Distributed Multi-Hop Traffic Engineering via Stochastic Policy Gradient Reinforcement Learning

Pinyarash Pinyoanuntapong, Minwoo Lee, Pu Wang
Department of Computer Science
University of North Carolina at Charlotte, Charlotte, NC 28223, U.S.A.
{ppinyoan, minwoo.lee, pu.wang}@uncc.edu

***Abstract*—Multi-hop networks (e.g., mesh, ad-hoc, and sensor networks) are important and cost-efficient communication backbones. Over the last few years wireless data traffic has drastically increased due to the changes in the way today's society creates, shares, and consumes information. This demands the efficient and intelligent utilization of limited network resources to optimize network performance. Traffic engineering (TE) optimizes network performance and enables optimal forwarding and routing rules to meet the quality of service (QoS) requirements for a large volume of traffic flows. This paper proposes a distributed model-free TE solution based on stochastic policy gradient reinforcement learning (RL), which aims to learn an stochastic routing policy for each router so that each router can send a packet to the next-hop router according to the learned optimal probability. The proposed policy-gradient solution naturally leads to multi-path TE strategies, which can effectively distribute the high traffic loads among all available routing paths to minimize the E2E delay. Moreover, a distributed software-defined networking architecture is proposed, which enables the fast prototyping of the proposed multi-agent actor-critic TE (MA-AC TE) algorithm and in-nature supports automated TE through multi-agent RL learning.** [1].

I. Introduction

Motivation. Over the last few years, data traffic has drastically increased due to the changes in the way today's society creates, shares, and consumes information. Recent Cisco's Visual Networking Index (VNI) [1] forecasts three-fold IP and Internet traffic growth and expect 2.6 Exabytes of daily use in 2022. They project over 30 percent of the compound annual growth rate of internet traffic. Furthermore, wireless multi-hop networks (e.g., mesh, ad-hoc, and sensor networks) emerge as important and cost-efficient communication backbones for establishing interconnections among Internet of things, 5G small cells, CubeSats (miniaturized satellites), roof access points of community networks, first responders in disaster zones, and soldiers in battlefield.

Fulfilling the necessity to accommodate demand growth, and utilizing limited network resources intelligently, traffic engineering (TE) [2] is one of the most important methods. TE optimizes network performance by measuring real-time network traffic, and designing optimal routing rules to meet the quality of service (QoS) requirements for a large volume of traffic flows. End-to-end (E2E) delay is one of the key TE metrics. Minimizing E2E delay, however, is very challenging in large-scale wired networks and wireless multi-hop networks due to the profound dynamics in traffic flow patterns, working conditions of routers, network topology, and link status.

Challenges. To minimize E2E delay, the TE solutions can be classified into three categories: (1) heuristic approach, (2) model-based optimization, and (3) model-free optimization. The heuristic approaches, such as OSPF, IEEE 802.11s [3], and B.A.T.M.A.N. [4], are generally based on shortest path routing protocol. Heuristic methods are simple and easy to implement. But, they cannot achieve the optimal E2E TE performance.

The model-based optimization approaches are generally based on network utility maximization (NUM) framework [5], [6], where the TE problem is formulated as constrained maximization problems of the utility function. However, these solutions suffer from the following three issues. First, they rely on strong assumptions on the network model, such as per-flow per-link queuing structure, unbounded buffer size for each queue, bounded variance of traffic arrivals, and the availability of instantaneous link rates. These assumptions do not hold for practical computer networks. Second, they cannot be used to minimize E2E delay because the E2E delay cannot be mathematically modeled as explicit functions of TE control parameters, such as traffic splitting ratio over each output link [5], [6], which, however, have to be included in the utility function in the NUM formulation. Third, they are not designed to handle non-stationarity caused by the time-varying network dynamics, such as the ever-changing traffic patterns. Because of above limitations, these model-based optimization solutions are barely implemented as the TE solutions to handle practical multi-hop computer networks.

To address the limitations of model-based approaches, recent advances in multi-agent reinforcement learning (MARL) have enabled the development of model-free distributed TE optimization schemes [7]–[10], where each router, acting as an agent, learns the optimal local TE policy in such a way that the collective TE policy of all routers can achieve the optimal E2E TE performance. The model-free TE does not rely on accurate network modeling and unrealistic assumptions made in model-based methods. Moreover, model-free TE is designed to work well under inherent dynamics, uncertainties, and non-stationarity in multi-hop networks. The research on distributed model-free TE mainly focuses on applying action-value methods, such as Q-learning and its variants [11]–[15],

[1]This work was supported in part by the US National Science Foundation (NSF) under Grant No. 1763182

which learn an deterministic target policy that maximizes the TE objective, i.e., E2E delay, by greedily selecting a action, i.e., next-hop forwarding router. The fundamental limitation of action-value RL methods is that they can only learn deterministic routing policies. This naturally leads to single-path TE solutions, where a single routing path is learned between a source-destination pair. The single-path TE solutions are simple and easy to implement. However, to improve the delay performance under high traffic load, multi-path TE solutions are generally preferred, where each source-destination pair is connected with multiple routing paths to better distribute the traffic load.

Proposed Solution. To address the aforementioned limitations, we propose a distributed model-free TE solution based on stochastic policy gradient RL, which aims to learn an stochastic routing policy for each router so that each router can send a packet to the next-hop router according to the learned optimal probability. The proposed policy-gradient solution naturally leads to multi-path TE strategies, which can effectively distribute the high traffic loads among all available routing paths to minimize the E2E delay. In particular, this paper the following three contributions.

- We first formulate the distributed multi-path TE problem as a multi-agent Markov decision process (MA-MDP).
- Then, to solve this MA-MDP problem, we employ the multi-agent actor-critic algorithm, where each router has its own actor and critic. The local critic uses exponential weighted average (EWA) to estimate the action-value functions to criticize the action selections. Based on critics inputs, the actor improves the routing policy by representing the policy as a linear parametric probability distribution for next-hop router selection.
- To demonstrate the feasibility of the proposed solution, we develop a distributed software-defined networking architecture, which enables the fast prototyping of the proposed MA-AC TE algorithm and in-nature supports automated TE through multi-agent RL learning.

II. Preliminaries

A. Distributed TE as MA-MDP

Distributed TE can be formulated as multi-agent extension of Markov decision process (MA-MDP) of N routers [16], which is defined as a tuple of $<\mathcal{S}, \mathcal{O}_{1:N}, \mathcal{A}_{1:N}, \mathcal{P}, r_{1:N}>$. In this formulation, $\mathcal{S}$ models a set of environmental states, which include (1) the network topology, (2) the source and destination (i.e., source and destination IP addresses) of each packet in each router, (3) the number of packets (queue size) of each router, and (4) the status of links of each router, e.g., signal-to-interference-plus-noise ratio (SINR). $\mathcal{O}_i$ is a set of observations for each router i, which include local network states available at router i. $\mathcal{A}_i$ is a set of actions for each router i, which include the next-hop routers the current router can forward the packets to. The transition probability $\mathcal{P}$ models the environment dynamics. r_i is the reward function of each router i, which is the (negative) 1-hop delay.

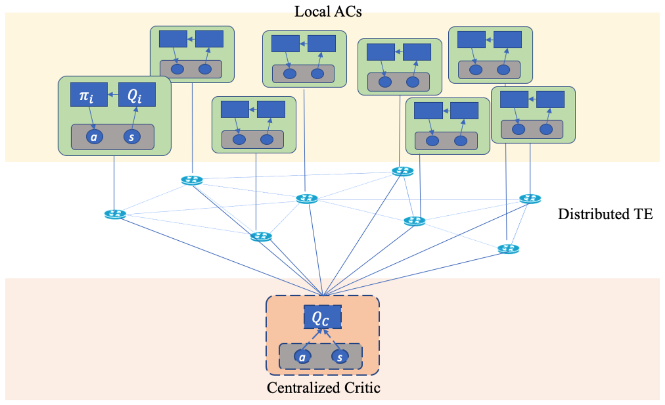


Fig. 1. Multi-agent actor-critic (AC) architecture. Each router's actor updates its policy function while critic updates the function approximation of state-action Q values.

When a packet enters a router i, from its local observation $o \in \mathcal{O}_i$ of the network states, the router determines where to send this packet to ($a \in \mathcal{A}_i$). As a result, the router receives a reward r_i (i.e., (negative) one-hop delay) when the packet arrives at its next-hop router $i+1$, which has its own local observation $o' \in \mathcal{O}_{i+1}$. Each router selects actions based on a local policy π_i, which specifies how the router chooses its action given the observation. The policy is stochastic $\pi_i(a|o) : \mathcal{O}_i \times \mathcal{A}_i \mapsto [0,1]$, where given current observation $o \in \mathcal{O}_i$, the router sends a packet to the next-hop router $a \in \mathcal{A}_i$ according to the probability $\pi_i(a|o)$ with $\sum_{a \in \mathcal{A}_i} \pi_i(a|o) = 1$. The return $G_i = \sum_{k=i}^{T} r_k$ is the total reward from intermediate state s_i to final state s_T, where s_i and s_T are the states when a packet arrives at the intermediate router i and destination router T, respectively. Let s_1 be the initial state when a packet enters the network from its source router. The goal is to find the optimal policy π_i for each router i so that the expected return $J(\boldsymbol{\pi})$ (i.e., expected E2E delay) from the initial state is maximized,

$$J(\boldsymbol{\pi}) = E[G_i|\boldsymbol{\pi}] = E[\sum_{i=1}^{T} r_i|\boldsymbol{\pi}] \qquad (1)$$

where $\boldsymbol{\pi} = \pi_1, ..., \pi_N$. In TE problems, the state is uniquely defined by the observations of all routers. Thus, in the following sections, for simplicity of notation, we use represent an observation o as a state s.

B. Multi-agent Actor-Critic Architecture

To solve the above MA-MDP problem, we adopt the multi-agent actor-critic-executor (MA-ACE) architecture as shown in Fig. 1, which proposed in our previous work [16], which is originally designed for implementing action-value based TE. In this paper, we extend it to implement the policy-based TE, multi-agent actor-critic traffic engineering (MA-AC TE). In MA-AC TE architecture, each router has its own actor and critic running locally. The local critic uses a variety of methods, such as exponential moving average and function approximation, to estimate the action-value functions $q_i^{\pi_i}(s,a)$, which criticize the action selections. Using critic's inputs, the actor improves the stochastic target policy.

In particular, the stochastic target policy for each router i is represented by a parametric probability distribution $\pi_{\theta_i}(a|s)$. The parametric policy $\pi_{\theta_i}(a|s)$ stochastically selects action a in state s according to parameter vector θ_i. As a result, the expected return in eq. (1) also becomes parametric, $J(\boldsymbol{\pi}) = J_{\boldsymbol{\theta}}(\boldsymbol{\pi}_{\boldsymbol{\theta}})$, where $\boldsymbol{\theta} = \theta_1, ..., \theta_N$. Then, the optimal joint policy $\boldsymbol{\pi}$ can be learned by updating the policy parameter $\boldsymbol{\theta}$ in the direction of the gradient $\nabla_{\boldsymbol{\theta}} J_{\boldsymbol{\theta}}(\boldsymbol{\pi}_{\boldsymbol{\theta}})$. In the multi-agent setting, by adopting the policy gradient theorem [17], [18], each router i updates its policy parameter θ_i along the direction of the partial derivative $\nabla_{\theta_i} J_{\boldsymbol{\theta}}(\boldsymbol{\pi}_{\boldsymbol{\theta}})$,

$$\nabla_{\theta_i} J_{\boldsymbol{\theta}}(\boldsymbol{\pi}_{\boldsymbol{\theta}}) = E\left[q_i^{\pi_i}(s,a)\nabla_{\theta_i}\log(\pi_{\theta_i}(a|s))\right]. \quad (2)$$

Intuitively, eq. (2) indicates that the policy parameter θ_i is updated in such a direction that the action with larger action-value or expected return is conservatively assigned with a higher probability.

III. Policy-gradient Distributed TE

A. Critic with EWA Policy Evaluation

As shown in eq. (2), the performance of the policy π is measured by the action-value $q_i^{\pi}(s,a)$, which is a E2E TE metric. Thus, there will be no direct training sample for policy evaluation until a packet forwarded by this router arrives at its destination. Inspired by temporal-difference prediction [18], we can apply *spatial-difference (SD) prediction* to quickly update the estimation of $q_i^{\pi}(s,a)$ only using local information exchanged between adjacent routers. In particular, the action-value $q_i^{\pi}(s,a)$ of router i can be recursively rewritten as the sum of 1-hop reward of router i and the action-value of the next-hop router $i+1$, i.e.,

$$q_i^{\pi_i}(s,a) = E\left[r_i + q_{i+1}^{\pi_{i+1}}(s',a')\right]. \quad (3)$$

This equation (3) indicates $q_i^{\pi_i}(s,a)$ can be estimated by averaging the samples of $r_i + q_{i+1}^{\pi_{i+1}}(s',a')$. This leads to a simple SD predication method based on *exponential weighted average (EWA)*, which iteratively updates the estimate of $q_i^{\pi_i}(s,a)$, denoted by $Q_i^{\pi_i}(s,a)$, based on 1-hop experience tuples (s,a,r_i,s',a') and the estimate of $q_{i+1}^{\pi_{i+1}}(s',a')$ of next-hop router, denoted by $Q_{i+1}^{\pi_{i+1}}(s',a')$, i.e.,

$$Q_i^{\pi_i}(s,a) \leftarrow Q_i^{\pi_i}(s,a) + \alpha[r_i + Q_{i+1}^{\pi_{i+1}}(s',a') - Q_i^{\pi_i}(s,a)] \quad (4)$$

where $\alpha \in (0,1]$ is the learning rate of the critic.

B. Actor with Time-average Stochastic-Gradient Policy Improvement

One of key challenges faced by policy-gradient methods is how to properly parameterize the policy without inducing unnecessary learning complexity and instability. Since our action space is discrete and not large, for each router i, we form a parameterized preference function $h_i(s,a,\theta_i)$ for each state-action pair (s,a). Then, the policy π_i is parameterized through exponential softmax distribution,

$$\pi_{\theta_i}(a|s) = \frac{\exp(h(s,a,\theta_i))}{\sum_{b\in\mathcal{A}_i}\exp(h_i(s,b,\theta_i))} \quad (5)$$

which indicates that the action with higher preference in each state is assigned with higher probability. The preferences $h_i(s,a,\theta_i)$ themselves can be parameterized nonlinearly through deep neural networks, where the θ_i are the network weights. However, since the local state space is not large and wireless routers do not have GPU-accelerated computing units, we will adopt computation-light linear parameterization for preference $h_i(s,a,\theta_i)$, i.e.,

$$h_i(s,a,\theta_i) = \theta_i^T \mathbf{x}(s,a) \quad (6)$$

where $\mathbf{x}(s,a)$ is the feature vector. Each component $x_j(s,a)$ of $\mathbf{x}(s,a)$ is a function of the state-action pair (s,a). In this paper, we employ the simplest feature function, which is the indicator function. This leads to a feature vector $\mathbf{x}(s,a) = (\mathbf{1}[(s,a)=(s_1,a_1)], ..., \mathbf{1}[(s,a)=(s_n,a_n)])$. As a result, the **target policy** in (6), which is the policy we want to learn and optimize, becomes

$$\pi_{\theta_i}(a|s) = \frac{e^{\theta_i^T \mathbf{x}(s,a)}}{\sum_{b\in\mathcal{A}_i} e^{\theta_i^T \mathbf{x}(s,b)}}.$$

The logarithmic gradient of the parametric policy gradient is computed by

$$\nabla_{\theta_i}\log\pi_{\theta_i}(a|s) = \mathbf{x}(s,a) - \sum_b \mathbf{x}(s,b)\pi_{\theta_i}(s|b). \quad (7)$$

Then, according to eq. (2), to find the policy that minimizes E2E delay, the local policy of each router is updated along the gradient direction defined in eq. (2), where the new θ_i^{t+1} is updated based on the previous θ_i^t, the action value estimate from local critic and logarithmic policy gradient $\log\pi_{\theta_i^t}(a|s)$, i.e.,

$$\theta_i^{t+1} = \theta_i^t + \beta Q_i^{\pi_i}(s,a)\nabla_{\theta_i^t}\log\pi_{\theta_i^t}(a|s) \quad (8)$$

where $\beta \in (0,1]$ is the learning rate of the actor. To further reduce the variance in the learned policy, we calculate time-average value of the parameter $\bar{\theta}_i^{t+1} = \frac{1}{t+1}\sum_{k=1}^{t+1}\theta_i^k = \frac{t+1}{t}\theta_i^t + \frac{1}{t}\theta_i^{t+1}$. The **behavior policy**, which generates the actual action, adopts the time-averaged parameter $\bar{\theta}_i^{t+1}$, i.e.,

$$\bar{\pi}_{\theta_i}(a|s) = \frac{e^{\bar{\theta}_i^T \mathbf{x}(s,a)}}{\sum_{b\in\mathcal{A}_i} e^{\bar{\theta}_i^T \mathbf{x}(s,b)}}.$$

IV. Distributed SDN for Distributed Policy-Gradient TE

A. Design Principles

The multi-agent learning framework for distributed TE calls for new network architecture, which provides programmable control and programmable measurement capabilities. Software-defined networking (SDN) technologies provide such capability. In particular, the main ideas of SDN are (i) separating the data plane from the control plane, (ii) employing the programmable forwarding table through an open and standardized interface, e.g., Openflow, and (iii) using a centralized network controller to change the networking behavior.

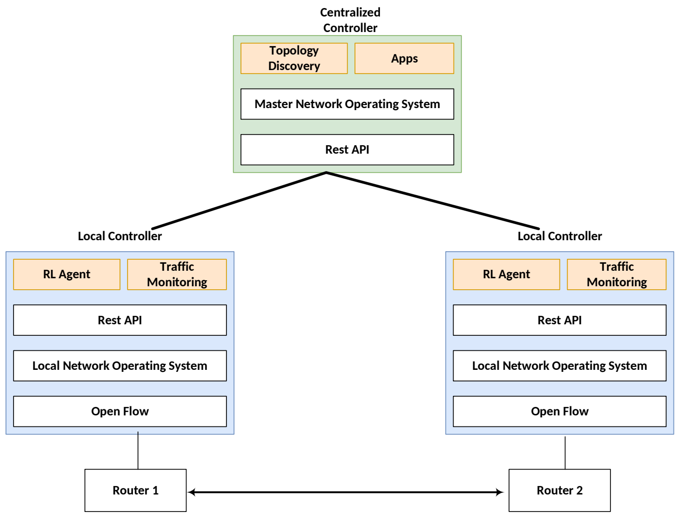


Fig. 2. Distributed software-defined network (D-SDN) architecture. Our prototype of the wireless mesh network that adopts D-SDN is at http://www.softmeshnet.com/

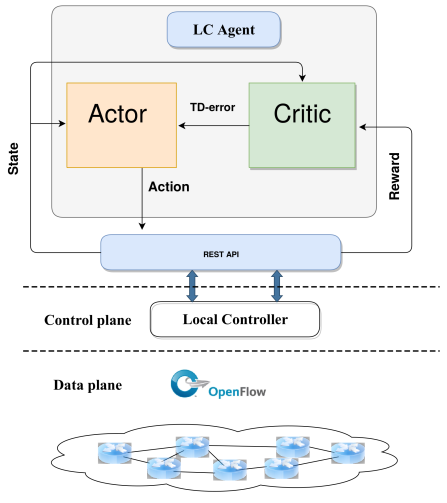


Fig. 3. Local reinforcement learning (actor-critic) agent for D-SDN

Despite its advantages, current SDN paradigm cannot be directly applied in wireless multi-hop networks, because they are mainly designed for high-speed wired networks, where centralized control and measurement are feasible. To address such challenge, we develop a distributed software-defined network (D-SDN) architecture (Fig. 2), which is particularly suitable for wireless multi-hop networks, where distributed control is highly desirable due to the lack of reliable wireless control channel. In particular, D-SDN has two types of network controllers: the centralized controller running on a server, and local controller running on each software-defined router. The centralized controller provides (i) the southbound APIs to enable real-time programmable control of the physical-, MAC-, and network-layer functions of wireless mesh routers, (ii) the high-level network applications (such as topology discovery and mobility management) for simplified and automatic network management, and (iii) the northbound APIs for easy access to the high-level network services and the global network state information (such as network topology and router health conditions). The local controller include programmable flow table (i.e., openflow table), which can be either updated by the centralized controller or by the local reinforcement learning engine (Fig. 3), which implements the proposed actor-critic algorithm.

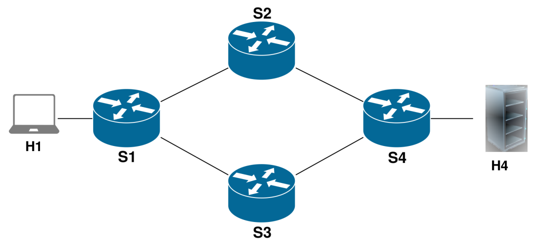


Fig. 4. Mesh Network Topology

In particular, whenever a packet comes to a router, a forwarding rule is executed, which includes MATCH and ACTION fields. If there is a MATCH entry for the packet header fields, e.g., destination IP, then the corresponding ACTION (e.g., forwarding the packet to a specfic output port) will be executed for that packet. In our case, the forwarding action (i.e., output port) is modified for every N packets according to the learned probability $\pi_{\theta_i}(a|s)$.

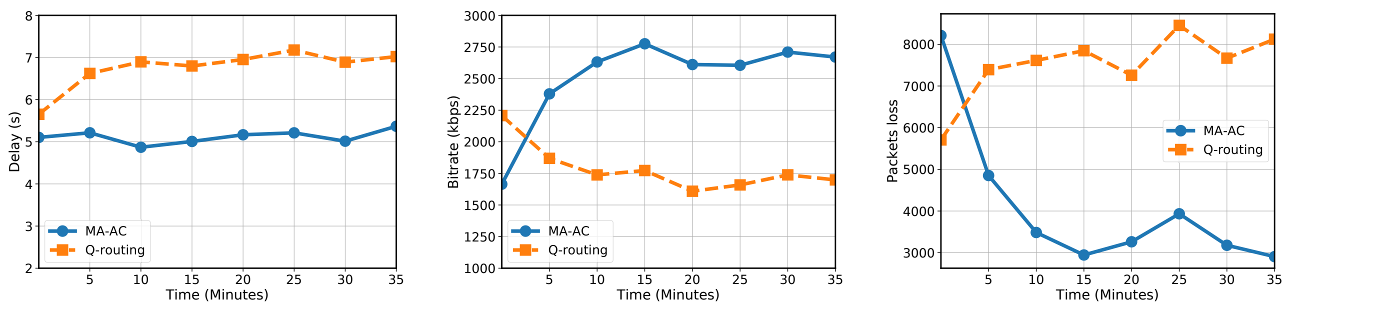
B. Feasibility Analysis

In this paper, we implement the distributed SDN using Mininet SDN emulator [19], where each software-define router is implemented by a software switch called openvswtich [20]. Both centralized and local controllers are implemented by modifying RYU SDN controller [21]. To verify the behavior of the MA-AC TE algorithm, we establish a small-scale multi-hop network with four routers as shown in Fig. 4. The terminal computer H_1 injects a traffic flow to the destination of computer H_4 with an Poisson distributed packet inter-departure time characterized by an average packet rate of 300 packets per second and packet sizes of 1500 Bytes. The traffic runs for 35 minutes. Thus, the data rate of the traffic flow has 3.6 Mbps. The link rate is 1.5 Mbps and the link delay is 50 ms. We compare the performance of our policy-gradient TE with the action-value based TE, i.e., the celebrated Q-learning based routing [11]. In particular, the action-value control algorithms aim to learn an deterministic routing policy, which maximizes the performance objective $J(\boldsymbol{\pi})$ in eq. (1) by greedily selecting a fixed action. This can be done by letting each router i greedily improve its current policy π_i, i.e., select the action with the maximum estimated action-value,

$$\pi_i(s) \leftarrow \arg\max_a Q_i^{\pi_i}(s,a).$$

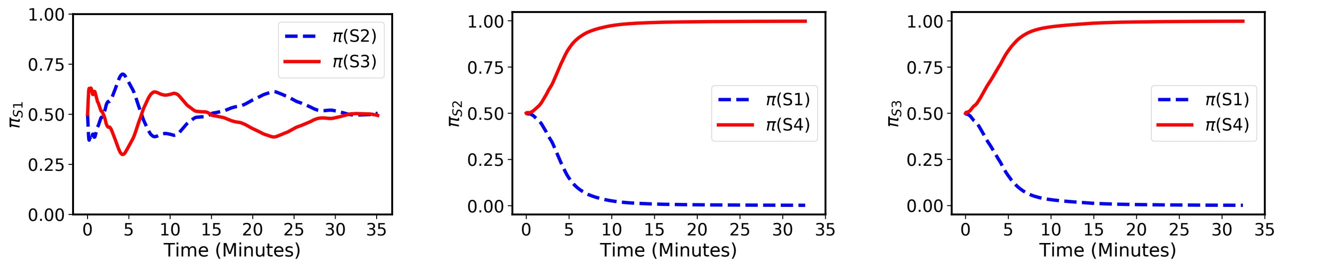
Since action-value based methods can only learn deterministic policies, this naturally leads to single-path TE solutions.

The Fig. 5 shows the average delay, throughput, and packet loss rate for every 5 minutes of policy-gradient TE solution (i.e., MA-AC TE), compared with the value-based TE solution (i.e., Q-learning TE). It can be observed that the policy-gradient TE achieves almost two times higher throughput than



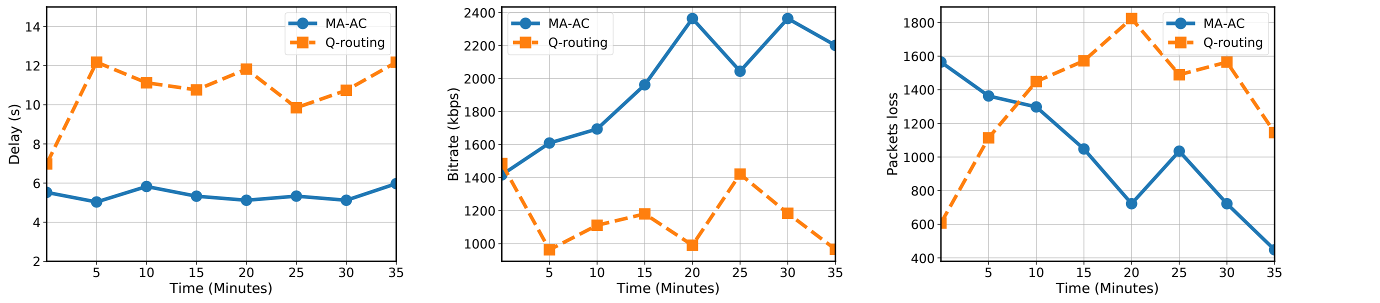
(a) E2E Delay (b) Throughputs (c) Packet Loss

Fig. 5. Performance comparison between MA-AC TE routing and Q-learning based routing.



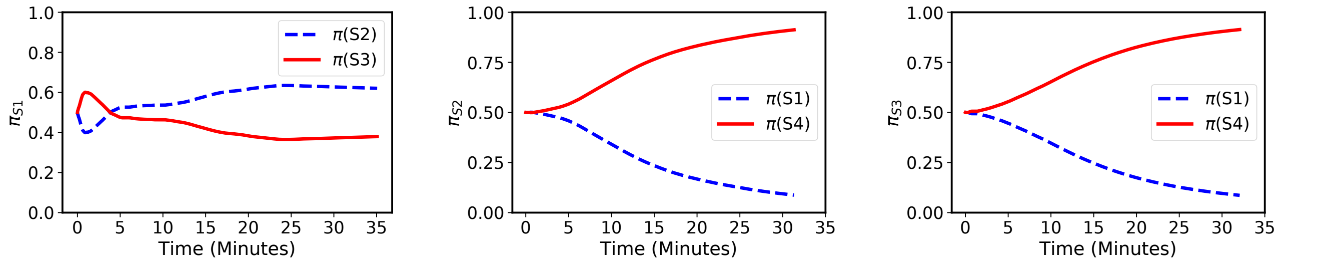
(a) Policy of edge router S1 (b) Policy of core router S2 (c) Policy of core router S3

Fig. 6. Learned routing policies of each router under MA-AC TE. For each router, each curve shows the probability of selecting a particular next-hop router.



(a) E2E Delay (b) Throughputs (c) Packet Loss

Fig. 7. Performance comparison of uneven links between MA-AC TE routing and Q-learning based routing.



(a) Policy of edge router S1 (b) Policy of core router S2 (c) Policy of core router S3

Fig. 8. Learned routing policies of each router under MA-AC TE with uneven bandwidth and link delay. For each router, each curve shows the probability of selecting a particular next-hop router.

Q-learning TE, with much less delay and comparably small packet loss rate. This is because policy-gradient TE enables multi-path forwarding, where two routing paths $S1-S2-S4$ and $S1-S3-S4$ are used, while one routing path is learned through Q-learning TE. More specifically, it is shown in Fig. 6 that the learned forwarding policy of MA-AC TE converges as time proceeds. When the policy converges, for the edge-router, it selects the next-hop routers $S2$ and $S3$ with equal probability (i.e., 0.5) as shown in Fig. 6(a). As shown in Fig. 6(b) and 6(c), both core router $S2$ and $S3$ select the $S4$ router on the forward path with a probability around 1, while selecting $S1$ router on the backward path with a probability around 0. It is evident that by combining the learned local policies of each router, we obtain the collective routing policy, which balances the traffic flow between two routing paths that have the same end-to-end link rate. Apparently, such policy is optimal in terms of minimizing E2E delay and maximizing network throughput. To show the adaptability of our algorithm to network dynamics, we increase the link rate of $S1-S2$ and $S2-S4$ to $2Mbps$ and link delay to 35 ms and decrease the link rate of $S1-S3$ and $S3-S4$ to $1Mbps$ and link delay to 65 ms. As shown in Fig. 7, policy gradient TE can still maintain much lower delay, higher throughput, and smaller packet loss, compared with value-based Q-routing. The key reason is that the edge router $S1$ and core router $S2$ collaboratively learned to allocate more traffic volume (around 65%) to the path $S1-S2-S4$ that has higher end-to-end data rate ($2.0Mbps$), compared the other path $S1-S3-S4$ that has lower end-to-end data rate ($1.0Mbps$).

V. Conclusions

In this paper, we propose a distributed model-free TE solution based on stochastic policy gradient RL, which aims to minimize the E2E delay by allowing each router to send a packet to the next-hop router according to the learned optimal probability. The proposed policy-gradient solution naturally leads to multi-path TE strategies, which can balance the traffic loads among all available routing paths to minimize the E2E delay and maximize network throughput. Moreover, a distributed software-defined networking architecture is proposed, which enables the fast prototyping of the proposed MA-AC TE algorithm and in-nature supports automated TE through multi-agent RL learning. In the future work, we will implement and test the proposed policy-gradient TE in our software-defined wireless mesh network testbed (www.softmeshnet.com).

References

[1] Cisco visual networking index: Forecast and trends. [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white-paper-c11-741490.html

[2] S. Agarwal, M. Kodialam, and T. Lakshman, “Traffic engineering in software defined networks,” in *2013 Proceedings of IEEE INFOCOM*, 2013, pp. 2211–2219.

[3] G. R. Hiertz, D. Denteneer, S. Max, R. Taori, J. Cardona, L. Berlemann, and B. Walke, “Ieee 802.11 s: the wlan mesh standard,” *IEEE Wireless Communications*, vol. 17, no. 1, 2010.

[4] The open mesh networks consortium. [Online]. Available: http://www.open-mesh.org

[5] D. P. Palomar and M. Chiang, “A tutorial on decomposition methods for network utility maximization,” *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 8, pp. 1439–1451, 2006.

[6] S.-C. Lin, P. Wang, I. F. Akyildiz, and L. Min, “Utility-optimal wireless routing in the presence of heavy tails,” *IEEE Transactions on Vehicular Technology*, 2018.

[7] 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 *2016 IEEE International Conference on Services Computing (SCC)*, 2016, pp. 25–33.

[8] G. Stampa, M. Arias, D. Sanchez-Charles, V. Muntés-Mulero, and A. Cabellos, “A deep-reinforcement learning approach for software-defined networking routing optimization,” *arXiv preprint arXiv:1709.07080*, 2017.

[9] Z. Fadlullah, F. Tang, B. Mao, N. Kato, O. Akashi, T. Inoue, and K. Mizutani, “State-of-the-art deep learning: Evolving machine intelligence toward tomorrows intelligent network traffic control systems,” *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2432–2455, 2017.

[10] Z. Xu, J. Tang, J. Meng, W. Zhang, Y. Wang, C. Liu, and D. Yang, “Experience-driven networking: a deep reinforcement learning based approach,” in *Proceedings of IEEE INFOCOM*, 2018.

[11] J. A. Boyan and M. L. Littman, “Packet routing in dynamically changing networks: A reinforcement learning approach,” in *Advances in Neural Information Processing Systems (NIPS)*, 1994, pp. 671–678.

[12] L. Peshkin and V. Savova, “Reinforcement learning for adaptive routing,” in *Proceedings of the 2002 International Joint Conference on Neural Networks (IJCNN’02)*, vol. 2, 2002, pp. 1825–1830.

[13] Y. Shilova, M. Kavalerov, and I. Bezukladnikov, “Full echo q-routing with adaptive learning rates: a reinforcement learning approach to network routing,” in *2016 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus)*, 2016, pp. 341–344.

[14] M. Kavalerov, Y. Shilova, and Y. Likhacheva, “Adaptive q-routing with random echo and route memory,” in *2017 20th Conference of Open Innovations Association (FRUCT)*, 2017, pp. 138–145.

[15] M. V. Kavalerov, Y. A. Shilova, and I. I. Bezukladnikov, “Preventing instability in full echo q-routing with adaptive learning rates,” in *2017 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus)*, 2017, pp. 155–159.

[16] P. Pinyoanuntapong, M. Lee, and P. Wang, “Delay-optimal traffic engineering through multi-agent reinforcement learning,” in *2019 IEEE INFOCOM Workshop: NI 2019: Network Intelligence: Machine Learning for Networking*, April 2019.

[17] 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 (NIPS)*, 2000, pp. 1057–1063.

[18] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[19] Mininet. [Online]. Available: http://mininet.org/

[20] Open vswitch. [Online]. Available: https://www.openvswitch.org/

[21] Ryu. [Online]. Available: https://osrg.github.io/ryu/