Pareto-Optimal Multi-Agent Cooperative Caching Relying on Multi-Policy Reinforcement Learning

Boyang Guo, Youjia Chen, *Member, IEEE*, Peng Cheng, *Member, IEEE*,
Ming Ding, *Senior Member, IEEE*, Jinsong Hu, *Member, IEEE*, Lajos Hanzo, *Fellow, IEEE*

***Abstract*—Given the popularity of flawless telepresence and the resultants explosive growth of wireless video applications, besides handling the traffic surge, satisfying the demanding user requirements for video qualities has become another important goal of network operators. Inspired by this, cooperative edge caching intrinsically amalgamated with scalable video coding is investigated. Explicitly, the concept of a Pareto-optimal semi-distributed multi-agent multi-policy deep reinforcement learning (SD-MAMP-DRL) algorithm is conceived for managing the cooperation of heterogeneous network nodes. To elaborate, a multi-policy reinforcement learning algorithm is proposed for finding the Pareto-optimal policies during the training phase, which balances the tele-traffic vs. the user experience trade-off. Then the optimal policy/solution can be activated during the execution phase by appropriately selecting the associated weighting coefficient according to the dynamically fluctuating network traffic load. Our experimental results show that the proposed SD-MAMP-DRL algorithm 1) achieves better performance than the benchmark algorithms; 2) obtains a near-complete Pareto-front in various scenarios and selects the optimal solution by adaptively adjusting the above-mentioned pair of objectives.**

***Index Terms*—Edge caching, multi-objective optimization, Pareto-front, scalable video coding, multi-agent reinforcement learning.**

I. Introduction

Edge caching has emerged as an efficient technique for tackling the escalation of wireless video traffic. By pushing popular contents near mobile users, it can dramatically reduce the transmission distance and hence the latency as well as the network cost. Conventionally, the major concern of edge caching is how to efficiently reuse the limited storage resources for enhancing the cache hit ratio. Recently, how to satisfy the heterogeneous user preferences for video qualities with limited storage resources has attracted substantial attention. In [1], maximizing the user's quality of experience (QoE) serves as the objective of the caching design.

This work was supported by the National Natural Science Foundation of China (Grant No. 62271150 and 62001116), the Engineering and Physical Sciences Research Council projects EP/W016605/1, EP/X01228X/1 and EP/Y026721/1, and the European Research Council's Advanced Fellow Grant QuantCom (Grant No. 789028).

Boyang Guo, Youjia Chen and Jinsong Hu are with Fujian Key Lab for Intelligent Processing and Wireless Transmission of Media Information, College of Physics and Information Engineering, Fuzhou University, 350108, China (e-mail: {201120067, youjia.chen, jinsong.hu}@fzu.edu.cn). Peng Cheng is with the Department of Computer Science and Information Technology, La Trobe University, Melbourne, VIC 3086, Australia (e-mail: peng.cheng@sydney.edu.au). Ming Ding is with Data61, CSIRO, NSW, 2015, Australia. (e-mail: ming.ding@data61.csiro.au). Lajos Hanzo is with the School of Electronics and Computer Science, University of Southampton, UK (e-mail: lh@ecs.soton.ac.uk).

Correspondence author: Youjia Chen.

As a further advance, the specifics of video coding techniques were also considered in the design of the caching strategy [2]. Specifically, the scalable video coding (SVC), mode of the H.264 standard is capable of dynamically reconfiguring the video qualities [3]. It encodes the original video into a basic layer and multiple enhancement layers that can be flexibly added and reduced for upgrading and reducing the video qualities. Such a design shows an intrinsic advantage in caching scenarios, where the storage resources are limited. To elaborate, SVC is introduced in the design of caching strategies for meeting personalized user requirements [4, 5], and also for the design of cooperative caching [6].

Although cooperative caching fundamentally enlarges the available storage resource for improving the edge caching performance, it increases the scheduling and design complexity [7]. The key issue of cooperative caching is to avoid repeated caching and content exchange among cooperative units. Hence the extraordinary decision-making capability of multi-agent reinforcement learning (MARL) makes it an excellent choice for cooperative cache design. However, a fully distributed structure is vulnerable to non-stationary environments and may even fail to converge due to the associated vulnerable information exchange [8]. By contrast, the centralized structure generally leads to high computational complexity, communication cost and decision delay. Hence the semi-distributed concept was introduced, which strikes an attractive performance vs. complexity trade-off [9].

When aiming for striking an attractive QoE vs. transmission cost trade-off, a multi-objective optimization problem (MOP) arises [10]. In this context, deep reinforcement learning (DRL) is capable of acquiring the optimal action by finding the maximum objective function value. Hence, when dealing with a MOP to be solved by DRL, transforming them into a single-objective one by scalarization is the commonly-used approach [11]. This kind of approach relies on a predetermined scalarization function during the training, which cannot be changed during the execution phase. This lack of flexibility makes the corresponding approaches less suitable for dynamic wireless networks.

Against this backdrop, we aim to reduce the wireless traffic and enhance the user's QoE simultaneously, and we investigate the benefits of cooperative caching design intrinsically amalgamated with SVC. A semi-distributed multi-agent multi-policy deep reinforcement learning (SD-MAMP-DRL) algorithm is proposed. As illustrated in Fig. 1, the main process consists of a training phase and an execution phase. During the training phase, multi-agents are jointly trained with the aid of an

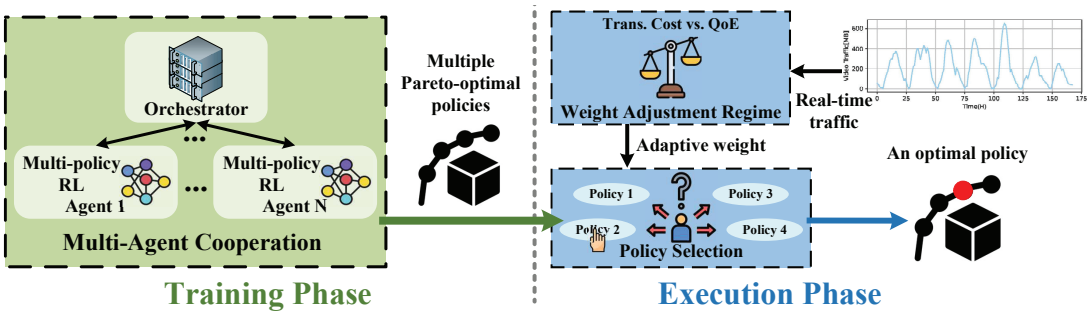

Fig. 1. The main process of proposed SD-MAMP-DRL.

TABLE I
RELATED LITERATURE USING REINFORCEMENT LEARNING IN COMPARISON TO OUR WORK

References	Heterogeneous agents	Distributed computing	Cooperation among multiple agents	Multi-objective problem	Pareto policies	Adaptive decisions to wireless environments
[12]	✓		✓			
[13]			✓			
[14]		✓				
[15, 16]		✓	✓			
[17]		✓	✓	✓		
[18, 19]				✓		
[20–22]				✓	✓	
Proposed	✓	✓	✓	✓	✓	✓

orchestrator. In contrast to the traditional RL that learns an optimal policy of every possible state, the proposed solution aims to develop multiple Pareto-optimal policies that balance the transmission cost and the users' QoE. Subsequently, in the execution phase, a weight to balance the two objectives is generated based on the real-time traffic load. Each agent then uses the weight to select the corresponding optimal policy from those Pareto-optimal policies.

The main contributions of this paper are three-fold,

- We introduce a semi-distributed multi-agent structure that leverages the benefits of edge servers while taking into account the heterogeneity of cooperative devices.
- A multi-policy DRL approach relying on maximizing the hypervolume of the Pareto-front is used to obtain the Pareto-optimal policies during the training, which strikes a trade-off between the user's QoE and transmission cost.
- To adapt to the wireless environment, we propose a traffic-based weight adjustment regime, relying on which of the most preferred policy can be dynamically selected from the Pareto-optimal policies during the execution phase.

The rest of the paper is organized as follows. Section II introduces the current state-of-the-art. Section III describes the system model and formulates the cooperative caching problem, while Section IV elaborates on the proposed SD-MAMP-DRL algorithm for the training phase. The execution phase, including the traffic-based weight adjustment regime and the corresponding policy selection process, is covered in Section V. Our simulation results are presented in Section VI, and we conclude with a summary of our findings in Section VII.

II. Related Work

A. Multi-Agent Reinforcement Learning

In recent years, DRL has been successfully used for solving decision-making problems, such as games, robotic tracking control, autonomous driving, as well as edge caching problems [12]. When the decision-making problem involves multiple intelligent cooperative or competitive individuals in a common environment, it may be modeled as a multi-agent reinforcement learning (MARL) problem.

A fully centralized architecture having a global view is capable of improving the overall caching performance, but suffers from excessive computational complexity when the network scale escalates [13]. In the spirit of distributed algorithms, a decentralized structure was developed, such as independent Q-learning (IQL), where each agent independently improves its strategy according to its state and reward [14]. To improve the cooperative performance, flawless communications of the agents are required in the distributed structure, which however imposes extra communication overhead [15].

Again, to strike a performance vs. complexity trade-off, semi-distributed structures have also received research attention recently. In [16], a remote server evaluated the actions of the agents and sent the evaluation results back to agents for model updates. In [17], a value decomposition network (VDN) was adopted, where the state-action value from each agent was accumulated in the edge server to calculate the gradient for neural network updates in each agent.

B. Multi-policy Reinforcement Learning

For MOPs that involve multiple potentially conflicting objectives, a commonly used approach is to transform these objectives into a single-objective optimization problem (SOP). Apart from the most popular scalarization method [18, 19], converting other objectives into constraints is also an attractive approach. In [23], the ϵ-constraint approach was adopted to achieve optimality. However, in time-varying scenarios, frequent retraining is required for refreshing the policy.

By contrast, Pareto-optimality reflects the trade-off among different objectives, which has multiple solutions, where none of the associated metrics can be improved in value without degrading some of the other metrics. To find the Pareto-optimal set of solutions in a single run, the non-dominated sorting genetic algorithm-II (NSAG-II) was adopted in [24].

Similarly, to obtain the Pareto solutions, the straightforward approach is that of transforming a MOP into multiple SOPs with multiple scalarization preferences, which however results in excessive computational cost [20, 21].

The authors of [20, 21] found that linear scalarization can only find the solution set in a convex hull, while nonlinear scalarization has no guarantee of convergence. To avoid the extra complexity of multiple training processes, the multi-policy concept was proposed in [22], which aims to learn the Pareto solutions in a single training process.

In Table I, we boldly contrast our novel contributions to the related literature of reinforcement learning techniques mentioned in Sections II-A and II-B.

III. System Model and Problem Formulation

A. Network Scenario

We consider a video caching system based on the architecture of edge-based next-generation cellular networks [25], which is shown in Fig. 2. Multiple small-cell and/or macro-cell base stations (BSs) equipped with mobile edge computing/caching units are connected and served by an edge server. The edge server generally consists of multiple management modules, which can also offer computing and caching functionality. Multiple edge servers are connected to a remote video server through the wireless core network.

We then embark on investigating a cooperative caching framework with the aid of the edge server. To reduce the transmission delay and cost of video delivery from the remote video server through the backbone and core network, the popular video files are pre-cached in the BSs and the caching module of the edge server. The edge server and its associated BSs compose a cooperative caching cluster, where the cached contents can be shared among the cluster units under the control of the edge server.

Overall, the cooperative caching cluster consists of the following network nodes having different functions.

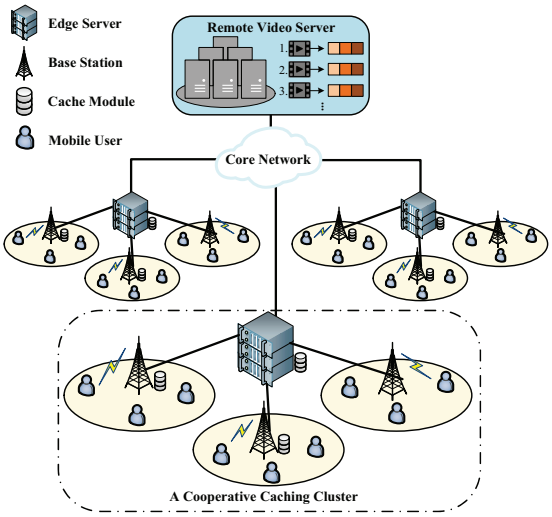

Fig. 2. An illustration of cooperative caching networks.

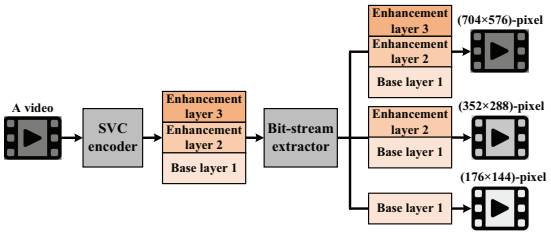

Fig. 3. An illustration of the scalable video coding.

- *Non-caching BSs*: Such BSs only work as basic access points of the cellular network and serve their associated users.
- *Caching-enabled BSs*: Such BSs work as wireless access points, but they are also equipped with extra storage for video caching.
- *Edge Server*: It serves as the controller which coordinates the content sharing among the associated BSs. It also has sufficient storage resources to cache popular video files.
- *Mobile Users*: Under the basic one-user-one-BS association criterion, we assume that each user can only request one video from its serving BS at a given time instance.

We assume that there are M BSs and a single edge server in a cooperative caching cluster, denoted by $\mathcal{M} = \{0, 1, 2, \ldots, M\}$, where the index 0 represents the edge server. Moreover, for the m-th BS, $\mathcal{U}_m, m \in \{1, \ldots, M\}$ denotes the set of associated mobile users.

B. Video Delivery in the Cooperative Caching Scheme

Based on the cooperative caching scheme, the video delivery involves four cases:

- *from the associated BS*: when the requested video is cached in the user's serving BS. Similar to [26], we define ω_1 (cost/Mbit) to represent the transmission cost of unit traffic from a BS to its associated user.
- *from the edge server*: when the requested video is cached in the edge server but not in the associated BS. The transmission cost of unit traffic is denoted by ω_2. As shown in Fig. 2, compared to the above case, the backhaul link between the edge server and the serving BS is involved in the video delivery, thus we have $\omega_2 > \omega_1$.
- *from cooperative BSs*: when the requested video is only cached in one of the cooperative BSs. The corresponding transmission cost of unit traffic is denoted by ω_3. Following the network topology shown in Fig. 2, two backhaul transmissions are involved, and hence we have $\omega_3 > \omega_2$.
- *from the remote video server*: when the requested video is not cached in any cooperative cache unit. The transmission cost of unit traffic denoted by ω_4, and obviously $\omega_4 > \omega_3 > \omega_2 > \omega_1$.

C. SVC-Based Cooperative Caching Scheme

To accommodate the users' requirements concerning various video qualities, and to fully exploit the caching resources, we incorporate the SVC technique into our cooperative caching scheme (SVC-CCS). The video stream is encoded with the

aid of the SVC technique into multiple layers, including a base layer that provides basic video quality and multiple enhancement layers that increase the video quality, as and when required and/or affordable [3].

For example, as illustrated in Fig. 3, SVC generates the base layer-1 and two enhancement layers (2 and 3). Base layer-1 offers a (176×144)-pixel Quarter Common Intermediate Format (QCIF) video, while the combination of base layer-1 and enhancement layer-2 offers a (352×288)-pixel CIF resolution. Moreover, all three layers may be combined to offer a (704×576)-pixel representation.

We further assume each video v in the library $\mathcal{V} = \{1, \ldots, V\}$ is encoded into L layers providing L various video qualities. Upon denoting the size of the l-th layer of the v-th video by b_{vl}, and the video quality provided to the u-th user by p_{uv}, the overall video size delivered can be expressed by

$$\sum_{l=1}^{p_{uv}} b_{vl}. \tag{1}$$

Furthermore, an SVC-encoded video layer is viewed as a caching object in SVC-CCS. We define $\delta_{mvl} = 1$ to indicate that the l-th layer of the v-th video is indeed cached in the m-th BS, and $\delta_{mvl} = 0$ otherwise. Additionally, $\delta_{0vl} = 1$ indicates that the l-th layer of the v-th video is cached in the edge server. It should be noted that, considering the disparity in video popularity and the importance of the base layer for a video in SVC, a video layer is allowed to be cached in multiple nodes within a caching cluster.

D. Performance and Optimization of SVC-CCS

We now embark on jointly optimizing two parameters, δ_{mvl} and p_{uv}, namely the caching decision and the user-specific video quality, while aiming for minimizing the transmission cost and maximizing the user's quality of experience (QoE).

1) Transmission Cost Reduction: Let us assume that the u-th user requests the v-th file, and it is served by the m-th BS at a video quality p_{uv}. Then, upon considering the four video delivery cases mentioned in Section III-B, the transmission cost can be calculated as

$$\sum_{l=1}^{p_{uv}} \Big[\delta_{mvl}\omega_1 + \Big(1-\delta_{mvl}\Big)\Big(\delta_{0vl}\omega_2 + \big(1-\delta_{0vl}\big)\big(\delta_{m'vl}\omega_3 + (1-\delta_{m'vl})\omega_4\big)\Big)\Big] b_{vl}, \tag{2}$$

where $\sum_{l=1}^{p_{uv}}[\cdot]$ represents the cumulative layer transmissions for a given video quality p_{uv}, and m' denotes the index of a cooperative BS in the same cooperative cluster. To elaborate a little further,

1) the first item $\delta_{mvl}\omega_1 b_{vl}$ of (2) represents the transmission cost when the video layer is delivered directly from the associated BS;
2) the second item $(1-\delta_{mvl})\delta_{0vl}\omega_2 b_{vl}$ represents the cost when the layer is delivered from the edge server;
3) the third item $(1-\delta_{mvl})(1-\delta_{0vl})\delta_{m'vl}\omega_3 b_{vl}$ represents the cost of delivery when the layer is delivered from a cooperative BS;
4) $(1-\delta_{mvl})(1-\delta_{0vl})(1-\delta_{m'vl})\omega_4 b_{vl}$ represents the cost of delivery from the remote video server.

Compared to a scenario with no caching, where all the video layers are delivered from the remote video server at a transmission cost of $\sum_{l=1}^{p_{uv}} \omega_4 b_{vl}$, the average transmission cost reduction ratio, o_1, resulting from cooperative caching for all the users can be expressed as

$$o_1 = \frac{1}{U} \sum_{\substack{m \in \mathcal{M} \\ m' \neq m, m' \in \mathcal{M}}} \sum_{u \in \mathcal{U}_m} \sum_{v \in \mathcal{D}_u} \Bigg[1 - \frac{1}{\sum_{l=1}^{p_{uv}} \omega_4 b_{vl}} \sum_{l=1}^{p_{uv}} \Big[\delta_{mvl}\omega_1 + \Big(1-\delta_{mvl}\Big)\Big(\delta_{0vl}\omega_2 + \big(1-\delta_{0vl}\big)\big(\delta_{m'vl}\omega_3 + (1-\delta_{m'vl})\omega_4\big)\Big)\Big] b_{vl}\Bigg]. \tag{3}$$

Here, $\mathcal{D}_u$ denotes the video requested by the u-th user.

2) Quality of User Experience: In this paper, we use the bitrate of the received video to quantify the user's QoE on the video quality [27]. Upon relying on the definition of mean opinion score (MOS) [28], and considering the various user requirements, we calculate the user QoE as

$$o_2 = \frac{1}{U} \sum_{\substack{m \in \mathcal{M} \\ m \neq 0, m' \in \mathcal{M}}} \sum_{u \in \mathcal{U}_m} \sum_{v \in \mathcal{D}_u} \min\left\{ \exp\left(-\beta_1 \Big(\frac{\sum_{l=1}^{p_{uv}} b_{vl}}{\sum_{l=1}^{\hat{p}_{uv}} b_{vl}} \Big)^{-\beta_2} + \beta_1 \right), 1 \right\}. \tag{4}$$

Here, $\hat{p}_{uv}$ denotes the video quality that user-u requests for video-v; $\frac{\sum_{l=1}^{p_{uv}} b_{vl}}{\sum_{l=1}^{\hat{p}_{uv}} b_{vl}}$ represents the ratio of obtained video quality to the user's requested video quality; β_1 and β_2 are the coefficients representing the overall quality reduction and quality gain vs. bitrate; $\min\{\cdot, 1\}$ limits the QoE within the range of $(0, 1]$, where the maximum QoE, $o_2 = 1$, means that the user's requested video quality is satisfied.

3) Problem Formulation: To simultaneously minimize the video transmission cost and maximize the video quality, we formulate the following multi-objective optimization problem:

$$\max_{\{p_{uv}, \delta_{mvl}, \delta_{0vl}\}} \mathbf{O} \triangleq [o_1, o_2], \tag{5}$$

$$\text{s.t.} \quad \sum_{v \in \mathcal{V}} \sum_{l \in \mathcal{L}} \delta_{mvl} b_{vl} \leq g_m, \tag{5a}$$

$$\sum_{v \in \mathcal{V}} \sum_{l \in \mathcal{L}} \delta_{0vl} b_{vl} \leq g_0, \tag{5b}$$

$$\delta_{mvl} \in \{0, 1\}, m \in \mathcal{M}, v \in \mathcal{V}, l \in \mathcal{L}, \tag{5c}$$

$$\delta_{0vl} \in \{0, 1\}, v \in \mathcal{V}, l \in \mathcal{L}, \tag{5d}$$

$$1 \leq p_{uv} \leq L, \tag{5e}$$

where g_m and g_0 represent the cache capacity in BS-m and the edge server, respectively. Note that, for non-caching BSs, the cache capacity equals 0. The constraint (5e) represents the range of video qualities.

In essence, the formulation in (5) represents a multi-objective nonlinear integer programming problem, which is typically NP-hard. Although some traditional techniques, such as the branch-and-bound and heuristic algorithms can solve

this problem, they will impose excessive computational complexity and are inefficient in nonlinear scenarios [29, 30]. To address this problem, we resort to DRL, which constitutes a powerful technique for handling a dynamic decision-making problem.

Explicitly, a multi-agent DRL-based approach is proposed in Section IV, where a Pareto-optimization-based multi-policy DRL algorithm is designed to deal with the multi-objective optimization problem considered and a semi-distributed multi-agent framework is utilized for supporting cooperation among BSs and the edge server.

IV. Semi-distributed Multi-agent Multi-policy Deep Reinforcement Learning

In this section, we introduce the proposed semi-distributed multi-agent multi-policy deep reinforcement learning (SD-MAMP-DRL) technique for a cooperative caching cluster. Firstly, in contrast to the traditional DRL aiming for finding the optimal action in each state, the goal of our algorithm termed as multi-policy DRL is to obtain the Pareto solutions, i.e. the set of actions in each state that achieves the Pareto-front. Secondly, under the umbrella of semi-distributed multi-agent DRL, we define the state and action spaces for different kinds of agents and introduce the detailed double deep Q-network (DDQN) [17] and QMIX [9] algorithms adopted in the agents and the orchestrator, respectively. Furthermore, we highlight the detailed information exchange between them.

A. Semi-distributed MADRL Framework

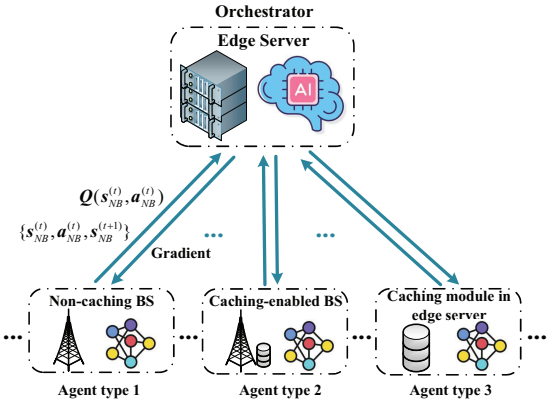

Fig. 4. Multi-agent caching cooperative framework.

As illustrated in Fig. 4, the proposed SD-MADRL framework consists of multiple types of agents and an orchestrator. In our scenario, the agents include non-caching BSs, caching-enabled BSs, and the caching module of the edge server, where the edge server acts as the orchestrator. Each agent deploys an independent DDQN to make a decision according to its local observations. The orchestrator deploys a mixing network, which leverages the global information collected from the agents for adjusting the DDQN parameters of each agent.

B. State and Action Spaces for Different Agents

By taking into account the different types of agents, we design different state and action spaces as follows.

1) State Space: The state of an agent depends on its local observations of the environment [31].

- For the non-caching BS-m, the state is represented by the video requests received from its associated mobile users, and the corresponding requirements on video quality, denoted by $\boldsymbol{s}_{\text{NB}}^{(t)} \triangleq \left\{\mathcal{D}_u^{(t)}, \hat{p}_{uv}^{(t)}, \forall u \in \mathcal{U}_m, v \in \mathcal{D}_u^{(t)}\right\}$, where (t) denotes the t-th time episode.
- For the caching-enabled BS-m, the state includes two parts: 1) the video requests from its associated users and the corresponding requirements on video quality, and 2) the video layers cached in its storage, denoted by $\boldsymbol{s}_{\text{CB}}^{(t)} \triangleq \left\{\mathcal{D}_u^{(t)}, \hat{p}_{uv}^{(t)}, \forall u \in \mathcal{U}_m, v \in \mathcal{D}_u^{(t)}; \delta_{mvl}^{(t-1)}, \forall v, \forall l\right\}$.
- For the caching module of the edge server, its state only involves the video layers cached in its storage, denoted by $\boldsymbol{s}_{\text{EC}}^{(t)} \triangleq \left\{\delta_{0vl}^{(t-1)}, \forall v, \forall l\right\}$.

2) Action Space: The action of an agent is based on its observation and experience [31].

- For the non-caching BS-m, its action only involves the video quality provided for its associated users, denoted by $\boldsymbol{a}_{\text{NB}}^{(t)} \triangleq \left\{p_{uv}^{(t)}, \forall u \in \mathcal{U}_m, v \in \mathcal{D}_u^{(t)}\right\}$.
- For the caching-enabled BS-m, its action includes both the video quality provided for its users and the video layers to be cached at the next time episode. Therefore, we have $\boldsymbol{a}_{\text{CB}}^{(t)} \triangleq \left\{p_{uv}^{(t)}, \forall u \in \mathcal{U}_m, v \in \mathcal{D}_u^{(t)}; \delta_{mvl}^{(t)}, \forall v, \forall l\right\}$.
- For the caching module of the edge server, its action only involves the caching decision, hence its definition is $\boldsymbol{a}_{\text{EC}}^{(t)} \triangleq \left\{\delta_{0vl}^{(t)}, \forall v, \forall l\right\}$.

C. Procedures in Each Agent

In (5), there are two optimization objectives, therefore we harness a pair of DDQNs in each agent [32], where each DDQN consists of an evaluation network and a target network. It is worth noting that the proposed algorithm can also be extended to address a multi-objective optimization problem by increasing the number of DDQNs. Overall, for different types of agents, their algorithmic structures are similar, although their detailed states and actions may be different.

For the i-th agent, $i \in \mathcal{I} \triangleq \{0, 1, \cdots, M\}$, given the state s_i and the action a_i, in the evaluation networks having the parameters $\theta_{i,1}^{(t)}$ for Objective-1 and parameters $\theta_{i,2}^{(t)}$ for Objective-2, the Q-value vector, $\mathbf{Q}_i^{(t)}$, can be calculated as

$$\begin{aligned}\mathbf{Q}_i^{(t)} &= [q(s_i^{(t)}, a_i^{(t)}; \theta_{i,1}^{(t)}), q(s_i^{(t)}, a_i^{(t)}; \theta_{i,2}^{(t)})] \\ &\triangleq q(s_i^{(t)}, a_i^{(t)}; \boldsymbol{\Theta}_i^{(t)}),\end{aligned} \tag{6}$$

where $q(s_i^{(t)}, a_i^{(t)}; \boldsymbol{\Theta}_i^{(t)})$ is the state-action value function denoting the expecting value of actions, namely Q-value vector, and $\boldsymbol{\Theta}_i^{(t)} \triangleq [\theta_{i,1}^{(t)}, \theta_{i,2}^{(t)}]$.

Definition 1. *(State-action value function) In RL, the state-action value function is defined as the expected return $G^{(t)}$ starting from state s, which can be expressed as*

$$q_\pi(s, a) = \mathbb{E}_\pi\left[G^{(t)} | s^{(t)} = s, a^{(t)} = a\right], \tag{7}$$

where π is the policy mapping the observed state s to a probability distribution over all possible actions.

While in DRL, the deep neural network is considered a function approximator to estimate the state-action value function, namely the Q-value function in DDQN, which can be expressed as

$$q(s, a; \theta), \tag{8}$$

where θ is the parameter of the neural network.

The action for the next state is then selected by

$$a_i^{*(t+1)} = \underset{a_i^{(t+1)}}{\operatorname{argmax}} \mathbf{ActSel}\big(q(s_i^{(t+1)}, a_i^{(t+1)}; \mathbf{\Theta}_i^{(t)})\big), \tag{9}$$

where the action selection scheme is designed and specified in Section IV-E, based on the characteristics of the Pareto-front constructed by the above pair of objectives.

Meanwhile, in the target networks having the parameters $\overline{\mathbf{\Theta}}_i^{(t)}$, which includes $\overline{\theta}_{i,1}^{(t)}$ for Objective-1 and parameter $\overline{\theta}_{i,2}^{(t)}$ for Objective-2, the Q-value vector with chosen $\boldsymbol{a}_i^{*(t+1)}$ is calculated by

$$\overline{\mathbf{Q}}_i^{(t)} = q\left(s_i^{(t+1)}, a_i^{*(t+1)}; \overline{\mathbf{\Theta}}_i^{(t)}\right). \tag{10}$$

Then, the messages consisting of $\{s_i^{(t)}, a_i^{(t)}, s_i^{(t+1)}\}$ and $\{\mathbf{Q}_i^{(t)}, \overline{\mathbf{Q}}_i^{(t)}\}$ are sent from the i-th agent to the orchestrator. The i-th agent then waits for the gradient vector to be returned from the orchestrator, while the goal of updating the network parameters $\mathbf{\Theta}_i$ of the two evaluation networks by

$$\mathbf{\Theta}_i^{(t+1)} = \mathbf{\Theta}_i^{(t)} + \zeta \nabla \mathbf{y}^{(t)}, \tag{11}$$

where ζ denotes the learning rate. The parameters $\overline{\mathbf{\Theta}}_i$ of the target network are periodically updated by copying $\mathbf{\Theta}_i$.

D. Procedures in the Orchestrator

1) Calculation of the Global Reward: In the t-th training time episode, the orchestrator receives the states and the actions of each agent. Here, the joint state and joint action are defined as

$$\mathbf{S}^{(t)} = \{s_i^{(t)}, \forall i \in \mathcal{I}\}. \tag{12}$$

$$\mathbf{A}^{(t)} = \{a_i^{(t)}, \forall i \in \mathcal{I}\}. \tag{13}$$

Then, the corresponding reward vector can be calculated using (2) and (3), which are defined as

$$\mathbf{r}^{(t)} = \mathbf{O}^{(t)} = [o_1^{(t)}, o_2^{(t)}]. \tag{14}$$

2) Calculation of the Joint Q-value: As shown in Fig. 5, the mixing network in the orchestrator uses the Q-values of agents and the joint state as input to calculate the joint Q-value. In more detail, the two-layer hyper-networks generate the weights $\{\mu_1^{(t)}, \mu_2^{(t)}\}$ and biases $\{\eta_1^{(t)}, \eta_2^{(t)}\}$.

Given the received set $\{\mathbf{Q}_i^{(t)}, \forall i\}$, the joint Q-value $\mathbf{Q}_{\text{tot}}^{(t)}$ can be calculated as

$$\mathbf{Q}_{\text{tot}}^{(t)} = \frac{1}{|\mathcal{I}|} \sum_{i \in \mathcal{I}} \mu_2^{(t)} \cdot \text{elu}(\mathbf{Q}_i^{(t)} \cdot \mu_1^{(t)} + \eta_1^{(t)}) + \eta_2^{(t)}, \tag{15}$$

where $\text{elu}(\cdot)$ is the active function. Similarly, with the received set $\{\overline{\mathbf{Q}}_i^{(t)}, \forall i\}$, $\overline{\mathbf{Q}}_{\text{tot}}^{(t)}$ can be obtained.

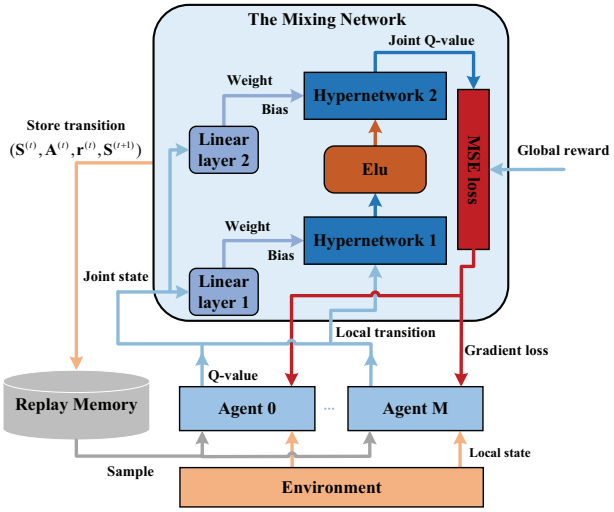

Fig. 5. The detail of QMIX deployed in the orchestrator.

3) Gradient Return to Each Agent: Upon combining the joint Q-value vector $\mathbf{Q}_{\text{tot}}^{(t)} = [Q_{\text{tot},1}^{(t)}, Q_{\text{tot},2}^{(t)}]$ obtained, as well as $\overline{\mathbf{Q}}_{\text{tot}}^{(t)} = [\overline{Q}_{\text{tot},1}^{(t)}, \overline{Q}_{\text{tot},2}^{(t)}]$, and the global reward $\mathbf{r}^{(t)}$, we can obtain the loss vector $\mathbf{y}^{(t)} = [y_1^{(t)}, y_2^{(t)}]$ and its corresponding gradient vector as follows

$$\mathbf{y}^{(t)} = \left[\mathbf{r}^{(t)} + \gamma \overline{\mathbf{Q}}_{\text{tot}}^{(t)} - \mathbf{Q}_{\text{tot}}^{(t)}\right]^2, \tag{16}$$

where $\gamma \in (0, 1)$ denotes the discount factor, and the squaring operation is applied for each element of the vector. Then, the gradient vector can be formulated as

$$\nabla \mathbf{y}^{(t)} = \left[\frac{\partial y_1^{(t)}}{\partial Q_{\text{tot},1}^{(t)}}, \frac{\partial y_2^{(t)}}{\partial Q_{\text{tot},2}^{(t)}}\right], \tag{17}$$

where ∇ denotes the gradient operator on each element of the vector.

The message consisting of $\{\nabla \mathbf{y}^{(t)}\}$ will be sent to each agent for updating the evaluation networks, as shown in (11). Given a sufficient number of training episodes, the losses of the DDQNs in each agent would converge, i.e. be capable of carrying out the optimal action maximizing the global reward.

The detailed procedure of the proposed SD-MAMP-DRL is formulated in Algorithm 1.

E. Hypervolume-based Action Selection Scheme

In this subsection, we outline the design of an action selection scheme for the agent. This scheme allows the agent to obtain the optimal action as indicated in line 12 of Algorithm 1, namely function **ActSel** in (9). We then present the specific formulation, represented by (20), summarized in Algorithm 2.

In the traditional DDQN algorithm of single-objective optimization problems, the action is selected according to the maximum Q-value, formulated as

$$a_i^{*(t+1)} = \underset{a_i^{(t+1)}}{\operatorname{argmax}}\, q(s_i^{(t+1)}, a_i^{(t+1)}; \theta_i^{(t)}).$$

However, when dealing with a multi-objective optimization problem, the Q-value obtained is a vector, namely $\mathbf{Q}$ in (7).

Algorithm 1 SD-MAMP-DRL

Initialization:
  Initialize the replay memory $\mathcal{Z}$.
  Initialize all parameters for evaluation networks and target networks of all agents and the mixing network.
  Initialize the learning rate ζ, and explore probability ψ.
for Step $t = 1, ..., T$ **do**
  Procedures in Each Agent:
  for Agent i **do**
    Update $\mathbf{\Theta}_i^{(t+1)} = \mathbf{\Theta}_i^{(t)} + \zeta \nabla \mathbf{y}^{(t)}$.
    Every period time, $\overline{\mathbf{\Theta}}_i = \mathbf{\Theta}_i$.
    Observe $s_i^{(t)}$ from environment.
    Choose $a_i^{(t)}$ as:
    $a_i^{(t)} = \begin{cases} \text{randomly action, with probability } \psi. \\ \text{(9) (c.f. Algorithm 2), with probability } 1-\psi. \end{cases}$
    Execute $a_i^{(t)}$ and receive $s_i^{(t+1)}$.
    Obtain $\mathbf{Q}_i^{(t)}, \overline{\mathbf{Q}}_i^{(t)}$ according to (6), (10).
    Send $s_i^{(t)}, a_i^{(t)}, s_i^{(t+1)}, \mathbf{Q}_i^{(t)}, \overline{\mathbf{Q}}_i^{(t)}$ to the orchestrator.
  end for
  Procedures in the Orchestrator:
  Construct $\mathbf{S}^{(t)}$, $\mathbf{A}^{(t)}$, and $\mathbf{S}^{(t+1)}$.
  Calculate $\mathbf{r}^{(t)}$.
  Store transition $(\mathbf{S}^{(t)}, \mathbf{A}^{(t)}, \mathbf{r}^{(t)}, \mathbf{S}^{(t+1)})$ to $\mathcal{Z}$.
  Obtain the $\mathbf{Q}_{\text{tot}}$, $\overline{\mathbf{Q}}_{\text{tot}}$ according to (15).
  Calculate $\mathbf{y}$ according to (16).
  Perform a gradient descent step according to (17).
  Send the gradient to each agent.
  Update evaluation networks in the orchestrator.
  Reset target networks as evaluation networks every period time correspondingly.
end for
Execution Phase: See Section V for detail.
return result

Hence, in the action selection, the traditional 'maximum' criterion cannot be applied here. In this case, we design the following action selection scheme based on the hypervolume of the multi-objective Pareto-optimal solutions.

1) Pareto-Front: For a pair of Q-value vectors, $\mathbf{Q}$ dominates $\mathbf{Q}'$ denoted by $\mathbf{Q} \succ \mathbf{Q}'$, which means that $\mathbf{Q}$ is at least as good as $\mathbf{Q}'$ for all the elements, and strictly better than $\mathbf{Q}'$ in at least one element. And $\mathbf{Q}' \not\succ \mathbf{Q}$ represents that $\mathbf{Q}$ is non-dominated by $\mathbf{Q}'$.

Given a set $\mathcal{Q}$ containing all Q-value vectors, the Pareto-front is defined as follows.

$$\mathcal{Q}^* \triangleq \{\mathbf{Q} \in \mathcal{Q} : \mathbf{Q}' \not\succ \mathbf{Q}, \forall \mathbf{Q}' \in \mathcal{Q}\}. \tag{18}$$

It indicates that the Pareto-front $\mathcal{Q}^*$ is a set consisting of all non-dominated Q-value vectors.

2) Hypervolume Measurement for the Pareto-Front: The goal of the proposed SD-MADRL algorithm is to obtain all the efficient (optimal) actions constituting the Pareto-front during the training phase. To obtain an accurate Pareto-front, we design the action selection scheme based on its hypervolume.

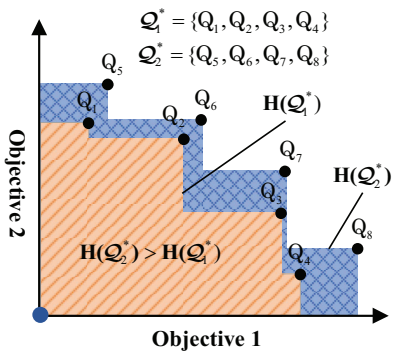

Fig. 6. The hypervolume of the Pareto-front $\mathcal{Q}_1^*$ and $\mathcal{Q}_2^*$.

Definition 2. *(Hypervolume) Explicitly, the hypervolume of the Pareto-front represents the volume of objective subspace dominated by the non-dominated Q-value vectors and delimited from a reference point [33]. The hypervolume for the optimization problem with a pair of objectives is illustrated in Fig. 6. In our scenario, we choose the origin as the reference point, and the hypervolume of the set $\mathcal{Q}^*$ is defined as*

$$\mathbf{H_o}(\mathcal{Q}^*) \triangleq \Lambda\Big(\{\mathbf{Q}' \mid \exists \mathbf{Q} \in \mathcal{Q}^* : \mathbf{Q} \succ \mathbf{Q}' \textit{ and } \mathbf{Q}' > 0\}\Big), \tag{19}$$

where $\Lambda(\cdot)$ denotes the Lebesgue measure [34].

The hypervolume is the only known scalar indicator for the Pareto-front to be strictly monotonic [33]. That is, if an approximated Pareto-front $\mathcal{Q}_1^*$ dominates another front $\mathcal{Q}_2^*$, then we have $\mathbf{H_o}(\mathcal{Q}_1^*) > \mathbf{H_o}(\mathcal{Q}_2^*)$. Hence, the proposed action selection scheme chooses the optimal action, which contributes to the maximum hypervolume.

3) Proposed Action Selection Scheme: For the i-th agent, the Pareto Q-value vector obtained is stored in the Pareto-front $\mathcal{Q}_i^*$ during the training phase. In the t-th episode, the contribution of each action to $\mathcal{Q}_i^{*(t)}$, i.e. the hypervolume, is calculated, and the action associated with the maximum hypervolume is selected (c.f. (9)). That is,

$$\begin{aligned} a_i^{*(t+1)} &= \underset{a_i^{(t+1)}}{\operatorname{argmax}} \textbf{ActSel}\big(q(s_i^{(t+1)}, a_i^{(t+1)}; \mathbf{\Theta}_i^{(t)})\big), \\ &= \underset{a_i^{(t+1)}}{\operatorname{argmax}} \mathbf{H_o}\left(\mathbf{F}\left(\mathcal{Q}_i^{*(t)} \bigcup q\big(s_i^{(t+1)}, a_i^{(t+1)}; \mathbf{\Theta}_i^{(t)}\big)\right)\right), \end{aligned} \tag{20}$$

where the function $\mathbf{F}(\cdot)$ represents the operation of selecting the Pareto-front from a vector set.

After selecting the optimal action, the Pareto-front $\mathcal{Q}_i^*$ will be updated simultaneously. That is,

$$\mathcal{Q}_i^{*(t+1)} = \mathbf{F}\left(\mathcal{Q}_i^{*(t)} \bigcup q\big(s_i^{(t+1)}, a_i^{*(t+1)}; \mathbf{\Theta}_i^{(t)}\big)\right). \tag{21}$$

The detailed workflow of the proposed action selection scheme is specified in Algorithm 2.

V. Policy Selection and Weight Adjustment Regime

Following the workflow of multi-objective optimization, in this section, we first describe the policy selection which maps

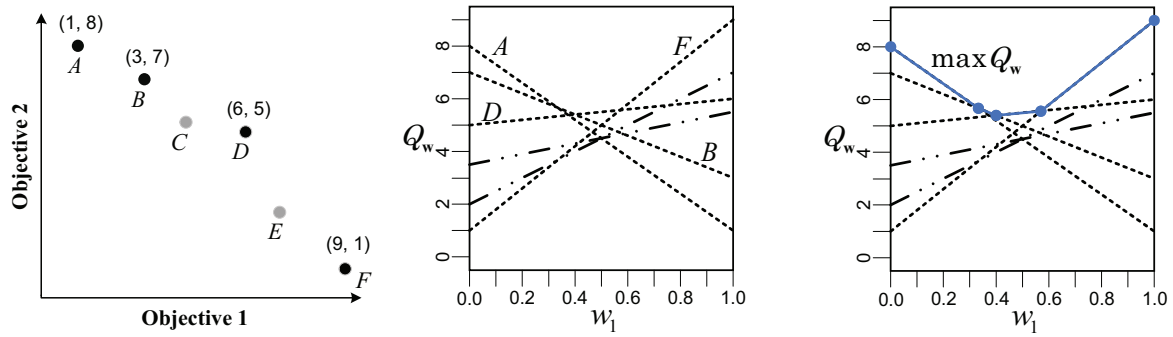

Fig. 7. An example of convex coverage set. (a) Q-value vectors in the Pareto-front; (b) The corresponding linear-scalarized Q-value $Q_{\mathbf{w}}$; (c) The piece-wise function $\max Q_{\mathbf{w}}$ vs. $\mathbf{w}_1$.

Algorithm 2 Proposed Action Selection Scheme Based-on Hypervolume

Input: $s^{(t)}, \boldsymbol{\mathcal{Q}}_s^{*(t-1)}, \boldsymbol{\Theta}^{(t)}, \mathcal{H} = \emptyset$
for each $a^{(t)}$ under the state **do**
  Calculate the Q-value vector: $\mathbf{Q}^{(t)}{=}q\big(s^{(t)}, a^{(t)}; \boldsymbol{\Theta}^{(t)}\big)$.
  Obtain $\boldsymbol{\mathcal{Q}}^{(t)} = \boldsymbol{\mathcal{Q}}_s^{*(t-1)} \bigcup \mathbf{Q}^{(t)}$.
  Delete the dominated vectors in the set $\boldsymbol{\mathcal{Q}}^{(t)}$.
  Calculate the hypervolume $hv = \mathbf{H_o}(\boldsymbol{\mathcal{Q}}^{(t)})$.
  Store hv to $\mathcal{H}$
  Clear $\boldsymbol{\mathcal{Q}}^{(t)}$.
end for
Obtain $a^{*(t)} = \text{argmax}_{a^{(t)}}\, \mathcal{H}$
Obtain $\boldsymbol{\mathcal{Q}}_s^{*(t)} = q\big(s^{(t)}, a^{*(t)}; \boldsymbol{\Theta}^{(t)}\big) \bigcup \boldsymbol{\mathcal{Q}}_s^{*(t-1)}$
Update $\boldsymbol{\mathcal{Q}}_s^{*(t)}$ by deleting the dominated vectors.
Output: $a^{*(t)}$.

the weights of two objective function parameters to the optimal solution. Then, we introduce a weight adjustment regime on the basis of real-time network traffic.

A. Policy Selection

1) Most Preferred Pareto Solution: Again, in the semi-distributed multi-agent structure, each agent carries out its own action. Relying on the training process introduced in Section IV, for each state, each agent can obtain the Pareto-front of the Q-value vectors and the corresponding action is referred to as the Pareto solution.

Without loss of generality, for the i-th agent associated with the state s, the Pareto-front $\boldsymbol{\mathcal{Q}}^*$ in (18) has been obtained. The corresponding Pareto solution set is

$$\boldsymbol{\mathcal{A}}^* \triangleq \Big\{ a : \mathbf{Q}(a) \in \boldsymbol{\mathcal{Q}}^* \Big\}, \tag{22}$$

where $\mathbf{Q}(a) \triangleq q\big(s, a; \boldsymbol{\Theta}\big)$ is the Q-value vector for action a with the trained DDQN.

In practice, this necessitates determining one solution from the Pareto solution set. Here, the solution determined is referred to as the most preferred Pareto solution, while this determination process is referred to as policy selection. Mathematically, we have

$$\pi : s \to a^*, a^* \in \boldsymbol{\mathcal{A}}^*. \tag{23}$$

2) Linear Scalarization and Convex Coverage Set: In this work, we use linear scalarization for choosing the most preferred Pareto solution in (23).

With an arbitrary weight vector $\mathbf{w}$, the linear-scalarized Q-value can be expressed as

$$Q_{\mathbf{w}}(a) = \mathbf{w}^{\mathrm{T}} \mathbf{Q}(a). \tag{24}$$

The policy selection in (23) represents the selection of the action which corresponds to the maximum linear-scalarized Q-value, where

$$\begin{aligned} a^* &= \underset{a \in \boldsymbol{\mathcal{A}}^*}{\text{argmax}}\, Q_{\mathbf{w}}(a) \\ &= \underset{a \in \boldsymbol{\mathcal{A}}^*}{\text{argmax}}\, \mathbf{w}^{\mathrm{T}} \mathbf{Q}(a). \end{aligned} \tag{25}$$

Note that the Pareto-front $\boldsymbol{\mathcal{Q}}^*$ obtained may contain several concave points, indicated by the grey points in Fig. 7(a). The corresponding Pareto solutions of these concave points never be the most preferred solution, regardless of the weight $\mathbf{w}$.

A convex coverage set (CCS) is defined [35], which is the subset of the Pareto solution set containing all the most preferred Pareto solutions associated with an arbitrary $\mathbf{w}$, defined as follows

$$\boldsymbol{\mathcal{A}}^c{=}\Big\{ a \in \boldsymbol{\mathcal{A}}^* : \exists \mathbf{w},\ \forall a' \in \boldsymbol{\mathcal{A}}^*,\ \mathbf{w}^{\mathrm{T}} \mathbf{Q}(a) \geq \mathbf{w}^{\mathrm{T}} \mathbf{Q}(a') \Big\}. \tag{26}$$

where we have $\boldsymbol{\mathcal{A}}^c \subseteq \boldsymbol{\mathcal{A}}^*$.

3) Policy Selection by Weight Segmentation: In our scenario of two-dimensional Q-value vectors, the function $Q_{\mathbf{w}}(w_1)$ can be represented by the lines seen in Fig. 7(b), where each line corresponds to a Q-value vector on the Pareto-front. Hence, the solid blue line in Fig. 7(c) represents the piece-wise function $\max Q_{\mathbf{w}}(w_1)$. Observe from this figure that the most preferred Pareto solution a^* is also represented by a piece-wise function corresponding to the solid blue line.

To identify the break points defining the segments (such as the point A in Fig. 7(a)) of this piece-wise function, we calculate the intersection point between every two lines. Given the pair of actions a and a' in the CCS, the corresponding Q-

Algorithm 3 CCS and the Break Point Acquisition

Input: $\boldsymbol{\mathcal{A}}^c = \emptyset$, the state s, $\boldsymbol{\mathcal{W}} = \{0, 1\}$.
Obtain the Pareto-front $\boldsymbol{\mathcal{Q}}^*$ and corresponding Pareto solution set $\boldsymbol{\mathcal{A}}^*$ from DDQN.
Sort the vectors in the set $\boldsymbol{\mathcal{Q}}^*$ according to the value of the second element, and obtain the sorted $\boldsymbol{\mathcal{Q}}^* = \{\mathbf{Q}_1, \mathbf{Q}_2, ..., \mathbf{Q}_N\}$ and the corresponding sorted $\boldsymbol{\mathcal{A}}^* = \{a_1, a_2, ..., a_N\}$.
for $n = 2, ..., N-1$ **do**
  for $m = 1, ..., n-1$ **do**
    Calculate $\mathbf{Q}_n - \mathbf{Q}_m = (I_1, I_2)$.
    for $k = n+1, ..., N$ **do**
      Calculate $\mathbf{Q}_k - \mathbf{Q}_m = (J_1, J_2)$.
      if $J_1 I_2 - J_2 I_1 > 0$ **then**
        Delete the $\mathbf{Q}_n$ in $\boldsymbol{\mathcal{Q}}^*$ and the a_n in $\boldsymbol{\mathcal{A}}^*$.
        End this loop and jump to $n+1$ turn.
      end if
    end for
  end for
end for
Obtain $\boldsymbol{\mathcal{A}}^c = \boldsymbol{\mathcal{A}}^*$.
for $j = 2, .., |\boldsymbol{\mathcal{Q}}^*|$ **do**
  Obtain $\mathbf{Q}_{j-1}$=$[Q_{j-1,1}, Q_{j-1,2}]$ from $\boldsymbol{\mathcal{Q}}^*$.
  Obtain $\mathbf{Q}_j$=$[Q_{j,1}, Q_{j,2}]$ from $\boldsymbol{\mathcal{Q}}^*$.
  Calculate $w_j = \frac{Q_{j-1,2} - Q_{j,2}}{(Q_{j-1,2} - Q_{j,2}) - (Q_{j-1,1} - Q_{j,1})}$.
  Store w_j in $\boldsymbol{\mathcal{W}}$.
end for
Reorder $\boldsymbol{\mathcal{W}}$.
Output: $\boldsymbol{\mathcal{A}}^c$ and $\boldsymbol{\mathcal{W}}$.

value vectors are $\mathbf{Q}(a) = [Q_1, Q_2]$ and $\mathbf{Q}(a') = [Q_1', Q_2']$. Upon assuming that $Q_2 > Q_2'$ (i.e. $Q_1 < Q_1'$), we have the most preferred solution as

$$a^* = \begin{cases} a, & \text{if } w_1 \leq \frac{Q_2 - Q_2'}{(Q_2 - Q_2') - (Q_1 - Q_1')}; \\ a', & \text{otherwise.} \end{cases} \tag{27}$$

By determining all the break points of w_1, we can obtain the break point set $\boldsymbol{\mathcal{W}} = \{w_1, \cdots, w_{|\boldsymbol{\mathcal{A}}^c|+1}\}$ having ascending elements. Then the most preferred solution can be readily decided. Given the weight vector $\mathbf{w} = [w_1, w_2]$, we have

$$a^* = a_j, \text{ if } w_j \leq w_1 \leq w_{j+1}. \tag{28}$$

The detailed workflow of obtaining the CCS $\boldsymbol{\mathcal{A}}^c$ and the break point set $\boldsymbol{\mathcal{W}}$ is presented in Algorithm 3.

B. Weight Adjustment Regime and Execution

In essence, the weight vector balances the preference of the two objectives: reducing the transmission cost and improving the quality of video service. As w_1 denotes the weight for the reduced traffic in the network, a larger $0 < w_1 < 1$ means that more transmission cost can be reduced by adopting the corresponding action, and hence a lower video quality is provided for the users since $w_2 = 1 - w_1$.

It is worth noting that wireless traffic tends to exhibit periodicity in the time domain [36], yielding increased traffic congestion probability at peak time. Hence, in this work, a dynamic transmission cost reduction weight w_1 is adopted by closely following the real-time network traffic. In detail, we consider a pair of criteria:

- The transmission cost reduction weight obeys $w_1 \to 1$ when the real-time traffic of the BS is approaching its maximum capacity.
- The transmission cost reduction weight follows $w_1 \to 0$ when the network traffic is quite low, aiming for improving the user's QoE.

Considering that the variation of QoE under small and large-scale traffic is slower than in medium-scale, we use a shifted *sigmoid* function to generate the weight of transmission cost reduction [37]. The proposed dynamic weight for the transmission cost reduction is as follows.

$$w_1(k) = \frac{1}{1 + \exp[-a(\frac{k}{\hat{k}} - \frac{1}{2}))]}, \tag{29}$$

where k denotes the current traffic and $\hat{k}$ is the threshold of traffic, while $a \geq 12$ is an experiential parameter controlling the slope of the curve [37]. Given the real-time w_1, in the execution phase, the most preferred solution can be directly obtained according to (27).

Above we have given the detailed decision process on the basis of real-time network traffic. In particular, the different weight segmentation and corresponding decisions are listed in Section VI-B.

VI. Simulation Result and Analysis

In this section, we first demonstrate the performance of our proposed SD-MAMP-DRL algorithm and show the impact of various video and caching parameters. Then, we characterize the Q-value of different solutions, i.e. the Pareto-front associated with various video and caching parameters, and show the weight segmentation, the executed actions and the rewards obtained for validating the proposed algorithm. Finally, we compare the performance of the algorithm conceived to that of the benchmark algorithms.

In our simulations, PyTorch 1.9 was used to execute our algorithm, and the RMSProp optimizer was employed for updating the parameters of DDQN and the mixing network. The discount factor γ and the learning rate ζ are set as 0.99 and 0.0005, respectively. The 1000 episodes are considered for the agent's training. Additionally, we set the bitrate of the five SVC layers used to 600 kbps, 2200 kbps, 4800 kbps, 6760 kbps, and 11200 kbps, respectively [38]. The number of users for the m-th BS in the cooperative caching cluster is $U_m = 3$, and the transmission cost coefficients $\omega_1, \omega_2, \omega_3, \omega_4$ in (3) are 0, 1, 3, and 5, respectively. The popularity of videos in the video library follows the Zipf distribution with the exponent of 0.3, 0.8, and 1.5.

In the following, we set the number of agents to $M = 3$ and 5, the number of videos to $V = 5$, 10, 15, 20, and 25, the number of video layers to $L = 3$, 4 and 5, while the storage size of the caching-enabled BS to range from 1 to 3 for comparison, respectively. Except otherwise stated, all users' requirement on video quality is set as $\hat{p}_{uv} = L$.

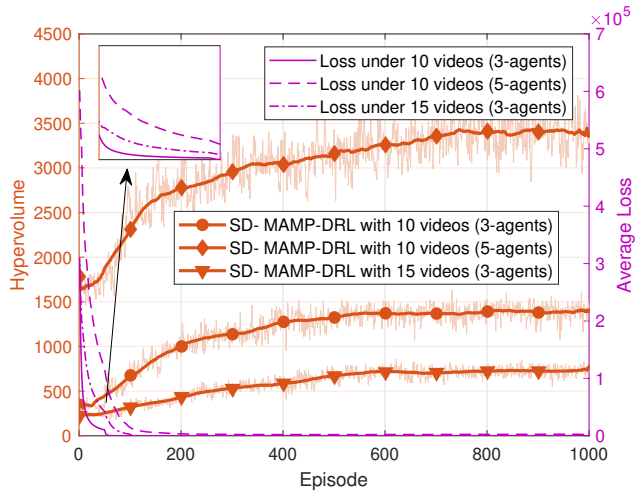

Fig. 8. The total hypervolume achieved by multiple agents during training vs. the episode index.

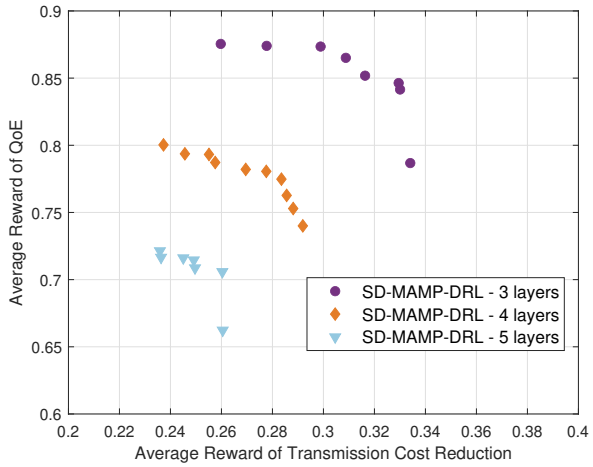

Fig. 9. Pareto-front of average reward for 3, 4 and 5 video layers.

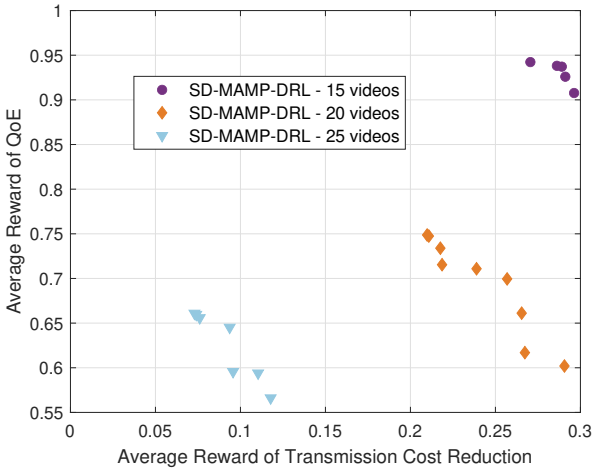

Fig. 10. Pareto-front of average reward for 15, 20, 25 video numbers.

A. *Performance of the Proposed SD-MAMP-DRL*

Again, in our proposed algorithm, each cooperative agent aims for maximizing the hypervolume of the Pareto-front of Q-value vectors. Hence, for characterizing the performance of this multi-agent algorithm, i.e. the overall performance of the cooperative caching cluster, in Fig. 8, we plot the accumulated hypervolume achieved vs. the episode index during the training. Specifically, we consider the cases, where the number of agents is $M = 3$ and 5, and the number of videos is $V = 10$ and 15.

Observe from Fig. 8 that: 1) first, the accumulated hypervolume increases with the number of training episodes, and then saturates in all cases. Again, the hypervolume represents the space dominated by the non-dominated Q-value vectors, and it is a monotonic characteristic of the Pareto-front. Hence, the increase of the hypervolume indicates that the Pareto-front obtained during the training is approaching the true Pareto-front, and the saturation of the hypervolume implies the convergence of the algorithm for all cooperative agents; 2) Given the three agents, the hypervolume associated with 15 videos is smaller than that with 10 videos. The reason behind this is that given the same storage size, a larger video library leads to degraded caching performance, i.e. lower transmission cost reduction and a lower QoE. This results in a reduced Q-value vector in the Pareto-front, which leads to a smaller hypervolume; 3) Given $V = 10$ videos, the hypervolume associated with three agents saturates at around 600 episodes, while that of five agents saturates at around 800 episodes. Furthermore, the loss value of five agents decreases slower than that of three agents. Hence, it may be surmised that increasing the number of agents in this multi-agent algorithm leads to a moderate degradation of the convergence speed.

To show the Pareto-front achieved by the proposed semi-distributed algorithm, in Fig. 9, we plot the non-dominated average reward of QoE vs. transmission cost reduction achieved by 3, 4 and 5 video layers. 1) Observe from Fig. 9 the trade-off between the two objectives, namely the average reward of QoE vs. that of transmission cost reduction. In all scenarios, a higher QoE requires a more minor transmission cost reduction. A higher QoE requires better video qualities and hence more video layers transmitted at higher bit rates, leading to increasing the network's wireless traffic. 2) Upon comparing the three Pareto-fronts, it is observed that the non-dominated average reward vectors obtained in the scenario of three video layers generally have higher average rewards in terms of both QoE and transmission cost reduction than those in the scenario of four or five video layers. More specifically, having fewer video layers results in a smaller requested video rate $\sum_{l=1}^{\hat{p}_{uv}} b_{vl}$ (c.f. the denominator in (4)), and hence a higher QoE can be satisfied with lower transmission cost. Furthermore, a smaller number of video layers leads to a smaller number of caching candidates. Hence, the same storage size can achieve better caching performance, i.e. resulting in higher transmission cost reduction.

In Fig. 10, we show the average reward of QoE vs. transmission cost reduction parameterized by the number of videos. This allows us to characterize the Pareto-front achieved after the training using various numbers of videos. We can observe

TABLE II
PARETO-OPTIMAL SOLUTIONS AND CORRESPONDING REWARD VECTOR

Weight Segment $w_j \sim w_{j+1}$	Video Quality Served p_{uv}	Trans. Cost Reduction o_1	User's QoE o_2
$0.00 \sim 0.33$	$\{3,2,3,3,2,2\}$	0.41	1.00
$0.33 \sim 0.45$	$\{2,2,3,3,2,2\}$	0.43	0.92
$0.45 \sim 0.65$	$\{3,2,3,3,2,1\}$	0.48	0.88
$0.65 \sim 0.85$	$\{2,1,2,3,2,2\}$	0.57	0.84
$0.85 \sim 0.88$	$\{2,1,2,3,2,1\}$	0.63	0.73
$0.88 \sim 0.90$	$\{2,1,2,3,1,1\}$	0.67	0.61
$0.90 \sim 0.91$	$\{1,1,1,3,2,1\}$	0.68	0.56
$0.91 \sim 0.95$	$\{1,1,1,3,1,1\}$	0.72	0.45
$0.95 \sim 1.00$	$\{1,1,1,2,1,1\}$	0.73	0.37

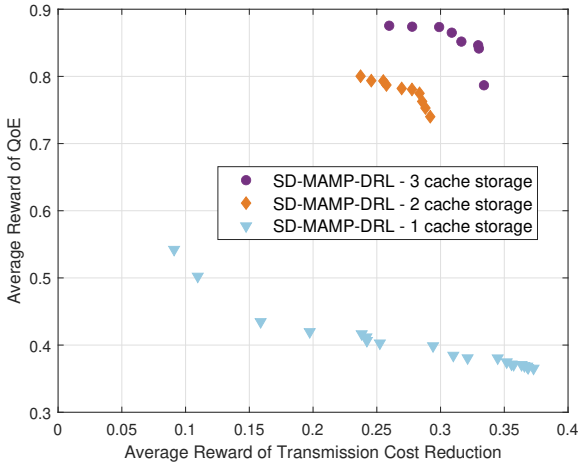

Fig. 11. Pareto-front of average reward for 3, 2 and 1 cache storage.

that the non-dominated average rewards in the scenario of 25 videos generally are smaller in terms of both QoE and transmission cost reduction than those of 15 or 20 videos. Specifically, the performance in the scenario of 15 videos is higher around 7.38% and 300% than those of 20 or 25 videos. The reason behind this is that a degraded caching performance is achieved due to having a larger number of caching candidates for given a storage size.

In Fig. 11, we enlarge the storage size of the caching-enabled BSs from 1 to 3 units and plot the Pareto-front of the average reward obtained after training. The performance improvement of increasing the storage size can be clearly observed, especially in terms of QoE. Specifically, when the cache size is increased from 1 to 2, the average reward of QoE increases from around 0.4 to nearly 0.8, although when the cache size is further increased from 2 to 3, the increase in QoE becomes moderate. Moreover, when the cache size is only 1, the improvement of user QoE brings a large degradation in transmission cost reduction.

B. Weight Segmentation for Decision

To show the most preferred actions corresponding to different weights and the resultant rewards, in Table II we list the optimal action p_{uv} of BSs, i.e. the video quality provided for user-u requesting video-v. As we can see, upon relying on the Pareto-front obtained after training, the region $w_1 \in (0,1)$ is unequally divided into 9 segments, and the most preferred action, p_{uv}, varies across segments.

Since the weight w_1 represents the importance of the transmission cost reduction, we can observe from the table that the video quality provided for the associated 6 users, namely p_{uv}, gradually decreases with the growth of w_1. Explicitly, the agent degrades the video quality provided for their users to reduce the video traffic, when the traffic load is high, i.e. w_1 in (29). On the other hand, we can see that the QoE decreases simultaneously due to the poorer video quality provided by the BS. These results verify the rationale of the optimal action decided in the execution phase and the trade-off between the transmission cost reduction and QoE in wireless caching.

Furthermore, considering that the users' requested video quality is $\{3,1,3,3,2,2\}$, when the weight for transmission cost reduction lies within the range of $(0, 0.33]$, all the users are served with their requested video quality, i.e. $p_{uv} = \{3,2,3,3,2,2\}$, resulting in a QoE reward of $o_2 = 1$. As ω_1 increases, the video quality experienced is progressively reduced. Explicitly, one user is not satisfied when $\omega_1 \in (0.33, 0.65]$; two users are not satisfied when $\omega_1 \in (0.65, 0.85]$; and as ω_1 continues to increase, the number of unsatisfied users grows accordingly.

C. Performance Comparison with Benchmarks

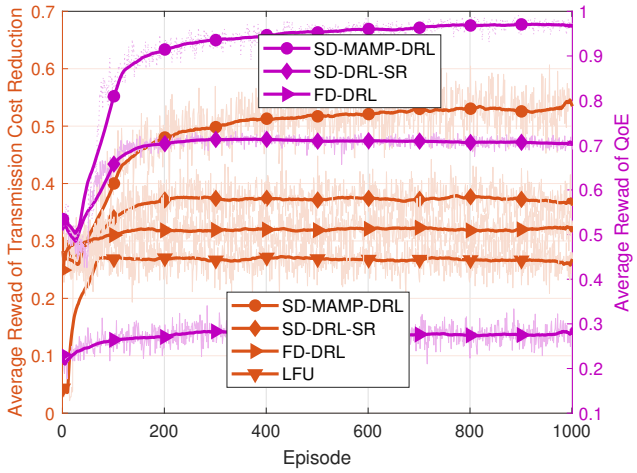

Fig. 12. Comparison for transmission cost reduction in different algorithms

To further validate the benefits of the semi-distributed multi-agent structure in the global optimization of the cooperative caching cluster, we also compared it to the following benchmark algorithms.

- **Fully distributed DRL (FD-DRL)** [8]: Each agent employs an independent DDQN without the orchestrator for learning the optimal strategy from its local state, and formulates its own caching and video service strategy.
- **Semi-Distributed DRL with global reward (SD-DRL-GR)** [39]: In contrast to FD-DRL, SD-DRL-GR facilitates state-information exchange among the agents

and relies on a centralized node for calculating the global/system reward, which is used in each agent for gradient update.

- **Least frequently used (LFU)** [40]: This is a traditional caching algorithm, where the video file requested least frequently is replaced in the storage.

In Fig. 12, it is observed that the performance of the traditional LFU algorithm is inferior to the other learning-based algorithms, and LFU cannot provide dynamic decisions concerning the video qualities. More importantly, among the three multi-agent DRL algorithms: 1) the FD-DRL achieves the poorest performance, which is due to the lack of global information or message exchange; 2) the proposed SD-MAMP-DRL algorithm outperforms SD-DRL-GR since the neural network of the orchestrator calculates not only the global rewards as the SD-DRL-GR but also the joint Q-value and the corresponding gradient, which is then returned to each agent for network updates.

VII. Conclusion

Pareto-optimal cooperative edge caching of popular videos relying on SVC techniques was investigated. With the goals of reducing transmission cost caused by video delivery and improving QoE, we proposed the novel SD-MAMP-DRL algorithm, in which the semi-distributed structure achieves global optimization of the cooperative caching cluster and the multi-policy DRL obtains multiple Pareto policies suitable for different network conditions. Our simulation results validated the convergence of the proposed algorithm and portrayed the Pareto-front in various scenarios, Our solution outperformed the benchmark algorithms. Moreover, the dynamic weight setting expedited the final phase of policy selection in an efficient manner.

References

[1] Z. Qu, B. Ye, B. Tang, S. Guo, S. Lu, and W. Zhuang, "Cooperative caching for multiple bitrate videos in small cell edges," *IEEE Transactions on Mobile Computing*, vol. 19, no. 2, pp. 288–299, 2020.

[2] B. Jedari, G. Premsankar, G. Illahi, M. Di Francesco, A. Mehrabi, and A. Ylä-Jääski, "Video caching, analytics, and delivery at the wireless edge: a survey and future directions," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 1, pp. 431–471, 2021.

[3] S. Rimac-Drlje, O. Nemcic, and M. Vranjes, "Scalable video coding extension of the H.264/AVC standard," in *Elmar, International Symposium*, 2009.

[4] J. Meng, H. Lu, and J. Liu, "Joint quality selection and caching for SVC video services in heterogeneous networks," in *IEEE WCNC 2020 - IEEE Wireless Communications and Networking Conference (WCNC)*, 2020, pp. 1–6.

[5] X. Zhang, T. Lv, Y. Ren, W. Ni, N. C. Beaulieu, and Y. J. Guo, "Economical caching for scalable videos in cache-enabled heterogeneous networks," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 7, pp. 1608–1621, 2019.

[6] K. Poularakis, G. Iosifidis, A. Argyriou, I. Koutsopoulos, and L. Tassiulas, "Caching and operator cooperation policies for layered video content delivery," in *IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications*, 2016, pp. 1–9.

[7] R. Wu, G. Tang, T. Chen, D. Guo, L. Luo, and W. Kang, "A profit-aware coalition game for cooperative content caching at the network edge," *IEEE Internet of Things Journal*, vol. 9, no. 2, pp. 1361–1373, 2022.

[8] J. Foerster, N. Nardelli, G. Farquhar, T. Afouras, P. H. Torr, P. Kohli, and S. Whiteson, "Stabilising experience replay for deep multi-agent reinforcement learning," in *International conference on machine learning*. PMLR, 2017, pp. 1146–1155.

[9] T. Rashid, M. Samvelyan, C. Schroeder, G. Farquhar, J. Foerster, and S. Whiteson, "QMIX: Monotonic value function factorisation for deep multi-agent reinforcement learning," in *International conference on machine learning*, 2018, pp. 4295–4304.

[10] D. Liu, J. Zhang, J. Cui, S.-X. Ng, R. G. Maunder, and L. Hanzo, "Deep-learning-aided packet routing in aeronautical Ad Hoc networks relying on real flight data: From single-objective to near-Pareto multiobjective optimization," *IEEE Internet of Things Journal*, vol. 9, no. 6, pp. 4598–4614, 2022.

[11] Parisi, Simone, Restelli, Marcello, Pirotta, and Matteo, "Multi-objective reinforcement learning through continuous pareto manifold approximation," *The Journal of Artificial Intelligence Research*, 2016.

[12] G. Qiao, S. Leng, S. Maharjan, Y. Zhang, and N. Ansari, "Deep reinforcement learning for cooperative content caching in vehicular edge computing and networks," *IEEE Internet of Things Journal*, vol. 7, no. 1, pp. 247–257, 2020.

[13] Z. Yang, Y. Liu, Y. Chen, and L. Jiao, "Learning automata based Q-learning for content placement in cooperative caching," *IEEE Transactions on Communications*, vol. 68, no. 6, pp. 3667–3680, 2020.

[14] T. Ming, "Multi-agent reinforcement learning: Independent versus cooperative agents," in *Proceedings of the Tenth International Conference on Machine Learning (ICML)*, 1993, pp. 330–337.

[15] S. Chen, Z. Yao, X. Jiang, J. Yang, and L. Hanzo, "Multi-agent deep reinforcement learning-based cooperative edge caching for ultra-dense next-generation networks," *IEEE Transactions on Communications*, vol. 69, no. 4, pp. 2441–2456, 2021.

[16] Y. Zhang, B. Feng, W. Quan, A. Tian, K. Sood, Y. Lin, and H. Zhang, "Cooperative edge caching: A multi-agent deep learning based approach," *IEEE Access*, vol. 8, pp. 133 212–133 224, 2020.

[17] Y. Chen, Y. Cai, H. Zheng, J. Hu, and J. Li, "Cooperative caching for scalable video coding using value-decomposed dimensional networks," *China Communications*, 2022.

[18] X. Hu, Y. Zhang, X. Liao, Z. Liu, W. Wang, and F. M. Ghannouchi, "Dynamic beam hopping method based on multi-objective deep reinforcement learning for next generation satellite broadband systems," *IEEE Transactions on Broadcasting*, vol. 66, no. 3, pp. 630–646, 2020.

[19] Y. Yu, J. Tang, J. Huang, X. Zhang, D. K. C. So, and K.-K. Wong, "Multi-objective optimization for UAV-assisted wireless powered IoT networks based on extended DDPG algorithm," *IEEE Transactions on Communications*, vol. 69, no. 9, pp. 6361–6374, 2021.

[20] K. V. Moffaert, M. M. Drugan, and A. Nowé, "Scalarized multi-objective reinforcement learning: Novel design techniques," in *2013 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL)*, 2013.

[21] G. Duflo, G. Danoy, E.-G. Talbi, and P. Bouvry, "Automating the design of efficient distributed behaviours for a swarm of UAVs," in *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, 2020, pp. 489–496.

[22] K. Van Moffaert and A. Nowé, "Multi-objective reinforcement learning using sets of pareto dominating policies," *J. Mach. Learn. Res.*, vol. 15, no. 1, p. 3483–3512, Jan. 2014.

[23] Z. Zhao, J. Shi, Z. Li, J. Si, P. Xiao, and R. Tafazolli, "Multiobjective resource allocation for mmWave MEC offloading under competition of communication and computing tasks," *IEEE Internet of Things Journal*, vol. 9, no. 11, pp. 8707–8719, 2022.

[24] A. Akbar, M. Ibrar, M. A. Jan, A. K. Bashir, and L. Wang, "SDN-enabled adaptive and reliable communication in IoT-fog environment using machine learning and multiobjective optimization," *IEEE Internet of Things Journal*, vol. 8, no. 5, pp. 3057–3065, 2021.

[25] A. Kabir, G. Rehman, S. M. Gilani, E. J. Kitindi, Z. Ul Abidin Jaffri, and K. M. Abbasi, "The role of caching in next generation cellular networks: A survey and research outlook," *Transactions on Emerging Telecommunications Technologies*, vol. 31, no. 2, p. e3702, 2020.

[26] X. Li, X. Wang, and V. Leung, "Weighted network traffic offloading in cache-enabled heterogeneous networks," in *IEEE International Conference on Communications*, 2016.

[27] B. Jedari, G. Premsankar, G. Illahi, M. Di Francesco, A. Mehrabi, and A. Ylä-Jääski, "Video caching, analytics, and delivery at the wireless edge: a survey and future directions," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 1, pp. 431–471, 2020.

[28] H. Hu, X. Zhu, Y. Wang, R. Pan, J. Zhu, and F. Bonomi, "Proxy-based multi-stream scalable video adaptation over wireless networks using subjective quality and rate models," *IEEE Transactions on Multimedia*, vol. 15, no. 7, pp. 1638–1652, 2013.

[29] M. Qi, M. Wang, and Z.-J. Shen, "Smart feasibility pump: Reinforcement learning for (mixed) integer programming," *arXiv preprint arXiv:2102.09663*, 2021.

[30] Y. Sun, M. Peng, and S. Mao, "Deep reinforcement learning-based mode selection and resource management for green fog radio access networks," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 1960–1971, 2018.

[31] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 26–38, 2017.

[32] L. Xi, L. Yu, Y. Xu, S. Wang, and X. Chen, "A novel multi-agent DDQN-AD method-based distributed strategy for automatic generation control of integrated energy systems," *IEEE Transactions on Sustainable Energy*, vol. 11, no. 4, pp. 2417–2426, 2019.

[33] C. Audet, J. Bigeon, D. Cartier, S. Le Digabel, and L. Salomon, "Performance indicators in multiobjective optimization," *European journal of operational research*, vol. 292, no. 2, pp. 397–422, 2021.

[34] A. P. Guerreiro, C. M. Fonseca, and L. Paquete, "The hypervolume indicator: Problems and algorithms," *arXiv preprint arXiv:2005.00515*, 2020.

[35] D. M. Roijers, S. Whiteson, and F. A. Oliehoek, "Computing convex coverage sets for faster multi-objective coordination," *Journal of Artificial Intelligence Research*, vol. 52, pp. 399–443, 2015.

[36] B. Yan, Y. Zhao, X. Yu, W. Wang, Y. Wu, Y. Wang, and J. Zhang, "Tidal-traffic-aware routing and spectrum allocation in elastic optical networks," *Journal of Optical Communications and Networking*, vol. 10, no. 11, pp. 832–842, 2018.

[37] X. Yin, J. Goudriaan, E. A. Lantinga, J. Vos, and H. J. Spiertz, "A flexible sigmoid function of determinate growth," *Annals of botany*, vol. 91, no. 3, pp. 361–371, 2003.

[38] C. Kreuzberger, D. Posch, and H. Hellwagner, "A scalable video coding dataset and toolchain for dynamic adaptive streaming over HTTP," in *the 6th ACM Multimedia Systems Conference*, 2015, pp. 213–218.

[39] N. Naderializadeh, J. J. Sydir, M. Simsek, and H. Nikopour, "Resource management in wireless networks via multi-agent deep reinforcement learning," *IEEE Transactions on Wireless Communications*, vol. 20, no. 6, pp. 3507–3523, 2021.

[40] K. Arora and D. R. Ch, "Web cache page replacement by using LRU and LFU algorithms with hit ratio: a case unification," *Int. J. Comput. Sci. Inf. Technol*, vol. 5, no. 3, pp. 3232–3235, 2014.

Boyang Guo received his B.S. degree in electronic and information engineering from Jiangsu University of Science and Technology, Zhenjiang, China. He is currently pursuing his Ph.D. degree in information and communication engineering with Fuzhou University, China. His research interests include wireless caching/computing, resource allocation and reinforcement learning in wireless networks.

Youjia Chen (Member, IEEE) received the B.S. and M.S. degrees in communication engineering from Nanjing University, Nanjing, China, and the Ph.D. degree in wireless engineering from the University of Sydney, Camperdown, NSW, Australia, in 2005, 2008 and 2017, respectively. From 2008 to 2009, she was with Alcatel-Lucent ShanghaiBell. Then she was with the College of Photonic and Electrical Engineering, Fujian Normal University, Fuzhou, China, from August 2009. She is currently a Professor with the College of Physics and Information Engineering, Fuzhou University, Fuzhou, China. She has authored or co-authored more than 40 research papers in leading international journals and conference, and contributed to a Wiley-IEEE Press book. Her research interests include wireless caching/computing, intelligent reflecting surfaces, internet of video things, and wireless AI.

Peng Cheng (Member, IEEE) received the B.S. and M.S degrees with great honors in communication and information systems from University of Electronic Science and Technology of China (UESTC), Chengdu, China, in 2006 and 2009 and the Ph.D. degree from Shanghai Jiao Tong University, Shanghai, China, in 2013. From 2014 to 2017, he was a Postdoctoral Research Scientist in CSIRO, Sydney, Australia. From 2017 to 2020, he was an ARC DECRA Fellow/Lecturer at the University of Sydney, Australia. He is currently an ARC DECRA Fellow and a Senior Lecturer (Tenured Associate Professor in U.S. systems) in Department of Computer Science and Information Technology, La Trobe University, Australia, and is affiliated with the University of Sydney, Australia. He has published over 70 peer-reviewed research papers in leading international journals and conferences. His current research interests include wireless AI, machine learning, IoT, millimeter-wave communications, and compressive sensing theory.

Ming Ding (Senior Member, IEEE) is currently a Senior Research Scientist with Data61, CSIRO, Sydney, NSW, Australia. He has authored more than 150 papers in IEEE journals and conferences, all in recognized venues, and around 20 3GPP standardization contributions, and also two books. His research interests include information technology, data privacy and security, and machine learning and AI. He also holds 21 U.S. patents and has co-invented another more than 100 patents on 4G/5G technologies. He is currently the Editor of IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS and IEEE COMMUNICATIONS SURVEYS AND TUTORIALS. He was the guest Editor/Co-Chair/Co-Tutor/TPC Member for multiple IEEE top-tier journals/conferences. He was the recipient of several awards for his research work and professional services.

Jinsong Hu (Member, IEEE) received the B.S. and Ph.D. degrees from the School of Electronic and Optical Engineering, Nanjing University of Science and Technology, Nanjing, China, in 2013 and 2018, respectively. From 2017 to 2018, he was a Visiting Ph.D. Student with the Research School of Engineering, The Australian National University, Canberra, ACT, Australia. He is an Associate Professor with the College of Physics and Information Engineering, Fuzhou University, Fuzhou, China. His research interests include covert communications, physical-layer security, and array signal processing. He is currently serving as a Review Editor of Frontiers in Communications and Networks.

Lajos Hanzo (Fellow, IEEE) received his Master degree and Doctorate in 1976 and 1983, respectively from the Technical University (TU) of Budapest. He was also awarded the Doctor of Sciences (DSc) degree by the University of Southampton (2004) and Honorary Doctorates by the TU of Budapest (2009) and by the University of Edinburgh (2015). He is a Foreign Member of the Hungarian Academy of Sciences and a former Editor-in-Chief of the IEEE Press. He has served several terms as Governor of both IEEE ComSoc and of VTS. He has published 2000+ contributions at IEEE Xplore, 19 Wiley-IEEE Press books and has helped the fast-track career of 123 PhD students. Over 40 of them are Professors at various stages of their careers in academia and many of them are leading scientists in the wireless industry. He is also a Fellow of the Royal Academy of Engineering (FREng), of the IET and of EURASIP.