Routing in Optical Transport Networks with Deep Reinforcement Learning

José Suárez-Varela[1,*], Albert Mestres[1], Junlin Yu[2], Li Kuang[2], Haoyu Feng[2], Albert Cabellos-Aparicio[1], and Pere Barlet-Ros[1]

[1] *Computer Architecture Department, Universitat Politècnica de Catalunya, Spain*
[2] *Huawei Technologies Co., LTD.*
[*] *Corresponding author: jsuarezv@ac.upc.edu*

Compiled February 19, 2020

Deep Reinforcement Learning (DRL) has recently revolutionized the resolution of decision-making and automated control problems. In the context of networking, there is a growing trend in the research community to apply DRL algorithms to optimization problems such as routing. However, existing proposals failed to achieve good results, often under-performing traditional routing techniques. We argue that the reason behind this poor performance is that they use straightforward representations of networks. In this paper, we propose a DRL-based solution for routing in Optical Transport Networks (OTN). Contrary to previous works, we propose a more elaborated representation of the network state that reduces the level of knowledge abstraction required to DRL agents and eases to capture the singularities of network topologies. Our evaluation results show that using our novel representation, DRL agents achieve better performance and learn how to route traffic in OTNs significantly faster compared to state-of-the-art representations. Additionally, we reverse engineered the routing strategy learned by our DRL agent and, as a result, we found a routing algorithm that outperforms well-known traditional routing heuristics.

http://dx.doi.org/10.1364/jocn.XX.XXXXXX

1. INTRODUCTION

In the last few years, we have witnessed significant advances in Deep Reinforcement Learning (DRL) that are revolutionizing the way we can resolve decision-making and automated control problems [1, 2]. In this context, there is a growing interest in the computer network community to apply DRL-based solutions to network optimization problems. All this, with the goal of building self-driving networks [3].

In this paper, we address the application of DRL to perform online routing in Optical Transport Networks (OTN). This is an optimization problem where DRL-based solutions may be appropriate given their ability to both make fast decisions (i.e., routing every traffic demand as it arrives) and devise smart strategies to save network resources in the long-term. All this, dealing with the uncertainty in the generation of future traffic, which has a stochastic nature.

Recent works have already used reinforcement learning to address related problems such as routing in optical networks [4], IP routing [5] or QoS provisioning [6]. However, they failed to achieve good results given their lack of generalization capability. This means that they are not able to make correct decisions when facing network scenarios not explored during the training phase.

In DRL algorithms, two main elements must be defined: (i) the *observation space* and (ii) the *action space*. The observation space describes the state of the environment (i.e., the current state of the network in our case). The action space, on the other hand, describes the modifications that the DRL agent makes over the environment. In our case, the action represents changes to be applied to the routing configuration. The network state is typically represented as a matrix containing the per-link utilization [4]. Likewise, existing proposals usually limit the dimensionality of the action space by using straightforward representations, such as the per-link weights for link-state routing algorithms (e.g., OSPF) [5, 7]. In contrast, we argue that, in order to outperform existing routing solutions, it is not enough to leverage recent advances in DRL algorithms as done in previous works, but it is even more important to carefully design more elaborated representations of the observation and action spaces that can better represent the singularities of network topologies and simplify the learning process to the DRL agent.

In the light of the above, in this paper we propose a representation that enables to reduce the level of knowledge abstraction required to the agent and facilitates to better capture the singularities of network topologies. Consequently, it contributes to achieve better performance. Our approach is to design state and action representations that convert the challenging problem of routing traffic over OTN to an easier problem to resolve. This makes it easier for the agent to learn the overall utilization and the dependencies among the end-to-end paths in the network topology. As a result, this also facilitates the detection of possible singularities in the network, such as potential bottlenecks.

The remainder of this paper is structured as follows. In Section 2, we describe the DRL-based routing problem addressed in this paper. Section 3 presents a review of state-of-the-art DRL-based state/action representations for network-related problems and describes the representation proposed in this paper. Section 4 includes a description of the DRL-based solution implemented. In Sections 5, 6 and 7, we make an extensive evaluation our DRL-based solution in some realistic OTN scenarios. This includes a systematic hyperparameter evaluation to optimize the performance of our DRL agent, and a comparison with previous DRL-based solutions [4, 5, 7] and well-known routing heuristics in the state-of-the-art. Our evaluation experiments use specific traffic models that simulate real-world network scenarios in a custom-built simulator. Lastly, in Section 8 we reverse engineer the routing policy learned by our DRL agent and, based on this analysis, we find a routing algorithm that outperforms traditional heuristics. Note that portions of this work were previously presented at the Optical Fiber Communications Conference (OFC) [8] and the IEEE International Conference on Communications (ICC) [9] in 2019. However, this paper includes a formal description of our DRL-based solution (Section 4), a considerably more extensive evaluation (in Sections 5, 6 and 7) and a reverse engineering analysis of the routing policy learned by the DRL agent (in Section 8).

2. DRL-BASED ROUTING SCENARIO IN OTN

In this section, we describe an OTN scenario where a DRL agent makes routing decisions at the electrical domain. Although the scenario was already introduced in an earlier conference version of this paper [9], we include it here for completeness as it provides necessary background to understand the remainder of this paper. In this scenario, the agent operates over a logical topology composed by Reconfigurable Optical Add-Drop Multiplexer (ROADM) nodes and some predefined lightpaths connecting them (see Fig. 1). Then, the role of the DRL agent is to route incoming traffic demands through particular sequences of lightpaths (i.e., end-to-end paths). Since the agent works at the electrical domain, traffic demands are considered requests of Optical Data Units (ODUk) signals defined in the ITU-T Recommendation G.709 [10]. These ODUk signals, that may belong to different clients, are then multiplexed into Optical Transport Units (OTUk), which are data frames including Forward Error Correction (FEC). The OTUk frames are finally transmitted over sequences of lightpaths (optical channels) in the OTN. Note that in this scenario the DRL agent acts oblivious of the mechanisms related to the optical domain (e.g., physical impairments).

In this paper, we define the routing problem as devising a certain strategy to route new source-destination traffic demands with the aim of saving network resources in the long-term. Note that this is a challenging task for the DRL agent since it should learn during the training phase some singularities of network topologies such as potential bottlenecks as well as understand the underlying dependencies among end-to-end paths. Moreover, this learning process is also hampered by the uncertainty in the generation of future traffic demands, which is stochastic.

This problem can be modeled as a Markov Decision Process (MDP). A MDP is defined by the tuple $\{s, a, T, r, \gamma, s_0\}$. The State

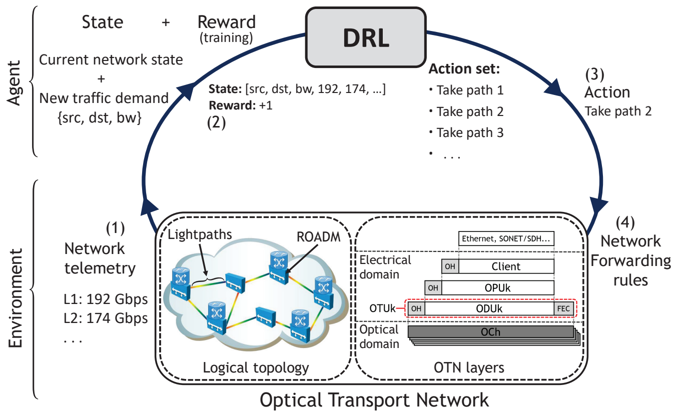


Fig. 1. Schematic representation of the DRL agent's operation in the OTN routing scenario.

(s) must represent the environment unambiguously according to the action space defined. In our scenario, it must represent the current network state and some information about the traffic demand(s) to be routed. The Action (a) stands for the set of actions the agent can apply (i.e., the changes to the routing configuration). The Transition distribution ($T(s, a, s')$) defines how the environment, the network in our case, evolves after applying an action. In our scenario, the transition function models the stochastic behavior of changes in the network state as well as the generation of new traffic demands to be routed. The Reward (r) is the incentive that the agent obtains after making a decision. Its purpose is to steer the learning process towards the achievement of the optimization goal. Then, the objective is to find a policy $\pi(s)$ mapping input states to actions that maximizes the *discounted cumulative reward* $R = \sum_{t=0}^{T} \gamma^t r(s_t, a_t)$. The actions can be deterministic or stochastic (i.e., probability distribution over actions), and the discount factor $\gamma \in [0, 1)$ defines the importance of the reward obtained in future decisions. Finally, the initial state (s_0) represents an empty network with a first traffic demand to be routed.

Solving the MDP requires to evaluate all the possible combinations of state-action pairs, and this is computationally very expensive when the state has high dimensionality. In our scenario, the number of possible network states is finite and proportional to the number of lightpaths, their capacity and the granularity considered for the lightpaths' utilization (i.e, number of utilization intervals). For standard networks, this is typically an extremely high number of states. For instance, in the experiments we perform in Sections 5, 6 and 7 the number of possible network states is above 10^{100}.

An alternative to solve the MDP is using Reinforcement Learning (RL) together with sophisticated generalization techniques. This makes it possible to extract knowledge from visited states that can be used for unexplored states. In order to achieve this level of generalization, recent DRL algorithms propose the use of Deep Neural Networks (DNN). Thus, with a proper training, such neural networks are able to model how to act successfully in regions not explored in advance.

Fig. 1 represents the basic operation of the DRL agent in the OTN routing scenario. We assume that the DRL agent has access to telemetry information of the network, which is aligned with current architectural trends, such as Software-Defined Networking (SDN) [11] or Overlay Networking. Thus, when there is a new traffic demand to be routed (ODUk signal), this is communicated to the agent (step 1). Then, the agent generates a new state

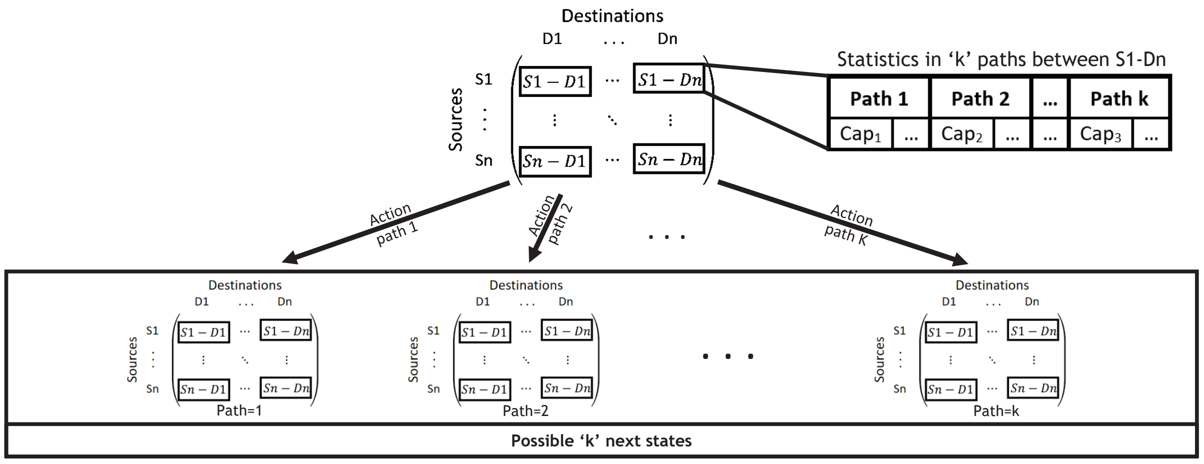


Fig. 2. Scheme of the state representation proposed.

representation that will be the input of its decision model (i.e., a neural network model). This state representation should include information about the current network state and the new traffic demand (step 2). With this input, the DRL agent selects an action that involves making a routing decision for the new demand (step 3). Lastly, the resulting action is translated into a set of forwarding rules that are installed in some (ROADM) network devices (step 4). Additionally, during the training phase the DRL agent explores different routing strategies and receives a reward after applying every action. This enables to learn the routing policy that leads to better cumulative reward in the long-term.

3. PROPOSED REPRESENTATION

In this section, we propose a novel representation to perform DRL-based online routing in OTNs. The design of this representation involves both how to define the network state (i.e., the *observation space*) of the DRL agent and the set of actions that the agent can apply (i.e., the *action space*). Note that this representation was already proposed in [9], hence much of the content in this section is already included in that paper.

A. State-of-the-art representations

Before describing our representation, we review the brief related work addressing network routing based on Deep Learning techniques and the representations they proposed. Some works like [3, 5, 7], represent the network state directly with a traffic matrix (i.e., the traffic of every source-destination pair). This information allows the agent to define a global routing policy considering the overall traffic demand in the network. Then, the action of the agent is to select the link weights of an external algorithm (e.g., softmin routing [5], OSPF-like [7]) that defines the final routing policy. Although these representations obtain reasonable performance in simple routing problems (e.g., link-weight selection), they exhibited poor results when applied to more complex problems, such as flow-based routing, even in some cases falling behind more classical routing algorithms.

Other approaches propose making routing decisions for every new traffic demand considering the current state of the network. For instance, to the best of our knowledge Deep-RMSA [4] is the only work addressing DRL-based online routing specifically for optical networks. To do this, they represent the network state with the links' utilization. Particularly, they model fiber links as binary arrays with a number of frequency slots that can be available (1) or occupied (0). Note that they address the Routing, Modulation and Spectrum Assignment (RMSA) problem in Elastic Optical Networks, while in this paper we address routing at the electrical domain over a logical topology with lightpaths already provisioned. Alternatively, [12] proposes to represent the network state with a matrix containing the traffic demand aggregated in every router for a number of time intervals. Both proposals, Deep-RMSA [4] and [12], define a discrete action space for the agent where each option represents the selection of a path among a number of candidate paths. The main drawback of these representations for the problem addressed in this paper is that the DRL agent must abstract knowledge from the link-level features represented in the observation space to the path-level options present in the action space.

We claim that, using straightforward representations of the network state, such as the links' utilization or the traffic matrix it is not feasible to achieve high performance in network routing. For example, these representations do not include information about the network topology and the interdependencies between the links that form an end-to-end path, which is critical to routing. Note that the alternative of including this information as an adjacency matrix would not solve the problem, as it would be very difficult for the DRL agent to learn these relationships from a raw matrix (e.g., the network paths and the links they share).

B. Description of the proposed representation

In contrast to state-of-the-art proposals, our approach is to propose a more elaborated state/action representation that facilitates the agent to learn how to efficiently route the traffic. In other words, our representation should help the DRL agent to achieve generalization.

A key aspect to consider in the design of such a representation is that, in network scenarios, we can provide a simple estimate of how the network state will change after routing a new traffic demand. For instance, if we assume that we know the bandwidth request of an incoming traffic demand (as in the OTN scenario in Sec. 2), it is easy to estimate the resulting utilization of the links after allocating that demand to a specific end-to-end path. This means that we can leverage this information and provide this knowledge directly to the agent. This considerably simplifies the problem, because otherwise the agent would have to learn these relationships from exploration. However, there is still the challenge of how to choose the best routing policy considering the uncertainty of future traffic demands, which follow a stochastic generation process. In this context, after proper training, the DRL agent should acquire some knowledge from the network and learn routing strategies that effectively deal with such uncertainty in the traffic. For instance, it may detect potentially critical links and try to prevent bottlenecking them.

In the light of the above, we propose the following representation for the DRL agent. Instead of considering link-level statistics as in previous works (e.g., utilization of the links in the network [4]), we propose the use of statistics at the level of *end-to-end paths*. This way, the agent does not need to infer knowledge from the link-level to the path-level. Regarding the action space, we propose a set of discrete actions where each action corresponds to the selection of a specific end-to-end path. Particularly, we consider that, for each new traffic demand $\{source, destination, bandwidth\}$, the agent can select one path among a list of "k" candidate paths (e.g., "k" shortest paths) that connect the source and the destination of such demand. With respect to the state representation, the agent is provided with some relevant statistics of the "k" candidate paths of all the source-destination pairs in the network. In Fig. 2, the top matrix represents a scheme of the current network state. This matrix contains the current statistics of the "k" end-to-end paths for each source-destination pair. With this representation, it is possible to compute all the next states that can be reached after applying every possible action in the current action set and provide them to the DRL agent. That is, in each epoch the input of the DRL agent will be "k" matrices (as shown in the bottom of Fig. 2), where each matrix represents the estimated path statistics (e.g., available capacity) after allocating the current traffic demands to each of the "k" candidate paths. In other words, the proposed representation provides the agent with a set of matrices that describe the consequences of applying every possible action. Note that the cost to compute the input state is proportional to the number of possible actions considered. However, limiting the actions to "k" paths allows us to control the dimensionality and the cost to compute the state. Lastly, note that it is also required to include some information of the current traffic demand in the state (i.e., source, destination, bandwidth...).

To the best of our knowledge, there are not previous proposals that use state representations with statistics of end-to-end paths to address DRL-based networking problems.

4. DESCRIPTION OF THE DRL-BASED SOLUTION

This section describes the complete DRL solution integrating the representation proposed in Section 3-B. The DRL agent receives as input the current network state including a new traffic demand, and the objective is to select a discrete action $a(t) \in [0..k]$ that represents a specific end-to-end path for the demand. Note that the number of candidate paths (k) and the criteria to select them (e.g., shortest paths) may have an impact on the final performance achieved by the agent. Formally, the DRL agent aims to find a policy $\pi_\theta(s|a)$ modeled by a Deep Neural Network (DNN) with some weights and biases (θ) that are updated during training to maximize the discounted cumulative reward.

Besides the design of a good state/action representation for the DRL agent, it is also important the selection of a proper DRL algorithm that well suits the nature of the problem. This involves for instance the exploration bias of the algorithm, which controls the trade-off between the performance that can be potentially achieved and the training time to converge to the solution. The more biased is the exploration, the faster should the agent converge. However, having high exploration bias may result in policies whose performance is very far from the optimal solution of the Markov Decision Process (MDP) problem. Note that the

```
Input: Initial parameters for policy $\pi_\phi$ and value $V_\phi$
1  for $i = 1, 2, 3, \ldots$ until convergence do
2      - Collect trajectories $\mathcal{D}_i$ on policy $\pi_i = \pi(\phi_i)$
3      - Compute advantages $\hat{A}_i^{GAE}$ using the critic estimates $V_{\phi_i}$
4      - Compute the policy gradient $\hat{g}_i$ using $\hat{A}_i^{GAE}$
5      - Compute the KL-divergence Hessian-vector product function:
           $f(V) = \hat{H}_i v$
6      - Apply conjugate gradient method to calculate:
           $x_i \approx \hat{H}_i^{-1} \hat{g}_i$
7      - Compute the proposed policy update step:
           $\Delta_i \approx \sqrt{\frac{2\delta}{x_i^T \hat{H}_i x_i}} x_i$
8      - Backtracking line search to obtain the final policy update (actor):
           $\theta_{i+1} = \theta_i + \alpha^j \Delta_i$
9      - Update the critic neural network ($\phi_{i+1}$) using $\hat{A}_i^{GAE} \in \mathcal{D}_i$ with
         an Adam optimizer
10 end
```

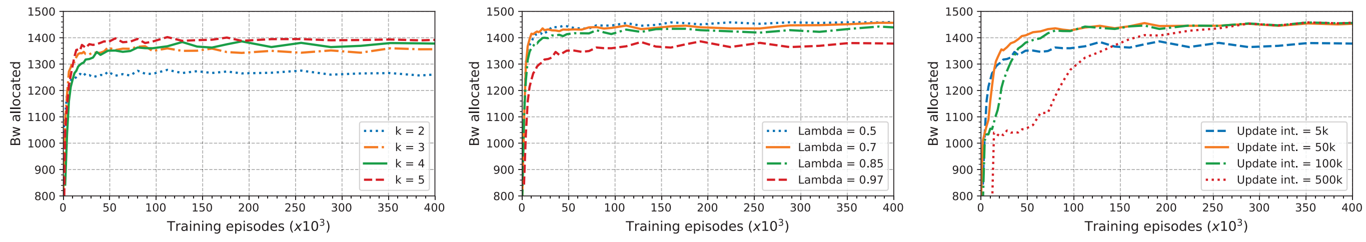
Algorithm 1: TRPO Actor-critic training process

optimal MDP solution represents an upper-bound of the performance that DRL agents can achieve.

In order to select a DRL algorithm for our agent, we made some preliminary experiments with different agents implementing the following algorithms: TRPO [13], PPO [14], DDPG [15], PCL [16], A3C [17] and ACER [18]. To this end, we used the default implementations of these algorithms in ChainerRL (v0.3.0) [19]. Lastly, we found that the *Trust Region Policy Optimization* (TRPO) algorithm clearly outperformed the other algorithms in terms of performance and time to converge to the solution in the OTN routing scenario presented in this paper (Sec. 2). Note that this does not necessarily mean that the other algorithms tested may not potentially achieve similar or even better performance than TRPO after a fine-tuning process.

TRPO is a recent reinforcement learning algorithm based on the classic Natural Policy Gradient [20] algorithm. Unlike primary policy gradients (e.g., REINFORCE [21]), TRPO introduces a number of sophisticated mechanisms that provide stability to the training and avoids the well-known vanishing or exploding gradient problems. This algorithm can be applied to both continuous and discrete action spaces. In our case, we consider that the agent aims to learn a stochastic policy $\pi_\theta(s|a) = P[a|s;\theta]$ based on the discrete action space proposed in Section 3-B. Additionally, the implementation used in this paper includes two more mechanisms that further contribute to stabilize the training and, consequently, achieve better performance: (i) it implements an *actor-critic model* where the policy and the value estimates are modeled by separate neural networks, and (ii) it uses the *Generalized Advantage Estimator* (GAE) [22] as advantage function, which enables to reduce the variance of policy gradient estimates at the expense of some bias.

Algorithm 1 describes the DRL agent's training process. Firstly, a number of episodes is executed (line 2) following the current policy (π_θ) modeled by the actor neural network. Thus, for each timestep in these episodes a tuple (s_t, a_t, r_t, s_{t+1}) is stored in a buffer ($\mathcal{D}_i$). Subsequently, advantage estimates are computed for each timestep sample in $\mathcal{D}_i$ (line 3) using the Generalized Advantage Estimator $\hat{A}_t^{GAE} = \sum_{l=0}^{\infty} (\lambda\gamma)^l \delta_t^V$. Where $\lambda \in [0,1]$ adjusts the bias-variance trade-off of the estimator, $\gamma \in [0,1)$ is the discount factor and δ_t^V is the temporal difference error that can be computed with the following expression: $\delta_t^V = r_t + \gamma V(s_{t+1}) - V(s_t)$. At this point, the value estimates of the critic network $V_\phi(s)$ are used to compute δ_t^V. Then, using these advantage estimates the actor network (π_ϕ) is updated via the TRPO update algorithm [13] (lines 4-8). Finally, the critic is



(a) Number of paths **(b)** λ exploration parameter of TRPO **(c)** Update interval

Fig. 3. Hyperparameter evaluation in NSFNET. The y-axis represents the avg. bandwidth allocated over 4,000 evaluation episodes.

updated by minimizing the Mean Squared Error (MSE) between the current value predictions V_{ϕ_i} and the values updated with the new advantage estimates $V_{\phi_i} + \hat{A}_i^{GAE}$. To this end, it uses an Adam optimizer [23], which is an extension of the classic Stochastic Gradient Descent (SGD) method that incorporates adaptive learning rate.

5. SCENARIO AND PARAMETRIZATION

In this section we first describe the scenario where we evaluate the DRL-based solution described in Section 4, which includes the state/action representation proposed in Section 3-B, to route traffic demands in OTNs. Then, we perform an evaluation of some relevant hyperparameters to optimize the performance achieved by the DRL agent.

A. Evaluation scenario

The objective of the DRL agent is to efficiently route traffic demands in OTNs. Each traffic demand is defined by the tuple $\{source, destination, bandwidth\}$ and must be allocated in an end-to-end path in the OTN. We assume that the DRL agent manages the routing over a logical topology where the nodes are ROADM devices and the edges represent lightpaths connecting the ROADM nodes. Thus, the agent operates at the electrical domain to allocate ODUk traffic demands to sequences of lightpaths, which form end-to-end paths. For the sake of simplicity, we consider 5 different types of traffic demands (ODU0 to ODU4) whose bandwidth requirements are expressed in terms of multiples of ODU0 signals[1]. We consider that a demand is properly allocated if there was enough available capacity in all the lightpaths forming the end-to-end path selected. Traffic demands do not expire during an episode, hence episodes end when a demand do not fit into the path selected. Likewise, in order to maximize the total bandwidth allocated in the network, we define the immediate reward of the agent as the bandwidth (in ODU0 bandwidth units) of the current traffic demand if it was properly allocated, otherwise the reward is 0.

In our experiments, we train the DRL agent described in Section 4. We used two independent fully-connected neural networks respectively for the actor and critic models of the agent. Each network has two hidden layers, each one with 64 units. We made some experiments increasing the number of layers and units, but we did not see any relevant performance improvement. Note that the number of units in the input and output layers varies depending on the size of the observation and action spaces in the different experiments we perform. For the discount factor $\gamma \in [0, 1)$, we selected a value of γ=0.995. Note that γ represents the importance of the rewards obtained in future decisions (Sec. 2). Since the optimization objective in our scenario represents a long-term planning strategy, γ must be considerably high. For the DRL agent with our representation (Sec. 3-B), we pre-compute the 'k" shortest paths (by number of hops) of all the source-destination pairs in the network. Thus, each step the action space contains the "k" shortest paths that connect the source and the destination of the new traffic demand to be routed. In all the cases, we use vectors with one-hot encoding to represent the source, destination and ODUk type (i.e., bandwidth requirement) of traffic demands.

B. Hyperparameters evaluation

We perform an evaluation of some relevant hyperparameters of the DRL implementation to optimize the performance of the agent specifically for our problem environment. Particularly, we consider the following three hyperparameters: (i) the number of paths considered in the action space, (ii) the λ parameter used in the DRL algorithm, and (iii) the update interval of the actor and critic networks.

We evaluate, with a custom-built simulator, the DRL agent using our state/action representation in the 14-node NSFNET topology [24], where the topology edges represent lightpaths with a capacity of 200 ODU0 bandwidth units in both directions. We generate new traffic demands with an uniform distribution for the source, destination and ODUk type.

In order to optimize each hyperparameter, we perform independent parametric evaluations using the following initial values: (i) Number of candidate paths = 4, (ii) λ = 0.97, (iii) Update interval = 5,000 steps. The selection of these initial values is based on some preliminary experiments we made in a simpler scenario with a 6-node network topology. Note that despite each hyperparameter is optimized independently, there may be some co-dependencies among them. In this process, the selection of proper initial values is important to avoid large variations on some parameters during their independent fine-tuning process.

Number of candidate paths

We vary the number of "k" shortest paths considered in the action space. Intuitively, the more paths available, the more flexibility has the DRL agent to allocate the traffic. However, it is important to maintain a reduced number of paths since more paths involves larger dimensionality in the state, and this implies more processing cost to update the input state and higher RAM memory consumption. Also, it may imply a more complex learning process as there is a larger action space to explore.

Fig. 3a presents the results of this evaluation. The y-axis shows the average amount of bandwidth allocated (in ODU0 bw units) properly allocated by the agent over 4,000 evaluation episodes with different seeds to generate the traffic. The x-axis indicates the number of training episodes of TRPO. Here, we

[1]According to the ITU-T Recommendation G.709 [10], we define the bandwitdth requirements as follows: ODU1=2 ODU0 Bandwidth Units (BU), ODU2=8 ODU0 BUs, ODU3=32 ODU0 BUs , and ODU4=64 ODU0 BUs.

can observe that the curves saturate after few training episodes, which means that the agent converges fast to its best strategy in all the cases. It is noteworthy that, in this particular case, the learning process does not necessarily slow down as the number of candidate paths grows. Based on these results, we consider that a value of k=4 paths is sufficient, since further increasing the number of paths implies to enlarge the state (more processing cost) and it does not improve significantly the performance. Note that the optimal value of "k" may also depend on some graph-level properties of the topology (e.g., diameter, connectivity) where the DRL agent operates.

λ parameter of TRPO

As mentioned earlier in Section 4, our DRL agent uses the Generalized Advantage Estimator (GAE) to compute the advantage estimates for the TRPO policy and value updates. It enables to reduce considerably the variance of the gradient estimates, but this comes at the expense of some bias. To control the trade-off between bias and variance in the estimates there is a tunable parameter $\lambda \in [0, 1]$ in the GAE calculation $\hat{A}_t^{GAE} = \sum_{l=0}^{\infty} (\lambda\gamma)^l \delta_t^V$. Thus, λ=1 provides an unbiased estimate but introduces high variance. Conversely, λ=0 produces much lower variance, but may induce a lot of bias.

Fig. 3b shows the results of our parametric evaluation of λ. Here, we observe that reducing the λ value from 0.97 to 0.7 both improves the performance and slightly accelerates the learning process. Likewise, decreasing the value beyond λ=0.7 does not further improve the performance.

Update interval

The update interval defines the number of training steps that are executed and maintained in memory before the DRL agent updates the policy and the value functions (i.e., the actor and critic neural network models). The larger this interval is, the more data has to be stored until the neural networks are updated. Consequently, this typically implies higher consume of RAM memory.

Fig. 3c depicts the evaluation results varying the update interval. From these results, we can infer that the optimal value is an update interval of 50,000 steps in our case. With this value the agent achieves the same performance than using higher values (100,000 and 500,000). However, it learns faster and consumes less RAM memory.

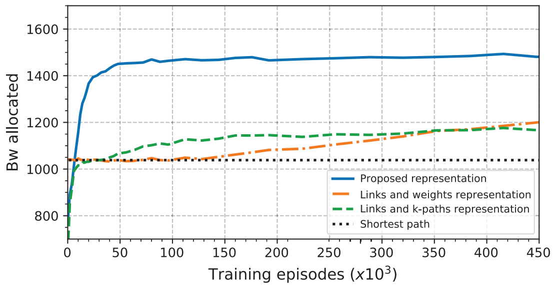
6. EVALUATION

In this section, we compare the performance of our DRL agent with respect to other DRL agents using different state/action representations commonly present in the state-of-the-art [4, 5, 7] and a traditional Shortest Path routing policy.

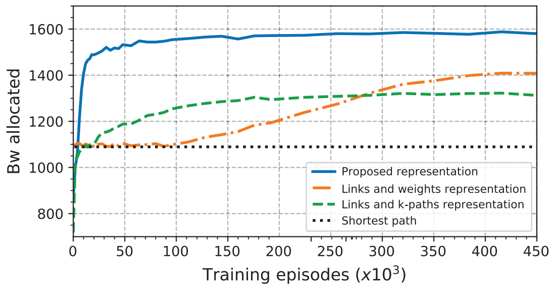
A. Evaluation against state-of-the-art representations

We evaluate the agent in two real-world network topologies: the 14-node NSFNET (used in Sec. 5-B) and the 17-node German Backbone Network (GBN) [25]. As in the previous experiments, we consider that every edge in both topologies represents lightpaths with capacity for 200 ODU0 demands on both directions.

In our evaluation, we consider the following state-of-the-art representations: (i) *Links and k-paths:* This is the simplest representation. It uses the available capacity of the lightpaths (edges in the logical topology) as the state representation and a discrete action space with k=4 candidate shortest paths (as in our representation). This representation is similar to the one they use in Deep-RMSA [4], although they address the Routing, Modulation and Spectrum Assignment (RMSA) problem in Elastic Optical Networks. (ii) *Links and weights:* This representation uses also the available capacity of the lightpaths to represent the network, but the actions consist of defining weights for the lightpaths; then, the path with lowest weight is selected (as in OSPF). This second representation uses the same action space as [5, 7].



(a) Uniform traffic distribution



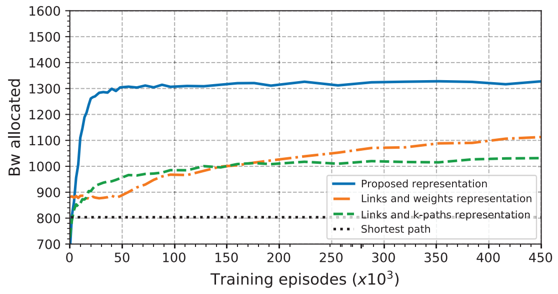
(b) Realistic traffic distribution

Fig. 4. Evaluation against state-of-the-art representations in NSFNET. The y-axis represents the avg. bandwidth allocated over 4,000 evaluation episodes.

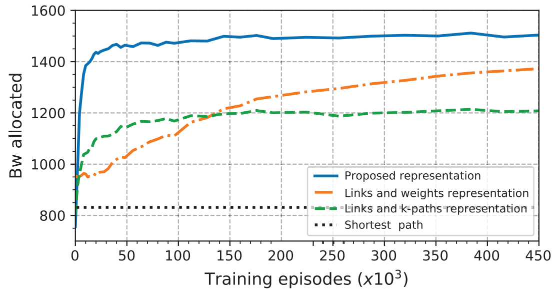
We train the agent in two scenarios with different traffic profiles. In the first scenario, we used an uniform distribution for sources, destinations and ODUk types (as in Sec. 5-B). This represents the most challenging case for the DRL agent, given that it cannot exploit particular characteristics from the traffic to direct the exploration during training. The second scenario represents a more realistic traffic distribution similar to the one we can find in real-world networks. We generate traffic with a bimodal distribution [26], in which 20% of nodes generate 80% of the traffic. Also, the distribution of the ODUk requests follows an elephant-mice distribution [27], where there is a high number of low bandwidth requests and the bulk of the traffic is generated by a reduced number of big traffic demands.

Based on the hyperparameters evaluation (Sec. 5-B), we select a value of λ=0.7 and an update interval of 50,000. For those representations with discrete path-based actions, we use k=4.

Fig. 4 shows the average bandwidth properly allocated w.r.t. the number of training episodes for the two traffic scenarios in the NSFNET topology. Each figure depicts the performance achieved by our representation, the two representations based on the state-of-the-art and the application of a traditional Shortest Path routing policy. In the scenario with realistic traffic, the agent achieves slightly higher performance, since there are more low bandwidth requests, which are easier to be allocated properly. Likewise, for both traffic distributions, we observe a similar behavior: (i) the simplest representation (Links and k-paths) outperforms the shortest path policy but its performance is quite poor compared to our representation. The proposed representa-



(a) Uniform traffic distribution



(b) Realistic traffic distribution

Fig. 5. Evaluation against state-of-the-art representations in GBN. The y-axis represents the avg. bandwidth allocated over 4,000 evaluation episodes.

tion is able to allocate approximately 26% more bandwidth with an uniform traffic distribution and 22% with realistic traffic. (*ii*) the "Links and weights" representation is able to surpass the shortest path policy and "Links and k-paths", but it learns much slower than using the other representations. For instance, in the case with realistic traffic, the representation proposed in this paper needs only 10,000 episodes to reach the same performance achieved by the weights representation after 450,000 episodes.

Fig. 5 shows the same experiments for the GBN topology. In this scenario, we can observe a similar behavior for the three representations, but also an increase in performance compared to the results obtained by the shortest path policy. This can be explained by the different distributions of the betweeness centrality of edges (i.e., lightpaths) in both topologies. For example, in the GBN topology, we observe that some edges are included in a high number of shortest paths connecting different source-destination pairs. This makes these edges more prone to become congested. We further discuss this issue in Section 6-B.

In all the evaluations we performed with link-based state representations (i.e., "Links and k-paths" and "Links and weights"), the agent achieves better performance when it applies actions to select the weights on the lightpaths ("Links and weights") than when it selects directly end-to-end paths ("Links and k-paths"). This suggests that, when representing the state with link-level features, it may be more beneficial using also link-level actions (e.g., links' weights) than applying path-level actions. Additionally, we observe that the learning process of the agent is much faster when it uses discrete path-level actions than when it defines weights at the link-level. In this context, our representation uses path-level features for both the state and action spaces, which avoids the agent to abstract knowledge from the link to the path-level. What is more, the state includes explicit information about the resulting states after applying all the possible actions, which makes the problem less complex for the agent.

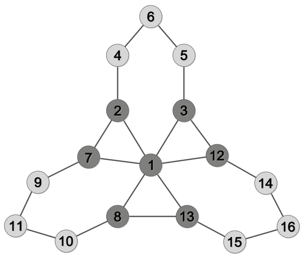


Fig. 6. Adverse network topology for shortest path routing.

B. Evaluation in an adverse scenario for Shortest Path routing

In Section 6-A, we discussed that in the GBN scenario the DRL agent achieves better performance with respect to the shortest path policy than in the NSFNET scenario. In order to further investigate this issue, in this section we analyze these topologies from a graph theory perspective and make an evaluation in a scenario that is specifically adverse to the application of the shortest path policy.

To this end, we first introduce the concept of *betweenness centrality*. This is a metric of centrality used in graph theory based on the configuration of shortest paths. Particularly, the betweenness centrality of an edge is the ratio of shortest paths that pass through that edge with respect to the total number of node pairs in the graph. Thus, in network topologies using shortest path routing, edges with high betweenness centrality are more likely to become saturated if their capacity is not scaled to that factor. That is the case of the GBN topology in Section 6-A, where all the lightpaths have the same capacity and some of them (the top 5) are included in [8.8-14.3%] of all the shortest paths connecting all the node pairs in the topology. In contrast, in the NSFNET topology, the top 5 lightpaths are within the range [6.5-8.2%] of betweenness centrality.

To show the ability of our DRL-based solution to adapt to these adverse scenarios to the shortest path policy, we make an evaluation in the topology depicted in Fig. 6. This topology is inspired by widely deployed real-world networks that form rings to connect some regions (e.g. country-level networks) and are inter-connected by a central core. Particularly, this topology contains three identical rings that are connected via a central node and has some alternative links that permit to offload traffic through the closest nodes of the other rings (e.g., from node 2 to 7). Note that all the lighpaths have a capacity of 200 ODU0 demands. In this topology, the top 5 lightpaths have a beetweenness centrality in the range [9.1-10%].

For the evaluation, we maintain the same hyperparameter values used in Section 6-A and train the DRL agent in two sce-

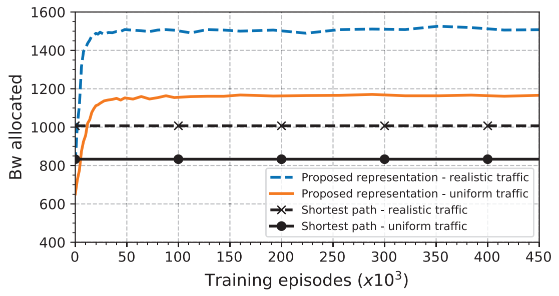
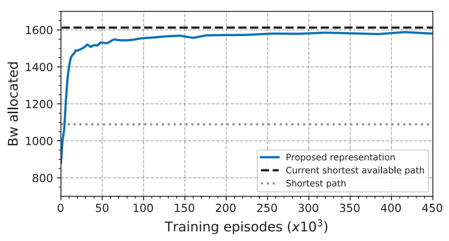
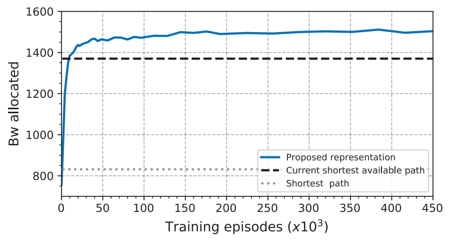


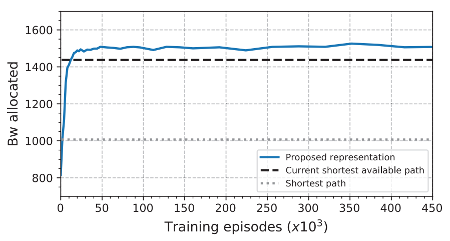
Fig. 7. Evaluation in an adverse scenario for shortest path routing. The y-axis represents the avg. bandwidth allocated over 4,000 evaluation episodes.



(a) NSFNET topology



(b) GBN topology



(c) Adverse topology for shortest path

Fig. 8. Evaluation of the DRL-based solution proposed against the SAP policy with realistic traffic distributions. The y-axis represents the avg. bandwidth allocated over 4,000 evaluation episodes.

narios with different traffic profiles. In the first scenario, we generate traffic following an uniform distribution for sources, destinations and ODUk types. In the second scenario, we further exacerbate the effect of lightpaths with high betweenness centrality by using a realistic bimodal traffic distribution in which 30% of node pairs generate 80% of the total traffic volume. In this case, the set of node pairs generating more traffic contains nodes directly connected to the top 5 lightpaths with higher betweenness centrality. This further hampers the possibility of the shortest path policy to succeed. Particularly, these pairs are (in both directions): {(1-2), (1-3), (1-7), (1-8), (1-12), (1-13), (2-3), (7-8), (12-13)}. In this latter scenario, ODUk demands follow an elephant-mice distribution [27] as in Section 6-A.

Fig. 7 shows the evaluation results. We can observe that, with uniform traffic the DRL agent achieves ≈40% more bandwidth allocated than the shortest path policy. Likewise, as we expected the DRL agent further outperforms the shortest path policy in the scenario with realistic traffic by allocating ≈52% more bandwidth. In this latter scenario, we analyzed the behavior of the DRL agent to get some hints of the strategy learned. Thus, we could observe that in some cases the agent decides to take longer paths to avoid using those lightpaths with high betweenness centrality, given that they are more prone to be congested. For instance, when there is a new traffic demand between nodes 2 and 3, the agent always selects the path [2-4-6-5-3] instead of following the shortest path [2-1-3], which traverses two lightpaths with higher betweenness centrality. This is a smart strategy given that the amount of traffic generated among node pairs in the set {2,4,6,5,3} is considerably lower in this scenario.

7. EVALUATION AGAINST CURRENT SHORTEST AVAILABLE PATH

In this section, we evaluate the DRL agent using our representation against the application of a more sophisticated heuristic we call "current Shortest Available Path" (hereafter SAP). In particular, it consists of dynamically filtering from the set of "k" candidate shortest paths those with enough capacity to support the new traffic demand and select from this subset the path with lower number of hops. This policy typically represents a performance very close to the optimal MDP solution in our OTN routing scenario. Note that it does not necessarily represents an efficient strategy in other related problems considering optical-level constraints (e.g., wavelength continuity).

We make the evaluation in the setup described in Section 5-A using the same hyperparameter configuration as in Section 6-A for our DRL-based solution. For a fair comparison, we use the same action set with k=4 candidate shortest paths for the DRL agent and the SAP policy. The evaluation is made in the three network topologies used previously: (i) NSFNET topology, (ii) GBN topology, and (iii) adverse topology for the shortest path policy.

In our experiments, we use realistic traffic distributions. For the NSFNET and GBN topologies we generate traffic with the bimodal distribution described in Section 6-A, whereas in the adverse topology for shortest path routing we use the bimodal distribution described in Section 6-B. Figs. 8a, 8b and 8c show the evaluation results respectively in the three different topologies. As we can observe, the DRL agent achieves similar performance to SAP in the NSFNET topology. However, it clearly allocates more bandwidth than SAP in the GBN topology (≈10%) and the adverse topology for shortest path (≈6%). This evidences that the DRL agent devised in these cases a smarter strategy than SAP by exploiting some knowledge acquired from topology singularities and traffic distributions.

In order to better understand the performance achieved by the SAP policy, we calculate the optimal solution in a simple scenario. Particularly, we solve the MDP problem by exploring all the possible states and compare the performance achieved with respect to SAP and the DRL agent using our representation.

Since the calculation of the MDP solution is computationally very expensive, we made the evaluation in a simple scenario with the 6-node topology used in Deep-RMSA [4]. Here, the edges correspond to lightpaths with capacity for 3 ODU0 demands in both directions. For the traffic generation, we consider that all the traffic demands are ODU0 and the sources and destinations follow an uniform distribution.

Fig. 9 depicts the evaluation results. As we expected, the SAP policy obtains practically the same performance as the optimal MDP solution. This figure also evidences that, in this scenario the DRL agent with our representation achieves a performance very close to the optimal policy. Likewise, we observe that the shortest path policy is quite far from the other strategies. Particularly, it allocates ≈50% less bandwidth on average.

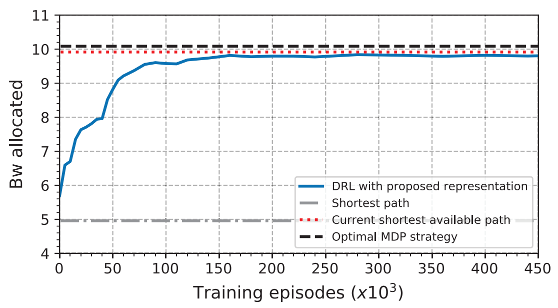


Fig. 9. Evaluation of the current Shortest Available Path (SAP) policy w.r.t. the optimal MDP solution in a simple scenario.

Table 1. Statistics summary of our reverse engineering analysis.

#	Statistic	$\frac{\%\text{ of actions}}{\text{Total actions}}$	$\frac{\%\text{ of actions}}{\text{Different actions}}$	Description
1	Different actions	59.94%	-	% of the DRL agent's actions that differ from the SAP policy (w.r.t. total number of actions over 10,000 episodes)
2	Longer path	56.97%	95.04%	% of cases where the DRL agent selects a longer path (in number of hops) than the selection of SAP
3	More available capacity	27.95%	46.62%	% of cases where the DRL agent selects a path with more available capacity than in the selection of SAP
4	Lower weighted betweenness	24.30%	40.54%	% of cases where the DRL agent selects a path with lower maximum weighted betweenness than in the selection of SAP
5	Longer path and more av. cap.	26.06%	43.47%	% of cases where the DRL agent selects a path that is longer and has more available capacity than in the selection of SAP
6	Longer path and lower wgt. btw.	22.62%	37.74%	% of cases where the DRL agent selects a path that is longer and has lower weighted betweenness than in the selection of SAP
7	Higher $min\left(\frac{av.cap}{wgt\_betweenness}\right)$	32.34%	53.96%	% of cases where the DRL agent selects a path whose minimum $av.\,cap/wgt.\,betweeness$ (over all its links) is higher than in SAP

Note that, in these latter experiments traffic demands follow an uniform distribution, hence it is not possible to exploit any meaningful information from the traffic distribution. However, in the presence of non-uniform distributions (e.g., bimodal) the optimal MDP solution may achieve better performance with respect to SAP. We do not provide results for this latter case given that the computation of the MDP solution turns to be considerably more costly.

8. REVERSE ENGINEERING OF THE ROUTING POLICY LEARNED BY THE DRL AGENT

Reverse engineering of DRL-based solutions may have two main advantages: (i) it enables to make more efficient implementations of the policy learned by DRL agents (e.g., new heuristics), and (ii) having an implementation with deterministic behavior avoids facing the uncertainty that characterizes machine learning-based solutions when acting over input states not considered during training and evaluation.

In this section we aim to gain insight into the policy learned by our DRL agent. As we could observe in Section 7, there are some cases where the DRL agent clearly outperformed the SAP policy, particularly in the GBN scenario and the adverse topology for shortest path. In this context, our approach is to focus on these scenarios and analyze the cases where the actions of our DRL agent differ from SAP. To this end, we make experiments in the adverse topology for shortest path and use the DRL agent already trained for 450,000 episodes in the scenario with realistic traffic (Fig. 8c).

For the evaluation, we run 10,000 episodes and collect some relevant statistics considering the cases where the DRL agent differs from SAP. We mainly focus on analyzing the long-term strategy of the agent. Accordingly, we only analyze routing scenarios where the network is not excessively congested so that the agent has enough flexibility to allocate the new traffic demand in multiple paths. Otherwise, when the network becomes considerably congested, decisions are not meaningful to infer the long-term strategy as there are often very few paths with enough available capacity. Thus, in our experiments we consider that episodes begin with an empty network and end when there is at least one lightpath that cannot support an ODU4 demand (i.e., minimum available capacity of 64 ODU0 bandwidth units).

Table 1 summarizes some statistics we extracted from the experiments. The first remarkable point is that the DRL agent took on average 59.94% of actions different from SAP. This suggests that the strategy followed by the agent may not be similar to the SAP policy. Additionally, we can observe that the DRL agent selected a longer path (in number of hops) than SAP in 56.97% of the cases. In other words, 95.04% of the times that the DRL agent differs from SAP, it selects a longer path (see Table 1, #2). Considering the significance of this statistic, we further analyze these cases. We then compute the following combinations of occurrences: (i) longer path with higher available capacity and (ii) longer path with lower weighted betweenness[2], (see Table 1 #5 and #6). Here, we can observe that in ≈40% of the cases the agent selects a larger path that also has higher available capacity or lower weighted betweeness.

Based on these results, we compute a more elaborated statistic that considers the ratio between the available capacity and the weighted betweeness (see Table 1, #7). Then, we can see that in 53.96% of the cases where the DRL agent differ from SAP, it takes a path with higher minimum $(av.\,cap/wgt.\,betweenness)$ ratio over all the links of the path. This suggests that the behavior of the DRL agent may be partially explained by a policy that selects the path that maximizes the minimum $(av.\,cap/wgt.\,betweenness)$ ratio over the k candidate paths. We show below an analytic expression describing such policy:

$$Path = \max_{P_k \in \mathcal{P}} \left(\min_{l \in P_k} \left(\frac{Av.\,cap\ (P_k(l))}{wgt.\,btw\ (P_k(l))} \right) \right) \qquad (1)$$

Where $\mathcal{P}$ is the the set with k candidate paths and l indexes all the links on the k-th path (P_k).

Note that this policy implicitly includes a prediction of the potential traffic that may carry each link to compute the weighted betweenness. An alternative method that does not involve traffic prediction mechanisms would be to consider instead the classic (non-weighted) betweenness metric (i.e., considering only the number of paths that may traverse a link). This would be equivalent to apply the policy in Equation 1 and assume that the traffic

[2]We define the weighted betweenness of a link as the potential number of end-to-end paths that traverse such link weighted by the amount of traffic that may carry all these paths.

Table 2. Evaluation of the heuristic based on our DRL agent

Avg. bandwidth allocated (5,000 episodes)	NSFNET	GBN	Adverse topology to shortest path
Heuristic	1642.28	1574.52	1536.81
SAP	1611.71	1368.36	1470.15
Improvement	1.90%	15.07%	4.53%

is uniformly distributed over all the src-dst pairs in the network. In order to evaluate the performance of this latter heuristic, we run 5,000 evaluation episodes in the three topologies used in previous sections with the realistic traffic distributions. Table 2 shows the average bandwidth allocated (in ODU0 units) by the heuristic and compares it with the performance achieved by the SAP policy over the same experiments. As we can observe, our heuristic inspired by the actions of the DRL agent outperforms in all the cases the SAP policy. The best case is in the GBN topology, where it allocated 15.07% more bandwidth than SAP.

9. CONCLUSION

In this paper, we address the use of Deep Reinforcement Learning (DRL) to route traffic demands in Optical Transport Networks (OTN). Contrary to the dominant trend in other domains, where the current approach is to use deeper neural networks with less elaborated features, we argue that a different approach is to use a more careful representation of the network state. This is explained by the complexity of describing link-level interdependencies and the stochastic nature of the network traffic. Conversely, recent efforts in the field of networking went in the direction of applying DRL as a black-box using simple representations of the observation/action space. We proposed a DRL-based solution that includes a representation of the network state that still has reasonable dimensionality, but can better capture the crucial relationships among the lightpaths and paths in OTN topologies. This more "engineered" representation allows the agent to learn more easily and faster. Our evaluation results, using different real-world network topologies and traffic profiles, show that our representation significantly outperforms previous proposals. Likewise, we made a reverse engineering analysis of our DRL agent in order to better understand the routing policy learned. As a result, we implemented a heuristic based on the actions of the DRL agent that outperforms well-known routing heuristics.

ACKNOWLEDGMENTS

This work has been supported by the Spanish MINECO under contract TEC2017-90034-C2-1-R (ALLIANCE) and the Catalan Institution for Research and Advanced Studies (ICREA).

REFERENCES

1. V. Mnih *et al.*, "Human-level control through deep reinforcement learning," Nature (2015).
2. D. Silver *et al.*, "Mastering chess and shogi by self-play with a general reinforcement learning algorithm," arXiv:171201815 (2017).
3. A. Mestres, A. Rodriguez-Natal, J. Carner, P. Barlet-Ros, and E. Alarcón, et al., "Knowledge-defined networking," SIGCOMM Comput. Commun. Rev. **47**, 2–10 (2017).
4. X. Chen, J. Guo, Z. Zhu, R. Proietti, A. Castro, and S. Yoo, "Deep-RMSA: A deep-reinforcement-learning routing, modulation and spectrum assignment agent for elastic optical networks," in *Optical Fiber Communications Conference and Exposition (OFC),* (2018).
5. A. Valadarsky, M. Schapira, D. Shahaf, and A. Tamar, "Learning to route," in *Proceedings of HotNets,* (2017).
6. S. C. Lin, I. F. Akyildiz, P. Wang, and M. Luo, "Qos-aware adaptive routing in multi-layer hierarchical software defined networks: A reinforcement learning approach," in *IEEE International Conference on Services Computing (SCC),* (2016), pp. 25–33.
7. G. Stampa, M. Arias, D. Sanchez-Charles, V. Muntés-Mulero, and A. Cabellos, "A deep-reinforcement learning approach for software-defined networking routing optimization," CoNEXT Stud. Work. (2017).
8. J. Suárez-Varela, A. Mestres, J. Yu, L. Kuang, H. Feng, P. Barlet-Ros, and A. Cabellos-Aparicio, "Routing based on deep reinforcement learning in optical transport networks," in *Optical Fiber Communication Conference (OFC),* (Optical Society of America, 2019), p. M2A.6.
9. J. Suárez-Varela, A. Mestres, J. Yu, L. Kuang, H. Feng, P. Barlet-Ros, and A. Cabellos-Aparicio, "Feature engineering for deep reinforcement learning based routing," in *IEEE International Conference on Communications (ICC),* (IEEE, 2019).
10. "ITU-T Recommendation G.709/Y.1331: Interface for the optical transport network," http://www.itu.int/rec/T-REC-G.709/ (2016).
11. D. Kreutz, F. M. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," Proc. IEEE **103**, 14–76 (2015).
12. B. Mao, Z. M. Fadlullah, F. Tang, N. Kato, O. Akashi, T. Inoue, and K. Mizutani, "Routing or computing? the paradigm shift towards intelligent computer network packet transmission based on deep learning," IEEE Transactions on Comput. **66**, 1946–1960 (2017).
13. J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *Proceedings of ICML,* (2015), pp. 1889–1897.
14. J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," arXiv preprint arXiv:1707.06347 (2017).
15. T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," arXiv preprint arXiv:1509.02971 (2015).
16. O. Nachum, M. Norouzi, K. Xu, and D. Schuurmans, "Bridging the gap between value and policy based reinforcement learning," in *Advances in Neural Information Processing Systems,* (2017), pp. 2775–2785.
17. V. Mnih, A. P. Badia *et al.*, "Asynchronous methods for deep reinforcement learning," in *International conference on machine learning,* (2016), pp. 1928–1937.
18. Z. Wang, V. Bapst, N. Heess, V. Mnih, R. Munos, K. Kavukcuoglu, and N. de Freitas, "Sample efficient actor-critic with experience replay," arXiv preprint arXiv:1611.01224 (2016).
19. "ChainerRL," https://github.com/chainer/chainerrl. Accessed: Jan 2019.
20. S. M. Kakade, "A natural policy gradient," in *Advances in neural information processing systems,* (2002), pp. 1531–1538.
21. R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Advances in neural information processing systems,* (2000), pp. 1057–1063.
22. J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," arXiv preprint arXiv:1506.02438 (2015).
23. D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," arXiv preprint arXiv:1412.6980 (2014).
24. H. Beyranvand and J. A. Salehi, "A quality-of-transmission aware dynamic routing and spectrum assignment scheme for future elastic optical networks," J. Light. Technol. **31**, 3043–3054 (2013).
25. J. Pedro, J. Santos, and J. Pires, "Performance evaluation of integrated otn/dwdm networks with single-stage multiplexing of optical channel data units," in *Proceedings of ICTON,* (2011), pp. 1–4.
26. A. Medina, N. Taft, K. Salamatian, S. Bhattacharyya, and C. Diot, "Traffic matrix estimation: Existing techniques and new directions," SIGCOMM Comput. Commun. Rev. **32**, 161–174 (2002).
27. L. Guo and I. Matta, "The war between mice and elephants," in *Proceedings of ICNP,* (2001), pp. 180–188.