Reliable and Accurate Autonomous Flow Operation based on Off-line Trained Reinforcement Learning

Sima Barzegar, Marc Ruiz, and Luis Velasco*

Optical Communications Group (GCO), Universitat Politècnica de Catalunya (UPC), Barcelona, Spain
e-mail: luis.velasco@upc.edu

Abstract: A RL agent trained offline for reliability and able to refine its policies during online operation is proposed. Results for three illustrative flow automation use cases show remarkable performance with extraordinary adaptability to changes.

1. Introduction

Autonomous network operation evolves from software-defined networking and promises to reduce operational expenditures by implementing closed-loops based on data analytics. Such control loops can be implemented based on policies that specify the action to be taken under some circumstance (*policy-based* management). Although such policies can be modified, they do 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) [1]. 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 is in charge of implementing and enforcing those objectives, often with the help of Machine Learning (ML).

In our previous work in [2], we proposed an IBN solution based on Reinforcement Learning (RL), which demonstrated the ability to learn the optimal policy based on the specified operational objectives related to the allocated capacity or the delay. The key here is the time needed to learn such optimal policies, as exploration entails *low reward* decision making (i.e., far from optimal operation). To solve this issue, the authors in [3] proposed a general learning life-cycle that included both offline training and online learning, in the context of supervised ML. The objective of that work was to accelerate autonomous operation by deploying accurate models that are firstly trained offline and fine-tuned while in operation.

In this paper, we apply the main lessons learnt from [3] to IBN agents based on RL. Here, a policy-based management is used to generate the initial database for the offline learner. Note that the policy-based operation is highly reliable, as it is based on specific rules that can be defined and understood by human operators. From such an operation, we show how the proposed offline+online learner can learn the operational rules and is able to improve the quality of the operational decisions through online learning. In addition, we show that by iterating the offline and online cycle, the proposed RL agent is able to adapt to changes that would otherwise require manually reprogramming the rules under the policy-based management approach.

2. Autonomous operation

Fig. 1a represents the simplest form of autonomous operation of a *flow* based on some policy, e.g., some thresholds; an *agent* is in charge of collecting monitoring data from the network and taking actions. Let us assume that the monitoring data includes the bit count since the last monitoring sample (amount of traffic) and the actual capacity allocated to the flow, and 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*. The value of the threshold should be selected to absorb *variations* in the traffic from one monitoring sample to the next one, plus the time to increase the allocated capacity. For instance, imagine a virtual link (*vlink*) supported by one lightpath (i.e., the capacity allocated to the vlink is 100Gb/s) and that the threshold to increase capacity is set to 80%. Then, if the measured traffic is 81Gb/s, the agent would request increasing the capacity, which, in this case, entails setting-up a new parallel lightpath and increasing to 200Gb/s the total capacity allocated to the vlink. When the traffic decreases, the action of deallocating capacity can be taken if the traffic is below the threshold.

Although this simple approach enables dynamic flow capacity allocation, fixing the right values for the thresholds that minimize overprovisioning (defined as *capacity* - *traffic*) while avoiding traffic loss is not a straightforward task, since it depends on the variability of the flow traffic. Here is where RL can help, as the agent is able to learn automatically from the environment, so it can take actions without being explicitly programmed. Fig. 1b depicts a RL-based approach with three interrelated modules: *i*) the *environment adaptation* is in charge of representing the state (s) based on the monitoring data and of computing the reward; *ii*) the *agent* that contains the models, thus linking states to actions (*a*); and *iii*) the *learner* that stores operation in a buffer and is able to learn effective policies, which are used to update the models (online learning). This approach can potentially improve the performance of the policy-based operation, once the policies that avoid traffic losses and minimize overprovisioning are learned; however, this requires time. In addition, there are several issues that can impact on the aforementioned online learning performance, e.g.: 1) changes in traffic

The research leading to these results has received funding from the Spanish MINECO TWINS project (TEC2017-90097-R) and by the ICREA institution.

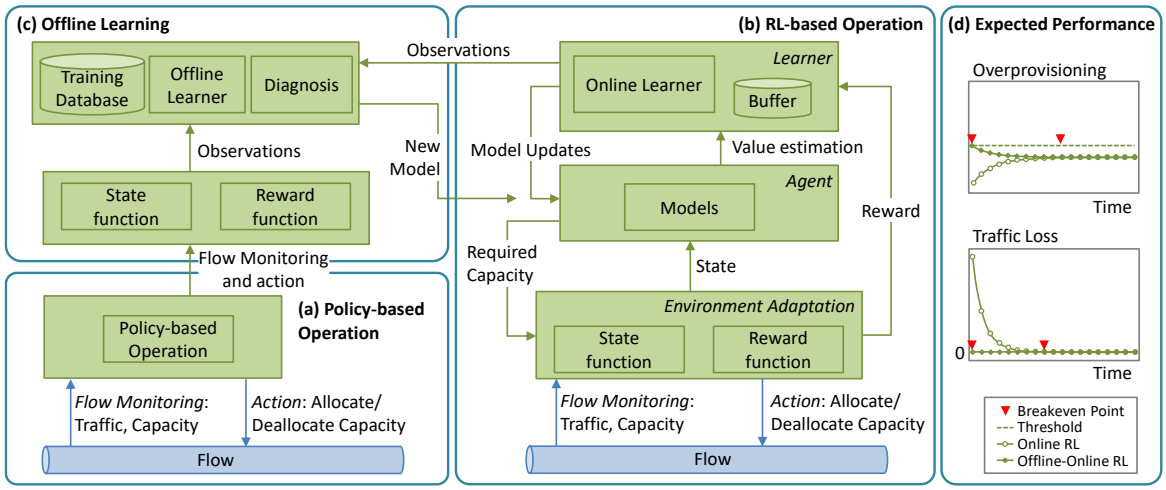

Fig. 1. Policy-based Flow autonomous operation (a), based on RL with online learning (b), and based on RL with offline learning (c). Expected performance (d).

variability that might produce loss before new policies are learned (denoted as *Issue 1 -I1*); 2) smooth model fine tuning could not be enough to mitigate persistent errors in taking some specific actions (*I2*); and 3) online learning tends to forget valuable learning in the long run, thus reducing the accuracy of the model (*I3*) [4].

In light of the above, it seems of paramount importance to periodically analyze offline the evolution of the traffic variation to detect changes, as well as the reward obtained by the decisions made to detect persistent low reward actions and a reduction of the accuracy of the model. Therefore, this work proposes an operation scheme based on two phases: *i*) pre-training the models (*offline learning*): an initial model can be trained with the observations obtained while the conservative policy-based approach is in operation; to generate the observations, the state and reward functions in the *environment adaptation* module are used (Fig. 1c); and *ii*) once accurate models are obtained, the operation is governed by the RL approach that, starting from those models, can progressively perform an online fine tuning. We expect that the offline learning agent will start operating with a performance like that of the policy-based approach, thus avoiding the initial poor-performance of the online learning phase, and continue learning to reach a similar performance to the online learning agent (Fig. 1d). Note that the offline+online cycle can be completed several times during the operation to improve the learned models based on the collected observations, which will also enable adaptability to changes.

3. Offline+Online cycle based on Q-learning

In this section, we start from our previous work on Q-learning modelling for capacity flow management [2]. In summary, we define the state as the ratio of traffic over capacity and a reward function that highly penalizes those actions that cause traffic loss and moderately penalizes those leading to capacity overprovisioning. The action selected for a given state (i.e., the capacity units to be added/subtracted) is that returning the maximum value of the *Q-function*, defined as a table of states and actions (*Q-table*).

Let us consider that a policy-based operation based on a conservative threshold-based capacity allocation is firstly run to generate a significant number of observations for offline training. Then, Q-table is first initialized with *high* reward values and next, is trained without exploration from the set of observations in the training dataset. Due to the absence of exploration, some states might remain unseen. Therefore, for the sake of robustness, the cells corresponding to the *safest action* for every unseen state are updated with a high reward.

Once the model is in operation, the online learner smoothly adapts the model to improve performance. Note that offline learning is mainly devoted to ensuring no loss while having moderated capacity overprovisioning. Thus, by fixing the learning rate (*lr*) and the discount factor (*df*) to low values, the model will smoothly adapt to reduce capacity overprovisioning during online learning, while avoiding traffic loss.

Although this operational regime based on online learning can work well most of the time, the issues anticipated in the previous section (*I1* - *I3*) need to be accurately detected, as they require large, deep changes of the model. To that end, the *Diagnosis* block, in the offline learning system, executes Algorithm I every time a new observation (a tuple with a timestamp, the measured traffic, and the reward value) is received from online operation (see Fig. 1b-c); it also receives the database with the decisions that got reward value below a reward bound (DB_{low}), the observed traffic is stored in a time series database (*Y*), and the current traffic variance observed (σ_{cur}). The algorithm analyzes the decisions made in terms of its reward value and returns the diagnosis needed by the offline learner for a proper model re-learning; it first computes the variance σ^2 observed in the last δ time units in *Y* that is used to statistically corroborate the hypothesis H_0 of equality [5] with the current traffic variance σ_{cur}. If such a hypothesis is not accepted, diagnosis of traffic variance change is returned with the updated current variance (lines 1-3 in Algorithm I). Next, it checks the reward in the received observation (line 4) and, if it is below the bound, it looks for a persistent low rewarded action (lines 5-7) and for a long record of

low rewarded actions in the last δ time units (lines 8-10). *Null* diagnosis is returned if no offline action is needed (line 12).

The offline learner proceeds when the diagnosis ends with an identification; in case of persistence of a pair <*s*,*a*>, all the observations of that pair are selected while other observations are sampled; *lr* is set to 0 for this specific pair to force keeping during online operation the decisions obtained during offline learning. For the other two cases, all the received observations within the last δ time units are selected together with others belonging to states that are not observed recently.

Algorithm I. Model diagnosis

INPUT: *obs*, DB_{low}, *Y*, σ_{cur}
OUTPUT: *diagnosis*, *results*

1: $\sigma \leftarrow$computeTrafficVariance(*Y*)
2: **if** $H_0[\sigma_{cur}==\sigma]$ = FALSE **then** (I1)
3: **return** *'trafficChange', σ*
4: **if** *obs*.reward < *bound*
5: $<s,a>_{max}$, *count*←max(count(DB_{low},*by*=<*s*,*a*>)
6: **if** *count* > *m* **then** (I2 and I3)
7: **return** *'persistent', <s,a>max*
8: *DB'*←select(DB_{low}, 'time'>*obs*.time-δ)
9: **if** |*DB'*|>*n* **then** (I3)
10: **return** *'decay', DB'*
11: **return** null, -

4. Results and discussion

For evaluation purposes, the systems in Fig. 1, including threshold- and RL- based operation, were implemented and realistic synthetic traffic traces were generated for three different use cases according to the methodology in [6]. Use case 1 emulates an agent managing a flow of an application, so capacity increments are with a granularity of 1Gb/s, whereas in use case 2, the agent manages a vlink supported by lightpaths in the optical layer, so capacity increments are with a granularity of one lightpath (100Gb/s). Finally, use case 3 considers an agent managing the capacity of a vlink with granularity of 1Gb/s and that such capacity is used to decide the number of lightpaths needed.

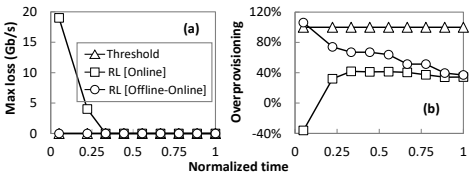

Fig. 2. Performance for use case 1

For the first use case, traffic flows in the range [20, 90] Gb/s were generated. Fig. 2a-b shows the results in terms of traffic loss and average relative overprovisioning (threshold = 80%), respectively, against normalized time (0 is the starting time of RL-based online operation and 1 the time when online RL converged to the best model). As expected, the RL-based agent with online learning starts operation with inadmissible traffic loss, which hinders its eligibility for reliable operation. On the contrary, both threshold and RL with offline+online learning (*lr*=0.1 and *df*=0.4) agents ensure no loss, being overprovisioning reduced by the latter as soon as operation allows improving models online. At normalized time 1, both RL-based agents reduced overprovisioning to about 40%.

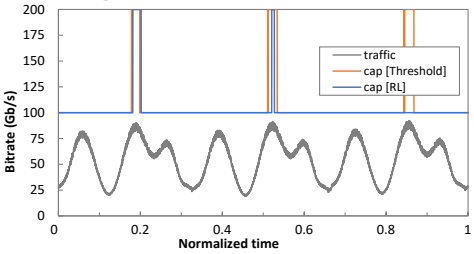

Fig. 3. Capacity allocation for use case 2

Fig. 3 illustrates the incoming traffic and the capacity allocated by the threshold-based and the RL agent with offline+online learning for use case 2; the gain of the RL- based method is obtained in terms of the number of lightpaths. Although both methods start with similar performance, RL learns a more efficient capacity usage, thus avoiding setting up additional lightpaths.

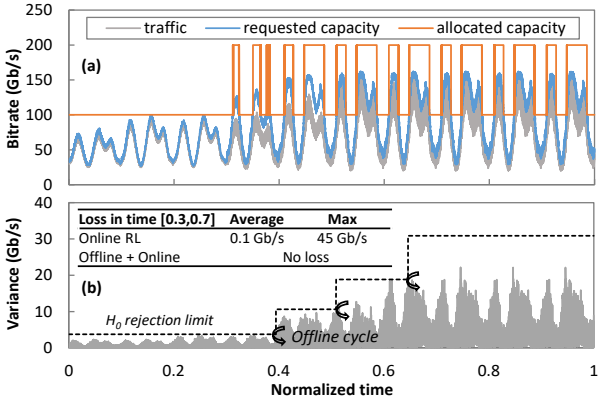

Fig. 4. Capacity and offline-online cycles for use case 3

Finally, in use case 3, traffic gradually doubles from time 0 to 1 and increases its variance 4 times. Algorithm I was configured with δ=120 minutes and ran to detect when an offline learning cycle needed to be triggered. Fig. 4a shows the traffic and the requested and allocated capacity for the flow. Fig. 4b shows the computed variance over time and the limit of accepting hypothesis H_0. We observe that three cycles were triggered, each one increasing the reference variance and consequently, the limit of the hypothesis test. This operation allows good capacity adjustment ensuring no traffic loss, even during the hard traffic changing period (from time 0.3 to 0.7). Note that the same scenario in a pure online RL-based operation with no offline RL support produced loss as high as 45 Gb/s in the same period (see inset in Fig. 4b).

To sum up, the RL-based flow operation extended with the offline phase showed to be as robust as that of a threshold-based one, while reducing overprovisioning. Results on three use cases with static and evolutionary traffic validate the applicability of the proposed offline-online learning cycle.

References

[1] A. Clemm *et al*., "Intent-Based Networking - Concepts and Definitions," IRTF draft, work-in-progress, 2020.
[2] S. Barzegar *et al*., "Reinforcement Learning -based Autonomous Multilayer Network," ECOC 2020.
[3] L. Velasco *et al*., "A Learning Life-Cycle to Speed-up Autonomic Optical Transmission and Networking Adoption," JOCN, 2019.
[4] R. Sutton and A. Barto, *Reinforcement learning: an introduction*, MIT Press, 1998.
[5] P. Brockwell and R. Davis, *Introduction to Time Series and Forecasting*, Springer, 2016.
[6] M. Ruiz *et al*., "CURSA-SQ: A Methodology for Service-Centric Traffic Flow Analysis," JOCN, 2018.