Feature Engineering for Deep Reinforcement Learning Based Routing

José Suárez-Varela*, Albert Mestres*, Junlin Yu†, Li Kuang†, Haoyu Feng†,
Pere Barlet-Ros* and Albert Cabellos-Aparicio*
* Universitat Politècnica de Catalunya. E-mail: {jsuarezv, amestres, pbarlet, acabello}@ac.upc.edu
† Huawei Technologies Co., LTD. E-mail: {yujunlin2, kuangli, fenghaoyu}@huawei.com

***Abstract*—Recent advances in Deep Reinforcement Learning (DRL) techniques are providing a dramatic improvement in decision-making and automated control problems. As a result, we are witnessing a growing number of research works that are proposing ways of applying DRL techniques to network-related problems such as routing. However, such proposals failed to achieve good results, often under-performing traditional routing techniques. We argue that successfully applying DRL-based techniques to networking requires finding good representations of the network parameters: *feature engineering*. DRL agents need to represent both the state (e.g., link utilization) and the action space (e.g., changes to the routing policy). In this paper, we show that existing approaches use straightforward representations that lead to poor performance. We propose a novel representation of the *state* and *action* that outperforms existing ones and that is flexible enough to be applied to many networking use-cases. We test our representation in two different scenarios: (i) routing in optical transport networks and (ii) QoS-aware routing in IP networks. Our results show that the DRL agent achieves significantly better performance compared to existing state/action representations.**

I. Introduction

Recent advances in the field of Deep Reinforcement Learning (DRL) are showing a dramatic improvement in decision-making and automated control problems [1]. As a result, the networking community has recently started to investigate the potential of such techniques to solve network-related problems, such as IP routing [2], optical routing [3] or QoS provisioning [4]. These recent proposals aim to build self-driving networks [5] by designing ML-based agents that operate the network autonomously.

Network routing is a particularly interesting problem for the application of DRL techniques. Indeed, the configuration of optimal routes in a network is arguably one of the most fundamental and well studied problems in the field of networking. This problem has been addressed using traditional traffic engineering techniques [6], which rely on complex optimization algorithms with simple delay models. In contrast, DRL and modern ML techniques enable to operate under complex and non-linear models. However, at the time of this writing, existing DRL-based proposals still fail to achieve good results and often under-perform traditional routing techniques based on simple heuristics. In this paper, we argue that the main reason behind this poor performance is that these proposals apply DRL as a black-box using *straightforward representations* for both the observation and the action space.

The representation of the observation space describes the state of the environment, the network in our case. This is typically achieved by representing it as a matrix containing the per-link utilization. The action space, on the other hand, describes the modifications that the DRL agent performs on the environment. In our context, the action corresponds to changes to be applied to the routing configuration of a network. Given the high dimensionality of the output, existing proposals usually limit the action space to straightforward representations, such as the per-link weights for link state routing algorithms [2], [7].

We argue that networking problems are fundamentally different from other recent problems where DRL techniques have been successfully applied (e.g., [8]). We show that, in order to outperform traditional algorithms in the networking context, 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 good representations of the state and action spaces that can better represent the singularities of network topologies and simplify the learning process to the DRL agent. We refer to this design process as *feature engineering*.

An important limitation of existing representations is that they do not achieve a good generalization. The power of modern DRL techniques relies on their ability to make good decisions in scenarios where they may have not been trained in advance. A good representation is crucial to simplify the learning process by reducing the level of abstraction required by the ML model. In other words, the more advanced the representation is, the easier and faster is the learning process.

In this paper, we propose a novel representation for the network state (observations) that captures both the overall utilization of the network and the critical inter-dependencies among the paths resulting from the network topology in a way that is simple and that can be more easily exploited by the agent to learn better and faster. In addition, one of the main advantages of this representation is that it is flexible enough to be applied to a wide set of networking use-cases.

We experimentally test our representation against state-of-the-art DRL techniques in two separate use-cases to show its flexibility. First, we evaluate the performance of the DRL agent routing traffic demands in Optical Transport Networks (OTN). And second, we perform QoS-aware routing in IP networks including traffic with different delay requirements.

 DOI 10.1109/ICC.2019.8761276

II. Deep Reinforcement Learning in Routing

The objective of Deep Reinforcement Learning is to learn the policy that leads to a maximum *cumulative reward*. The learning is achieved by iteratively exploring the state and action spaces. In this paper, we focus on optimizing the routing policy in a network. In particular, we define the routing problem as choosing a certain strategy to route new source-destination flows with the aim of making the best of the network resources in the long-term.

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 ($S$) must represent the environment unambiguously regarding the nature of the action space. In our scenario, it must represent the state of the network and the information of the flow(s) to be routed. The Action (A) stands for the set of actions that the agent can take and can be either continuous or discrete (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 taking an action. In our scenario, the transition distribution models the stochastic behavior of changes in the network state as well as the generation of new traffic flows to be routed. The Reward (R) is the incentive that the agent obtains after making decisions. Its purpose is to steer the learning process of the agent towards the achievement of its final objective. The discount factor ($\gamma \in [0, 1)$) represents the importance of future states. In our scenario, it must be considerably high, since the objective represents a long-term planning strategy. Finally, the initial state (s_0) represents an empty network with a first traffic demand to be routed.

Solving the MDP directly is typically computationally very expensive given the high dimensionality of the problem state and the uncertainty in the transitions. The number of possible network states is proportional to the number of links, their capacity and the granularity considered for the link utilization. For standard networks, this is typically an extremely high number of states. For example, in the experiments we perform in Section IV-A, the number of possible network states is close to 10^{100}. An alternative to solve the MDP is using Reinforcement Learning together with 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. Thus, with a proper training, such neural networks are able to model how to act successfully in regions not explored before.

Fig. 1 represents the basic operation of the DRL agent in a network routing scenario. When there is a new flow to be routed, this is communicated to a network controller (1). Then, the controller generates a new state representation that will be the input of the DRL agent (2). This state representation includes information about the state of the network and the new traffic request. With this input, the DRL agent selects an action that involves making a routing decision (3). Lastly, the controller translates the resulting action to a particular set of network rules that are installed into some network devices (4). This way, during the training phase, the DRL agent learns how to properly act over the network by iteratively exploring different routing strategies and considering the reward obtained.

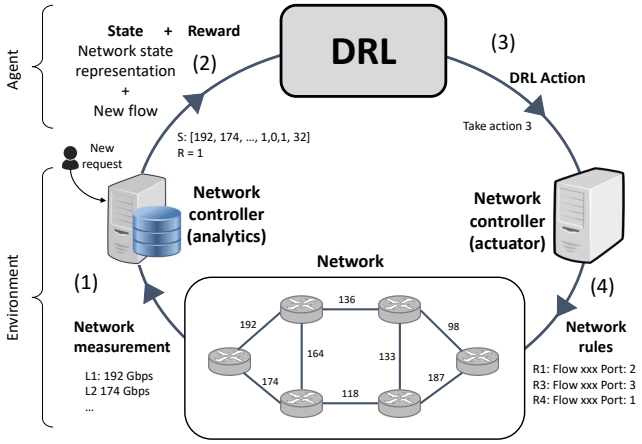

Fig. 1. Schematic representation of the DRL operation.

III. Proposed representation

In this section, we propose a novel representation for DRL to specifically tackle optimization problems in networking scenarios, such as network routing. 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*).

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 [2], [7], [5], opt for using directly the traffic matrix (i.e., the traffic demand between each source-destination pair) as the representation of the network state. 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 [2], 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. Thus, in [9] the network state is represented as a matrix that contains the traffic demand aggregated in every router for a number of time intervals. Alternatively, [3] represents the network state with the links' utilization. Particularly, they model links as arrays with a number of slots that can be full or empty. Both proposals, [9] and [3],

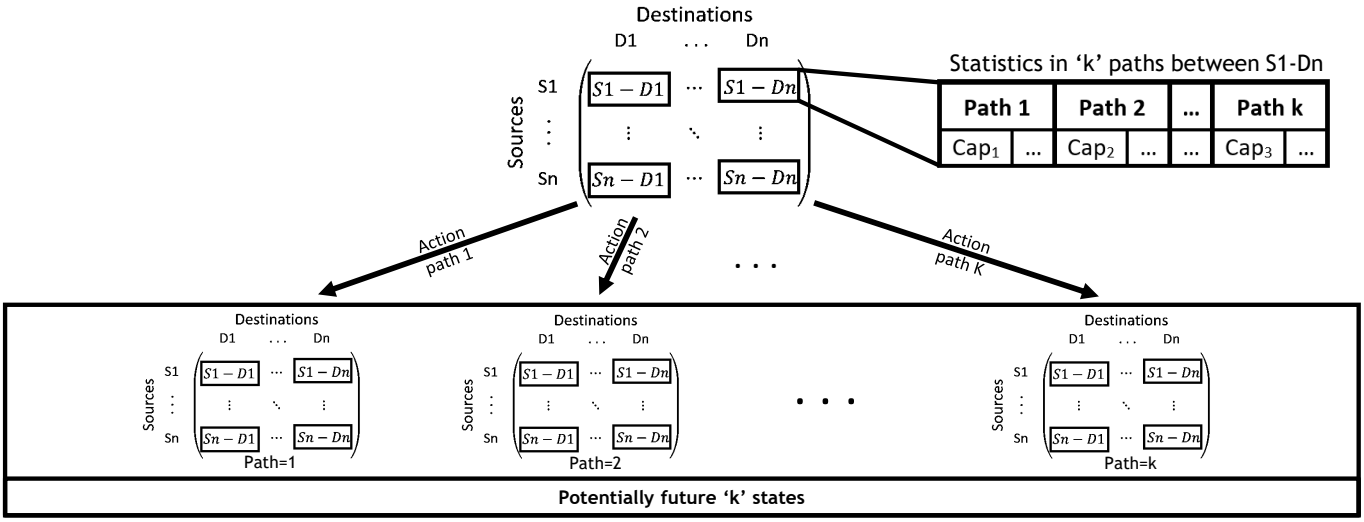

Fig. 2. Scheme of the state representation proposed.

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 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, in networking scenarios, it is not feasible to achieve good performance by using only straightforward representations of the network state, such as the links' utilization or the traffic matrix. 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 our proposed representation

In contrast to state-of-the-art proposals, our approach is to perform *feature engineering* to achieve 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 demand, it is easy to estimate the resulting link utilization 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 requests, which are stochastic. In this context, after proper training, the 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/paths 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 statistics, as in previous works, we propose the use of statistics at the *path level*. 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 $\{source,\ destination,\ bandwidth\ demand, \ldots\}$, the agent can select one path among a list of "k" candidate paths (e.g., "k" shortest paths) that connect the source and the desination of such demand. With respect to the state representation, the agent is provided with 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 maintained in the network controller. This matrix contains the current statistics of the "k" end-to-end paths (e.g., available capacity) for each source-destination pair. With this representation, the controller can 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 after allocating the current traffic request to each of the "k" candidate paths. In other words, the proposed representation provides the agent with a set of matrices that describe relevant statistics for the consequences of applying every possible action. Note that limiting the actions to "k" paths allows us to control the dimensionality of the state representation.

This representation assumes that the DRL agent has access to telemetry information of the network, which is aligned with current architectural trends, such as Software-Defined Networking or Overlay Networking. Also note that, depending on the particular problem to address, the path-level statistics present in the state representation may vary (e.g., available capacity, end-to-end delay). Additionally, it may be required to include some information about the current traffic request (e.g., bandwidth demand, delay constraints).

IV. Results

In this section we show that our proposal can be successfully applied to different networking use-cases. To this end, we evaluate its performance in two different routing scenarios: (i) traffic routing in Optical Transport Networks (OTN), and (ii) QoS-aware routing in IP networks.

A. Routing in Optical Transport Networks

In this use-case, the objective of the agent is to efficiently route new service requests in OTNs. Each service request is defined by the tuple $\{source, destination, bandwidth\}$ and must be allocated in the OTN, i.e., it has to be routed through a concatenation of lightpaths to reach the destination. We assume that the DRL agent manages the routing over the logical topology. That is, we consider that all the nodes in the logical topology are reconfigurable optical add-drop multiplexer (ROADM) devices, which have enough flexibility to route the traffic over any of the lightpaths connected to them. Additionally, we consider that the agent operates at the electrical domain to allocate traffic demands to end-to-end paths. Thus, we consider there are only 5 types of traffic requests (from ODU0 to ODU4) with different bandwidth demands that can be expressed as multiples of ODU0 signals[1].

We assume that these service requests do not expire, hence the objective is to postpone the event that a request cannot be allocated. We consider that a request is properly allocated to an end-to-end path if there was enough available capacity in all the lightpaths included in it. Consequently, we define the immediate reward as the bandwidth of the current service request if it was properly allocated, otherwise it is 0. Likewise, an episode ends when a service request cannot be allocated.

In our experiments we trained a DRL agent using an implementation of the Trust Region Policy Optimization (TRPO) algorithm [10] provided by ChainerRL (v0.3.0) [11]. We modeled the agent with a neural network with two hidden layers, each one with 64 neurons. The number of neurons in the input and the output layers varies in the different scenarios we tested. We selected a value of 0.995 for the *discount factor* (γ) and 0.97 for the *exploration parameter* (λ), which obtained the best results.

We evaluated the agent in two real-world topologies: the 14-node NSFNET used in [12] and the 17-node German Backbone Network (GBN) used in [13], with an ad-hoc simulator. For the sake of simplicity, we consider that all the links in both networks have a capacity equal to 200 ODU0 demands. We compared three different representations of the observation and action spaces. The simplest one uses the available capacity of the links as network representation and the selection of one path among 4 candidate shortest paths for the (discrete) action space. This representation (hereafter referred to as "Link and k-paths") is similar to the approach followed by [3]. The second approach (referred to as "Links and weights representation") uses also the available capacity of the links to represent the network, but the actions consist of defining weights for the links; then, the path with lowest weight is selected. This second representation uses the same action space as in [2]. Finally, we implement the representation described in Sec. III using only the available capacity per path as path-level metric. In all the cases, we use vectors with one-hot encoding to represent the source, destination and ODUk type (i.e., bandwidth requirement) of the traffic demands.

We trained the agent in two scenarios with different traffic profiles. In the first scenario, we generated a bimodal synthetic traffic distribution [14], in which 20% of nodes generate 80% of the traffic. The distribution of the ODUk requests follows an elephant-mice distribution [15], where there is a high number of low bandwidth requests and the bulk of the traffic is generated by a reduced number of big requests. In the second scenario, we used an uniform distribution for sources, destinations and ODUk types. This latter scenario represents the most challenging case for the DRL agent, given that it cannot exploit particular characteristics of the traffic profile to efficiently direct the exploration during training.

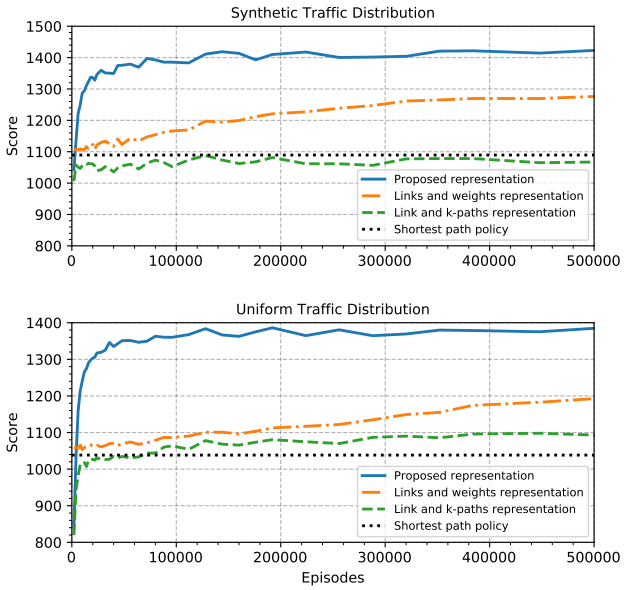

Fig. 3. Average score as a function of the training episodes for different state/action representations in the NSFNET topology.

Fig. 3 shows the score (i.e., the accumulated reward) achieved by the DRL agent. This represents the average bandwidth properly allocated w.r.t. the number of training episodes for the two traffic scenarios in the NSFNET topology.

[1]Note that an ODU0 can carry 1.244 Gbit/s approximately. The largest is ODU4, which corresponds to the bandwitdth carried by 64 ODU0 signals.

Each figure shows the performance achieved by the tree observation/action spaces and by a traditional Shortest Path (SP) routing policy. In the scenario with synthetic traffic, the agent achieves a slightly higher score, 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 ("Link and k-paths") achieves a similar score than the SP policy, (ii) the "Links and weights" representation is able to outperform the SP policy, but it learns much slower than using the other representations. For example, in the case with synthetic traffic, the representation proposed in this paper needs only 8k episodes to achieve the same score obtained by the weights representation after 500k episodes. Finally, the proposed representation is able to allocate 30% more bandwidth than using the "Link and k-paths" representation.

Fig. 4 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 score obtained by the SP policy. This may be due to the different distributions of the node degree in both topologies. For example, in the GBN topology, we observed there are more nodes with low degree. This makes the links connected to these nodes more prone to become congested. Note also that, for all the evaluations we performed with link-based state representations, the agent achieves better performance when it uses link weights in the action space than when it applies discrete path-level actions. This suggests that, when representing the state with link-level features, it may be more beneficial using link-based actions.

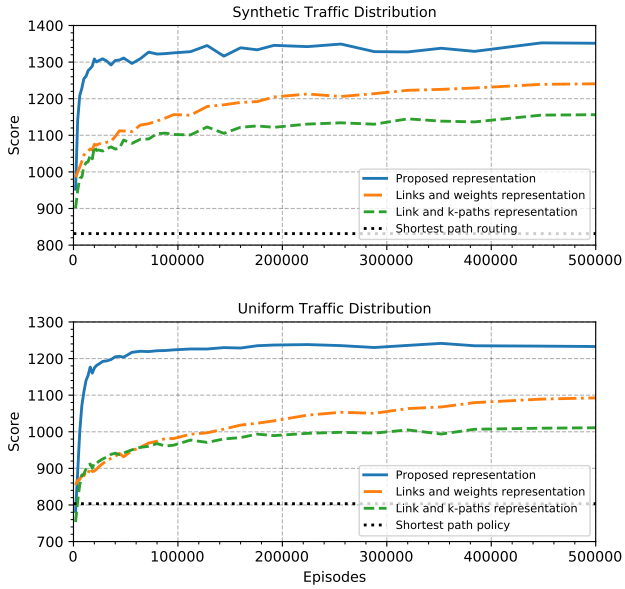

Fig. 4. Average score as a function of the training episodes for different state/action representations in the GBN topology.

Lastly, we conclude that the DRL agent learns much faster when using discrete actions to select one of the "k" candidate paths than when it defines link weights. Moreover, our representation using the paths' available capacity of the possible next states achieves much higher performance than using only the utilization of the links (aprox 30% better).

B. QoS-aware routing

In this use-case, we test the representation proposed in Sec. III to perform routing in IP networks focusing on Quality of Service (QoS) provisioning. Here, the objective of the DRL agent is to efficiently route traffic flows so that their QoS requirements are met. In this context, we consider a scenario with 4 different applications that generate flows with different bandwidth and end-to-end delay requirements (Table I).

TABLE I
APPLICATIONS IN THE QOS-AWARE ROUTING USE-CASE.

Application	Bandwidth	Maximum tolerable delay
App 1	300 kbps	10 ms
App 2	500 kbps	15 ms
App 3	1.5 Mbps	20 ms
App 4	3 Mbps	30 ms

Note that this scenario is aware of the delays of the network paths. For this purpose, we extended the simulator used in Sec. IV-A to model also the path delays apart from the link utilization. Moreover, each epoch, the simulator randomly generates a new flow with a particular source, destination and application type (Table I).

For the DRL agent, we use the same implementation of TRPO as in the previous use-case (Sec. IV-A), with the same neural network and the same configuration parameters (γ, λ).

We trained the agent in the 6-node topology depicted in Fig. 5, where all the links have a capacity of 50 Mbps and a base delay of 5 ms.

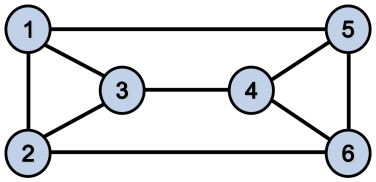

Fig. 5. Topology of the QoS-aware routing use-case.

For this use-case, the state representation includes both the available capacity and the end-to-end delay of paths. Similarly to the previous use-case, we define new flows using vectors with one-hot encoding to represent the source, the destination and the application type. Moreover, the action set consists of selecting one of the 4 shortest paths for a specific source-destination pair. We calculate the immediate reward of the agent as the ratio of flows whose delay requirement is satisfied w.r.t. the number of flows routed at the current time.

Fig. 6 shows the evaluation results of the agent during training. Here, the score (y-axis) represents for each episode the maximum number of flows that the agent was able to route satisfying the delay requirements. Note that each evaluation involves the execution of 120 episodes with different flow sets randomly selected. We compare these results with the

average score achieved by an agent applying a *Lowest Delay Path* (LDP) policy. We define the LDP policy as selecting the current path with lowest delay. As we can observe, after 2k training episodes the DRL agent beats the LDP policy and, by episode 25k, it achieves around 14% better performance.

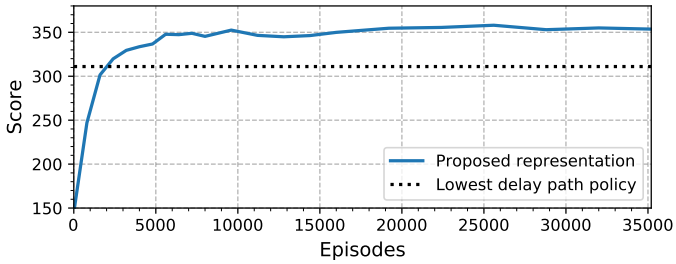

Fig. 6. Training of the QoS-aware routing use-case.

In order to further analyze the behavior of the DRL agent with our representation, we show in Fig. 7 a step-by-step evaluation during an example episode. This figure represents the number of flows whose delay requirement was satisfied (y-axis) with respect to the total number of flows already routed by the agent (x-axis). It is noteworthy that, at the beginning of the episode, since the network is empty, all the flows can be satisfied regardless of the path that is selected. However, there is a point around flow #170 where DRL agent begins to outperform the baseline LDP policy. This proves that the DRL agent planned from the beginning a routing strategy that performs better in the long-term. Likewise, we observe that, when the network suffers from severe congestion (around flow #355), the DRL agent is still able to minimize the impact of such congestion and maintains a higher rate of flows satisfied than LDP.

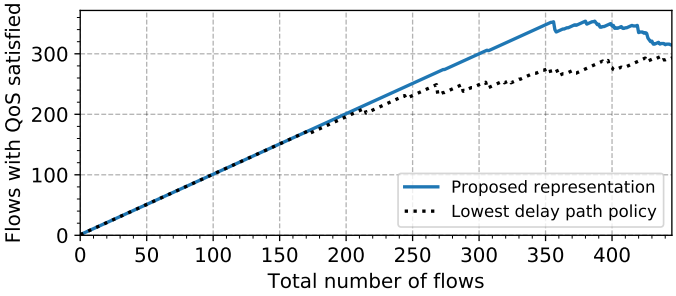

Fig. 7. Step-by-step evaluation during an episode.

V. Conclusion

In this paper, we discussed the challenges involved in the application of Deep Reinforcement Learning (DRL) to networking. Contrary to the dominant trend in other domains, where the current approach is to use deeper neural networks with less elaborated features, we argued that the key to the success of DRL in networking is a more careful representation of the network state (i.e., *feature engineering*). This is explained by the complexity of describing link 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 rough representations of the observation/action space.

In this paper, we proposed an alternative representation of the network state that still has reasonable dimensionality, but can better capture the crucial relationships among the links that form the network topology. This more "engineered" representation allows the agent to learn more easily and faster. Our results, using different network topologies and traffic profiles, show that our representation significantly outperforms previous proposals in two paradigmatic use-cases.

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, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, 2015.

[2] A. Valadarsky, M. Schapira, D. Shahaf, and A. Tamar, "Learning to route," in *Proceedings of HotNets*, 2017.

[3] 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.

[4] 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.

[5] A. Mestres, A. Rodriguez-Natal, J. Carner, P. Barlet-Ros, and E. Alarcón, et al., "Knowledge-defined networking," *SIGCOMM Comput. Commun. Rev.*, vol. 47, no. 3, pp. 2–10, Sep. 2017.

[6] J. Rexford, "Route optimization in ip networks," *Handbook of Optimization in Telecommunications*.

[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 Student Workshop*, 2017.

[8] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, T. Lillicrap, K. Simonyan, and D. Hassabis, "Mastering chess and shogi by self-play with a general reinforcement learning algorithm," *arXiv:1712.01815*, 2017.

[9] 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 Computers*, vol. 66, no. 11, pp. 1946–1960, 2017.

[10] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *Proceedings of ICML*, 2015, pp. 1889–1897.

[11] "ChainerRL," https://github.com/chainer/chainerrl, Accessed: March 2018.

[12] H. Beyranvand and J. A. Salehi, "A quality-of-transmission aware dynamic routing and spectrum assignment scheme for future elastic optical networks," *Journal of Lightwave Technology*, vol. 31, no. 18, pp. 3043–3054, 2013.

[13] 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.

[14] A. Medina, N. Taft, K. Salamatian, S. Bhattacharyya, and C. Diot, "Traffic matrix estimation: Existing techniques and new directions," *SIGCOMM Comput. Commun. Rev.*, vol. 32, no. 4, pp. 161–174, 2002.

[15] L. Guo and I. Matta, "The war between mice and elephants," in *Proceedings of ICNP*, 2001, pp. 180–188.