Reinforcement Learning for Autonomous Traffic Flow Capacity Management

Sima Barzegar, Marc Ruiz, and Luis Velasco*
Optical Communications Group (GCO), Universitat Politècnica de Catalunya (UPC), Barcelona, Spain
**e-mail: luis.velasco@upc.edu*

2023 23rd International Conference on Transparent Optical Networks (ICTON) | 979-8-3503-0303-2/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICTON59386.2023.10207531

ABSTRACT
As the dynamicity of the traffic increases, the need for self-network operation becomes more evident. One of the solutions that might bring cost savings to network operators, is that of the dynamic capacity management of large packet flows, especially in the context of packet over optical networks. Machine Learning, and particularly Reinforcement Learning (RL), seem to be an enabler for autonomicity, as a result of its inherent capacity to learn from experience. In this tutorial, we introduce RL and review its application for autonomous capacity management of traffic flows.
Keywords: reinforcement learning, network automation.

1. INTRODUCTION

Autonomous network operation evolves from software-defined networking (SDN) and promises to reduce operational expenditures by implementing closed-loops based on data analytics [1]. Such control loops can be put into practice based on policies that specify the action to be taken under some circumstance (policy-based management), e.g., allocate the capacity of a packet flow so that the ratio traffic volume over capacity is under 80%. In packet flows this ratio is related to the average delay that the packets in the flow will experience because of queuing, and thus to the Service Level Agreement (SLA) in the case that the packet flow is related to some customer connection. Although such policies can be modified they are purely reactive. Under high traffic variations they might entail either poor Quality of Service (QoS) (e.g., high delay or even traffic loss) and SLA breaches or poor resource utilization, which in both cases represent large costs for network operators. Note that policy-based management does not define the desired performance and thus, agents implementing those policies are unable to learn the best actions to be taken. Another approach for network automation is Intent-Based Networking (IBN) [2]. IBN allows the definition of operational objectives that a network entity, e.g., a traffic flow, has to meet without specifying how to meet them. IBN implements and enforces those objectives, often with the help of Machine Learning (ML) [3].

Many ML techniques could be potentially applied for autonomous traffic flows capacity management. For instance, deep learning techniques can be applied to predict complex future events, like future traffic conditions. In particular, the use of feed-forward neural networks (FFNN) [4] allows considering complex nonlinear relations among input features and the predicted future event. Moreover, they facilitate working with a mix of numerical and categorical inputs, as well as making predictions for several steps ahead, i.e., multi-step prediction. However, in this paper, we consider the application of Reinforcement Learning (RL) [5].

2. REINFORCEMENT LEARNING

RL considers the paradigm of an intelligent *agent* that takes actions in an *environment*. At every discrete time step t, with a given state s, the agent selects action a with respect to a policy, and it receives from the environment a reward r and the new state s'. The objective is to find the optimal policy that maximizes a cumulative reward function. RL fits perfectly as part of intent agents, as the related problems can be usually stated in the form of a Markov decision process and they can be solved RL using dynamic programming techniques. In addition, in contrast to supervised learning, RL does not need labeled datasets and it can correct sub-optimal actions through exploration.

The simplest RL is Q-learning [5], which is a model-free discrete RL method that uses a Q-table to represent the learned policy, where every pair $<s, a>$ contains a q value. Being at state s, the action a to be taken is the one with the highest q value (or it is chosen randomly). Once the action is implemented and the new state s' and the gained reward r are received from the environment, the agent updates the corresponding q value in the Q-table. Q-learning works efficiently for problems where both states and actions are discrete and finite. However, it usually introduces overestimation, which leads to suboptimal policies, and the Q-table grows with the number of states. *Deep Q-learning* (DQN) substitutes the Q-table by a FFNN that receives a continuous representation of the state and returns the expected q value for each discrete action [5]. However, the FFNN tends to make learning unstable, so a *replay buffer* can be used to retrain the FFNN. *Double DQN* [6] uses two different FFNNs (*learning* and *target*) to avoid overestimation, which happens when a non-optimal action is quickly biased (due to noise or exploration) with a high q value that makes it preferably selected. The learning model is updated using the q values retrieved from the target model, which is just a simple copy of the learning model and it is periodically updated. Finally, *dueling double DQN* (D3QN) [7] uses two different estimators to compute the q value of a pair $<s, a>$: *i*) the *value* estimator, an average q value of any action taken at state s; and *ii*) the *advantage* estimator, which is the specific state-action dependent component. The sum of both components

returns expected *q* values. DQN-based methods assume a finite discrete action space. Nonetheless, other approaches, such as *Actor-Critic* methods [8], use continuous state and action spaces. *Actor-Critic* methods train two different types of models separately: *i*) *actors*, which compute actions based on states, and *ii) critics* that evaluate the actions taken by actors, i.e., compute *q* values. Both actor and critic models can be implemented by means of FFNNs. Aiming at reducing overestimation, the Twin Delayed Deep Deterministic Policy Gradient (TD3) method [8] considers one single actor and two different critic models, where the minimum value from the two critics is used for learning the optimal policy.

3. RL FOR AUTONOMOUS NETWORK OPERATION

Several works have proposed the application of RL algorithms for autonomous network operation. In the context of optical networking, the authors in [9] proposed the use of *Actor-Critic* -based algorithms running in the SDN controller for dynamic lightpath provisioning. They showed that changes in the traffic profile impact on the obtained performance. The authors in [10] studied the application of several deep RL algorithms and reward functions to increase network availability in SDN-controlled wide area networks. In the context of IBN, the work in [11] applied RL to the autonomous operation of optical connections based on digital subcarrier multiplexing. Their RL algorithm adapted the capacity of the optical connection to the upper layer packet traffic. The authors in [12] proposed an IBN solution based on RL, which demonstrated the ability to learn the optimal policy based on the specified operational objectives related to the allocated capacity or the delay. Finally, the work in [13] proposed an RL algorithm for autonomous bandwidth allocation in the context of multilayer optical networks. They proposed a centralized algorithm running in the SDN controller that supervises the network performance. When the network performance degrades, the centralized algorithm tunes parameters in the RL algorithms.

4. FLOW CAPACITY AUTONOMOUS OPERATION

Let us now focus on the flow capacity autonomous operation. A *flow manager* might collect monitoring data from the network and expose some interface, so an RL algorithm can take actions. For illustrative purposes, Fig. 1 shows a typical RL framework [2], where the learning agent is separated into two different blocks, the *learner* and the *agent*.

Let us assume that the monitoring data includes the byte count since the last monitoring sample (amount of traffic) and the actual capacity allocated to the flow. The actions to be taken are related to the actual capacity allocated to the flow, which can be increased or decreased as needed with some *granularity* to meet the required QoS. For instance, a customer connection can manage the capacity of the flow with granularity 1 Gb/s by configuring some packet node, whereas in a virtual link supported by the optical layer, the capacity can be increased/decreased by establishing or tearing down parallel lightpaths, each with a capacity of 100s Gb/s. It seems clear that the time to change the capacity is also different, ranging from seconds to minutes. The RL algorithm should then decide the capacity to be allocated to the flow to absorb *variation* in the traffic from one monitoring sample to the next one, plus the time to increase the allocated capacity, with the objective to avoid any traffic loss and ensure some additional QoS metric. Then, the *traffic variation* becomes a major feature for a flow, together with the *traffic pattern*, i.e., the evolution of the mean traffic with time.

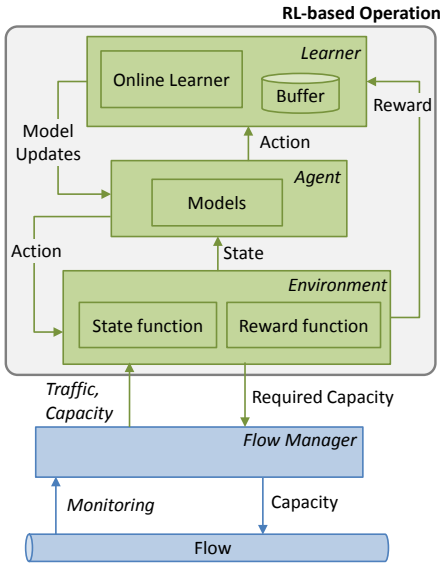

Figure 1. Flow capacity autonomous operation.

This approach can provide excellent performance once the policies that avoid traffic losses, meet the desired QoS, and minimize overprovisioning are learned; however, online learning of such policies requires time. In addition, there are several issues that can impact the aforementioned online learning performance, e.g.: 1) changes in traffic variability might produce loss before new policies are learned; 2) smooth model fine tuning could not be enough to mitigate persistent errors in taking some specific actions; and 3) online learning tends to forget valuable learning in the long run, thus reducing its accuracy [5].

Figure 2a represents a possible evolution of the traffic variation (the traffic pattern is omitted for simplicity) and the obtained performance - overprovisioning, traffic loss, and some other QoS metric. The path supporting the flow is established at time *t0* and the desired QoS is specified, so the RL algorithm needs time to learn the traffic variation (and the traffic pattern); meanwhile (until *ta* in Fig. 2a), poor performance, including traffic loss, can be expected. Once a good model is obtained, it is expected that a RL algorithm can provide the target performance. However, a steep change in the variation of the traffic (times *t1* to *t2*) can impact the performance until the new variation is learn, although the performance might not converge to the desire level even after learning.

It seems clear that the above behavior is unacceptable for network operators, as it would provide poor performance and might incur penalties due to SLA breaches. Specifically, it seems of paramount importance to

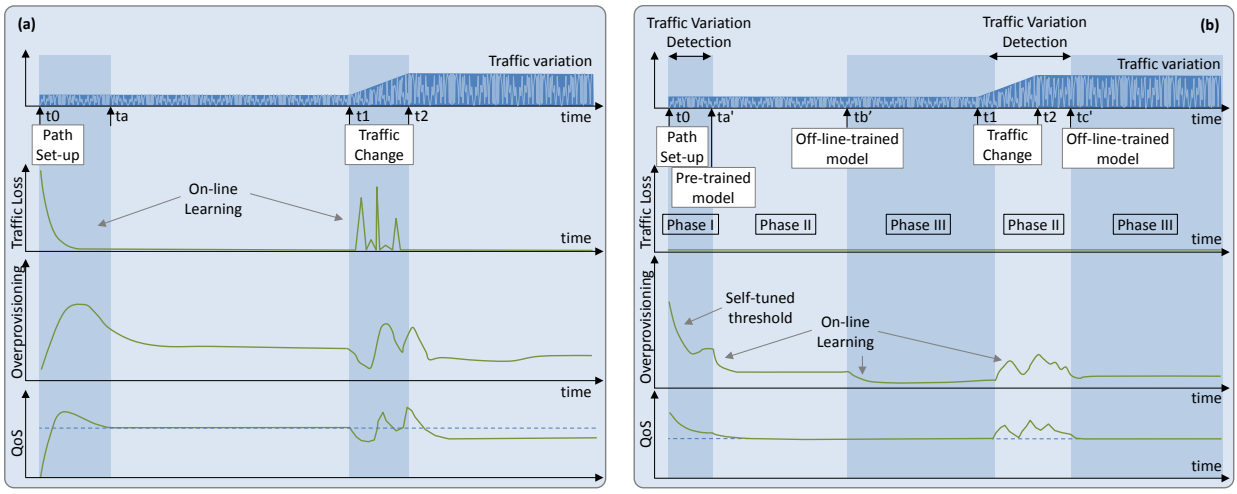

Figure 2. Operation lifecycle. (a) online learning RL operation, (b) off-line training with online fine tuning RL operation.

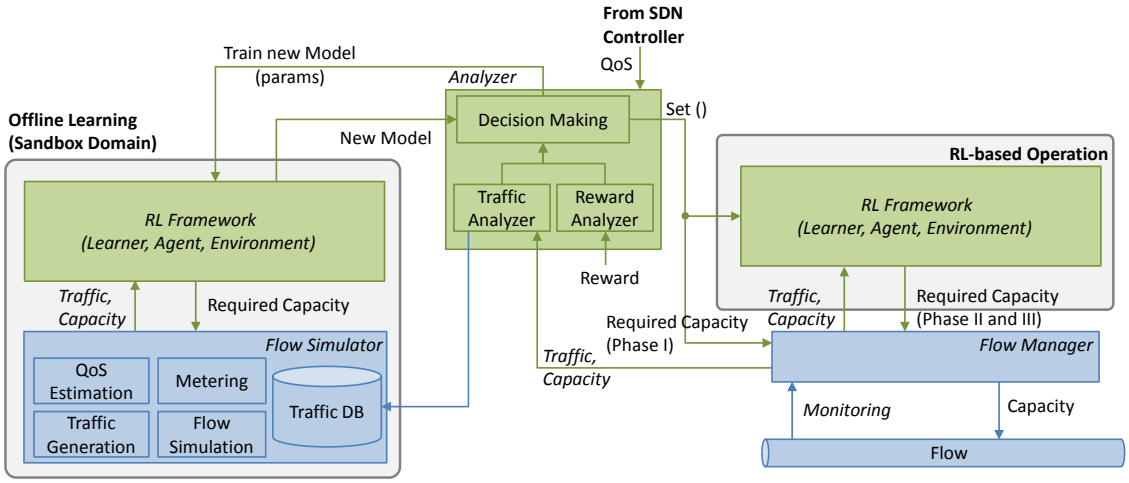

Figure 3. Extended architecture for flow capacity autonomous operation with offline learning.

start the operation with already trained models. To that end, an initial model can be trained offline using a network simulator in a sandbox domain. Once in operation, the model will be improved by the online learner. However, there are traffic characteristics, e.g., traffic pattern, that are observed after a long period of time, e.g., several days. Therefore, some alternatives are needed to operate the flow during that initial time.

Our solutions go beyond training offline and propose implementing offline-online learning cycles to deal with large changes in traffic flow, i.e., to provide guaranteed performance during the whole lifetime of the traffic flow (Fig. 2b). Specifically: *i*) a policy-based management implemented as a self-tuned *threshold-based* algorithm is in charge of managing the flow capacity during the time immediately after the path is set-up (*Phase I*: time interval [*t0*, *ta'*], where *ta'-t0* should be short, e.g., 1 hour). That algorithm tunes a threshold for the flow capacity and accurately determines the traffic variation. This approach enables dynamic flow capacity allocation by fixing the right values for the threshold that minimize overprovisioning. However, avoiding traffic loss and guaranteeing that the required QoS is met is not a straightforward task, as it depends on the variance of the traffic flow -defined as the difference between maximum and minimum amount of traffic during some period. Therefore, during this period, the threshold is set conservatively to avoid *underprovisioning* (i.e., traffic exceeds capacity, and some traffic is loss) at the expense of large overprovisioning; *ii*) Once the variation of the traffic has been determined, a pre-trained *generic model* can be used for flow operation (*Phase II*: time interval [*ta'*, *tb'*]). The model is general as it has been pre-trained assuming a given traffic pattern, e.g., sinusoidal with daily periodicity, but supporting the measured traffic variation. Once in operation, the pre-trained model starts to fine tune with the observed samples; *iii*) Once enough measurements are available to determine the characteristics of the flow, including the traffic pattern, a specific model can be trained in a sandbox domain by using a simulator set to operate at time *tb'* (*Phase III*). That model should improve the performance or be easier to operate than the pre-trained one; *iv*) assuming that the traffic pattern does not change, any change in the traffic variation that cannot be absorbed by the current model can trigger returning to Phase II for more intensive parameter tuning, while a new specific model is trained and is set to operate at time *tc'* [Phase II - Phase III cycle].

In view of this, the basic RL-based flow capacity operation (Fig. 1) can be extended to consider a scheme based on (Fig. 3): *i*) analyzing the traffic to obtain meaningful traffic characteristics; *ii*) making decisions

regarding the allocated capacity when no model is in operation (Phase I); *iii*) selecting pre-trained models that fit with the observed traffic characteristics; *iv*) training new models in a sandbox domain (*offline learning*) [14], where real traffic measurements are used to generate traffic in a simulation environment and the QoS can be realistically estimated. A replica of the RL algorithm in operation is used here for training new models; and *v*) once accurate models are obtained, they are used for flow operation and will be progressively fine-tuned online. As in Fig. 1, a *Flow Manager* collects monitoring data from the forwarding plane and enforces flow capacity.

The models include some parameters that need to be tuned as a function of the traffic, to provide the desired performance while meeting the required QoS. Such parameter tuning can be carried out during offline learning, as well as during online operation to deal with small traffic changes. Based on the analysis of the traffic and the reward, the analyzer block decides when to tune parameters and when to update the model with an offline learned one (labeled *Set*() in Fig. 3) to meet the given QoS. Note that both, parameter tuning, and the offline-online cycle can be completed several times during the operation to improve the learned models, which will also enable adaptability to changes.

5. CONCLUSION

Main conclusions extracted from the numerical evaluation are summarized in Table 1. As expected, online RL produces from moderate to high loss (reaching peaks of 1-10 Gb/s) at the beginning of network operation. The proposed lifecycle leads to several conclusions. Firstly, zero traffic loss and QoS assurance is guaranteed from path set-up regardless of the chosen RL method. Secondly, *Phase II* allows a very efficient and robust operation based on pre-trained generic models that were tuned with specific traffic characteristics. *Phase II* clearly outperformed threshold-based operation in terms of capacity utilization since it remarkably reduced overprovisioning (up to 45%). We investigated the application of Q-Learning, TD3, and D3QN, and observed that all the methods reached outstanding capacity efficiency (more than 50% of capacity reduction with respect to threshold-based operation) without losing QoS performance *in Phase III*; Q-Learning and TD3 behaved slightly better than D3QN. Finally, the continuous analysis and tracking conducted during *Phase III* to detect traffic changes allowed a prompt detection of sharp changes (between 1 and 650 minutes), triggering *Phase II* from several hours to days before online RL operation suffered any significant degradation.

Table 1. Summary of results for policy-based and RL operation w/ and w/o offline learning.

Approach	Concept		Threshold-based	RL
Policy-based	Traffic loss		Zero traffic loss	
	QoS assurance		Since path set-up	
	Over-provisioning		Very large	
Online RL Operation	Traffic loss			High loss
	QoS assurance			After 5 days
	ρ range for QoS assurance			Narrow
	Over-provisioning	conservative ρ		Small
		optimal ρ		Small
Offline+ Online RL Operation	Traffic loss		Zero traffic loss	Zero traffic loss
	QoS assurance		Since path set-up	Since path set-up
	ρ fine tuning effectiveness			Large
	Over-provisioning Gain	Phase I	None	
		Phase I->Phase II		Large
		Phase II		Large
		Phase II-> Phase III		Small
	Reliability (Phase III-> Phase II)			High

ACKNOWLEDGEMENTS

This project has received funding from the Smart Networks and Services Joint Undertaking under the European Union's Horizon Europe research and innovation programme under Grant Agreement No. 101096120 (SEASON), from the MICINN IBON (PID2020-114135RB-I00) project and from the ICREA Institution.

REFERENCES

[1] L. Velasco *et al*., "MDA for optical networking: benefits, architectures, and use cases," *Netw. Mag*., vol. 33, 2019.
[2] L. Velasco *et al*., "Intent-based networking for optical networks," *JOCN*, vol. 14, 2022.
[3] D. Rafique *et al*., "ML for optical network automation," *JOCN*, vol. 10. 2018.
[4] P. Zhang and M. Qi, "Neural network forecasting for seasonal and trend time series," *EJOR*, vol. 160, 2005.
[5] R. Sutton and A. Barto, *Reinforcement Learning: An Introduction*, 2nd Ed., MIT Press, 2018.
[6] H. Hasselt *et al*., "Deep reinforcement learning with double Q-learning," in *Proc. AAAI Conference on AI*, 2016.
[7] Z. Wang *et al*., "Dueling network architectures for deep reinforcement learning," in *Proc. ICML*, 2016.
[8] S. Fujimoto *et al*., "Addressing function approximation error in actor-critic methods," in *Proc. ICML*, 2018.
[9] X. Chen *et al*., "DeepRMSA: A DRL for RMSA in EONs," JLT, vol. 37, 2019.
[10] S. Troia *et al*., "On deep reinforcement learning for traffic engineering in SD-WAN," *JSAC*, 2021.
[11] L. Velasco *et al*., "Autonomous and energy eff. lightpath op. based on DSCM," *JSAC*, vol. 39, 2021.
[12] S. Barzegar *et al*., "Packet flow capacity autonomous operation based on RL," *Sensors*, vol. 21, 2021.
[13] T. Panayiotou *et al*., "A data-driven bandwidth allocation for EONs," *JLT*, vol. 37, 2019.
[14] M. Ruiz *et al*., "Modeling and assessing services performance in a sandbox," *JLT*, vol. 38, 2020.