 and until the minimum value of 1% is reached. Finally, a target network is added (Mnih et al. 2015).

3.7. Testing Phase

Once the model is trained, it can be used to solve new instances of the dynamic RWA problem. This is illustrated in Algorithm 2 for a specific instance. First, the environment is built (Lines 6 and 7), and a value dedicated to store the number of accepted requests is initialized. Second, the episode is executed. When a request is issued, the agent will always select the action with the highest Q-value (Line 12). These values are obtained thanks to the neural network parametrized with the weights $\mathbf{w}$ during the training phase (Algorithm 1). The score is then updated with the reward signal (Line 15). When all the requests are issued, the final score (i.e. number of admitted incoming traffic connections) is returned (Line 16). This corresponds to the value of the optimization problem we want to solve.

4. Experimental Results

This section evaluates the performance of our approach and discusses the corresponding results obtained.

Algorithm 2: Pseudo-code of the algorithm for solving the dynamic RWA problem.

▷ **Pre:** G is the topology of the network considered.
▷ **Pre:** Θ is the size of an episode (i.e., the number of requests).
▷ **Pre:** K is the number of shortest paths to compute between each pair of nodes.
▷ **Pre:** $\mathcal{W}$ is the number of wavelengths available.
▷ **Pre:** $\mathbf{w}$ are the weights previously trained in Algorithm 1.
$G' := \mathsf{lineGraphTransformation}(G)$ ▷ *Building the line-graph transformation of the network*
$s_1 := \mathsf{InitializeEnvironment}(G')$ ▷ *Initializing the environment and getting the initial state*
$p := 0$
for t **from** 1 **to** Θ **do**
  $(n^1, n^2, \tau, \delta) := \mathsf{getRequest}(s_t)$ ▷ *Getting the request associated to the current state*
  $A := \left\{ \ell_\omega^k(n^1, n^2) \;\middle|\; k \in \{0, \ldots, K\} \wedge \omega \in \{0, \ldots, \mathcal{W}\} \right\} \cup a_\emptyset$ ▷ *Computing the action space*
  $a_t := \mathsf{argmax}_{a \in A} Q(s_t, a, \mathbf{w})$ ▷ *Selecting an action with an ε-greedy policy*
  $s_{t+1} := \mathsf{transition}(s_t, a_t)$ ▷ *Performing the transition to the next state*
  $r_t := \mathsf{getReward}(s_t, a_t)$ ▷ *Computing the reward associated to the action*
  $p := p + r_t$
return p

4.1. Setup

The neural network introduced in Section 3.5 is implemented in Python 3.8 using PyTorch (Paszke et al. 2019). Training is carried out on a single GPU (NVIDIA V100 Volta, 32GB memory). Prior to training, we built a *validation set* to keep track of the performances during the training phase and to compare the model with baselines. This validation set is comprised of 100 randomly generated instances. The mathematical programming formulation of the model is solved by Gurobi 9.5.0 with a system containing 16 GBs of RAM with 4 core Intel Core i7 processors.

The random instances are generated by producing stochastic arrival times and service hold times for incoming traffic requests. We generate arrival times based on a Poisson process with a mean of 0.6. The duration of requests follows an exponential distribution with a mean of 100. These values have been set experimentally, in order to maintain the load of the network at an acceptable level. Having a very low load or a very high load results in situations where the majority of traffic requests is admitted or rejected and which are then trivial to solve. We select these values to keep the problem challenging for the agent.

Prior to training, we solve these instances with widely used baselines introduced in the literature. For routing, we select the *shortest path selection* (SPFF) and *alternate routing* (ARFF) methods (Wang et al. 2011). For the wavelength selection, the *first-fit heuristic* is considered in

both cases. This strategy favors selecting more wavelengths and greedily aims to reduce congestion and the blockage of the incoming requests. We also developed a simple *random selection* strategy (RANDOM) to evaluate the other methods against it. Each time a request is issued, the lightpath selected (i.e. a route and a wavelength) is generated randomly among the possible options. In addition to these baselines, we considered the (unrealistic) situation where we have perfect information on the arrival time and duration for all the incoming traffic requests. Although this problem is different than the dynamic RWA problem as there is no stochasticity to handle, the solution obtained provides an upper bound of the performances that can be reached. We formalized this fully-observable variant as an integer program.

Let $G=(V,E)$ be a graph generated from an underlying topology of the network, where $v \in V$ is a node, and $\ell \in E$ is a fiber link, joining a pair of nodes. Let D be the set of all incoming traffic requests, and let Δ_d be the set of all possible lightpaths for a traffic request $d \in D$. We define a binary variable $x_{d,\delta}$ for each $d \in D$ and for each $\delta \in \Delta_d$. The value of $x_{d,\delta}$ is 1 if the traffic request d is assigned to lightpath δ and 0 otherwise. In addition, we add a binary variable $y_{d,t}$ for each $d \in D$ and for each $t \in \mathcal{T}$, where $\mathcal{T}$ is the set of all periods of the temporal horizon. The value of $y_{d,t}$ is 1 if the request d is active in the network at time t. For each request $d \in D$, we have complete information on their starting time s_d, and duration h_d. We define a parameter $q_{\ell,\delta}$ which is equal to 1 if the fiber link ℓ is required for the lightpath δ and 0 otherwise. In addition, parameter $r_{\omega,\delta}$ is equal to 1 if the wavelength $\omega \in \Omega$ is required for the lightpath δ and 0 otherwise. The corresponding mathematical programming model is formalized as follows.

$$\text{maximize} \sum_{d \in D} \sum_{\delta \in \Delta_d} x_{d,\delta} \tag{2}$$

$$\text{subject to} \sum_{\delta \in \Delta_d} x_{d,\delta} \leq 1 \qquad \forall d \in D \tag{3}$$

$$h_d \sum_{\delta \in \Delta_d} x_{d,\delta} = \sum_{t=s_d}^{s_d+h_d+1} y_{d,t} \qquad \forall d \in D \tag{4}$$

$$\sum_{d:s_d \in \{t-h_d,t\}} \sum_{\delta \in \Delta_d} q_{\ell,\delta}\, r_{\omega,\delta}\, x_{d,\delta}\, y_{d,t} \leq 1 \qquad \forall t \in \mathcal{T}, \forall \ell \in E, \forall \omega \in \Omega \tag{5}$$

$$x_{d,\delta} \in \{0,1\} \qquad \forall d \in D, \forall \delta \in \Delta_d \tag{6}$$

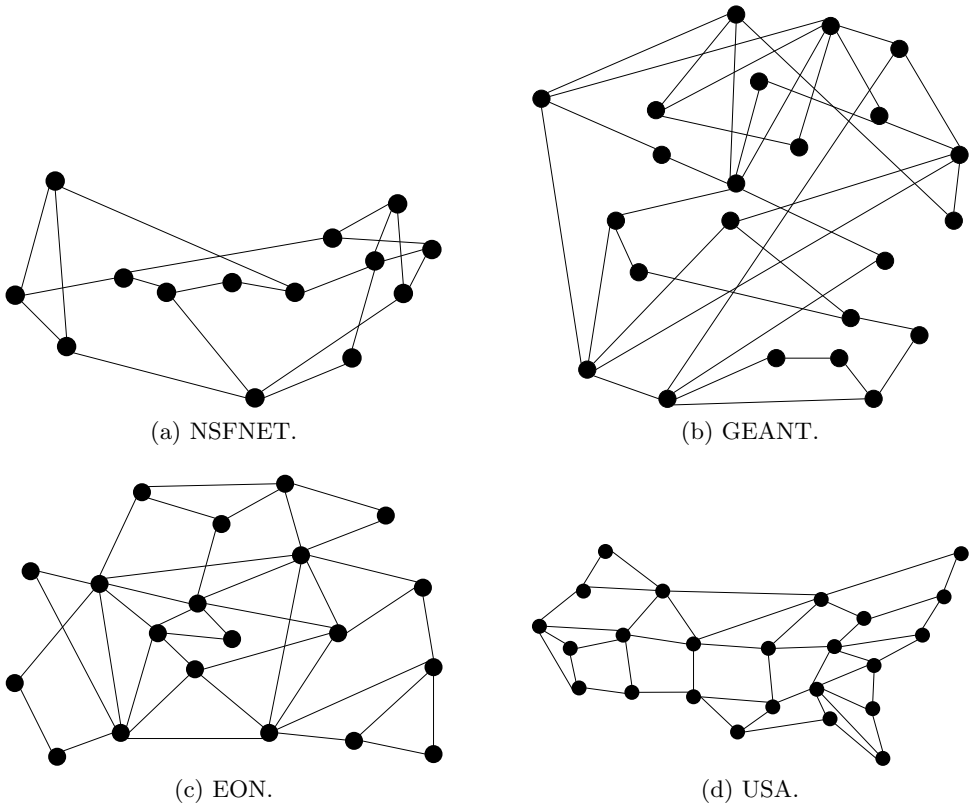
$$y_{d,t} \in \{0,1\} \qquad \forall d \in D, \forall t \in \mathcal{T} \tag{7}$$

The objective is to maximize the number of accepted requests (Equation (2)). Constraint (3) ensures that at maximum one lightpath is selected for each incoming traffic request. As we consider a dynamic RWA, where the traffic requests are released after their service time is finished, we need to keep track of the times when the requests are utilizing the network resources. It is carried out

by Constraint (4). Constraint (5) ensures that each fiber link and each wavelength is used by at most one lightpath for the requests currently on the network at each time period. The notation of the outer sum indicates that we sum over all the requests (d) having their start time (s_d) in the range between $t - h_d$ and t.

4.2. Results on the Learning Phase

Figure 5 presents the *National Science Foundation Network* (NSFNET) (Claffy et al. 1994), the *Gigabit European Academic Network* (GEANT), the *European Optical Network* (EON), and the *USA network* (USA). Characteristics of these topologies (number of nodes, number of links, and average number of connections of a node) are summarized in Table 2. For each incoming traffic connection request, the source and the destination are selected randomly. The arrival of incoming traffic connection requests follows a Poisson process and each request stays on the network for a duration that follows an exponential distribution.



(a) NSFNET. (b) GEANT.

(c) EON. (d) USA.

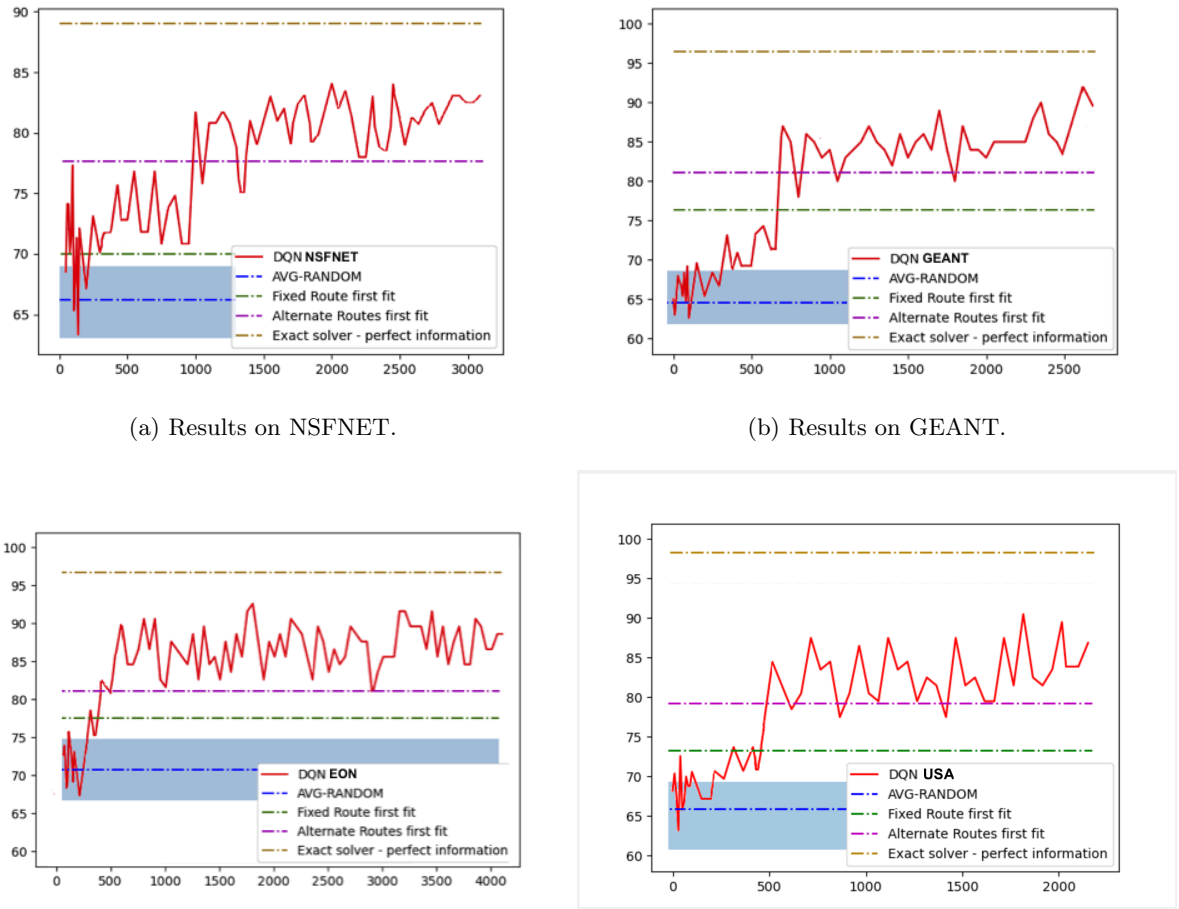
Figure 5 Illustration of the topologies used in the experiments.

A dedicated agent is trained for each topology. The learning curves are illustrated in Figure 6. These curves show how the average number of accepted requests on the validation set evolves during the training. Performances of the baselines are also reported. As no training is carried out for the

Table 2 Summary of the characteristics of each topology.

Topology	Number of nodes	Number of links	Average degree of a node
NSFNET	14	21	3.0
GEANT	24	37	3.1
EON	20	36	3.6
USA	24	44	3.7

baselines, the performances remain constant. Interestingly, we observe that the agent is consistently able to achieve better performances than the baselines when trying the maximize the number of accepted requests. A specific model is trained and saved for each topology. They are referred to as



(a) Results on NSFNET.

(b) Results on GEANT.

(c) Results on EON.

(d) Results on USA.

Figure 6 Evaluating the performance of the DQN algorithm (red) during training for the four topologies. The x-axis corresponds to the training epoch, and the y-axis corresponds to the average number of accepted requests on the 100 instances on the validation set.

DQN-NSFNET, DQN-GEANT, DQN-EON, and DQN-USA, respectively. For the selection, we took the model that is revealed to be the best performing on the validation set (i.e., when the red curve

is at the highest point). These four models are then reused for the next experiments. The exact number of iterations required to train these models depends on the specific topology. Looking at Figure 6, we can see three phases in each training curve.

- *Phase 1:* the model is oscillating close to a poor reward (episodes 0 to 1000 for DQN-NSFNET).

In this phase, the model is mainly exploring the state space to obtain relevant training samples.

- *Phase 2:* the model is improving fast and obtains better rewards (episodes 1000 to 1200 for DQN-NSFNET). In this phase, the model is able to infer promising actions.
- *Phase 3:* the model is stagnating on a higher reward quality (episodes 1200 to 3000 for DQN-NSFNET). In this phase, it is harder for the model to improve the reward.

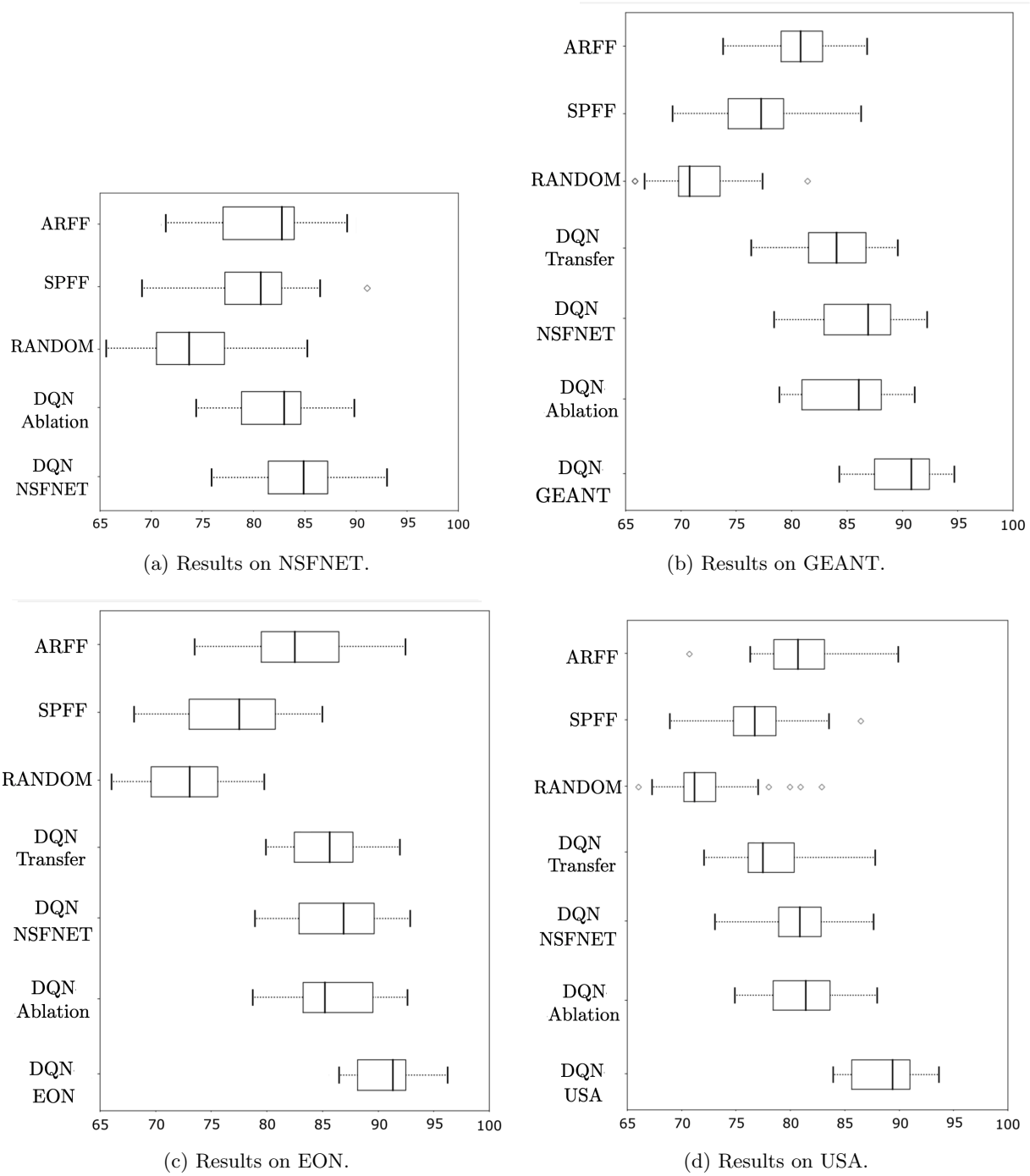
The models selected have always reached the third phase.

4.3. Performances on NSFNET

This experiment studies the performance of DQN-NSFNET on 100 new instances generated on NSFNET. The random distributions are similar than the ones considered for the training. We highlight that this still corresponds to scenarios that are not seen during the training. Results are presented in Figure 7(a) by means of blox-plots. Each plot shows the percentages (and the spread) of accepted traffic connection requests for the 100 simulation scenarios on NSFNET. The results obtained are aligned with what has been observed during the training. The trained agent (DQN-NSFNET) is able to accept more requests than the baselines and has the highest median value. In the best case, less than 10% of the requests are blocked. The median of the results of the DQN algorithm is roughly 3% better than one of the next best-performing approach (ARFF), and roughly 12% better than random policy (RANDOM). As a last comment, we also observe that always following the shortest path greedily (SPFF) is sub-optimal.

4.4. Performances on Other Networks

A similar experiment is carried out on the three other networks. The agent DQN-NSFNET is challenged on 100 new instances generated on GEANT, EON, and USA. The goal of this experiment is to analyze the ability of the agent to generalize to a network not seen during the training. Results are summarized in Figures 7(b), (c), and (d). Interestingly, DQN-NSFNET is still able to compete with the previous baselines (ARFF and SPFF). This shows that the methodology we propose is able to generalize to other networks, which was one the goal pursued by the use of graph neural networks. To further extend this analysis, we also considered the agents trained specifically on these networks (DQN-GEANT, DQN-EON, and DQN-USA, respectively). As expected, these agents reached the best results. This indicates that, when information about the topology of the network is available, it is better to use it for training the agent.



(a) Results on NSFNET.

(b) Results on GEANT.

(c) Results on EON.

(d) Results on USA.

Figure 7 **Evaluating the performances of the different approaches on 100 new instances with box plots. The x-axis shows the percentage of accepted requests for each method.**

4.5. Ablation Study: Removing the Attention Mechanism

To assess the relevance of the GAT architecture, we propose an ablation study consisting in removing this module and using a simple fully-collected neural network instead. This new baseline hence directly trains the agent without inputting any embedding of the topology and consequently does

not capture the relational information on the network structure. This variant is referred to as DQN-Ablation. Results are summarized in Figure 7 for each topology. As expected, it achieves inferior results compared to the GAT counterpart.

4.6. Analysis: Transfer Learning Ability

Transfer learning (Weiss et al. 2016) refers to the ability of a model trained for a specific task to be reused for another task without retraining it from scratch. To analyze this ability, we first took the agent trained on NSFNET (DQN-NSFNET) and saved all its learnable weights. Second, the agent was retrained on each other topology using the learned weights instead of a random initialization, and with a significantly lower number of episodes (capped at 1000). The results obtained are represented in Figure 7 for each network, and are referred to by the name DQN-Transfer. We observe that the results obtained are still better than the non-learned baselines but nevertheless slightly worse than the standard model DQN-NSFNET. A possible explanation is that the retraining time was not enough for the model to achieve better performances.

4.7. Analysis: Robustness against Stochastic Fluctuations

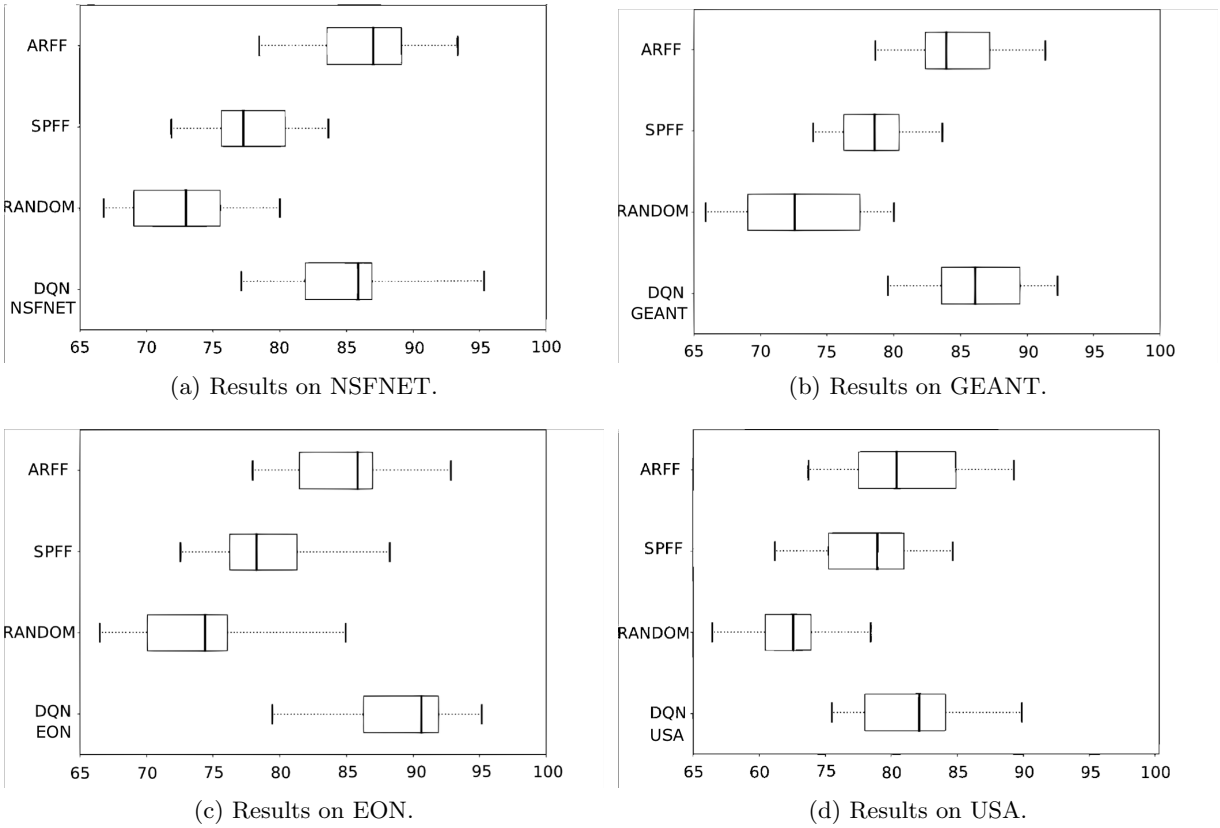
In the previous experiments, the stochasticity of the arrival and departure of the traffic requests followed the same distribution in both training and testing. To evaluate the robustness of the agent against distributional shifts, we considered a new simulated environment where the incoming traffic requests are subject to different random distributions in the test phase. This scenario is more representative of a realistic situation. To this end, we consider a scenario where the arrival times and duration of requests follow the distributions presented in Equation (8).

$$arrival \sim \mathsf{uniform}(5, 50), \quad duration \sim \mathsf{uniform}(5, 150) \tag{8}$$

Figure 8 shows the performance of DQN-NSFNET, DQN-GEANT, DQN-EON, and DQN-USA, on the four networks with the new random distributions. Although the general performances of the learned models have decreased, it remains competitive compared to the other baselines, which indicates the robustness of the approach to stochastic fluctuations.

4.8. Analysis: Characteristics of Learned Solutions

This last set of experiments aims to analyze the characteristics of the routing and wavelength assignation solution obtained by the agent. First, Figure 9 shows the average number of admitted incoming requests over the 100 instances considered for the evaluation. The models tested correspond to the results of DQN-NSFNET, DQN-GEANT, DQN-EON, and DQN-USA presented in Figure 7 for their specific topology. As one can expect, the network gets more congested over time as the load is increased and fewer possibilities of routing are present. Despite the fluctuations, we



(a) Results on NSFNET.

(b) Results on GEANT.

(c) Results on EON.

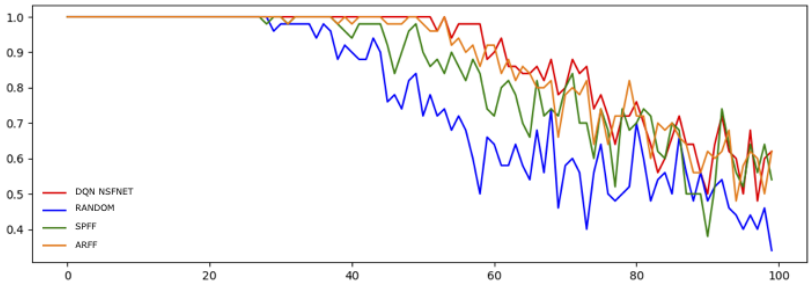
(d) Results on USA.

Figure 8 Evaluating the performances of the agent on 100 new instances with box-plots. The instances follow a different random generation than the ones seen during the training.

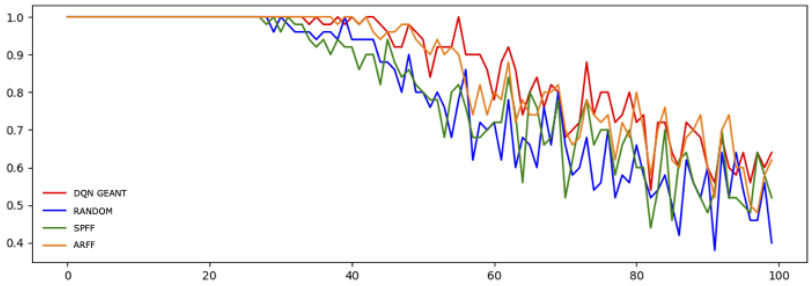
can observe that the trained agents are able to accept more requests, which is the goal intended. Second, Figure 10 shows the average length of lightpaths, in terms of the number of fiber links involved, selected by the agents over the 100 instances considered for the evaluation. A first observation is that the agent tends to select lightpaths that are not the shortest ones, compared to the shortest path heuristic (SPFF). As better results in terms of acceptance rate are obtained, this confirms the hypothesis that taking always the shortest path available is sub-optimal.

5. Conclusion

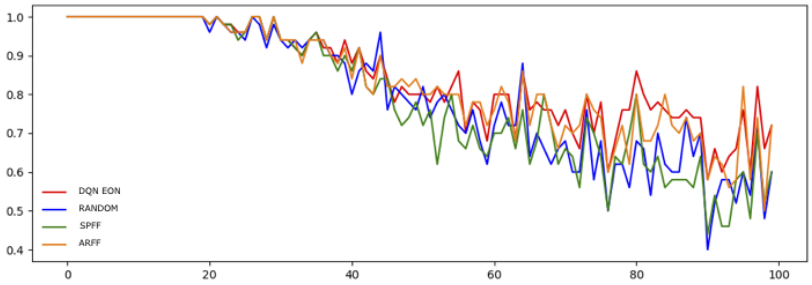
With the rapid developments in communication systems, all-optical networks are becoming increasingly complex. Unfortunately, routing and resource assignment within such networks are complex combinatorial optimization tasks. The difficulty is even more important when stochastic aspects are involved. Based on this context, the contribution of this paper is a method based on deep reinforcement learning and graph neural network to solve the dynamic routing and wavelength assignment problem in the context of wavelength-division multiplexing networks. A specificity of the method we introduce is that it simultaneously selects routes and wavelengths. By inferring the information from the elements and the topology of the physical network, the approach aims



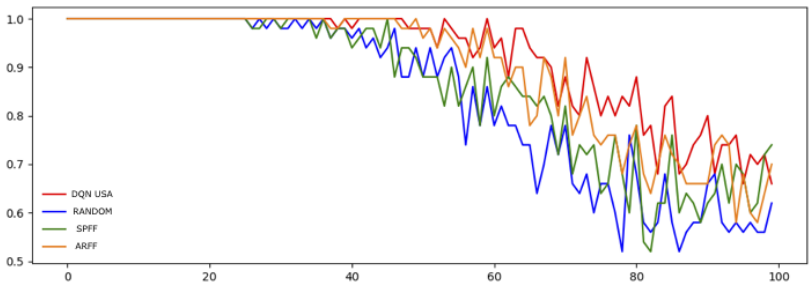
(a) Results on NSFNET.



(b) Results on GEANT.

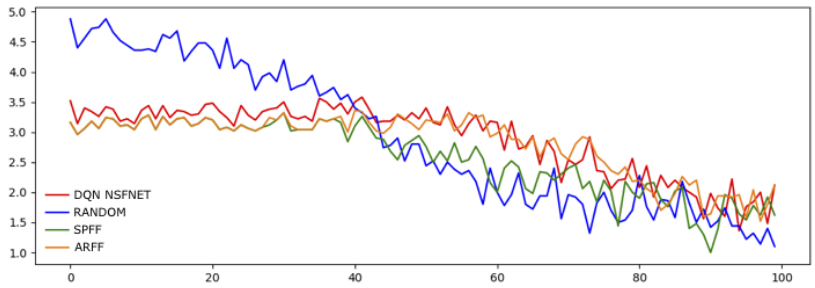


(c) Results on EON.

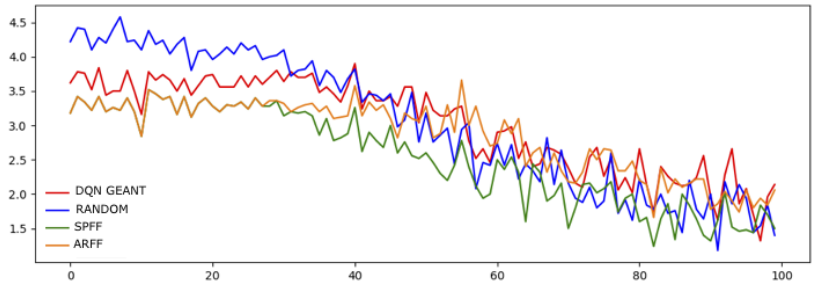


(d) Results on USA.

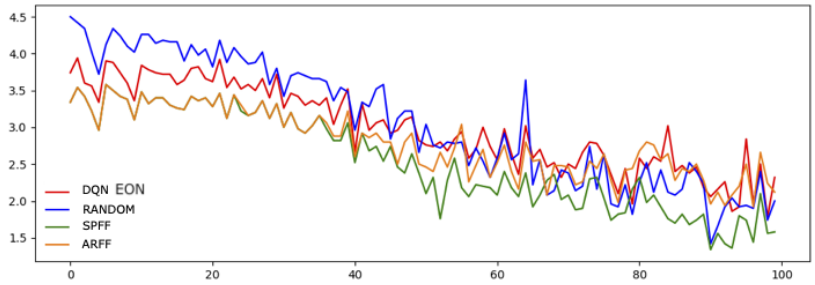
Figure 9 Visualization of the average admitted traffic for each request over 100 instances. The x-axis corresponds to the number of currently issued requests, and the y-axis shows the proportion of accepted requests.



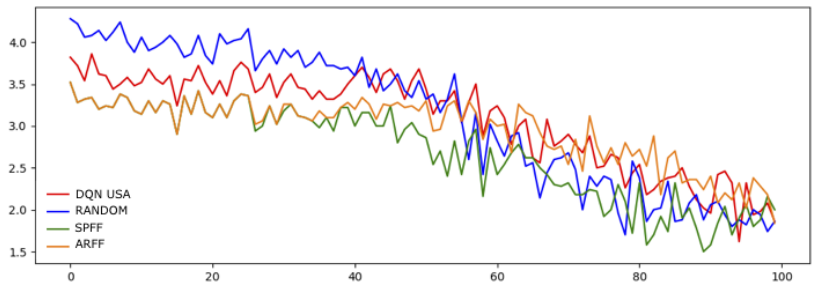
(a) Results on NSFNET.



(b) Results on GEANT.



(c) Results on EON.



(d) Results on USA.

Figure 10 Visualization of the average length of a lightpath for each request over 100 instances. The x-axis corresponds to the number of currently issued requests, and the y-axis shows the average length.

to minimize the blockage rate of incoming requests in the long-term. Empirical results show that the proposed algorithm performs superior to widely used heuristics and demonstrate the ability of the approach to generalize to unseen scenarios. This benefit has been enabled thanks to a graph attention network, a neural architecture able to extract knowledge from graphs given as input.

References

Almasan P, Suárez-Varela J, Rusek K, Barlet-Ros P, Cabellos-Aparicio A (2022) Deep reinforcement learning meets graph neural networks: exploring a routing optimization use case. *Computer Communications* 196:184–194.

Amin R, Rojas E, Aqdus A, Ramzan S, Casillas-Perez D, Arco JM (2021) A Survey on Machine Learning Techniques for Routing Optimization in SDN. *IEEE Access* 9:104582–104611, ISSN 2169-3536 VO - 9, URL http://dx.doi.org/10.1109/ACCESS.2021.3099092.

Cappart Q, Chételat D, Khalil E, Lodi A, Morris C, Veličković P (2021) Combinatorial optimization and reasoning with graph neural networks. *arXiv preprint arXiv:2102.09544* .

Chen X, Li B, Proietti R, Lu H, Zhu Z, Yoo SJB (2019) DeepRMSA: A Deep Reinforcement Learning Framework for Routing, Modulation and Spectrum Assignment in Elastic Optical Networks. *J. Lightwave Technol.* 37(16):4155–4163, URL http://jlt.osa.org/abstract.cfm?URI=jlt-37-16-4155.

Chu X, Li B, Zhang Z (2003) A dynamic RWA algorithm in a wavelength-routed all-optical network with wavelength converters. *IEEE INFOCOM 2003. Twenty-second Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE Cat. No.03CH37428)*, volume 3, 1795–1804, ISBN 0743-166X VO - 3, URL http://dx.doi.org/10.1109/INFCOM.2003.1209202.

Claffy KC, Braun HW, Polyzos GC (1994) Tracking long-term growth of the NSFNET. *Communications of the ACM* 37(8):34–45, ISSN 0001-0782.

Daryalal M, Bodur M (2020) Stochastic RWA and Lightpath Rerouting in WDM Networks URL http://arxiv.org/abs/2012.10084.

Di Ianni M (1998) Efficient delay routing. *Theoretical Computer Science* 196(1-2):131–151, ISSN 0304-3975, URL http://dx.doi.org/10.1016/S0304-3975(97)00198-9.

Duhovnikov S, Schupke DA, Lehmann G, Fischer T, Rambach F (2006) Dynamic RWA for All Optical Networks Using Linear Constraints for Optical Path Feasibility Assessment. *2006 European Conference on Optical Communications*, 1–2, ISBN 1550-381X VO -, URL http://dx.doi.org/10.1109/ECOC.2006.4801328.

Freeman LC (1977) A Set of Measures of Centrality Based on Betweenness. *Sociometry* 40(1):35–41, ISSN 00380431, URL http://dx.doi.org/10.2307/3033543.

Gilmer J, Schoenholz SS, Riley PF, Vinyals O, Dahl GE (2017) Neural message passing for quantum chemistry. *International conference on machine learning*, 1263–1272 (PMLR), ISBN 2640-3498.

Glorot X, Bengio Y (2010) Understanding the difficulty of training deep feedforward neural networks. *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, 249–256 (JMLR Workshop and Conference Proceedings).

Glorot X, Bordes A, Bengio Y (2011) Deep sparse rectifier neural networks. *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, 315–323 (JMLR Workshop and Conference Proceedings).

Gu R, Yang Z, Ji Y (2020) Machine learning for intelligent optical networks: A comprehensive survey. *Journal of Network and Computer Applications* 157:102576, ISSN 1084-8045, URL http://dx.doi.org/https://doi.org/10.1016/j.jnca.2020.102576.

Huimin G, Wen Y, Liang W, Zhengtang L (2017) Research on dynamic routing and wavelength assignment algorithm for optical networks. *2017 IEEE 2nd Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*, 221–224, ISBN VO -, URL http://dx.doi.org/10.1109/IAEAC.2017.8054010.

Jaumard B, Daryalal M (2017) Efficient Spectrum Utilization in Large Scale RWA Problems. *IEEE/ACM Transactions on Networking* 25(2):1263–1278, URL http://dx.doi.org/10.1109/TNET.2016.2628838.

Jaumard B, Meyer C, Yu X (2005) When is wavelength conversion contributing to reducing the blocking rate? *GLOBECOM '05. IEEE Global Telecommunications Conference, 2005.*, volume 4, 2078–2083, URL http://dx.doi.org/10.1109/GLOCOM.2005.1578031.

Jaumard B, Meyer C, Yu X (2006) How much wavelength conversion allows a reduction in the blocking rate? *J. Opt. Netw.* 5(12):881–900, URL http://dx.doi.org/10.1364/JON.5.000881.

Kamiński K, Ludwiczak J, Jasiński M, Bukala A, Madaj R, Szczepaniak K, Dunin-Horkawicz S (2022) Rossmann-toolbox: a deep learning-based protocol for the prediction and design of cofactor specificity in rossmann fold proteins. *Briefings in Bioinformatics* 23(1):bbab371.

Kingma DP, Ba JL (2015) Adam: A method for stochastic optimization. *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*.

Kiran YV, Venkatesh T, Ram Murthy CS (2007) A Reinforcement Learning Framework for Path Selection and Wavelength Selection in Optical Burst Switched Networks. *IEEE Journal on Selected Areas in Communications* 25(9):18–26, URL http://dx.doi.org/10.1109/JSAC-OCN.2007.028806.

Martin I, Troia S, Hernandez JA, Rodriguez A, Musumeci F, Maier G, Alvizu R, de Dios O (2019) Machine Learning-Based Routing and Wavelength Assignment in Software-Defined Optical Networks. *IEEE Transactions on Network and Service Management* 16(3):871–883, URL http://dx.doi.org/10.1109/TNSM.2019.2927867.

Mnih V, Kavukcuoglu K, Silver D, Graves A, Antonoglou I, Wierstra D, Riedmiller M (2013) Playing Atari with Deep Reinforcement Learning URL http://dx.doi.org/10.48550/arxiv.1312.5602.

Mnih V, Kavukcuoglu K, Silver D, Rusu AA, Veness J, Bellemare MG, Graves A, Riedmiller M, Fidjeland AK, Ostrovski G, et al. (2015) Human-level control through deep reinforcement learning. *nature* 518(7540):529–533.

Mohan N, Kaushal P (2020) Dynamic routing and wavelength assignment for efficient traffic grooming. *Journal of Optical Communications* URL http://dx.doi.org/doi:10.1515/joc-2019-0309.

Paszke A, Gross S, Massa F, Lerer A, Bradbury J, Chanan G, Killeen T, Lin Z, Gimelshein N, Antiga L, et al. (2019) Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems* 32.

Pinto A (2002) Optical Networks: A Practical Perspective, 2nd Edition. *Journal of Optical Networking* 1(6):219–220, URL http://opg.optica.org/jon/abstract.cfm?URI=jon-1-6-219.

Pointurier Y, Heidari F (2007) Reinforcement learning based routing in all-optical networks with physical impairments. *2007 Fourth International Conference on Broadband Communications, Networks and Systems (BROADNETS '07)*, 928–930, URL http://dx.doi.org/10.1109/BROADNETS.2007.4550535.

Pouya H (2018) *New Models and Algorithms in Telecommunication Networks*. Ph.D. thesis, Concordia University, URL https://spectrum.library.concordia.ca/id/eprint/984939/.

Ramaswami R, Sivarajan KN (1995) Routing and wavelength assignment in all-optical networks. *IEEE/ACM Transactions on Networking* 3(5):489–500, URL http://dx.doi.org/10.1109/90.469957.

Randhawa R, Sohal JS (2010) Blocking probability analysis of survivable routing in WDM optical networks with/without sparse-wavelength conversion. *Optik* 121(5):462–466, ISSN 0030-4026, URL http://dx.doi.org/https://doi.org/10.1016/j.ijleo.2008.08.007.

Rodriguez A, Fernández W, Ramírez L (2016) RWA: Novel Heuristic Algorithm for Optical Networks with Dynamic Traffic BT - Advanced Computer and Communication Engineering Technology. 1–10 (Cham: Springer International Publishing), ISBN 978-3-319-24584-3.

Rusek K, Suárez-Varela J, Almasan P, Barlet-Ros P, Cabellos-Aparicio A (2020) RouteNet: Leveraging Graph Neural Networks for Network Modeling and Optimization in SDN. *IEEE Journal on Selected Areas in Communications* 38(10):2260–2270, ISSN 1558-0008 VO - 38, URL http://dx.doi.org/10.1109/JSAC.2020.3000405.

Rusek K, Suárez-Varela J, Mestres A, Barlet-Ros P, Cabellos-Aparicio A (2019) Unveiling the Potential of Graph Neural Networks for Network Modeling and Optimization in SDN. *Proceedings of the 2019 ACM Symposium on SDN Research*, 140–151, SOSR '19 (New York, NY, USA: Association for Computing Machinery), ISBN 9781450367103, URL http://dx.doi.org/10.1145/3314148.3314357.

Scarselli F, Gori M, Tsoi AC, Hagenbuchner M, Monfardini G (2009) The Graph Neural Network Model. *IEEE Transactions on Neural Networks* 20(1):61–80, URL http://dx.doi.org/10.1109/TNN.2008.2005605.

Schaul T, Quan J, Antonoglou I, Silver D (2016) Prioritized experience replay. *4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings*.

Suárez-Varela J, Mestres A, Yu J, Kuang L, Feng H, Cabellos-Aparicio A, Barlet-Ros P (2019) Routing in optical transport networks with deep reinforcement learning. *J. Opt. Commun. Netw.* 11(11):547–558, URL http://dx.doi.org/10.1364/JOCN.11.000547.

Sutton RS, Barto AG (2018) *Reinforcement learning: An introduction* (MIT press), ISBN 0262352702.

Swaminathan A, Chaba M, Sharma DK, Ghosh U (2021) GraphNET: Graph Neural Networks for routing optimization in Software Defined Networks. *Computer Communications* 178:169–182, ISSN 0140-3664, URL http://dx.doi.org/https://doi.org/10.1016/j.comcom.2021.07.025.

Veličković P, Casanova A, Liò P, Cucurull G, Romero A, Bengio Y (2018) Graph attention networks. *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings*.

Wang Y, Cao X, Pan Y (2011) A study of the routing and spectrum allocation in spectrum-sliced Elastic Optical Path networks. *2011 Proceedings IEEE INFOCOM*, 1503–1511, ISBN 0743-166X, URL http://dx.doi.org/10.1109/INFCOM.2011.5934939.

Watkins CJCH, Dayan P (1992) Q-Learning. *Machine Learning* 8(3-4):279–292, ISSN 0885-6125.

Weiss K, Khoshgoftaar TM, Wang D (2016) A survey of transfer learning. *Journal of Big data* 3(1):1–40.

Xu Z, Tang J, Meng J, Zhang W, Wang Y, Liu CH, Yang D (2018) Experience-driven Networking: A Deep Reinforcement Learning based Approach. *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*, 1871–1879, URL http://dx.doi.org/10.1109/INFOCOM.2018.8485853.

Zang H, Jue JP, Mukherjee B (2000) A review of routing and wavelength assignment approaches for wavelength-routed optical WDM networks. *Optical networks magazine* 1(1):47–60.

Zang H, Jue JP, Sahasrabuddhe L, Ramamurthy R, Mukherjee B (2001) Dynamic lightpath establishment in wavelength routed WDM networks. *IEEE Communications Magazine* 39(9):100–108, ISSN 1558-1896 VO - 39, URL http://dx.doi.org/10.1109/35.948897.

Zhou J, Yuan X (2002) A study of dynamic routing and wavelength assignment with imprecise network state information. *Proceedings. International Conference on Parallel Processing Workshop*, 207–213, ISBN 1530-2016 VO -, URL http://dx.doi.org/10.1109/ICPPW.2002.1039732.

Zhou X, Lu W, Gong L, Zhu Z (2012) Dynamic RMSA in elastic optical networks with an adaptive genetic algorithm. *2012 IEEE Global Communications Conference (GLOBECOM)*, 2912–2917, URL http://dx.doi.org/10.1109/GLOCOM.2012.6503559.