Optimisation contrainte d'une politique d'équilibrage de charge

Ahmed Yassine Kamri, Pham Tran Anh Quang, Nicolas Huin, Jérémie Leguay

Optimisation contrainte d'une politique d'équilibrage de charge

Ahmed Yassine Kamri et Pham Tran Anh Quang et Nicolas Huin et Jérémie Leguay

[1]*Huawei Technologies Ltd., Paris Research Center, France*

L'équilibrage de charge consiste à répartir le trafic entre une paire de nœuds sur plusieurs chemins afin d'améliorer les performances du réseau. Idéalement, la politique d'équilibrage devrait s'adapter en fonction du trafic et anticiper son impact sur le réseau mais l'intégration de modèles pour la Qualité de Service est un défi. Nous proposons une solution basée sur l'apprentissage par renforcement profond, qui est capable d'apprendre la relation entre le trafic et la QoS, tout en offrant la sécurité nécessaire pour maximiser le débit et éviter de violer les contraintes de capacité des liaisons. Elle intègre un algorithme d'optimisation de politiques sous contraintes. Dans un scénario SD-WAN où les délais suivent le modèle de mise en file d'attente M/M/1, nous démontrons, à l'aide d'un programme non linéaire en nombres entiers, que notre solution peut atteindre un délai de bout en bout proche de l'optimum. Nous montrons également que notre solution apprend automatiquement les paramètres de récompense pour répondre aux contraintes de capacité.

Mots-clefs : Equilibrage de charge, apprentissage par renforcement, SD-WAN.

1 Introduction

Tighter Quality of Service (QoS) requirements coming from 5G require network operators to move away from best-effort management of their network and rethink their traffic engineering strategies. Popular static load-balancing policies such as equal-cost multi-path (ECMP) [TH00] or its weighted variant [Zho+14] cannot adapt to QoS measurements. They either waste resources or violate end-to-end performance requirements. Using Software Defined Network (SDN) for centralized monitoring and control, several adaptive solutions have been developed such as Niagara [Kan+15] or IRSR [Med+16]. However, they minimize a linear routing cost or the Maximum Link Utilization (MLU), respectively, and do not explicitly optimize QoS metrics such as the end-to-end delay.

The integration of accurate QoS models into routing optimization algorithms can be challenging, but model-free solutions might be a good practical solution. Proposed models are either too simple and lack accuracy, or they become intractable [BO06]. In a previous work [HLM20], we explored decomposition methods for integrating an M/M/1 model but in an single-path routing algorithm. Deep Reinforcement Learning (DRL) proved, under the umbrella of *experience-driven networking* [Xu+18], to be quite effective for routing problems. To evaluate the action of an agent, it requires the definition of a relevant reward function.

The definition of the reward is one the most crucial aspect of Reinforcement Learning. Bouacida and Shihada [BS18] tackled this issue within their LearnQueue framework where they defined a two-part reward. As we will see in Section 2, their approach requires a *manually* tunable static parameter. We propose to leverage Reward Constrained Policy Optimization (RCPO) [TMM18], explained in Section 3 to improve on their approach and get rid of the manual configuration. We then compare, in Section 4, RCPO and LearnQueue on an SD-WAN instance with an Non-Linear Program (NLP).

2 Problem statement

2.1 System architecture

The network is composed of a centralized network controller in charge of deciding the load balancing policy. It periodically updates the policy to minimize the end-to-end delay of all tunnels. Each tunnel $k \in K$ comprises a set of origin-destination flows, between s_k and t_k, that can be split between multiple paths. The set of paths P_k available for a tunnel is stable over time and is provided by either a local or a centralized path computation module. The controller decides, at each time step t, the load balancing weights of each tunnel to determine how $D_k(t)$ the bandwidth of tunnel k is split among paths.

Links in the network have a maximum capacity $c_e, \forall e \in E$, where E is the set of links, and we choose to mimic TCP's behavior in case of congestion using a max-min fairness rate allocation [BGH92]. The rate of each tunnel is adjusted to link capacity constraints with a water-filling algorithm. The *enqueueing rate* $er(t)$ is defined as $\sum_{k\in K} \widehat{D}_k(t) / \sum_{k\in K} D_k(t)$, where $\widehat{D}_k(t)$ is the amount of admitted traffic for tunnel k at time t.

We choose to model the link delay using the simple M/M/1 queuing model, given by $d_e(t) = d_e^{prop} + 1/(c_e - l_e(t))$ where d_e^{prop} is the propagation delay of the link and $l_e(t)$ is the load of the link at time t. The delay of a path p is given by the sum of the delay of its link (i.e., $\sum_{e\in p} d_e(t)$), and the delay of a tunnel d_k^t is given by the maximum delay over all its paths (i.e., $\max_{p\in P_k} \sum_{e\in p} d_e(t)$).

2.2 Markov decision process

We formulate our load balancing problem as a Markov decision process (MDP) defined by the tuple $(S, A, R, P, \mu, \gamma)$. An observable network state $s_t \in S$ comprises the bandwidth of each demand; the action space A comprises the split ratio decisions; the reward function $R : S \times A \times S \to \mathbb{R}$ evaluates the delay of the demands (we further develop the reward later); the transition matrix P gives the transition probability between states after an action is taken; the initial state distribution is given by $\mu : S \to [0, 1]$; and the discount factor for future rewards is given by $\gamma \in [0, 1)$.

The objective is to find an optimal policy π, i.e., a probability distribution over actions where $\pi(a|s)$ denotes the probability of taking action a at state s. The value of a policy π for a state s is given by $V_R^\pi(s) = \mathbb{E}^\pi[\sum_t \gamma^t R(s_t, a_t)|s_0 = s]$ which can be rewritten in a recursive Bellman equation as follows $V_R^\pi(s) = \mathbb{E}^\pi[R(s,a) + \gamma V_R^\pi(s')|s]$, where s' is the state after executing action a in state s.

The goal of our policy is to minimize the average delay of tunnels in the network; our reward function must reflect that goal. However, an agent can exploit loopholes to maximize its reward. In our case, since we are mimicking TCP's congestion behavior, the agent could overload some parts of the network to trigger congestion control and subsequently decrease the delay for some tunnels. Bouacida and Shihada [BS18] proposed, with LearnQueue, a reward function to account for this loophole by integrating both the delay and the traffic enqueuing rate (*er*) in the reward. The reward is parametrized by δ and is given by $(1-\delta) \times \mathtt{er}^t - \delta \times \sum d_k^t$. Unfortunately, finding the right value for δ can be quite cumbersome and we alleviate this issue by using the Reward Constrained Policy Optimization [TMM18] algorithm.

3 Reward Constrained Policy Optimization algorithm

To deal with the throttling loophole, we propose to add link capacity constraints to the model and to only consider the average delay in the reward. Our problem now becomes a Constrained Markov Decision Problem, which is more complicated to solve using Deep Reinforcement Learning. One solution is to use Lagrange relaxation to move the constraints into the reward.

Lagrange relaxation requires three parts to transform a constrained MDP into an MDP: a penalty signal $c(s_t, a_t)$, to evaluate the violation of the constraint for a given action-state pair; a constraint $C(s_t) = F(c(s_t, a_t), ..., c(s_N, a_N))$, to evaluate the violation of the constraints over time, and a threshold α, to indicate if the constraints are hard or soft. We then try to maximize the discounted reward of policy π, i.e., $J_R^\pi = \mathbb{E}^\pi_{s_0\sim\mu}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)\right]$, with respect to the expectation over the constraint $J_C^\pi \leq \alpha$ where $J_C^\pi = \mathbb{E}^\pi_{s_0\sim\mu}[C(s)]$. The objective of the unconstrained problem using Lagrange relaxation technique is $\min_{\lambda\geq 0} \max_{\theta} \left[J_R^{\pi_\theta} - \lambda\left(J_C^{\pi_\theta} - \alpha\right)\right]$,

where π_θ is the parametrized policy with parameters θ. The penalized reward corresponding to the unconstrained MDP is $r'(\lambda, s_t, a_t) = r(s_t, a_t) - \lambda c(s_t, a_t)$ and the value of the new discounted penalized reward is $V'^{\pi}(\lambda, s_t) = V_R^{\pi}(s_t) - \lambda V_C^{\pi}(s_t)$, where $V_C^{\pi}(s) = \mathbb{E}^{\pi}[\sum_{t=0}^{\infty} \gamma^t c(s_t, a_t)]$ is the discounted guiding-penalty.

Tessler et al. [TMM18] devised the Reward Constrained Policy Optimization (RCPO) algorithm (see Alg. 1) to solve a constrained MDP using Lagrange relaxation. For each episode k, RCPO follows the conventional procedure of Actor-Critic algorithms to update actor and critic networks based on its experience. The expectation of the constraint is computed at the end of each episode and the Lagrange multiplier is updated accordingly. Under mild assumptions, RCPO converges to a constraint satisfying solution.

Algorithm 1: RCPO

Input: penalty $c(.)$, constraint $C(.)$, threshold α, learning rates $\eta_1(k) < \eta_2(k) < \eta_3(k)$

Initialize actor parameters $\theta = \theta_0$, critic parameters $v = v_0$, and Lagrange multipliers $\lambda = 0$

for *k=0,1,...* **do**

  Initialize state $s_0 \sim \mu$

  for *t=0,1,...,T-1* **do**

    Sample action $a_t \sim \pi$, observe next state s_{t+1}, reward r_t, and penalties c_t

    $\hat{R}_t = r_t - \lambda_k c_t + \gamma \hat{V}(\lambda, s_t; v_k)$

    Critic update: $v_{k+1} \leftarrow v_k - \eta_3(k) \left[\frac{\delta \left(\hat{R}_t - \hat{V}(\lambda, s_t; v_k) \right)^2}{\delta v_k} \right]$

    Actor update: $\theta_{k+1} \leftarrow \Gamma_\theta \left[\theta_k + \eta_2(k) \nabla_\theta \hat{V}(\lambda, s) \right]$

  Lagrange multiplier update: $\lambda_{k+1} \leftarrow \Gamma_\lambda \left[\lambda_k + \eta_1(k) \left(J_C^{\pi_\theta} - \alpha \right) \right]$

return policy parameters θ

4 Results

We evaluated our algorithms on a typical SD-WAN network where three remote sites are multihomed to headquarters (HQ) with Multi-Protocol Label Switching (MPLS) and broadband Internet connectivity. Tunnels exist between the remote sites and HQ in both direction. Each tunnel can route traffic on two different paths: the MPLS path provide 6 Mb/s and the broadband Internet path provides 15 Mb/s.

We made the following choices for hyper-parameters and neural network architecture. For both LearnQueue and RCPO, we use a two-layer network 128 neurons with *relu* activation function for the actor and critic networks. We trained the networks using Deep Deterministic Policy Gradient (DDPG) [Lil+16] over one million time steps with a discount factor γ of 0.7. During training, we explore the action space using a noise following an Ornstein-Uhlenbeck process, with a mean of zero and a standard deviation of 0.5. For LearnQueue, we considered two extreme values of δ — 0.1 and 0.9 — to have solutions focusing on either delay minimization or throughput maximization. Finally, for RCPO, we initialize the Lagrangian parameter λ to 0.8 and set its learning rate to 0.01.

In Figure 1, we compare the average delay, the accepted throughput and the Maximum Link Utilization (MLU) of our algorithms with a Non-Linear Integer program (NLP), not shown due to lack of space. The NLP minimizes the average delay of all origin-destination flows in the network, taking into consideration the capacity constraints and the M/M/1 queuing model for delay. It provides an optimal solution, often missing in comparison found in the literature.

LearnQueue is clearly handicapped by the fact that δ must be defined in advance. When the focus is set on delay minimization ($\delta = 0.9$), LearnQueue drops traffic even though an optimal solutions without drops exist. When the focus is set on throughput maximization ($\delta = 0.1$), LearnQueue drops no traffic but the resulting delay is more important than the one in solutions from RCPO. RCPO generates solutions close to the optimal in terms of delay, without dropping any traffic.

Finally, we observe that solutions found with NLP have an MLU that is not always the lowest while having the lowest average delays. This observation reinforces the idea that the MLU is not a good target to

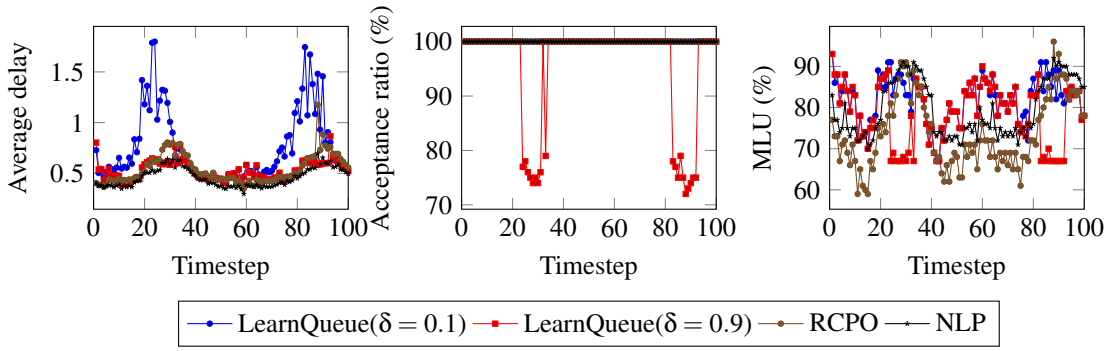

Figure 1: Comparison between the LearnQueue algorithm with $\delta \in \{0.1, 0.9\}$, the RCPO algorithm and the NLP algorithm. LearnQueue's algorithm shows a trade-off between accepted traffic and average delay while RCPO accepts 100% of the traffic with an average delay close to optimal.

minimize and that accurate QoS models needs to be explicitly integrated into load balancing optimization.

5 Conclusion

We have presented a deep reinforcement learning solution for load balancing to optimize the end-to-end average delay of a set of tunnels. To provide safety and avoid throughput degradation, we have enforced capacity constraints in the policy optimization using the RCPO algorithm. We have demonstrated that a close to optimal delay can be achieved while automatically learning reward parameters to meet capacity constraints. This approach outperforms the LearnQueue reward, which is difficult to parametrize.

References

[BGH92] Dimitri P Bertsekas, Robert G Gallager, and Pierre Humblet. *Data networks*. Vol. 2. Prentice-Hall International New Jersey, 1992.

[BO06] Walid Ben-Ameur and Adam Ouorou. "Mathematical models of the delay constrained routing problem". In: *Algorithmic Operations Research* 1.2 (2006).

[BS18] Nader Bouacida and Basem Shihada. "Practical and dynamic buffer sizing using LearnQueue". In: *IEEE Transactions on Mobile Computing* 18.8 (2018), pp. 1885–1897.

[HLM20] Nicolas Huin, Jeremie Leguay, and Sébastien Martin. "Génération de colonnes pour le problème de routage à délai variable". In: *ALGOTEL*. 2020.

[Kan+15] Nanxi Kang et al. "Efficient traffic splitting on commodity switches". In: *CoNEXT*. 2015.

[Lil+16] Timothy P. Lillicrap et al. "Continuous control wuth deep reinforcement learning". In: *ICLR* (2016).

[Med+16] Paolo Medagliani et al. "Global optimization for hash-based splitting". In: *Proc. IEEE GLOBECOM*. 2016.

[TH00] Dave Thaler and C Hopps. *RFC 2991: Multipath issues in unicast and multicast next-hop selection*. 2000.

[TMM18] Chen Tessler, Daniel J. Mankowitz, and Shie Mannor. "Reward Constrained Policy Optimization". In: *CoRR* abs/1805.11074 (2018). arXiv: `1805.11074`. URL: `http://arxiv.org/abs/1805.11074`.

[Xu+18] Zhiyuan Xu et al. "Experience-driven networking: A deep reinforcement learning based approach". In: *IEEE INFOCOM*. 2018.

[Zho+14] Junlan Zhou et al. "WCMP: Weighted Cost Multipathing for Improved Fairness in Data Centers". In: *Proc. ACM EuroSys*. 2014.