Multiagent Meta-Reinforcement Learning for Adaptive Multipath Routing Optimization

Long Chen, *Member, IEEE*, Bin Hu, *Member, IEEE*, Zhi-Hong Guan, *Senior Member, IEEE*, Lian Zhao, *Senior Member, IEEE*, and Xuemin Shen, *Fellow, IEEE*

***Abstract*— In this article, we investigate the routing problem of packet networks through multiagent reinforcement learning (RL), which is a very challenging topic in distributed and autonomous networked systems. In specific, the routing problem is modeled as a networked multiagent partially observable Markov decision process (MDP). Since the MDP of a network node is not only affected by its neighboring nodes' policies but also the network traffic demand, it becomes a multitask learning problem. Inspired by recent success of RL and metalearning, we propose two novel model-free multiagent RL algorithms, named multiagent proximal policy optimization (MAPPO) and multiagent metaproximal policy optimization (meta-MAPPO), to optimize the network performances under fixed and time-varying traffic demand, respectively. A practicable distributed implementation framework is designed based on the separability of exploration and exploitation in training MAPPO. Compared with the existing routing optimization policies, our simulation results demonstrate the excellent performances of the proposed algorithms.**

***Index Terms*— Adaptive routing, metapolicy gradient, multiagent, reinforcement learning (RL).**

I. Introduction

As a fundamental function of paramount importance in networked systems, routing is the problem of selecting paths to send data packets or goods from sources to destinations while meeting the quality-of-service (QoS) requirements under various system constraints. It has applications in a wide range of fields, such as goods transportation in logistics systems [1], power transmission in smart grids [2], traffic scheduling in intelligent transportation systems [3], and especially information exchange in communication networks [4]–[7]. Due to its intrinsic difficulty and the great benefits of optimal routing, routing has become a challenging issue attracting great research attention in recent years.

The performance metrics of routing strategy, such as end-to-end delay, throughput, packet delivery rate, link utilization ratio, and energy consumption, depend on network topology, nodes' processing capabilities, traffic pattern, and so on [8]–[10]. To analyze and improve routing protocols, a networked system is often modeled as a weighted graph, where weights are related to node and link properties, including but not limited to latency, capacity, reliability, and energy. The shortest path routing (SPR) is widely used due to its simplicity and effectiveness. However, finding the minimum cost paths that satisfy multiple constraints in a graph is NP-complete [9].

Many heuristic routing policies have been proposed to provide suboptimal solutions in both wired and wireless networks. When deploying the SPR protocol in practical IP networks, desirable performance can be observed with few link weight changes in scenarios of time-varying traffic demand matrices [10]. In large-scale complex networks, routing algorithms are proposed to balance traffic by minimizing the maximum generalized node betweenness centrality [11]. To increase throughput and end-to-end reliability, multipath approaches, such as equal-cost multipath [12] and optimization of link weights (OLW) [13], are adopted in multihop networks, where the source delivers the packets to their destinations via multiple paths, thereby supporting a higher transmission rate than what is possible with only one path. A heuristic approach, called generalized destination-based multipath routing (GDMR), can achieve the same high performance as explicit routing [14]. However, multipath routing also leads to some problems, such as energy balance and multipath cooperation. In addition, obtaining an accurate view of traffic demand and statistical characteristics of the communication channels is a nontrivial task in practice, as most networks lack the appropriate measurement infrastructure [15].

Reinforcement learning (RL), a bioinspired machine learning technique where an agent seeks an optimal behavior policy through repeated trial and error interactions with a dynamic environment [16], can provide a model-free solution for routing problems. Q-routing proposed in [17] is the first autonomous and distributed packet routing scheme based on Q-learning. With consideration of Q-value freshness, the predictive Q-routing [18] keeps the best Q-values and reuses them to predict the traffic trend. Based on collaborative RL, the mobile ad hoc network routing protocol proposed in [19] uses feedback in the selection of next-hop links to adapt

Manuscript received 11 December 2019; revised 17 August 2020 and 5 January 2021; accepted 25 March 2021. Date of publication 21 April 2021; date of current version 6 October 2022. This work was supported in part by the National Natural Science Foundation of China under Grant 61976100, Grant 61976099, Grant 61772086, Grant 61633011, and Grant 61873287. *(Corresponding author: Bin Hu.)*

Long Chen, Bin Hu, and Zhi-Hong Guan are with the School of Artificial Intelligence and Automation, Huazhong University of Science and Technology, Wuhan 430074, China (e-mail: chenclong@yeah.net; bhu@hust.edu.cn; zhguan@mail.hust.edu.cn).

Lian Zhao is with the Department of Electrical, Computer and Biomedical Engineering, Ryerson University, Toronto, ON M5B 2K3, Canada (e-mail: l5zhao@ryerson.ca).

Xuemin Shen is with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON N2L 3G1, Canada (e-mail: sshen@uwaterloo.ca).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TNNLS.2021.3070584.

Digital Object Identifier 10.1109/TNNLS.2021.3070584

its routing behavior, resulting in optimization of network throughput. As a model-based Q-learning approach, QELAR aims at maximizing the residual energy of the network nodes in underwater wireless broadcast channel environments [20]. By adding a penalty for dropping packets into the cost function, the channel-aware RL-based multipath adaptive routing (CARMA) [21] is proposed for optimizing route-long energy consumption. A self-learning routing protocol based on RL is designed for flying ad hoc networks in a complicated flight environment [22]. More RL-based routing algorithms can be found in [23]. Due to the well-known "curse of dimensionality" [16], the complexity of training RL with large state–action space prohibits a comprehensive representation of dynamic network states, thus limiting the potential benefit of RL-based packet routing policies.

Inspired by the recent success and thorough research on hybrid neural networks [24]–[30], the deep learning (DL) has been introduced in RL, named deep reinforcement learning (DRL), to represent large state–action space and execute automatic feature extraction in high-dimensional space. With deep neural network (DNN) as a powerful approximator of value function [31], DRL-based routing can achieve more efficient exploration and end-to-end decision-making. The Q-function of Q-routing is approximated via a neural network instead of a table, allowing agents to consider arbitrary information related to routing efficiency [32]. Deep Q-routing with communication (DQRC) proposed in [33] uses an independent long short-term memory (LSTM) recurrent neural network, which extracts routing features from high-dimensional information regarding backlogged packets and past actions, to approximate the Q-function effectively. Utilizing the router selection Markov decision process (MDP) to formulate the routing problem, two novel deep Q network (DQN)-based centralized online algorithms are designed to reduce the network congestion probability with short transmission paths [34]. In order to automatically adjust the weights of links dynamically, a deep deterministic policy gradient (DDPG) algorithm, a recently proposed efficient DRL algorithm, is applied in the software-defined networking (SDN), providing better routing configurations to minimize the network delay than traditional traffic engineering based on handcrafted feature selection [35]. Another algorithm [36] considers a similar network model while taking minimizing link utilization as the optimization target.

In this article, we consider optimal multipath routing schemes for distributed multihop networks. A motivating example is the multihop wireless sensor network (WSN) shown in Fig. 1. Several heterogeneous smart devices are scattered throughout an area of interest to observe a physical phenomenon, such as tracking and monitoring wild animals. The sensor nodes closer to wild animals are the current information sources. Within each sampling period, the real-time monitoring information generated at the sources is forwarded to the local edge servers (sinks) via other relay nodes (routers), where the edge servers have more powerful storage and computing capabilities and may be connected to a remote monitoring center and cloud servers. As the energy consumption of sending a data packet is much higher than that of receiving it, only constraints on the forwarding ability of nodes are considered. Since the behaviors of wild animals are uncertain and the smart nodes are heterogeneous and with limited capabilities, an adaptive multipath routing is needed to meet the required QoS. Similar routing problems also appear in the space and air segments of software-defined space–air–ground integrated vehicular networks [7].

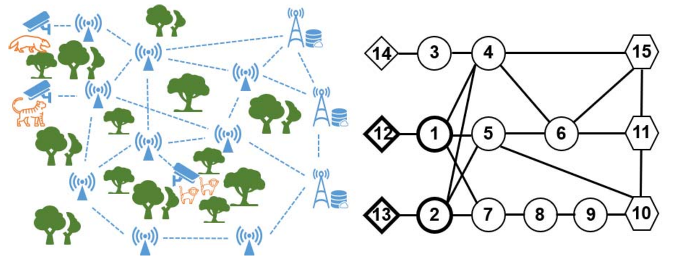


Fig. 1. WSN scenario and its network topology containing 15 nodes and 21 bidirectional links, where source nodes (diamond) $\mathcal{N}_s = \{n_{14}, n_{12}, n_{13}\}$ and sink nodes (hexagon) $\mathcal{N}_d = \{n_{15}, n_{11}, n_{10}\}$.

We leverage networked multiagent partially observable MDP (POMDP) to formulate the packet routing problem in a multihop network. Based on multiagent RL (MARL) and metalearning, two novel RL-based routing algorithms are designed to optimize network performance and resource utilization for uncertain and dynamic network environments. The twofold contributions of this article include the following.

1) In order to improve the communication performance of heterogeneous multihop networks with limited forwarding capabilities and energy supplies, we propose a novel multipath routing scheme based on an MARL algorithm, termed multiagent proximal policy optimization (MAPPO). Also, a practicable distributed implementation framework of MAPPO-based routing schemes is given. With a carefully designed reward function, the routing protocol can be energy-efficient and achieve a longer network lifetime in energy-limited noisy environments without sacrificing the packet delivery time.
2) By modeling a dynamic environment as a multitask learning problem, the multiagent metaproximal policy optimization (meta-MAPPO) is designed based on meta-reinforcement learning (meta-RL). Using the Kullback–Leibler divergence (KL-divergence) penalty to obtain a more robust metapolicy, the adaptive routing with meta-MAPPO can handle the dynamic traffic demand more effectively.

The remainder of this article is organized as follows. Section II reviews the background knowledge of RL and its applications in packet routing. The problem definitions of routing are described in Section III. Section IV and V present our design of MAPPO and meta-MAPPO routing schemes, respectively. The performance of the proposed algorithms is evaluated in Section VI with comparison to SPR, Q-routing, QELAR, and DQRC. Section VII concludes this work.

II. Background and Preliminary

In this section, we briefly review RL and its applications in routing problems and then put forward the necessity of distributed routing policies for real-world network scenarios.

A. Reinforcement Learning

RL is a trial and error learning process to solve sequential decision-making problems [16]. At each time step t, the agent receives a state s_t in a state space $\mathcal{S}$, and executes an action a_t from an action space $\mathcal{A}$, following a behavior policy π. Then, the agent receives a scalar reward r_t and transitions to the next state s_{t+1} according to the environment dynamics P. When an RL problem satisfies the Markov property, i.e., the future depends only on the current state and action, it can be formulated as an MDP, defined by a five-tuple $(\mathcal{S}, \mathcal{A}, P, R, \gamma)$, where $P(s_{t+1}|s_t, a_t)$ is the transition probability function, $R(s, a)$ is the reward function, and γ is the discount factor. Denoting the discounted cumulative reward by $G_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k}$, the goal of RL is finding an optimal policy $\pi(a|s)$ to maximize the expectation of G_0. As a fundamental concept in RL, a value function measures how good each state or state–action pair is. The state value function is defined as $V^\pi(s) = \mathbb{E}[G_t|s_t = s]$, and the state–action value function is $Q^\pi(s, a) = \mathbb{E}[G_t|s_t = s, a_t = a]$.

Because a ground-truth model of the environment is usually not available to the agent, model-free methods have been prevailing in RL in the past several years. There are two main approaches to representing and training agents with model-free RL, i.e., Q-learning and policy gradient [37]. Q-learning methods learn an approximator $Q_\phi(s, a)$ with parameter ϕ for the optimal action-value function $Q^*(s, a) = \max_\pi Q^\pi(s, a)$. Instead, policy gradient methods represent a policy explicitly as $\pi_\theta(a|s)$ with parameter θ. They optimize the parameter θ directly by gradient ascent on the performance objective $J(\theta) = \sum_{s\in\mathcal{S}} d^\pi(s)V^\pi(s)$ where $d^\pi(s)$ is stationary distribution of Markov chain $\{s_t\}_{t\geq 0}$ for π_θ. Besides, some policy gradient methods also learn an approximator $V_\phi(s)$ with parameter ϕ for the on-policy value function $V^\pi(s)$, which is used to update θ. More RL methods can be found in [37].

As a family of first-order policy gradient methods, proximal policy optimization (PPO) [38] bounds parameter updates to a trust region in each iteration to ensure stability and reliability. PPO achieves better performance on lots of continuous and discrete tasks and has better empirical sample complexity than the trust region policy optimization (TRPO). There are two primary variants of PPO: PPO-Penalty and PPO-Clip. PPO-Clip relies on specialized clipping in the objective function, whereas PPO-Penalty penalizes the KL-divergence. Due to its high performance and low computational complexity, PPO has become the default RL algorithm at OpenAI.

MARL focuses on models where several agents learn by dynamically interacting with a common environment to achieve a given global goal, while in single-agent RL scenarios the state of the environment changes solely as a result of the action of one agent, in MARL scenarios, the environment is subjected to the actions of all agents. In contrast to the problems involving centralized control in single-agent RL, decentralized control of MDP in MARL provably does not admit polynomial-time algorithms [39]. Due to the combinatorial nature, most MARL tasks are categorized as NP-hard problems [40]. Independently Q-learning and policy gradient methods such as PPO perform poorly in MARL [41], [46]. Nonstationarity of the environment is the main challenge in MARL problems. A newly emerging approach to address this issue is using a fully observable critic, which involves the observations and actions of all agents, making the environment stationary even though the policies of other agents change. As shown by multiagent DDPG (MADDPG) [41], the critic of each agent deals with a stationary environment in the training time to guarantee its convergence, and the distributed behavior policy only needs to access the local information in the inference time.

Another emerging research topic is meta-RL, which is to do metalearning in the field of RL. Meta-RL considers a distribution of different yet related tasks that differ in, for instance, the reward function or the transition probabilities from one state to another. Its objective is to enable agents to efficiently adapt to new tasks by learning from prior tasks (also known as learning to learn) [42]–[44]. Some optimization-based meta-RL algorithms, such as MAML [44], FOMAML, and Reptile [45], are proposed to formulate meta-RL as a bilevel optimization procedure. We denote the metaparameter of a metapolicy as $\boldsymbol{\mu}_{\text{meta}}$, which can quickly adapt to new tasks by performing a few gradient updates. The inner optimization obtains new policy $\boldsymbol{\mu}' = \mathcal{A}\text{lg}(\boldsymbol{\mu}, \omega)$ by adaptation to a given task ω based on policy $\boldsymbol{\mu}$, where $\mathcal{A}$lg is an RL algorithm. The outer aims to update $\boldsymbol{\mu}_{\text{meta}}$ based on the inner procedure. Unlike vanilla joint-training methods where $\boldsymbol{\mu}_{\text{meta}} \leftarrow \boldsymbol{\mu}'$, Reptile [45] updates the parameter by

$$\boldsymbol{\mu}_{\text{meta}} \leftarrow \boldsymbol{\mu}_{\text{meta}} + \xi(\boldsymbol{\mu}' - \boldsymbol{\mu}_{\text{meta}}) \tag{1}$$

where ξ is a learning rate. Due to taking advantage of metagradient information from the item $(\boldsymbol{\mu}' - \boldsymbol{\mu}_{\text{meta}})$, Reptile has better generalization performance for new tasks.

B. Q-Routing Approach and Its Extensions

Q-routing [17] is a variant of Q-learning [16]. As each node is viewed as an independent agent, the Q-routing is essentially a multiagent learning method. For node n_i, the state s_t^i represents the destination node n_d of the head-of-line (HOL) packet in its queue, and the action a_t^i represents the selected next-hop neighboring node n_j to relay the HOL packet to node n_d. The reward function is defined as $r_t^i = -(g_t^i + g_t^{ij})$, where g_t^i is the HOL packet's queuing delay at node n_i and g_t^{ij} is the transmission delay between nodes n_i and n_j. The Q-value $Q_i(n_d, n_j)$, stored in node n_i's Q-table, estimates the end-to-end delay for transmitting packets along node n_j to the destination n_d. When n_i has a packet to send, it selects a neighbor n_j with the largest Q-value. Upon sending the packet to node n_j, it receives a reward r_t^i and updates its Q-table as follows:

$$Q_i(n_d,n_j) = (1-\alpha)Q_i(n_d,n_j) + \alpha\Big[r_t^i + \max_k Q_j(n_d,n_k)\Big] \tag{2}$$

where $\alpha \in (0, 1]$ is a learning rate and n_k is a neighbor of n_j.

To prolong the lifetime of networks by making the residual energy of nodes more evenly distributed, QELAR [20] adds the residual energy e^i of each node as well as the average energy $\bar{e}^i$ among a group of its neighbor nodes into the reward function

$$r_t^i = -g - \alpha_1\left(e_t^i + e_t^j\right) + \alpha_2\left(\bar{e}_t^i + \bar{e}_t^j\right) \tag{3}$$

where g is the constant punishment representing the energy consumption and delay when forwarding a packet from current node n_i to next-hop n_j and α_1 and α_2 are the weight coefficients. CARMA [21] additionally introduces a penalty for dropping packets in the reward function, providing sufficient QoS of packet delivery ratio. With information sharing mechanism, DQRC [33] uses recurrent neural networks, which take more history and state information as inputs, to make a more accurate estimation for the transmission delay.

As the end-to-end delay and network lifetime are the performance indicators of main concern in this article, Q-routing, QELAR, and DQRC are chosen as baselines for comparison due to their preferable performances in ideal and noisy network environments.

C. Centralized and Distributed Routing Based on RL

The centralized routing schemes in [34]–[36] assume that there is a centralized controller that can collect all necessary information for the real-time routing decision-making, such as the current remaining buffer size of each node and the accurate traffic demand matrix. Thus, these centralized routing algorithms are hard to be applied in a realistic network. In contrast, Q-routing and its variants are typical fully distributed routing schemes. By sharing the Q values among adjacent nodes, Q-routing can be characterized as an asynchronous and online version of the Bellman–Ford shortest paths algorithm that measures path length not merely by the number of hops but rather by total delivery time [17]. However, Q-routing still suffers from Q value freshness, slow convergence, and parameter setting sensitivity [23]. With help of more information and independent LSTM networks, DQRC obtains a shorter average delivery time and improves the convergence, especially at heavy traffic regimes. Nonetheless, each node needs to send the sampled training batch to its neighbors and get the current Q value back to recalculate the remaining delivery time for updating parameters of its neural network, resulting in that the communication overhead is increased in the training process.

Since sampling data and optimizing policies are separable in training PPO, we proposed a novel practicable implementation framework of routing schemes with centralized learning in multhop networks. In the exploration (sampling data) phase, each router uses only local information for decision-making (decentralized execution) and encapsulates the current decision information in the HOL packet, which is transmitted to next-hop node immediately. In the exploitation (optimizing policies) phase, a virtual central node collects all the decision information from sink nodes and calculates each node's new policy parameters, which will be dispatched over the network to each node. In addition, PPO is chosen for two other reasons. First, the multipath routing allows nodes to forward packets of the same destination to different neighbors, which means that different actions can be taken for the same state. Obviously, probabilistic policies, such as TRPO and PPO, are naturally better suited to represent multipath routing schemes than deterministic policies used in DDPG, twin-delayed DDPG, and MADDPG. Second, in the process of training PPO, only $V(s)$ needs to be estimated. However, DDPG and soft actor–critic need to fit $Q(s, a)$, where the curse of dimensionality is more severe as the inputs include both the state and action of all nodes.

III. Problem Formulation and Mathematical Modeling

In this section, we establish the mathematical model of the routing problem on multihop networks and introduce the formulation of MARL with networked agents for packet routing.

A. Mathematical Model of Packet Routing

We consider a distributed multihop network made up of N nodes, where data packets generated at the ingress nodes (e.g., sensors) are delivered to the egress nodes (e.g., sinks) for further processing. An example scenario and its network topology are shown in Fig. 1, containing 15 nodes and 21 bidirectional links. The mathematical model of the routing problem is described as follows.

1) The network is modeled as a graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$, where $\mathcal{N}$ is defined as the finite set of nodes and $\mathcal{E}$ is the set of links between node pairs. Denoting the cardinality of $\mathcal{N}$ by $|\mathcal{N}|$, we have $N = |\mathcal{N}|$.
2) At the beginning of each sampling period T_s (e.g., 6 s), new data packets are generated at ingress nodes and needed to be forwarded to egress nodes. We denote the set of ingress and egress nodes by $\mathcal{N}_s$ and $\mathcal{N}_d$, respectively.
3) To describe the traffic demand, the traffic matrix (TM, denoted by η) is introduced [47]. η is a nonnegative $|\mathcal{N}_s| \times |\mathcal{N}_d|$ matrix whose (i, j)th entry $\eta_{i,j}$ represents the traffic volume between a source $n_s \in \mathcal{N}_s$ and a destination $n_d \in \mathcal{N}_d$ $(n_s \neq n_d)$ in one sampling period. In addition, let $\kappa = \sum_{i,j} \eta_{i,j}$ be the total traffic demand between sources and destinations.
4) Each node is set up with a dedicated packet buffer queue with a capacity much greater than κ, using a first-in, first-out (FIFO) discipline. The timeline in one sampling period is divided into lots of transmission time steps (e.g., radio frames with 10-ms duration each in LTE). Limited by the channel capacity and energy consumption for transmitting data, it is assumed that node n_i can send at most c_i data packets in one transmission time step.
5) Further assuming that the processing delay is negligible, only the queuing delay and transmission delay are considered.
6) We denote by ζ the delivery time it takes to send all newly generated packets to their destinations. We assume that at the end of each sampling period, all undelivered packets will be discarded, which means $\zeta \leq T_s$.

The network topology shown in Fig. 1 is the same as in [18]. We assume that $\mathcal{N}_s = \{n_{14}, n_{12}, n_{13}\}$ and $\mathcal{N}_d = \{n_{15}, n_{11}, n_{10}\}$. To make the problems nontrivial, the forwarding capacity c_i of node n_i is given by

$$c_i = \begin{cases} 3, & \text{if } n_i \in \{n_1, n_2, n_{12}, n_{13}\}, \\ 1, & \text{otherwise.} \end{cases}$$

Restricted by the limited forwarding capacities, network congestion will soon occur in central nodes with a high degree, such as nodes n_4, n_5, and n_6. As finding the optimal routing of this network to satisfy various constraints is a hard task [18], it is chosen as the basic topology in this article.

The mission of routing is to forward each packet to its destination through intermediate nodes. The delivery delay ζ is mainly used to evaluate the performance of routing schemes, while other QoS metrics, such as residual energy and link utilization ratio, may be also considered in different practical scenarios.

TM plays an important role in the optimization of network routing. Even for various traffic matrices with the same κ, the corresponding optimal routes are usually different [36]. For the scenario in Fig. 1, the TM is uncertain and dynamic as the behavior of wild animals is unpredictable. In practice, the reliable and accurate measurement of TM is challenging for large and dynamic IP networks. Though the TM may still be derived from indirect information (such as link load measurements), the routing algorithms that perform well under estimated traffic matrices may not have a satisfactory performance with real traffic matrices [15].

B. Networked Multiagent POMDP

Consider a system of N agents that are working in a common environment and connected by a communication network $\mathcal{G} = (\mathcal{N}, \mathcal{E})$. We then define the following Markov decision problems for networked agents inspired by the decentralized POMDP.

A networked multiagent POMDP is denoted with a tuple $(\{\mathcal{S}^i\}_{i\in\mathcal{N}}, \{\mathcal{A}^i\}_{i\in\mathcal{N}}, P, \{R^i\}_{i\in\mathcal{N}}, \mathcal{G})$, where $\mathcal{S}^i$ is the local state space of agent i and $\mathcal{A}^i$ is the set of actions that agent i can take. In addition, let $\boldsymbol{\mathcal{S}} = \prod_{i=1}^N \mathcal{S}^i$ and $\boldsymbol{\mathcal{A}} = \prod_{i=1}^N \mathcal{A}^i$ be the joint state space and action space of all agents. Moreover, $R^i : \boldsymbol{\mathcal{S}} \times \boldsymbol{\mathcal{A}} \rightarrow \mathbb{R}$ is the local reward function of agent i, and $P : \boldsymbol{\mathcal{S}} \times \boldsymbol{\mathcal{A}} \times \boldsymbol{\mathcal{S}} \rightarrow [0, 1]$ denotes the stochastic transition probability among the joint states. In a discrete-time environment, at time step t, suppose that the global state is $\boldsymbol{s}_t = (s_t^1, \ldots, s_t^N) \in \boldsymbol{\mathcal{S}}$ and the agents execute joint actions $\boldsymbol{a}_t = (a_t^1, \ldots, a_t^N) \in \boldsymbol{\mathcal{A}}$. Then, each agent $i \in \mathcal{N}$ receives a reward r_t^i, which is given by $R^i(\boldsymbol{s}_t, \boldsymbol{a}_t)$. Also, the MDP transits to a new state $\boldsymbol{s}_{t+1} \in \boldsymbol{\mathcal{S}}$ with probability $P(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t, \boldsymbol{a}_t)$. Note that an agent's reward does not only depend on its own action but also the actions of other agents. The global reward at t is defined as $r_t = \sum_{i\in\mathcal{N}} r_t^i$, the sum of each agent's local reward.

Let $\pi^i : \mathcal{S}^i \times \mathcal{A}^i \rightarrow [0, 1]$ be the probability that agent i selects action $a^i \in \mathcal{A}^i$ at local state $s^i \in \mathcal{S}^i$. When the state space is large, it is useful to represent policies as parametric functions (e.g., DNNs). For any agent i, we define the local policy as $\pi_{\theta^i}^i$, where $\theta^i \in \Theta^i$ is the parameter and Θ^i is a compact set of feasible parameters. We pack the parameters together by writing $\boldsymbol{\theta} = (\theta^1, \ldots, \theta^N) \in \boldsymbol{\Theta}$, where $\boldsymbol{\Theta} = \prod_{i=1}^N \Theta^i$. Let $\boldsymbol{\pi} : \boldsymbol{\mathcal{S}} \times \boldsymbol{\mathcal{A}} \rightarrow [0, 1]$ be the joint policy function, that is, $\boldsymbol{\pi}(\boldsymbol{a}|\boldsymbol{s})$ is the probability of choosing a joint action $\boldsymbol{a}$ at a global state $\boldsymbol{s}$. As each agent chooses its own action individually at each time step, the joint policy is given by $\boldsymbol{\pi}(\boldsymbol{a}|\boldsymbol{s}) = \prod_{i\in\mathcal{N}} \pi_{\theta^i}^i(a^i|s^i)$.

In the following, we make some regularity assumptions that are standard in the literature on actor–critic RL algorithms with function approximation [37]. It is assumed that for any $i \in \mathcal{N}$, $s^i \in \mathcal{S}^i$, and $a^i \in \mathcal{A}^i$, the policy function $0 \leq \pi_{\theta^i}^i(a^i|s^i) \leq 1$ for any $\theta^i \in \Theta^i$. Also, $\pi_{\theta^i}^i(a^i|s^i)$ is continuously differentiable with respect to the parameter θ^i over Θ^i, which is required by policy gradient methods and satisfied by well-known function classes such as DNNs. In addition, for any $\boldsymbol{\theta} \in \boldsymbol{\Theta}$, let $P_{\boldsymbol{\theta}}$ be the transition matrix of the Markov chain $\{\boldsymbol{s}_t\}_{t\geq 0}$ induced by policy $\boldsymbol{\pi}_{\boldsymbol{\theta}}$, that is, $P_{\boldsymbol{\theta}}(\boldsymbol{s}'|\boldsymbol{s}) = \sum_{\boldsymbol{a}\in\boldsymbol{\mathcal{A}}} \boldsymbol{\pi}_{\boldsymbol{\theta}}(\boldsymbol{a}|\boldsymbol{s}) P(\boldsymbol{s}'|\boldsymbol{s}, \boldsymbol{a})$, $\forall \boldsymbol{s}, \boldsymbol{s}' \in \boldsymbol{\mathcal{S}}$. We assume that the Markov chain $\{\boldsymbol{s}_t\}_{t\geq 0}$ is irreducible and aperiodic under any $\boldsymbol{\pi}_{\boldsymbol{\theta}}$ and it has a stationary distribution $d_{\boldsymbol{\theta}}(\boldsymbol{s})$ over $\boldsymbol{\mathcal{S}}$. Hence, the Markov chain of the state–action pair $\{(\boldsymbol{s}_t, \boldsymbol{a}_t)\}_{t\geq 0}$ has a stationary distribution $d_{\boldsymbol{\theta}}(\boldsymbol{s})\boldsymbol{\pi}_{\boldsymbol{\theta}}(\boldsymbol{a}|\boldsymbol{s})$ for any $\boldsymbol{s} \in \boldsymbol{\mathcal{S}}$ and $\boldsymbol{a} \in \boldsymbol{\mathcal{A}}$.

The objective of the networked agents is to collaboratively find an optimal joint policy $\boldsymbol{\pi}_{\boldsymbol{\theta}}$ that maximizes the global long-term return using only local information. We define $Q^{\boldsymbol{\pi}_{\boldsymbol{\theta}}}(\boldsymbol{s}, \boldsymbol{a}) = \sum_{t=0}^{\infty} \mathbb{E}[\gamma^t r_t | \boldsymbol{s}_0 = \boldsymbol{s}, \boldsymbol{a}_0 = \boldsymbol{a}; \boldsymbol{\pi}_{\boldsymbol{\theta}}]$ as the global state–action value function, where γ is the discount factor. The global state value function is represented by $V^{\boldsymbol{\pi}_{\boldsymbol{\theta}}}(\boldsymbol{s}) = \sum_{\boldsymbol{a}\in\boldsymbol{\mathcal{A}}} \boldsymbol{\pi}_{\boldsymbol{\theta}}(\boldsymbol{a}|\boldsymbol{s}) Q^{\boldsymbol{\pi}_{\boldsymbol{\theta}}}(\boldsymbol{s}, \boldsymbol{a})$. Accordingly, we denote the global advantage function by $A^{\boldsymbol{\pi}_{\boldsymbol{\theta}}}(\boldsymbol{s}, \boldsymbol{a}) = Q^{\boldsymbol{\pi}_{\boldsymbol{\theta}}}(\boldsymbol{s}, \boldsymbol{a}) - V^{\boldsymbol{\pi}_{\boldsymbol{\theta}}}(\boldsymbol{s})$. Then, the goal of the multiagent decision is to solve the optimization problem

$$\begin{aligned} \max_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) &= \sum_{\boldsymbol{s}\in\boldsymbol{\mathcal{S}}} d_{\boldsymbol{\theta}}(\boldsymbol{s}) V^{\boldsymbol{\pi}_{\boldsymbol{\theta}}}(\boldsymbol{s}) \\ &= \sum_{\boldsymbol{s}\in\boldsymbol{\mathcal{S}}} d_{\boldsymbol{\theta}}(\boldsymbol{s}) \sum_{\boldsymbol{a}\in\boldsymbol{\mathcal{A}}} \boldsymbol{\pi}_{\boldsymbol{\theta}}(\boldsymbol{a}|\boldsymbol{s}) Q^{\boldsymbol{\pi}_{\boldsymbol{\theta}}}(\boldsymbol{s}, \boldsymbol{a}). \end{aligned} \tag{4}$$

C. Formulation of Routing With Multiagent POMDP

The packet routing is formulated as an MARL problem based on networked multiagent POMDP, where each node is an independent agent that observes the local state and makes routing decisions according to its own policy. Therefore, we will illustrate the definitions of each element in MARL for a single agent.

1) State: We denote the state of agent i as $s^i : (d^i, \psi^i)$, where d^i is the destination of current HOL packet in node n_i's queue and ψ^i is some extra information related to agent i. We define ψ^i as an $|\mathcal{N}_d|$-dimensional vector whose jth element ψ_j^i is the number of caching packets destined to the node $n_j \in \mathcal{N}_d$. Hence, the current length of node n_i's packet buffer is $\sum_{j\in\mathcal{N}_d} \psi_j^i$.

2) Action: The action of agent i is defined as $a^i : n_j \in \Gamma_i$, where Γ_i is the set of neighbors of node n_i. Accordingly, for each agent, the size of action space equals the number of its adjacent nodes. Once a packet arrives at the head of queue at time step t, agent i observes a local state s_t^i and delivers the packet to a neighboring node a_t^i according to its policy π_{θ^i}.

3) Reward: We craft the reward to guide the agent toward an efficient policy for our target. To minimize the delivery time of all packets ζ, the reward r_t^i of agent i at time step t is set to be the negative of the total number of packets in its buffer queue. Also, the global reward is the sum $r_t = \sum_{i \in \mathcal{N}} r_t^i$. In this way, the problem of delayed rewards is partially alleviated because the effective reward is available at each time step.

IV. Routing Based on Networked Multiagent PPO

In this section, our focus is on optimizing routing schemes to provide better network performance in scenarios of fixed traffic matrices without knowledge about channel statistics of the network.

The explicit routing method for traffic engineering can achieve high network performance as it allows traffic flows of each source–destination pair to be distributed along predetermined paths. Moreover, GDMR proposed in [14] can convert an arbitrary explicit routing into a loop-free destination-based routing without any performance penalty for a given TM. The resulting destination-based routing only requires each router to maintain a simple forwarding table with at most $O(N)$ entries. Therefore, we can first obtain an explicit routing based on the previously described networked POMDP model through the MARL method and then convert it into an easy-to-deploy routing scheme by GDMR.

As the theoretical basis of the policy gradient algorithm and the actor–critic method in RL, the policy gradient theorem for single-agent RL in [16] is extended to networked MARL. Combining observation and action spaces of all agents in MARL and treating the problem as a single high-dimensional MDP, the gradient of $J(\boldsymbol{\theta})$ with respect to $\boldsymbol{\theta}$ is given by [16, Sec. 13.1]

$$\begin{aligned}\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) &\propto \mathbb{E}_{s\sim d_{\theta}, a\sim \pi_{\theta}}[Q^{\pi}(\boldsymbol{s},\boldsymbol{a}) \log \boldsymbol{\pi}_{\boldsymbol{\theta}}(\boldsymbol{a}|\boldsymbol{s})] \\ &= \mathbb{E}_{s\sim d_{\theta}, a\sim \pi_{\theta}}[A^{\pi}(\boldsymbol{s},\boldsymbol{a}) \log \boldsymbol{\pi}_{\boldsymbol{\theta}}(\boldsymbol{a}|\boldsymbol{s})] \\ &= \sum_{s\in\mathcal{S}} d_{\boldsymbol{\theta}}(s) \sum_{a\in\mathcal{A}} \boldsymbol{\pi}_{\boldsymbol{\theta}}(\boldsymbol{a}|\boldsymbol{s})[A^{\pi}(s,\boldsymbol{a}) \log \boldsymbol{\pi}_{\boldsymbol{\theta}}(\boldsymbol{a}|\boldsymbol{s})]. \quad (5)\end{aligned}$$

Recall that $\boldsymbol{\pi}_{\boldsymbol{\theta}}(\boldsymbol{a}|\boldsymbol{s}) = \prod_{i\in\mathcal{N}} \pi^i_{\theta^i}(a^i|s^i)$ in the networked multiagent POMDP, (5) can be rewritten as follows:

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) \propto \mathbb{E}_{s\sim d_{\theta}, a\sim \pi_{\theta}}\Big[A^{\pi}(\boldsymbol{s},\boldsymbol{a}) \nabla_{\boldsymbol{\theta}} \sum_{i\in\mathcal{N}} \log \pi^i_{\theta^i}(a^i|s^i)\Big].$$

Due to the independence of each θ^i, the partial derivative of $J(\boldsymbol{\theta})$ with respect to local parameter θ^i for networked multiagent POMDP becomes

$$\begin{aligned}\nabla_{\theta^i} J(\boldsymbol{\theta}) &\propto \mathbb{E}_{s\sim d_{\theta}, a\sim \pi_{\theta}}[A^{\pi}(\boldsymbol{s},\boldsymbol{a}) \nabla_{\theta^i} \log \pi^i_{\theta^i}(a^i|s^i)] \\ &= \sum_{s\in\mathcal{S}} d_{\boldsymbol{\theta}}(\boldsymbol{s}) \sum_{a\in\mathcal{A}} \boldsymbol{\pi}_{\boldsymbol{\theta}}(\boldsymbol{a}|\boldsymbol{s})\big[A^{\pi}(\boldsymbol{s},\boldsymbol{a}) \nabla_{\theta^i} \log \pi^i_{\theta^i}(a^i|s^i)\big]. \quad (6)\end{aligned}$$

Also, all the partial derivatives $\nabla_{\theta^1} J(\boldsymbol{\theta})$, $\nabla_{\theta^2} J(\boldsymbol{\theta})$, $\ldots$, $\nabla_{\theta^N} J(\boldsymbol{\theta})$ actually constitute the gradient of $J(\boldsymbol{\theta})$ at $\boldsymbol{\theta}$. If we guarantee stochastic gradient ascent of each agent's θ^i never leaves a sufficiently small neighborhood of $\boldsymbol{\theta}$, the joint policy will be improved from the global view.

Equation (6) shows that $\nabla_{\theta^i} J(\boldsymbol{\theta})$ can be obtained locally when given the advantage function $A^{\pi}(\boldsymbol{s},\boldsymbol{a})$. This allows us to take advantage of the centralized critic and distributed actor framework and extend the policy gradient algorithms in the single-agent RL to MARL. The centralized value function stabilizes training by accounting for states and actions of all agents, while this extra information is not required by distributed agents in the exploration and execution phase. As shown in Section VI, the distributed policies with a common centralized critic work very well in various multipath routing scenarios. Note that the framework still cannot avoid some open problems in single-agent RL, such as getting trapped into local optima. Also, due to its essential characteristics that each agent selects actions only according to the local state, there is no guarantee that the globally optimal policy can be represented by the distributed strategy. Considering an extreme scenario where the optimal action of agent i is solely decided by another agent j, this framework obviously cannot achieve the optimal solution.

We propose an MARL algorithm named MAPPO for optimizing routing schemes by extending PPO in the multiagent environment according to Equation (6). With the clipping of the probability ratio between old and new policies, stochastic gradient ascent of each agent's θ^i can stay in a sufficiently small neighborhood of $\boldsymbol{\theta}$ as much as possible.

In a training epoch of MAPPO, some trajectories $\mathcal{D}$ can be obtained from interaction with the environment, including the history of all nodes' states, actions, and rewards. We denote a trajectory by $\tau = \{\boldsymbol{s}_0, \boldsymbol{a}_0, r_0, \boldsymbol{s}_1, \ldots, r_{T-1}, \boldsymbol{s}_T\}$, where all packets are delivered at time step T. The estimated discounted return $\hat{G}_t$ on τ at time step t is computed by [16]

$$\hat{G}_t = \sum_{k=t}^{T-1} \gamma^{k-t} r_k + \gamma^{T-t} V_{\phi}(\boldsymbol{s}_T). \quad (7)$$

where $\gamma \in (0,1]$ is the discount factor and V_{ϕ} is the global value function parameterized with ϕ. The global advantage estimate $\hat{A}_t$ on τ at time step t can be given by a truncated generalized advantage estimation (GAE) [38]

$$\hat{A}_t = \hat{A}(\boldsymbol{s}_t, \boldsymbol{a}_t) = \delta_t + \gamma\lambda\delta_{t+1} + \cdots + (\gamma\lambda)^{T-1-t}\delta_{T-1} \quad (8)$$

where $\delta_t = r_t + \gamma V(\boldsymbol{s}_{t+1}) - V(\boldsymbol{s}_t)$ and $\lambda \in [0,1]$ is a hyperparameter to balance the bias–variance tradeoff.

Based on PPO-Clip in [38], the surrogate objective for agent i in MAPPO is defined as

$$J_i^{\mathrm{CLIP}}(\theta^i) = \mathbb{E}_{\boldsymbol{s}_t,\boldsymbol{a}_t}\Big[\min\Big(\rho_t^i(\theta^i)\hat{A}(\boldsymbol{s}_t,\boldsymbol{a}_t),\, g\big(\epsilon, \hat{A}(\boldsymbol{s}_t,\boldsymbol{a}_t)\big)\Big)\Big] \quad (9)$$

where

$$\rho_t^i(\theta^i) = \frac{\pi_{\theta^i}(a_t^i|s_t^i)}{\pi_{\theta^i_{\mathrm{old}}}(a_t^i|s_t^i)}, \quad g(\epsilon,\hat{A}) = \begin{cases}(1+\epsilon)\hat{A}, & \hat{A} \ge 0\\ (1-\epsilon)\hat{A}, & \hat{A} < 0\end{cases}$$

θ^i_{old} is the policy parameter that the agent i used to sample τ, and ϵ is a hyperparameter, say, $\epsilon = 0.2$ by default.

Given a set of trajectories $\mathcal{D} = \{\tau\}$, the policy of each agent i in MAPPO is updated by

$$\theta^i_{\mathrm{new}} = \arg\max_{\theta^i} \frac{1}{|\mathcal{D}|T} \sum_{\tau\in\mathcal{D}} \sum_{t=0}^{T-1} J_i^{\mathrm{CLIP}}(\theta^i). \quad (10)$$

In practice, θ^i can be updated iteratively via stochastic gradient ascent

$$\theta^i \leftarrow \theta^i + \alpha \nabla_{\theta^i} J_i^{\text{CLIP}}(\theta^i) \tag{11}$$

where α is the learning rate of the policy.

As the state value function $V_\phi(\boldsymbol{s}) = \mathbb{E}_\pi[\hat{G}_t | \boldsymbol{s}_t = \boldsymbol{s}]$, the global critic is updated by regression on mean-squared error

$$\begin{aligned}\phi_{\text{new}} &= \arg\min_\phi L(\phi) \\ &= \arg\min_\phi \frac{1}{|\mathcal{D}|T} \sum_{\tau \in \mathcal{D}} \sum_{t=0}^{T-1} \left(V_\phi(\boldsymbol{s}_t) - \hat{G}_t\right)^2. \end{aligned} \tag{12}$$

Thus, ϕ can be updated iteratively via stochastic gradient descent

$$\phi \leftarrow \phi - \beta \nabla_\phi L(\phi) \tag{13}$$

where β is the learning rate of the critic.

The steps of MAPPO are given in Algorithm 1. It is worth mentioning that MAPPO is not only suitable for routing decisions but also for a wider range of general multiagent POMDP problem-solving.

Algorithm 1 MAPPO

```
1: Input: initial policy parameter $\theta_0^i$ of each agent, initial
   global value function parameters $\phi_0$.
2: for $k = 0, 1, 2, \ldots, K-1$ do
3:     Collect set of trajectories $\mathcal{D}_k = \{\tau\}$ with policy $\boldsymbol{\pi}_{\boldsymbol{\theta}_k}$.
4:     Compute the estimated return $\hat{G}_t$ by Equation (7) for
       each $s_t$ based on the current value function $V_{\phi_k}$.
5:     Compute the advantage estimates $\hat{A}_t$ by Equation (8)
       based on $V_{\phi_k}$.
6:     Update the policy parameter $\theta_{k+1}^i$ of each agent $i$ by (11)
       for each state–action pair $(s_t^i, a_t^i)$.
7:     Fit the global value function by regression on
       mean-squared error and update the critic parameter $\phi_{k+1}$
       by Equation (13) for each global state $\boldsymbol{s}_t$.
8: end for
```

Since sampling data and optimizing policies are separable in training MAPPO, we proposed a novel practicable implementation framework of MAPPO to search for an optimal explicit routing in multihop networks. As shown in Fig. 2, the policy of agent i is approximated by an independent DNN with parameter θ^i, which is also called the actor network and located at the corresponding node n_i. In order to coordinate the learning of agents, we set up a virtual central node to deploy the critic network (value function), which is approximated by a DNN with parameter ϕ. This virtual node communicates with each sink node and can be located in any physical edge node. In order to eliminate the communication overhead caused by the transmission of massive trajectory and value function information, a shadow policy network with the same parameters θ^i is set up for each agent i at the central node.

The training of MAPPO alternates between sampling data through interaction with the environment (exploration) and optimizing the objective function using stochastic gradient ascent (exploitation) as follows.

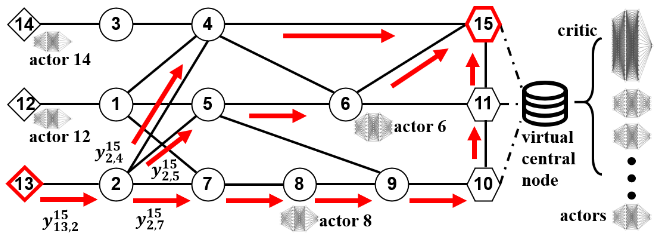


Fig. 2. Implementation of MAPPO for multipath routing.

1) In the exploration phase, each node uses only local information for decision-making (decentralized execution). An agent i selects next-hop node of its current HOL packet according to the local policy network independently and encapsulates the local decision information, such as s_t^i, a_t^i, $\log \pi_i(a_t^i | s_t^i)$, and r_t^i, into the payload of the HOL packet which is forwarded immediately. All the packets will be delivered to their destinations (edge nodes in Fig. 2).
2) In the exploitation phase, the virtual central node collects all the decision information from sink nodes to get trajectories $\mathcal{D}$. If necessary, it can also communicate directly with other nodes to get their information. Based on $\mathcal{D}$, the parameters of shadow actors and centralized critic are updated by Algorithm 1. Then, the parameters of shadow policies are dispatched to the distributed nodes for routing decisions to collect next $\mathcal{D}$.

An explicit routing can be constructed from the paths in each trajectory τ. The explicit routing with minimum delivery delay $\zeta_{\min}$ in the whole training stage should be memorized. After the training stage, the explicit routing is converted into a loop-free destination-based routing solution $\{y_{i,j}^d\}$ by using the routing conversion algorithm proposed in [14], where $y_{i,j}^d$ denotes the split ratio at node n_i to node n_j for the traffic destined to node n_d $(n_d \in \mathcal{N}_d, \langle n_i, n_j \rangle \in \mathcal{E})$. The calculated destination-based routing solution $\{y_{i,j}^d\}$ is further translated to forwarding entries, which will be installed in distributed nodes. Note that only the forwarding table is used by each node for routing decision-making in the execution stage of the multipath routing.

V. Routing With Meta-MAPPO

When the routing decision is modeled as an MARL task, the MDP of a node is not only affected by its neighboring nodes' strategies but also affected by the TM. Changes in the TM will affect the POMDP of each node, and the optimal routing policy for various traffic matrices may also be different [36]. Packet routing becomes a multitask learning problem for dynamic traffic demand.

Formally, we define a task $\omega(\eta)$ as solving the optimal routing scheme for a fixed TM η. We denote by Ω the probability distribution of all possible tasks ω with different traffic matrices and assume that Ω is a uniform distribution over the set $\{\omega(\eta) | \sum_{i,j} \eta_{i,j} = \kappa\}$. For the multitask learning problem, we are interested to find a policy (named metapolicy) that shows good performance in every task ω sampled from Ω. The joint-training and fine-tuning framework is used, the idea behind which is very simple: let agents learn from all possible tasks and environments to get a universal policy with metaparameter $\boldsymbol{\theta}_{\text{meta}}$, which will be fine-tuned with respect to a specific task before execution. The joint-training process works by repeatedly [45]: 1) sampling a task $\omega(\eta) \in \Omega$; 2) training on it by multiple gradient update steps from initial parameter $\boldsymbol{\theta}_{\text{meta}}$ to get a new parameters $\boldsymbol{\theta}$; and 3) and then moving the parameters of metapolicy toward new $\boldsymbol{\theta}_{\text{meta}}$ by taking advantage of $\boldsymbol{\theta}$.

Since joint-learning is a strong baseline for multitask RL setting [45], we first propose a novel algorithm named MAPPO with joint-learning (joint-MAPPO). Then, based on meta-RL, the meta-MAPPO is given to resolve multiagent multitask RL problems as a model-free method. For simplicity, we denote $\boldsymbol{\mu} = (\boldsymbol{\theta}, \phi)$, where $\boldsymbol{\theta} = (\theta^1, \theta^2, \dots, \theta^N)$ are the parameters of all actor networks, and ϕ is the parameters of the global critic network.

The joint-training phase is to obtain an initial $\boldsymbol{\mu}_{\text{joint}}$ based on a sampled batch of task ω, which is used in the fine-tuning phase and leads to fast adaptation to time-varying traffic demand matrix. To solve packet routing problems with dynamic traffic demand, MAPPO is adopted as the inner MARL algorithm to update the parameters $\boldsymbol{\theta}$ from initial parameter $\boldsymbol{\theta}_{\text{meta}}$. The steps of joint-MAPPO are given in Algorithm 2.

Algorithm 2 Joint-MAPPO

```
Phase 1: joint-training
 1: Initialize $\boldsymbol{\mu}_0 = (\boldsymbol{\theta}_0, \phi_0)$, composed of each node's policy
    parameters $\theta_0^i$ and the value function parameter $\phi_0$.
 2: for iteration $k = 0, 1, 2, \dots, K-1$ do
 3:   Sample task $\omega$ from the distribution $\Omega$ where tasks are
      given with different traffic matrices.
 4:   Compute $\boldsymbol{\mu}_{k+1}$ using MAPPO with $\boldsymbol{\mu}_k$ based on $\omega$.
 5: end for
 6: Set $\boldsymbol{\mu}_{\text{joint}} = \boldsymbol{\mu}_K$.
Phase 2: fine-tuning
    For the task $\omega$ with current TM $\eta$, update the parameter $\boldsymbol{\mu}$
    using MAPPO with initial parameter $\boldsymbol{\mu}_{\text{joint}}$.
```

As a metalearning method with good performance, Reptile [45] can be used here to solve the multitask problem, and we call the Reptile with MAPPO by reptile-MAPPO for short. However, after multiple gradient updates by (1), the differences between policies with parameters $\boldsymbol{\mu}'$ and $\boldsymbol{\mu}_{\text{meta}}$ become so large that the performance of the updated metapolicy deteriorates. In order to address this problem, a novel metagradient MARL algorithm, named meta-MAPPO, is proposed by adding a KL-divergence penalty in the inner objective function.

For the trajectories $\mathcal{D}$ sampled with parameters $\boldsymbol{\mu}$, we define the surrogate objective of meta-MAPPO's inner optimization procedure for agent i as

$$J_i(\theta^i) = \mathbb{E}_{\boldsymbol{s}_t,\boldsymbol{a}_t}\Big[\min\Big(\rho_t^i(\theta^i)\hat{A}(\boldsymbol{s}_t,\boldsymbol{a}_t),\ g\big(\epsilon, \hat{A}(\boldsymbol{s}_t,\boldsymbol{a}_t)\big)\Big) - \nu\,\mathrm{KL}\Big[\pi_{\theta^i_{\text{meta}}}\big(\cdot|s_t^i\big), \pi_{\theta^i}\big(\cdot|s_t^i\big)\Big]\Big] \quad (14)$$

where $\hat{A}_t$ is computed based on $\mathcal{D}$, $\mathrm{KL}(\cdot,\cdot)$ is the KL-divergence, ν is an adaptive coefficient, and $\nu = 3$ by default. Note that the clipping of ρ_t^i makes the updated θ^i not deviate too much from the old parameters θ^i_{old} used in sampling $\mathcal{D}$, while the KL-divergence penalty makes θ^i within the trust region around the metaparameters θ^i_{meta}. The parameter of inner policy θ^i for agent i in meta-MAPPO can be updated iteratively by

$$\theta^i \leftarrow \theta^i + \alpha \nabla_{\theta^i} J_i(\theta^i). \quad (15)$$

For a task ω sampled from Ω, we denote $\boldsymbol{\mu}_{\omega,K}$ as the result of K updates by (15), where $\boldsymbol{\mu}_{\omega,0} = \boldsymbol{\mu}_{\text{meta}}$. Given H tasks, the metaparameter $\boldsymbol{\mu}_{\text{meta}}$ is updated by

$$\boldsymbol{\mu}_{\text{meta}} \leftarrow \boldsymbol{\mu}_{\text{meta}} + \xi \frac{1}{H}\sum_{h=1}^{H}(\boldsymbol{\mu}_{h,K} - \boldsymbol{\mu}_{\text{meta}}) \quad (16)$$

where $\boldsymbol{\mu}_{h,K}$ is short for $\boldsymbol{\mu}_{\omega_h,K}$. The steps of meta-MAPPO are shown in Algorithm 3.

Algorithm 3 Meta-MAPPO

```
Phase 1: meta-learning
 1: Initialize parameters of meta-policy and critic networks
    $\boldsymbol{\mu}_{\text{meta}} = (\boldsymbol{\theta}_0, \phi_0)$, composed of each node's policy para-
    meter $\theta_0^i$ and the value function parameter $\phi_0$.
 2: for $l = 1, 2, 3, \dots, L$ do
 3:   for $h = 1, 2, 3, \dots, H$ do
 4:     Sample task $\omega_h$ from the task distribution $\Omega$.
 5:     Initialize $\boldsymbol{\mu}_{h,0} = \boldsymbol{\mu}_{\text{meta}}$.
 6:     for $k = 0, 1, 2, \dots, K-1$ do
 7:       Collect set of trajectories $\mathcal{D}_{h,k} = \{\tau\}$ with $\boldsymbol{\mu}_{h,k}$.
 8:       Compute the rewards-to-go $\hat{G}_t$ on $\mathcal{D}_{h,k}$ by Equa-
          tion (7) and update the critic parameter $\phi_{h,k+1}$ by
          Equation (13).
 9:       Compute the advantage estimates $\hat{A}_t$ on $\mathcal{D}_{h,k}$ by
          Equation (8) and update the policy parameter $\theta^i_{h,k+1}$
          of each agent $i$ by Equation (15).
10:     end for
11:   end for
12:   Update $\boldsymbol{\mu}_{\text{meta}}$ by Equation (16).
13: end for
Phase 2: fine-tuning
    For the task $\omega$ with current TM $\eta$, update the parameter $\boldsymbol{\mu}$
    using MAPPO with initial parameter $\boldsymbol{\mu}_{\text{meta}}$.
```

Like Reptile in [45], (16) actually utilizes the second-order gradient information of $\boldsymbol{\theta}_{\text{meta}}$ to update the metapolicy for $K \geq 2$. Thus, meta-MAPPO can be adapted to new tasks faster than joint-MAPPO. Moreover, the KL-divergence penalty in (14) makes $\boldsymbol{\mu}_{\omega,K}$ never leave a sufficiently small neighborhood of $\boldsymbol{\mu}_{\text{meta}}$, avoiding a sudden deterioration in the performance of the updated metapolicy. After the fine-tuning, we get an explicit routing with minimum delivery delay $\zeta_{\min}(\eta)$ for the current TM η. If the real-time TM does not change significantly, the expected delivery time of packets generated within a sampling period is $\zeta_{\min}(\eta)$.

In the implementation of both joint-MAPPO and meta-MAPPO for packet routing with dynamic TM, a key issue is how to trigger the fine-tuning. A event-triggered mechanism is used in this article. The event-triggered condition is designed based on the delivery time ζ. Regardless of whether the traffic demand varies gradually or abruptly, as long as the average delivery time $\bar{\zeta}$ of several consecutive sample periods in the execution stage deviates from the expected value $\zeta_{\min}(\eta)$ by more than 5%, a new fine-tuning process will be triggered.

VI. Simulation Results

In this section, we present simulation results of our routing schemes in four networked multiagent environments. For the network setting shown in Fig. 2, we first perform comparisons among MAPPO, SPR, Q-routing [17], QELAR [20], and DQRC [33] in an ideal environment with fixed but unknown TM. Then, we examine the performance of both delivery delay and energy consumption in noisy environments where the transmission attempt may fail and the energy efficiency is also a major design concern, such as WSNs described in Fig. 1. The effectiveness of meta-MAPPO is also evaluated in networks with dynamic TM. Finally, we show the network throughput performance of our algorithms in a real ISP network topology-Sprintlink network [48].

To represent each node's policy in our algorithms, we use a two-hidden-layer neural network with 32 units in each fully connected layer with a tanh nonlinearity and adopt the softmax activation function in the output layer. The global critic neural network is composed of two fully connected hidden layers with 64 units and a rectified linear unit (ReLU) nonlinearity, following a linear layer as the output layer. We use the adaptive moment estimation (Adam) optimizer with a learning rate of 0.003 for all of our experiments (i.e., $\alpha = \beta = \xi = 0.003$). The GAE in (8) uses $\gamma = 0.99$ and $\lambda = 0.95$. The meta-MAPPO uses $H = 5$ sampled tasks and $K = 2$ adaptations in each iteration.

A. Ideal Environment With Fixed Traffic Demand Matrix

First of all, we consider an ideal network scenario where there is no packet loss and assume that the unknown traffic demand matrix is fixed over a long period time. Given the fixed TM $\eta_0 = \text{diag}\{10, 10, 10\}$, the proposed algorithm MAPPO is used to search for the optimal routing scheme, with Q-routing, QELAR, and DQRC as baselines which are distributed routing polices based on RL. The minimum delivery delay $\zeta_{\min}$ during the training stage is the major design concern in the experiment.

The variation trend of cumulative reward (return) of MAPPO along with the training episode is shown in Fig. 3(a), where the actual reward value is multiplied by a scale factor to make it near 1 to speed up the convergence. We can

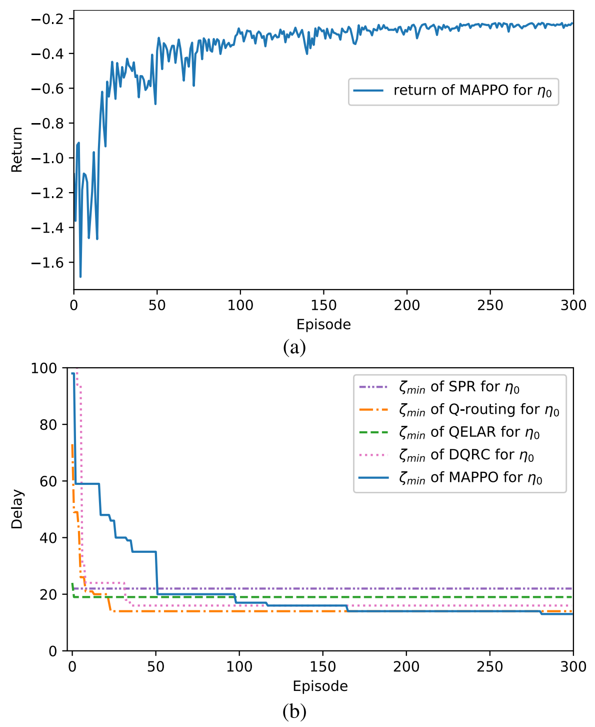


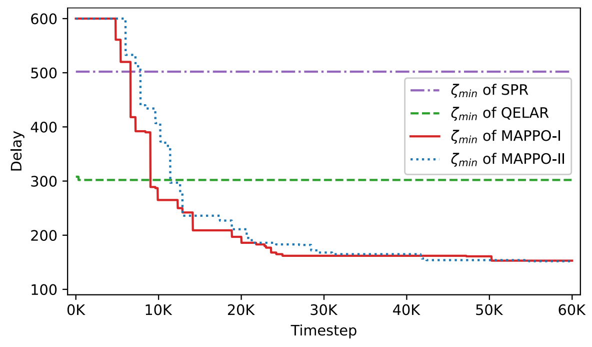
Fig. 3. (a) Return of MAPPO for η_1 and (b) delivery delay of five routings for η_1 in the training stage: SPR, Q-routing, QELAR, DQRC, and MAPPO.

clearly see that the reward converges to a stable value after a considerable decrease in the first 100 training episodes. Fig. 3(b) shows the minimum delivery delay $\zeta_{\min}$ of five routing schemes, i.e., SPR, Q-routing, QELAR, DQRC, and MAPPO, with respect to the episode of training, where $\zeta_{\min}$ is the minimum value of all delivery delays currently obtained during the training process. The delay of RL-based routing schemes drops sharply to a level below the SPR method in the first few episodes, and then gradually converges to a lower value. Though the delay of MAPPO grows down most slowly among these routing policies, MAPPO can reach the lowest delay $\zeta_{\min}$, which means that it obtains the best explicit routing. Q-routing achieves higher performance due to its better greedy exploration than its variant QELAR, which is mainly optimized for packet loss and energy consumption. As shown in Fig. 3(b), DQRC shows a comparable performance on $\zeta_{\min}$ as Q-Routing in our network scenarios, although it utilizes richer real-time information for decision-making.

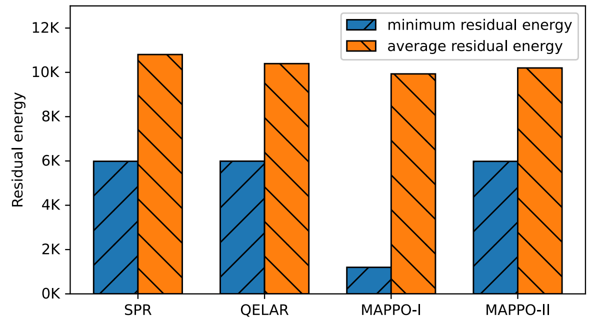
B. Noisy Environment With Energy Constrain

In order to showcase the flexibility and universality of MAPPO, we evaluated both delivery delay and energy consumption of various routing algorithms in a noisy environment, where links have heterogeneous channel quality and nodes have unbalanced initial energy distribution.

For the topology shown in Fig. 1, we assume that the packet loss rate of link $\langle n_2, n_4 \rangle$ is 0.8, while other links are 0.05, and the initial residual energy of node n_1 is 6000 units, while others are 12 000 units. When a node transmits and receives a packet and standby, it consumes 10 units, 3 units, and 0.03 unit of energy. Note that this energy consumption setting is consistent with the simulation in [20], in order

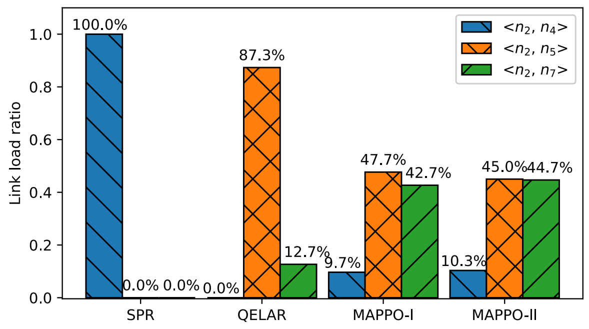


(a)

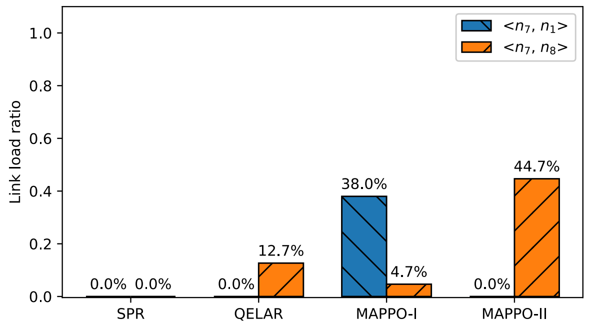


(b)

Fig. 4. (a) Delivery delay in the training stage and (b) residual energy in the packet forwarding stage of SPR, QELAR, MAPPO-I, and MAPPO-II.



(a)



(b)

Fig. 5. Traffic load ratio at key nodes of SPR, QELAR, MAPPO-I, and MAPPO-II for the traffic flow between source–destination pair (n_{13}, n_{15}) in the packet forwarding stage. (a) Node n_2. (b) Node n_7.

to facilitate comparison with QELAR. To simultaneously optimize the energy consumption, we integrate the minimum residual energy $e_{\min}$ of all nodes into the global reward function $r_t = \sum_{i\in\mathcal{N}} r_t^i - k_e e_{\min}$, where k_e is the weight coefficient and equal to 1 by default. We call the variant MAPPO-II for short and denote the routing scheme with global reward $r_t = \sum_{i\in\mathcal{N}} r_t^i$ in Section IV as MAPPO-I.

Fig. 4 shows the results of delivery delay and residual energy of the best trajectory of SPR, QELAR, MAPPO-I, and MAPPO-II. To simplify the description, we only consider the traffic flow between one source–destination pair (n_{13}, n_{15}) and set the total traffic volume $\kappa = 300$ here. $\zeta_{\min}$ at the current time step is the minimum value of all delivery delays already obtained during the training process. One can see that both MAPPO-I and MAPPO-II have the lowest delivery delay in Fig. 4(a), while QELAR is better than SPR due to its multipath characteristic. However, as shown in Fig. 4(b), MAPPO-I has the lowest minimum residual energy among these routing schemes as it does not make special adjustments to optimize energy consumption. MAPPO-II is energy-efficient and can achieve a longer network lifetime without sacrificing the packet delivery time. Note that the reason why SPR also has good performance is that the shortest path between n_{13} and n_{15} is $(n_{13}, n_2, n_4, n_{15})$, just bypassing the noisy link $\langle n_2, n_4\rangle$ and the node n_1 with lower residual energy.

To further illustrate the performance differences of these algorithms, we show the traffic load ratio at some key nodes in packet forwarding stage. We denote $l_{i,j}^d$ as the traffic load ratio at link $\langle n_i, n_j\rangle$ for the traffic flow destined to node n_d, which is slightly different from the traffic split ratio $y_{i,j}^d$ defined at Section IV as $l_{i,j}^d$ is the traffic load of link divided by the overall traffic volume. From Fig. 5(a), one can see that

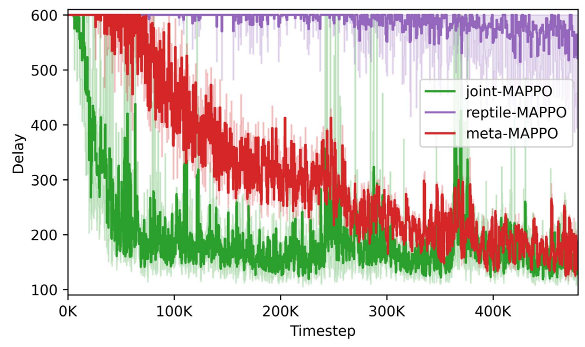


Fig. 6. Delivery delay of joint-MAPPO, reptile-MAPPO, and meta-MAPPO for randomly sampled traffic matrices in the joint-training phase.

QELAR, MAPPO-I, and MAPPO-II bypass the noisy link $\langle n_2, n_4\rangle$ at node n_2, while MAPPO-I and MAPPO-II retain very litter traffic flow on $\langle n_2, n_4\rangle$ as its packet loss rate is not equal to 1. Both MAPPO-I and MAPPO-II make the traffic between $\langle n_2, n_5\rangle$ and $\langle n_2, n_7\rangle$ more balanced, resulting in lower delivery delay. From Fig. 5(b), the traffic load ratio of MAPPO-II on link $\langle n_7, n_1\rangle$ is 0.0%, which means that MAPPO-II can bypass the node n_1 with low residual energy. However, the traffic load ratio of MAPPO-I on $\langle n_7, n_1\rangle$ is 38.0%, which increases the energy consumption of n_1.

C. Ideal Environment With Dynamic Traffic Demand Matrix

Then, we consider the network scenario where the traffic demand matrix is dynamic and random. For the time-varying TM, we want the routing strategy to be tuned to the optimal state as quickly as possible. The joint-MAPPO and meta-MAPPO methods are applied in the scenario. Q-routing is chosen for performance comparison as it is superior to QELAR and DQRC in ideal environments, as shown

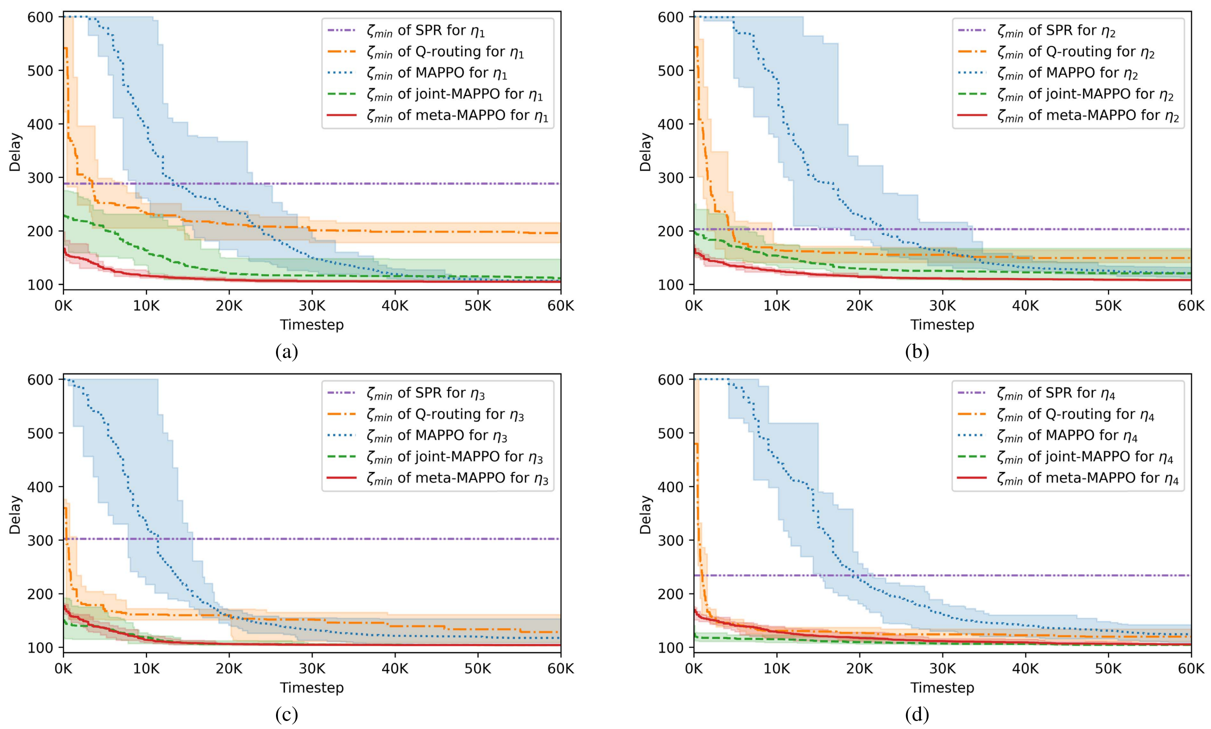


Fig. 7. Delivery delay of various routing schemes for four typical traffic matrices. (a) η_1. (b) η_2. (c) η_3. (d) η_4.

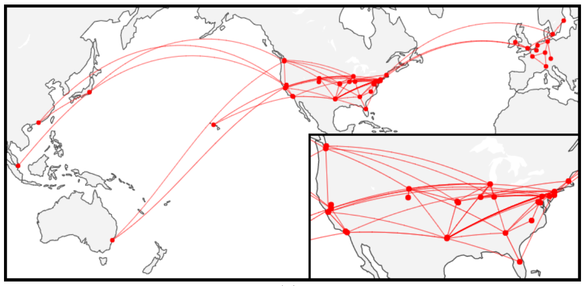
in Fig. 3(b). The dynamic TM η is chosen randomly while satisfying $\sum_{i,j}\eta_{ij}=\kappa=300$ in each training epoch. By using the joint-training framework, a good initial parameter $\boldsymbol{\mu}=\{\boldsymbol{\theta},\phi\}$ can be obtained. Fig. 6 shows the delivery delay ζ of joint-MAPPO, reptile-MAPPO, and meta-MAPPO in the joint-training phase, where we define the delivery delay of each time step as the same as that of the sampling period to which the time step belongs. The solid line is the average delivery delay of eight joint-training simulations and the bounds of the shaded region indicate the maximum and minimum values. The delay decreases along with the training time step, while the joint-MAPPO is the fastest among them. The reptile-MAPPO converges very slowly due to performance deterioration caused by overestimated metagradient in (1). By considering the KL-divergence penalty in (15), meta-MAPPO converges much faster than reptile-MAPPO.

After joint-training with randomly sampled traffic matrices, the fine-tuning is used to adapt to a specific TM. We examined several representative traffic matrices with the same total traffic volume $\kappa=300$, i.e.,

$$\eta_1=\operatorname{diag}\{15,270,15\},\quad \eta_2=\begin{bmatrix}0&0&0\\100&100&100\\0&0&0\end{bmatrix}$$

$$\eta_3=\begin{bmatrix}100&0&0\\100&0&0\\100&0&0\end{bmatrix},\quad\text{and } \eta_4=\begin{bmatrix}40&30&30\\30&40&30\\30&30&40\end{bmatrix}$$

where η_1 means the network scenario where most of the traffic is between one node pair, η_2 represents that all packets are generated at the same source node, η_3 represents that all packets have the same destination node, and η_4 is on behalf of the uniform packet flow. Fig. 7 shows the performance comparison of MAPPO, joint-MAPPO, meta-MAPPO, and Q-routing in terms of adaptation to different traffic demand matrices. The average value and maximum and minimum of the delivery delay $\zeta_{\min}$ are depicted by the solid line and the bounds of the shaded region around, respectively. One can see that meta-MAPPO has best performances on both convergence speed and final minimum delivery delay for η_1, η_2, and η_3. Moreover, the width of the shaded region of meta-MAPPO is the narrowest, meaning that it is more robust to random traffic demand. Note that joint-MAPPO converges



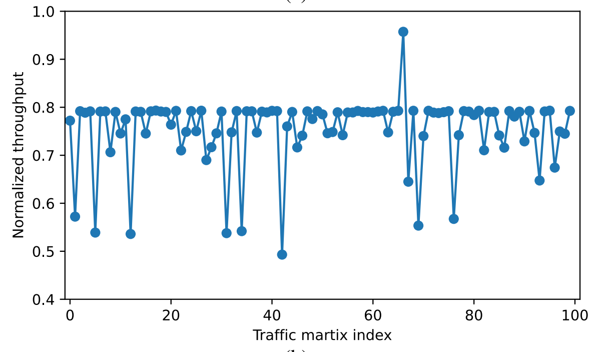


Fig. 8. (a) Topology of Sprintlink and (b) normalized throughput using meta-MAPPO under different traffic matrices.

faster than meta-MAPPO for η_4 because the joint-training of joint-MAPPO with randomly sampled traffic matrices is partially equivalent to fine-tuning for the uniform packet flow η_4.

D. Network Throughput Optimization by Meta-MAPPO

To illustrate the wide applicability of the proposed algorithms, the network throughput performance of meta-MAPPO in a real ISP network topology-Sprintlink network is shown in Fig. 8. The topology of Sprintlink provided by the Rocketfuel [48] has 44 nodes and 166 directed links. We use the gravity model [49] to generate 100 synthetic traffic matrices to evaluate the throughput performance of the algorithm. The resulted throughput is normalized by the optimal explicit routing described in [14]. One can see that the network normalized throughput of meta-MAPPO is 0.8 on average, which is the same as the OLW [13] and Static GDMR reported in [14], although it does not achieve the optimal solution. It can be further optimized, but a more in-depth discussion is beyond the scope of this article.

VII. Conclusion

We have modeled packet routing as a multiagent learning problem and proposed two novel routing algorithms and their distributed implementations, i.e., MAPPO and meta-MAPPO, based on MARL and metalearning. As shown by the results of the experiments, agents can learn adaptive routing policies in unknown environments and the packet delivery delay and energy consumption can be improved significantly, through interactions with the environment by MAPPO. With meta-MAPPO, the routing can maintain desirable performance under varying traffic demand matrices. By considering various techniques in deep RL, in our future work, we will further improve the performance of the proposed algorithms.

References

[1] N. Wang, M. Zhang, A. Che, and B. Jiang, "Bi-objective vehicle routing for hazardous materials transportation with no vehicles travelling in echelon," *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 6, pp. 1867–1879, Jun. 2018.

[2] X. Li, Y.-C. Tian, G. Ledwich, Y. Mishra, X. Han, and C. Zhou, "Constrained optimization of multicast routing for wide area control of smart grid," *IEEE Trans. Smart Grid*, vol. 10, no. 4, pp. 3801–3808, Jul. 2019.

[3] Y. Zhang, R. Su, G. G. N. Sandamali, Y. Zhang, C. G. Cassandras, and L. Xie, "A hierarchical heuristic approach for solving air traffic scheduling and routing problem with a novel air traffic model," *IEEE Trans. Intell. Transp. Syst.*, vol. 20, no. 9, pp. 3421–3434, Sep. 2019.

[4] J. Gao, M. Li, L. Zhao, and X. Shen, "Contention intensity based distributed coordination for V2V safety message broadcast," *IEEE Trans. Veh. Technol.*, vol. 67, no. 12, pp. 12288–12301, Dec. 2018.

[5] S. Zhang, P. He, K. Suto, P. Yang, L. Zhao, and X. Shen, "Cooperative edge caching in user-centric clustered mobile networks," *IEEE Trans. Mobile Comput.*, vol. 17, no. 8, pp. 1791–1805, Aug. 2018.

[6] Q. Li, J. Gao, H. Liang, L. Zhao, and X. Tang, "Optimal power allocation for wireless sensor powered by dedicated RF energy source," *IEEE Trans. Veh. Technol.*, vol. 68, no. 3, pp. 2791–2801, Mar. 2019.

[7] N. Zhang, S. Zhang, P. Yang, O. Alhussein, W. Zhuang, and X. S. Shen, "Software defined space-air-ground integrated vehicular networks: Challenges and solutions," *IEEE Commun. Mag.*, vol. 55, no. 7, pp. 101–109, Jul. 2017.

[8] J. Gao, L. Zhao, and X. Shen, "Network utility maximization based on an incentive mechanism for truthful reporting of local information," *IEEE Trans. Veh. Technol.*, vol. 67, no. 8, pp. 7523–7537, Aug. 2018.

[9] Z. Wang and J. Crowcroft, "Quality-of-service routing for supporting multimedia applications," *IEEE J. Sel. Areas Commun.*, vol. 14, no. 7, pp. 1228–1234, Sep. 1996.

[10] B. Fortz and M. Thorup, "Optimizing OSPF/IS-IS weights in a changing world," *IEEE J. Sel. Areas Commun.*, vol. 20, no. 4, pp. 756–767, May 2002.

[11] Z.-H. Guan, L. Chen, and T.-H. Qian, "Routing in scale-free networks based on expanding betweenness centrality," *Phys. A, Stat. Mech. Appl.*, vol. 390, no. 6, pp. 1131–1138, Mar. 2011.

[12] D. Thaler and C. Hopps, *Multipath Issues in Unicast and Multicast Next-Hop Selection*, document IETF RFC 2991, Nov. 2000.

[13] K. Holmberg and D. Yuan, "Optimization of Internet protocol network design and routing," *Networks*, vol. 43, no. 1, pp. 39–53, Jan. 2004.

[14] J. Zhang, K. Xi, and H. J. Chao, "Load balancing in IP networks using generalized destination-based multipath routing," *IEEE/ACM Trans. Netw.*, vol. 23, no. 6, pp. 1959–1969, Dec. 2015.

[15] M. Roughan, M. Thorup, and Y. Zhang, "Traffic engineering with estimated traffic matrices," in *Proc. 3rd ACM SIGCOMM Conf. Internet Meas.*, New York, NY, USA, Oct. 2003, pp. 248–258.

[16] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 2018.

[17] J. A. Boyan and M. L. Littman, "Packet routing in dynamically changing networks: A reinforcement learning approach," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, San Francisco, CA, USA, 1993, pp. 671–678.

[18] S. P. M. Choi and D.-Y. Yeung, "Predictive Q-routing: A memory-based reinforcement learning approach to adaptive traffic control," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, Cambridge, MA, USA, 1995, pp. 945–951.

[19] J. Dowling, E. Curran, R. Cunningham, and V. Cahill, "Using feedback in collaborative reinforcement learning to adaptively optimize MANET routing," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 35, no. 3, pp. 360–372, May 2005.

[20] T. Hu and Y. Fei, "QELAR: A machine-learning-based adaptive routing protocol for energy-efficient and lifetime-extended underwater sensor networks," *IEEE Trans. Mobile Comput.*, vol. 9, no. 6, pp. 796–809, Jun. 2010.

[21] V. D. Valerio, F. L. Presti, C. Petrioli, L. Picari, D. Spaccini, and S. Basagni, "CARMA: Channel-aware reinforcement learning-based multi-path adaptive routing for underwater wireless sensor networks," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 11, pp. 2634–2647, Nov. 2019.

[22] Z. Zheng, A. K. Sangaiah, and T. Wang, "Adaptive communication protocols in flying ad hoc network," *IEEE Commun. Mag.*, vol. 56, no. 1, pp. 136–142, Jan. 2018.

[23] Z. Mammeri, "Reinforcement learning based routing in networks: Review and classification of approaches," *IEEE Access*, vol. 7, pp. 55916–55950, 2019.

[24] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Netw.*, vol. 61, pp. 85–117, Jan. 2015.

[25] Z.-H. Guan, B. Hu, and X. Shen, *Introduction to Hybrid Intelligent Networks: Modeling, Communication, and Control*. Berlin, Germany: Springer, 2019.

[26] Z.-H. Guan and G. Chen, "On delayed impulsive hopfield neural networks," *Neural Netw.*, vol. 12, no. 2, pp. 273–280, Mar. 1999.

[27] Z.-H. Guan, J. Lam, and G. Chen, "On impulsive autoassociative neural networks," *Neural Netw.*, vol. 13, no. 1, pp. 63–69, Jan. 2000.

[28] Z.-H. Guan, G. Chen, and Y. Qin, "On equilibria, stability, and instability of hopfield neural networks," *IEEE Trans. Neural Netw.*, vol. 11, no. 2, pp. 534–540, Mar. 2000.

[29] B. Hu, Z.-H. Guan, G. Chen, and F. L. Lewis, "Multistability of delayed hybrid impulsive neural networks with application to associative memories," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 5, pp. 1537–1551, May 2019.

[30] B. Hu, Z.-H. Guan, T.-H. Qian, and G. Chen, "Dynamic analysis of hybrid impulsive delayed neural networks with uncertainties," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 9, pp. 4370–4384, Sep. 2018.

[31] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015.

[32] D. Mukhutdinov, A. Filchenkov, A. Shalyto, and V. Vyatkin, "Multi-agent deep learning for simultaneous optimization for time and energy in distributed routing system," *Future Gener. Comput. Syst.*, vol. 94, pp. 587–600, May 2019.

[33] X. You, X. Li, Y. Xu, H. Feng, J. Zhao, and H. Yan, "Toward packet routing with fully distributed multiagent deep reinforcement learning," *IEEE Trans. Syst., Man, Cybern., Syst.*, early access, Aug. 13, 2020, doi: 10.1109/TSMC.2020.3012832.

[34] R. Ding, Y. Xu, F. Gao, X. Shen, and W. Wu, "Deep reinforcement learning for router selection in network with heavy traffic," *IEEE Access*, vol. 7, pp. 37109–37120, 2019.
[35] G. Stampa, M. Arias, D. Sanchez-Charles, V. Muntes-Mulero, and A. Cabellos, "A deep-reinforcement learning approach for software-defined networking routing optimization," 2017, *arXiv:1709.07080*. [Online]. Available: http://arxiv.org/abs/1709.07080
[36] A. Valadarsky, M. Schapira, D. Shahaf, and A. Tamar, "Learning to route," in *Proc. 16th ACM Workshop Hot Topics Netw. (HotNets)*, New York, NY, USA, Nov. 2017, pp. 185–191.
[37] Y. Li, "Deep reinforcement learning," 2018, *arXiv:1810.06339*. [Online]. Available: http://arxiv.org/abs/1810.06339
[38] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017, *arXiv:1707.06347*. [Online]. Available: http://arxiv.org/abs/1707.06347
[39] D. S. Bernstein, R. Givan, N. Immerman, and S. Zilberstein, "The complexity of decentralized control of Markov decision processes," *Math. Oper. Res.*, vol. 27, no. 4, pp. 819–840, Nov. 2002.
[40] A. OroojlooyJadid and D. Hajinezhad, "A review of cooperative multi-agent deep reinforcement learning," 2019, *arXiv:1908.03963*. [Online]. Available: http://arxiv.org/abs/1908.03963
[41] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Proc. 31st Adv. Neural Inf. Process. Syst. (NeurIPS)*, Long Beach, CA, USA, Dec. 2017, pp. 6382–6393.
[42] J. X Wang *et al.*, "Learning to reinforcement learn," 2016, *arXiv:1611.05763*. [Online]. Available: http://arxiv.org/abs/1611.05763
[43] Y. Duan, J. Schulman, X. Chen, P. L. Bartlett, I. Sutskever, and P. Abbeel, "RL^2: Fast reinforcement learning via slow reinforcement learning," 2016, *arXiv:1611.02779*. [Online]. Available: https://arxiv.org/abs/1611.02779
[44] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *Proc. 34th IEEE Int. Conf. Mach. Learn. (ICML)*, Sydney, NSW, Australia, Aug. 2017, pp. 1126–1135.
[45] A. Nichol, J. Achiam, and J. Schulman, "On first-order meta-learning algorithms," 2018, *arXiv:1803.02999*. [Online]. Available: http://arxiv.org/abs/1803.02999
[46] K. Zhang, Z. Yang, H. Liu, T. Zhang, and T. Basar, "Fully decentralized multi-agent reinforcement learning with networked agents," in *Proc. IEEE Int. Conf. Mach. Learn. (ICML)*, Jul. 2018, pp. 5872–5881.
[47] M. Roughan, Y. Zhang, W. Willinger, and L. Qiu, "Spatio-temporal compressive sensing and Internet traffic matrices (extended version)," *IEEE/ACM Trans. Netw.*, vol. 20, no. 3, pp. 662–676, Jun. 2012.
[48] N. Spring, R. Mahajan, D. Wetherall, and T. Anderson, "Measuring ISP topologies with rocketfuel," *IEEE/ACM Trans. Netw.*, vol. 12, no. 1, pp. 2–16, Feb. 2004.
[49] M. Roughan, "First order characterization of Internet traffic matrices," in *Proc. 55th Session Int. Statist. Inst.*, Sydney, NSW, Australia, Apr. 2005, pp. 1289–1292.

Long Chen (Member, IEEE) received the M.S. degree in systems engineering from the Huazhong University of Science and Technology, Wuhan, China, in 2011, where he is currently pursuing the Ph.D. degree in control science and engineering.

His current research interests include cyber-physical systems, reinforcement learning, multiagent systems, and complex networks.

Bin Hu (Member, IEEE) received the Ph.D. degree in control science and engineering from the Huazhong University of Science and Technology (HUST), Wuhan, China, in 2015.

She is currently an Associate Professor with the School of Artificial Intelligence and Automation, HUST. Her current research interests include hybrid dynamical systems, complex networks, computational neuroscience, and artificial intelligence.

Zhi-Hong Guan (Senior Member, IEEE) received the Ph.D. degree in automatic control theory and applications from the South China University of Technology, Guangzhou, China, in 1994.

In 1994, he was a Full Professor with the Jianghan Petroleum Institute, Jingzhou, China. Since 1997, he has been a Full Professor with the Huazhong University of Science and Technology, Wuhan, China, where he has been a Huazhong Leading Professor since 2011. His research interests include complex systems and complex networks, impulsive and hybrid control systems, networked control systems, multiagent systems, networked robotic systems, neural networks, and artificial intelligence.

Dr. Guan received the Natural Science Award (First Class) from the Ministry of Education of China in 2005 and the Natural Science Award (First Class) from the Hubei Province of China in 2014. He is a Highly Cited Researcher in Control and Systems Engineering (Elsevier).

Lian Zhao (Senior Member, IEEE) received the Ph.D. degree from the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON, Canada, in 2002.

She is currently a Professor with the Department of Electrical, Computer, and Biomedical Engineering, Ryerson University, Toronto, ON, Canada. Her research interests are in the areas of wireless communications, resource management, mobile edge computing, caching and communications, and vehicular ad hoc networks.

Dr. Zhao received the Best Land Transportation Paper Award from IEEE Vehicular Technology Society in 2016, the Top 15 Editor Award in 2016 for IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, the Best Paper Award from the 2013 International Conference on Wireless Communications and Signal Processing (WCSP), and the Canada Foundation for Innovation (CFI) New Opportunity Research Award in 2005. She served as the Co-Chair for the Wireless Communication Symposium, IEEE Globecom 2020 and IEEE ICC 2018, the Local Arrangement Co-Chair for IEEE VTC Fall 2017 and IEEE Infocom 2014, and the Co-Chair for the Communication Theory Symposium, IEEE Globecom 2013. She has been serving as an Editor for IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS, IEEE INTERNET OF THINGS JOURNAL, and IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY. She has been an IEEE Communication Society (ComSoc) Distinguished Lecturer (DL).

Xuemin (Sherman) Shen (Fellow, IEEE) received the Ph.D. degree in electrical engineering from Rutgers University, New Brunswick, NJ, USA, in 1990.

He is currently a University Professor with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON, Canada. His research focuses on network resource management, wireless network security, Internet of Things, 5G and beyond, and vehicular ad hoc and sensor networks.

Dr. Shen is a fellow of the Engineering Institute of Canada, the Canadian Academy of Engineering, and the Royal Society of Canada, a Foreign Member of the Chinese Academy of Engineering, and a Distinguished Lecturer of the IEEE Vehicular Technology Society and Communications Society. He received the R.A. Fessenden Award in 2019 from IEEE, Canada, the Award of Merit from the Federation of Chinese Canadian Professionals (Ontario) in 2019, the James Evans Avant Garde Award in 2018 from the IEEE Vehicular Technology Society, the Joseph LoCicero Award in 2015 and the Education Award in 2017 from the IEEE Communications Society, the Technical Recognition Award from the Wireless Communications Technical Committee in 2019 and the AHSN Technical Committee in 2013, the Excellent Graduate Supervision Award from the University of Waterloo in 2006, and the Premier's Research Excellence Award (PREA) from the Province of Ontario, Canada, in 2003. He served as the Technical Program Committee Chair/Co-Chair for IEEE Globecom 2016, IEEE Infocom 2014, IEEE VTC 2010 Fall, and IEEE Globecom 2007, and the Chair for the IEEE Communications Society Technical Committee on Wireless Communications. He is the President-Elect of the IEEE Communications Society. He was the Vice President for Technical and Educational Activities, the Vice President for Publications, a Member-at-Large on the Board of Governors, the Chair of the Distinguished Lecturer Selection Committee, and a member of the IEEE Fellow Selection Committee of the ComSoc. He served as the Editor-in-Chief for the IEEE INTERNET OF THINGS JOURNAL, *IEEE Network*, and *IET Communications*. He is a registered Professional Engineer of Ontario, Canada.