Improving Inter-domain Routing through Multi-agent Reinforcement Learning

Xiaoyang Zhao*, Chuan Wu*, Franck Le†
*Department of Computer Science, The University of Hong Kong, Email: {xyzhao, cwu}@cs.hku.hk
†IBM T. J. Watson Research Center, Email: fle@us.ibm.com

***Abstract*— Border Gateway Protocol (BGP), the *de-facto* inter-domain routing protocol, allows Autonomous Systems (AS) to apply their own local policies for selecting routes and propagating routing information. However, BGP cannot make performance-based routing decisions, and instead often routes traffic through congested paths, resulting in poor performance. This paper presents an efficient and scalable multi-agent reinforcement learning (MARL) method for inter-domain routing. It allows ASes to achieve higher overall throughput for real-time traffic demand, with the following highlights: (1) it ensures that traffic is forwarded along policy compliant paths; (2) it satisfies partial observability and selfishness of each AS; (3) the proposed solution is scalable as it only requires ASes to share information within a limited radius; (4) the solution is incrementally deployable, requiring only tens of ASes in the entire network to run it to start reaping benefits. We conduct extensive evaluation on actual network topologies ranging from hundreds to tens of thousands of ASes. The results show throughput improvements of up to 17% as compared to default BGP routing.**

I. Introduction

Border Gateway Protocol (BGP) allows each Autonomous System (AS) to implement its own local policy. Despite its flexibility and remarkable success, BGP cannot make performance-based routing decisions. As a result, traffic is often routed through paths that are congested, leading to poor performance (e.g., low throughput, high latency). This limitation can severely impact the performance of emerging large-scale data science projects (e.g., Large Hadron Collider, Square Kilometre Array, and Linac Coherent Light Source) [1], where large datasets frequently need to be transferred across sites, and analyzed by distributed systems.

We argue there is ample room for improvement based on the following insight: Among equally preferred routes (e.g., multiple customer routes to a same destination prefix), the BGP best path selection algorithm implements multiple tie-breakers through subjective (e.g., shortest AS PATH length) and arbitrary rules (e.g., lowest neighbor BGP RID) [2]. Instead of relying on those arbitrary tie-breakers, among equally preferred routes, can a reinforcement learning (RL) approach help select the best route to maximize performance (e.g., maximize overall throughput)? Adopting a RL based approach to address that problem raises several questions and challenges:

This work was supported in part by grants from Hong Kong RGC under the contracts HKU 17204619, 17225516, C5026-18G(CRF).

(1) RL requires exploring, and the long learning period may raise concerns. In particular, can a RL solution provide improvements especially as Internet traffic patterns keep evolving?

(2) How much information should the ASes share? More information provide better visibility and can result in larger improvement. However, it can also raise concerns about scalability and privacy.

(3) Can the solution be incrementally deployable? Requiring every AS to adopt and deploy the solution before being able to observe benefits will make adoption challenging.

In this paper, we pursue a multi-agent reinforcement learning (MARL) framework for inter-domain routing that achieves higher system throughput of transferring real-time traffic, while satisfying partial observability and selfishness of each AS. We learn the routing model that will be deployed on each AS using MARL. The proposed model will encode network status implicitly in a neural network (NN) that maps flow observation to a routing policy (e.g., which next-hop to forward flows). We overcome challenges described above to improve inter-domain routing as follows:

▷ We design an efficient MARL model to improve system throughput, which is deployed on ASes. Each AS trains and executes its own routing policy in a distributed way, with the objective to maximize average throughput of by-passing traffic flows. We show that our solution can improve system throughput by 20% above in environments traced from reality and by 14% above in large-scale networks with heavy traffic. This is because an AS learns from partial observations, which greatly restricts the input dimension to the RL model.

▷ We compare performance by allowing ASes to share their observations with 1-hop neighbors, 2-hop neighbors, 3-hop neighbors and all other ASes. Results show that sharing more information can bring better benefits. However, we found that when information is only shared among 1-hop and 2-hop neighbors, our solution still achieves 22% and 24% improvement respectively.

▷ Our model is incrementally deployable and scalable. We cut down AS's action space and adopt multiple inferences [3] to ensure that the dimension of input/output is not related to the number of overall flows and ASes. With only twenty Tier 1 ASes deploying the model, it can offer significant performance improvement by 14% in a large-scale network.

II. Background and Related Work

Internet Routing Optimization has been substantially studied, e.g., using Multi-Protocol Label Switching (MPLS) or modifying BGP parameter settings [4] [5]. Software defined networking (SDN) based routing systems have been studied in recent years. With SDN, desired routing paths can be set up for particular applications. The Multi-dimension Link Vector (MLV) Network [6] uses OpenFlow in data plane and enables flexible inter-domain routing. Google and Facebook have SDN-based Internet routing systems, i.e., Espresso [7] and Edge Fabric [8], respectively. SDN-based systems are effective but not widely deployed; we therefore still focus on improving the existing networks.

RL has recently been used for sequential decision making in traiffic engineering problems. Iroko [9] is an emulator, where agent can quickly learn a fair distribution policy, despite the volatility of the network traffic. Zou *et al.* [10] adopt a sequence-to-sequence model to learn implicit forwarding paths based on empirical network traffic data. Both studies assume the agent has a global view of network state, which is impractical in Internet environment.

MARL has been used for routing in different networks. Ye *et al.* [11] use MARL for wireless sensor network routing, where an agent chooses a cooperative neighboring set and forwards packets using Q-learning algorithm, to reduce delivery latency. Pourpeighambar *et al.* [12] propose multi-agent routing solutions to minimize average end-to-end delay in a cognitive ratio network. Mao *et al.* [13] design an Accnet framework for intra-domain routing which requires information sharing among all agents. We are the first to study MARL for inter-domain routing, with distributed routing policies to optimize system throughput while satisfying partial observability and selfishness of ASes.

III. MARL-based Inter-doman Routing

A. System Overview

The Internet consists of multiple ASes, each of which interacts with others in the process of business negotiation and traffic management. We abstract business relationships between ASes to be *customer-provider* and *peer-peer*, as illustrated in Fig. 1. The routing advertisement between connected ASes follows basic BGP rules [2]: (i) customer advertises all known routing paths to its providers; (ii) provider advertises routing paths known from all customers to each of its customers; (iii) peer advertises routing paths known from all customers to each of its peers.

End-to-end traffic flows with indicated sources and destinations are cooperatively transferred by correlative ASes. Each AS follows routing policy produced by its RL model to choose next-hops for by-passing flows. The routing policy satisfies basic requirements of business relationship: (i) prefer customer ASes as next-hop to peer ASes; (ii) prefer peer ASes as next-hop to provider ASes.

We abstract the network to be a topology containing N nodes (ASs) and E edges (links between correlative ASs). The capacity of edge $e \in E$ is C_e. The traffic demand F is an $N \times N$ binary matrix, where element (f_s, f_d) with value 1 represents that there is a flow whose source node is f_s and destination node is f_d.

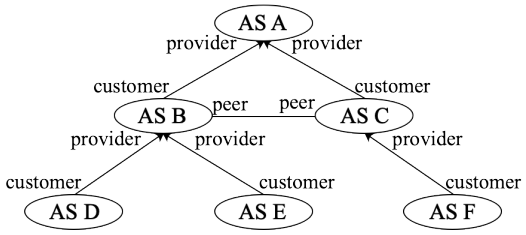

Fig. 1: Types of AS relationships: ASes D, E and F are customers of those above; ASes B and C are both providers of ASes below and customers of ASes above; ASes B and C are peers of each other; AS A is a provider to B and C.

In our MARL framework, each AS is an agent that aims to maximize average throughput of its own by-passing end-to-end flows. Training and executing are performed in a decentralized manner. We consider the system objective to be maximizing overall throughput, i.e., sum of all flows' throughput.

B. RL Model

We present detailed design of the RL model on each agent.

State Space. Each agent has its own observable field. We define the observation of agent i as $o_i = (\vec{w}, \mathbf{x}, \vec{h}, \vec{v})$:

- $\vec{w}$ is a 2-dimensional vector, indicating source and destination of the flow to be routed with the current inference. Source and destination of traversing flows can be extracted by by-passing agents through simple passive monitoring.
- $\mathbf{x}$ is an $L \times 2$ matrix encoding flows that the agent needs to transfer, where L is the maximum number of flows that can be transferred by an agent concurrently. Each vector $\vec{x}_n$ in $\mathbf{x}$, is a 2-dimensional vector, indicating source and destination of one flow.

Encoding this information helps agent produce routing policy considering potential traffic demand. We do not encode the entire traffic matrix because agents are partially observable and knowing traffic demand among the whole network is unrealistic.

- $\vec{h}$ is an m-dimensional vector where m is the number of neighboring agents. Each element represents traffic load status of the link connecting to one neighboring agent, computed as the number of flows on that link divided by its capacity. For each AS, connecting links are the most directly observable knowledge to learn network status.
- $\vec{v}$ is an L-dimensional vector. Each element represents observed throughput of the respective flow using routing decision from the last inference. Such throughput information helps an agent learn link status far beyond the immediate neighbors, useful for network load-balancing. A routing agent can derive throughput of an end-to-end flow through passive monitoring of the number of transferred packets with corresponding TCP flow header per unit time.

We define the input state of the RL model at each agent as $s_i = (o_k, a_{kj})_{k \in R, j \in F}$, where a_{kj} represents agent k's action for flow j (meaningful only if j tranverses k) and $R \subseteq N$. In reality, an AS observes status of its own by-passing flows and may also learn some flow information from neighboring ASes. Information sharing among ASes is limited because of partial observability and selfishness of the ASes. We consider different levels of information sharing in order to gain insights on how information exchange influences learning effectiveness.

(1) $R = N$. Agent i has observations and actions of all agents in the network. Knowing other agents' observations and actions allows an agent a maximum view of the whole network. Having global state is not realistic for inter-domain routing, but can be used as a baseline for performance evaluation.

(2) $R = M_i$, where M_i represents the neighboring agent set (including itself) of agent i. Compared with global state, neighboring state is more realistic to observe, for extended observable filed at each agent.

(3) $R = D_i$, where D_i represents 2-hop neighboring set of agent i.

(4) $R = T_i$, where T_i represents 3-hop neighboring set of agent i.

(5) $R = \{i\}$. Agent i only has its own observations, i.e., local state. In this case, individual agents take actions independently and regard others as part of the environment.

Action Space. After collecting s_i, agent i selects an action a_i based on policy $\pi_\theta^i(a_i|s_i)$, which is a probability distribution over action space. The policy is produced by a neural network (NN) with θ as the set of parameters. Naturally, an agent can transmit different flows concurrently. For flows with different destinations, candidate next-hop sets can be different because of BGP routing advertisement. Considering action space as a combination of concurrent flows' next-hop sets will lead to a large action space. To reduce the action space and expedite policy learning, we perform multiple inferences to produce action for each flow sequentially [3], and simplify the action space to be neighboring agent set M_i. In this way, the size of the action space is the degree of the agent.

To satisfy the common business peering relationship policies, in the output layer of the policy network, we mask the invalid actions, which choose next-hop AS not existing in routing path table, by setting its probability to 0 in the probability distribution. Then we re-scale the probabilities on all actions such that the sum still equals to 1 [3].

Reward. The whole system aims to maximize the sum of all flows' throughput in the traffic matrix. However, each agent is selfish, aiming to maximize its own reward. Our system works in a time-slotted fashion. In each interval t, an agent performs multiple inferences over its RL model to produce routing decisions for current by-passing flows. The reward r_t observed in an interval t is the average throughput of all concurrent flows traversing agent i in t, which are routed using actions produced by the inferences:

$$r_t = \frac{\sum_j v_j}{n_t}$$

where n_t represents the number of concurrent traversing flows at the agent in t and v_j is the throughput for each.

C. Model Training

Individual policy NN on each agent is trained by updating the NN parameters θ using policy gradients computed with samples. In each interval, we have one sample correponding to each flow routing inference, in the form of $< s, a, r, s' >$: a is the routing decision produced for the respective flow; s and s' are the input states before and after the routing action for the flow is taken; r is the reward computed in the current interval.

The goal of training policy NN is to maximize the expected cumulative discounted reward $J(\theta) = E[\sum_{t \geq 0} \gamma^t r_t]$, where $\gamma \in [0, 1]$ is the discount factor. The policy gradient used for NN update can be calculated as:

$$\nabla_\theta J(\theta) = E_{\pi_\theta}[\sum_t \nabla_\theta log(\pi_\theta(a_t|s_t)) Q^{\pi_\theta}(s_t, a_t)]$$

where t represents number of inferences done, the Q value indicates the 'quality' of action a taken in given state s following the policy π_θ, calculated as expected cumulative discounted reward to obtain after selecting a under s following π_θ.

However, high variance in Q values prevents convergence of the policy model. We adopt the *actor-critic* [14] algorithm, which introduces an advantage function, i.e., $Q^{\pi_\theta}(s, a) - V^{\pi_\theta}(s)$, where $V^{\pi_\theta}(s)$ is calculated as the expected cumulative reward following the policy π_θ from state s, over all possible actions in the state. Then this advantage function is used for policy gradient calculation instead of $Q^{\pi_\theta}(s, a)$.

Specifically, the *actor* is a policy network. Input state s is connected to a fully connected layer, and then to another fully connected layer before the output layer. The output of the policy network is a probability distribution over action space. The value function $V^{\pi_\theta}(s)$ is estimated by a value network (the *critic*) as the output, which has the same input and architecture as the policy network, except that the output layer is a linear neuron without any activation function. The value network is trained using the temporal difference method [14]. Fig. 2 gives an overview of our DRL architecture at each agent, in the case of using local state as input only.

We adopt a number of techniques to ensure exploration and expedite training convergence. We use an $\epsilon - greedy$ approach [14] to adequately explore the action space; or the system may converge to a poor local optimum. In each inference, an agent has probability ϵ to randomly choose a valid action in the action space, and probability $1 - \epsilon$ to choose the action produced by the current policy π_θ. In addition, we adopt experience replay to alleviate correlation in the samples sequence [14], to avoid the following situation: high reward after running one time interval will lead an agent to adopting actions following the same policy, which prevents it from exploring samples with higher rewards.

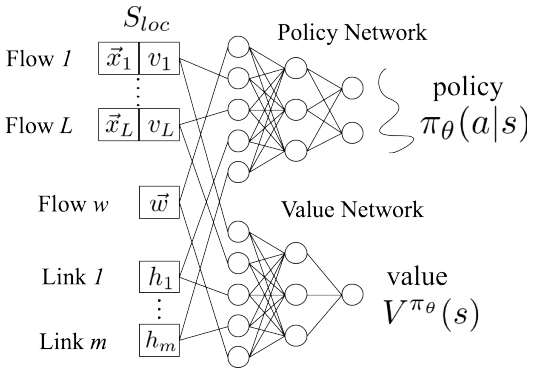

Fig. 2: DRL architecture at each agent. S_{loc} represents local state. Flow w is the current flow whose routing decision is to be inferenced.

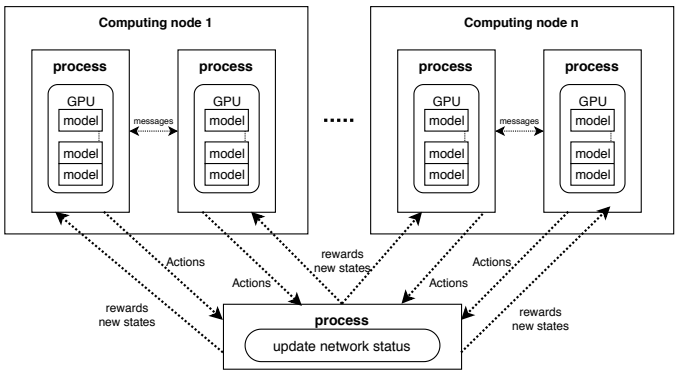

Fig. 3: Testbed Implementation

We perform decentralized, synchronized offline training on each agent, with multiple traffic matrices traced from reality. In each training epoch, the traffic matrices are divided into small batches and used for training one by one. We use the trained policy network for online routing decision making at the respective agent.

IV. Performance Evaluation

A. Experimental setup

Testbed Implementation. In reality, our proposed RL model is run at each AS in the Internet in a fully distributed manner, and each AS is equipped with at least one computing node to train and execute routing policy. To evaluate performance of our MARL based routing, we build a testbed with 20 1080Ti GPUs, with 2-4 GPUs on one computing node. Each computing node has one 8-core Intel Silver 4108 CPU@1.8GHz, 48GB RAM, and one 1Gbps NIC. We emulate multiple agents (ASes) by training and executing their RL models in parallel on the same GPU.

An overview of our implementation is shown in Fig. 3, where each model corresponds to one agent. In each computing node, we run one process per GPU, each responsible for training and executing multiple models. Messages (e.g., actions, observations) are transferred between processes using Message Queues when necessary (e.g., encoding neighboring or global state), to obtain input state s to each model. After receiving input states, agents produce actions in parallel. In addition, we design one special process in a particular computing node to be responsible for: collection of actions taken by all agents; status update of network environment; computation of rewards for all agents. Then, messages (new states s', rewards r) will be distributed to corresponding agents.

DRL Training. We implement NN training using libraries provided on Tensorflow. Each model's policy and value networks have one hidden layer with H neurons. We decide H as the square root of the product of input dimension and output dimension [15]. The rationale is that with too few neurons, we may not estimate an appropriate routing strategy while too many can easily lead to over-fitting and unnecessary feature acquisition. We choose *Leaky_ReLU* as activation function for the hidden layer, set normal distribution for kernel with $mean = 0, stddev = 0.05$, and set $bias$ with constant value 0.1. *Softmax* is used as the activation function in output layer for the policy network. Besides, we adopt Adam optimizer with initial learning rate 10^{-5} and discount factor $\gamma = 0.9$ for the reward. The exploration factor ϵ is set to 0.3 at the beginning and will decrease as the training proceeds.

B. Evaluation on PhEDEx traces

Flow-based Routing. We first do flow-level routing, i.e., perform model inference at each agent for each by-passing flow based on its source and destination. We evaluate it on traces from PhEDEx.

PhEDEx, Physics Experiment Data Export [16], is a project to manage the movement of data within CMS (Centers for Medicare and Medicaid Services). We use a topology with 131 nodes (including customer-provider, peer-to-peer relationships) and 200 traffic matrices (each containing 350 to 400 end-to-end flows) from PhEDEx. We split the traffic matrices into training dataset (90%, batch size is 16 traffic matrices) and testing dataset (10%). Capacity of each link is set to 100, and the flows have the same traffic demand of 1.

We compare the performance of our approach under different input states at each agent: local state only, one-hop neighbor state, two-hop neighbor state, three-hop neighbor state and global state (corresponding to cases (1)-(5) in Sec. III-B). We also compare with a random method by which each agent randomly chooses a valid next-hop for each flow. Fig. 4(a) shows the training curves, where the system throughput is averaged over those of all traffic matrices in each epoch. Intuitively, higher system throughput can be achieved when each agent has more information.

In Table I, we evaluate trained models using 20 testing traffic matrices, and present percentages of improvement as compared to the random method and the upper bound of optimal system throughput (computed by formulating the flow routing problem into a mixed integer problem as follows and solving it with MatLab-Yalmip).

$$\max \sum_{f \in F} v_f$$

subject to:

$$\sum_{j:(i,j)\in E} x_{ij}^f - \sum_{i:(j,i)\in E} x_{ji}^f = \begin{cases} 0, & \text{for } i = N^f \\ -1, & \text{for } i = t^f \\ 1, & \text{for } i = s^f \end{cases} \quad (1)$$

$$x_{ij}^f \in \{0,1\}, \forall (i,j) \in E, f \in F \quad (2)$$

$$0 \leq \sum_{f\in F} v_f x_{ij}^f \leq C_{ij}, \forall (i,j) \in E \quad (3)$$

$$v_f \geq \min_{(i,j)\in E: x_{ij}^f = 1} \left\{ \frac{C_{ij}}{\sum_{f'\in F} x_{ij}^{f'}} \right\}, \forall f \in F \quad (4)$$

where v_f is the throughput of flow f. Constraints 1 and 2 ensure flow conservation on each node, where s^f and t^f indicate source and destination of flow f, respectively. $N^f = N - t^f - s^f$. x_{ij}^f is a binary value denoting existence of flow f on edge (i,j), i.e., $x_{ij}^f = 1$ if flow f traverses edge (i,j), and $x_{ij}^f = 0$, otherwise. Constraint 3 ensures that the sum of throughput of flows passing through an edge can never exceed its capacity. Constraint 4 guarantees fairness among flows. Using TCP, flows traversing a link roughly share the link bandwidth.

We see that even with only local information (i.e., information is not shared among ASes), our system can quite closely approach the upper bound (more than 86%). Further, information sharing is helpful to achieve better performance. The rationale is that, encoding other agents' observations helps an agent capture much wider network status and encoding actions of others ensures environment stability at individual agents. Though equipping agents with global observation is unrealistic in the Internet environment, agents with 3-hop neighboring information are also able to achieve more than 90% of the upper-bound performance.

In practice, we can implement flow-based routing by taking advantage of the BGP AS SET to include union of paths for each flow during routes advertisement. For example, when an AS X receives two flows $F1$ and $F2$ with the same destination AS Z, by default it will select one next hop AS for both F_1 and F_2; if AS X adopts a flow-based RL model for routing, in the advertisement phase, it can select next hop AS A_1 for $F1$ and next hop AS A_2 for $F2$, and advertise the union of paths for $F1$ and $F2$ to its neighbors (i.e., $F1 \rightarrow Z$ and $F2 \rightarrow Z$).

Destination-based Routing. We next evaluate our approach following two important properties of default BGP routing:

(1) *Destination-based* routing. Each AS chooses the next-hop for each flow only based on the flow destination, but not the source.

(2) *Shortest AS path* based routing. When an AS has several candidate next-hops, BGP aims to choose the one with shortest length of AS-path to indicated destination.

To compare with default BGP routing (*destination-based* and *shortest path*), we adapt our model to produce destination-based routing decisions by modifying $\vec{w}$ in input state to our RL model at each agent to $\vec{w}'$, a value indicating destination of the flow to be routed only. In this way, for flows with different sources but the same destination, the same routing decision will be produced.

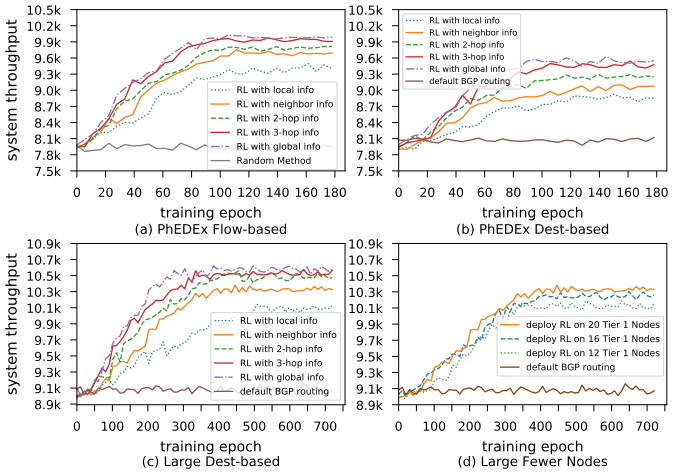

Fig. 4: Training curves under different input states and network topologies.

Fig. 4(b) shows the training convergence curves based on the same traces as used in flow-based routing. Table II gives the average improvement percentages as compared to default BGP, tested with 20 test traffic matrices using trained models. We observe that without control at the flow level, destination-based routing still brings improvement of up to 17%.

TABLE I: Performance of Flow-based MARL

Input States	Improvement as compared to random	Upper bound achieved
local info	20.1%	86.3%
1-hop neighbor info	22.5%	88.1%
2-hop neighbor info	24.1%	89.2%
3-hop neighbor info	25.8%	90.5%
global info	26.3%	90.8%

TABLE II: Performance of Destination-based MARL

Input States	Improvement as compared to default BGP
local info	10.2%
1-hop neighbor info	12.6%
2-hop neighbor info	14.6%
3-hop neighbor info	16.4%
global neighbor info	17.1%

C. Evaluation on Large Topology

We next investigate scalability of our approach. We build a topology containing 10,000 nodes (with customer-provider and peer-peer relationships between neighboring nodes). The topology is a hierarchical structure, with 20, 500, 3000 and 6480 nodes in four tiers, respectively. Nodes in a higher tier have larger degrees than nodes in a lower tier. We allow each AS to have multiple providers and path diversity to avoid congestion.

We use the gravity model [17] to produce 1000 traffic matrices for training and 100 traffic matrices for testing, which emulate realistic Internet traffic. Each traffic matrix contains about 20000 end-to-end flows, and the maximum number of concurrent flows traversing a tier-1 node is up to 1000. The capacity of each link is 100, and the flows have the same traffic demand of 1.

We use *destination-based* MARL and compare with default BGP routing. We only equip 20 tier-1 nodes with our RL approaches, and other nodes in the topology use the default BGP routing policy to forward traversing traffic flows. The reason is that nodes in tier-1 undertake most of the traffic management work; an efficient routing policy adopted by these nodes may lead to significant improvement of system throughput.

Fig. 4(c) shows the training convergence curves of MARL with 1000 training traffic matrices on the large topology. We evaluate the trained models using 100 testing traffic matrices, and show the average improvement of our MARL approaches with default BGP routing in Table III. With only 20 tier-1 nodes using our RL-based routing policies, the improvement in the overall system throughput can still be 16%, which exhibits good scalability and practicality of our approach: in real-world large networks, we only need to deploy RL on a small set of hub ASes, in order to achieve good improvement of global throughput.

Further, we evaluate impact of the number of nodes employing our RL-based routing in the entire network. The results are shown in Fig. 4(d) (training curves) and Table IV (testing results), where information sharing among the RL nodes is restricted to one hop. With fewer RL nodes in the network, the improvement is only slightly less, again showing good adoptability of our approach in practice.

TABLE III: Performance of Destination-based MARL on Tier-1 ASes only

Input States	Improvement as compared to default BGP
local info	11.6%
1-hop neighbor info	14.2%
2-hop neighbor info	15.8%
3-hop neighbor info	16.5%
global info	16.9%

TABLE IV: Performance of Destination-based MARL with Different Numbers of RL Agents

Number of RL Agents	Improvement as compared to default BGP
20 RL Nodes	14.2%
16 RL Nodes	13.3%
12 RL Nodes	11.9%

V. Conclusion and Future Work

This paper presents an efficient and scalable MARL framework for inter-domain routing, to achieve high system throughput for real-time traffic demand. It's particularly useful for federation of collaborative networks running large distributed analytics and frequently transferring large amounts of data between sites. Experimental evaluation based on realistic network topologies shows improvements of system throughput by up to 17%, as compared to default BGP routing.

For future work, we identify further efforts needed to improve our solution, as follows: i) we may explore GNN techniques to learn embeddings of the network, for better routing decisions; ii) we will compare our MARL framework with more inter-domain routing optimization approaches; and iii) we will explore benefits of our design with more network topologies and traffic patterns.

References

[1] A. O. Kazakci *et al.*, "Data science as a new frontier for design," in *DS 80-10 Proceedings of the 20th International Conference on Engineering Design (ICED 15) Vol 10: Design Information and Knowledge Management Milan, Italy, 27-30.07. 15*, 2015, pp. 189–198.

[2] Y. Rekhter, S. Hares, and T. Li, "A Border Gateway Protocol 4 (BGP-4)," RFC 4271, Jan. 2006. [Online]. Available: https://rfc-editor.org/rfc/rfc4271.txt

[3] Y. Bao, Y. Peng, and C. Wu, "Deep learning-based job placement in distributed machine learning clusters," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*. IEEE, 2019, pp. 505–513.

[4] P. B. Godfrey, M. Caesar, I. Haken, Y. Singer, S. Shenker, and I. Stoica, "Stabilizing route selection in bgp," *IEEE/ACM Transactions on Networking*, vol. 23, no. 1, pp. 282–299, 2014.

[5] E. Alabdulkreem, H. S. Al-Raweshidy, and M. F. Abbod, "Mrai optimization for bgp convergence time reduction without increasing the number of advertisement messages," *Procedia Computer Science*, vol. 62, pp. 419–426, 2015.

[6] Z. Chen, J. Bi, Y. Fu, Y. Wang, and A. Xu, "Mlv: A multi-dimension routing information exchange mechanism for inter-domain sdn," in *2015 IEEE 23rd International Conference on Network Protocols (ICNP)*. IEEE, 2015, pp. 438–445.

[7] K.-K. Yap, M. Motiwala, J. Rahe, S. Padgett, M. Holliman, G. Baldus, M. Hines, T. Kim, A. Narayanan, A. Jain *et al.*, "Taking the edge off with espresso: Scale, reliability and programmability for global internet peering," in *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*. ACM, 2017, pp. 432–445.

[8] B. Schlinker, H. Kim, T. Cui, E. Katz-Bassett, H. V. Madhyastha, I. Cunha, J. Quinn, S. Hasan, P. Lapukhov, and H. Zeng, "Engineering egress with edge fabric: Steering oceans of content to the world," in *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*. ACM, 2017, pp. 418–431.

[9] F. Ruffy, M. Przystupa, and I. Beschastnikh, "Iroko: A framework to prototype reinforcement learning for data center traffic control," *arXiv preprint arXiv:1812.09975*, 2018.

[10] Y. Zuo, Y. Wu, G. Min, and L. Cui, "Learning-based network path planning for traffic engineering," *Future Generation Computer Systems*, vol. 92, pp. 59–67, 2019.

[11] D. Ye, M. Zhang, and Y. Yang, "A multi-agent framework for packet routing in wireless sensor networks," *sensors*, vol. 15, no. 5, pp. 10 026–10 047, 2015.

[12] B. Pourpeighambar, M. Dehghan, and M. Sabaei, "Multi-agent learning based routing for delay minimization in cognitive radio networks," *Journal of Network and Computer Applications*, vol. 84, pp. 82–92, 2017.

[13] H. Mao, Z. Gong, Y. Ni, and Z. Xiao, "Accnet: Actor-coordinator-critic net for" learning-to-communicate" with deep multi-agent reinforcement learning," *arXiv preprint arXiv:1706.03235*, 2017.

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

[15] D. M. Bourg and G. Seemann, *AI for game developers*. " O'Reilly Media, Inc.", 2004.

[16] J. Rehn, T. Barrass, D. Bonacorsi, J. Hernandez, I. Semeniouk, L. Tuura, and Y. Wu, "Phedex high-throughput data transfer management system," in *Computing in High Energy and Nuclear Physics (CHEP)*, vol. 2006, 2006.

[17] M. Roughan, "Simplifying the synthesis of internet traffic matrices," *ACM SIGCOMM Computer Communication Review*, vol. 35, no. 5, pp. 93–96, 2005.