Evolutionary Multi-Objective Reinforcement Learning Based Trajectory Control and Task Offloading in UAV-Assisted Mobile Edge Computing

Fuhong Song, Huanlai Xing, *Member, IEEE*, Xinhan Wang, Shouxi Luo, *Member, IEEE*, Penglin Dai, *Member, IEEE*, Zhiwen Xiao, *Member, IEEE*, and Bowen Zhao

Abstract—This article studies the trajectory control and task offloading (TCTO) problem in an unmanned aerial vehicle (UAV)-assisted mobile edge computing system, where a UAV flies along a planned trajectory to collect computation tasks from smart devices (SDs). We consider a scenario that SDs are not directly connected by the base station (BS) and the UAV has two roles to play: MEC server or wireless relay. The UAV makes task offloading decisions online, in which the collected tasks can be executed locally on the UAV or offloaded to the BS for remote processing. The TCTO problem involves multi-objective optimization as its objectives are to minimize the task delay and the UAV's energy consumption, and maximize the number of tasks collected by the UAV, simultaneously. This problem is challenging because the three objectives conflict with each other. The existing reinforcement learning (RL) algorithms, either single-objective RLs or single-policy multi-objective RLs, cannot well address the problem since they cannot output multiple policies for various preferences (i.e., weights) across objectives in a single run. An evolutionary multi-objective RL (EMORL) algorithm is applied to address the TCTO problem. We improve the multi-task multi-objective proximal policy optimization of the original EMORL by retaining all new learning tasks in the offspring population, which can preserve promissing learning tasks. The simulation results demonstrate that the proposed algorithm can obtain more excellent non-dominated policies by striking a balance between the three objectives regarding policy quality, compared with two evolutionary algorithms, two multi-policy RL algorithms, and the original EMORL.

Index Terms—Mobile edge computing, multi-objective reinforcement learning, task offloading, trajectory control, unmanned aerial vehicle

1 Introduction

With the rapid development of Internet-of-Things (IoT) technology, smart devices (SDs) play an essential role in various applications, such as object detectors for autonomous control, high definition cameras for intelligent grazing, and meteorological sensors for environmental monitoring [1]. SDs can be deployed to monitor and collect data from areas of interest, thus providing new opportunities for emerging intelligent applications, e.g., industrial automation and smart city. These applications are usually computing-intensive, which results in dramatically increased demand for computing resources, posing a great challenge to SDs due to their limited computing resources and battery capacity [2].

The contradiction between computing-intensive applications and resource-constrained SDs creates a bottleneck when achieving satisfactory quality of experience (QoE) for end users. Fortunately, mobile edge computing (MEC) brings abundant computing resources to the edge of networks close to SDs [3]. Under this paradigm, SDs can offload computing-intensive applications to nearby terrestrial base stations (BSs), which reduces the processing delay of applications and saves the energy consumption of SDs. Migrating these applications to BSs for execution are also referred to as computation offloading. Although the traditional BS-based MEC promotes computing-intensive applications in many fields, including computation and communication, MEC with only BSs may not always results in satisfactory computation offloading performance [4]. A terrestrial BS has a fixed wireless communication coverage while users can be anywhere. It is not possible for a BS to connect to a user out of its coverage. Especially some BSs may be damaged by natural disasters or military attacks, causing computing resource scarcity and offloading performance degradation [5]. How to provide users with on-demand computing services is one of the main challenges BS-based MEC networks face. Thanks to its high mobility and excellent maneuverability, unmanned aerial vehicle (UAV) has been applied to terrestrial networks for communication coverage extension and deployment efficiency improvement [6], [7]. Generally, UAV-assisted MEC is more agile and can

- *The authors are with the School of Computing and Artificial Intelligence, Southwest Jiaotong University, Chengdu 611756, China. E-mail: {fhs, xhwang}@my.swjtu.edu.cn, hxx@home.swjtu.edu.cn, {sxluo, penglindai}@swjtu.edu.cn, xiao1994zw@163.com, cn16bz@icloud.com.*

Manuscript received 24 February 2022; revised 19 September 2022; accepted 19 September 2022. Date of publication 21 September 2022; date of current version 3 November 2023.
This work was supported in part by the Natural Science Foundation of Sichuan Province under Grant 2022NSFSC0568, and in part by the Fundamental Research Funds for the Central Universities, P. R. China.
Recommended for acceptance by C. Peng. *(Corresponding author: Huanlai Xing.)*
Digital Object Identifier no. 10.1109/TMC.2022.3208457

better support on-demand computing services than the traditional BS-based MEC.

1.1 Related Work

An increasing amount of research attention has been paid to various issues in UAV-assisted MEC networks. There are mainly two categories according to the number of objectives to optimize, namely single- and multi-objective optimization.

1.1.1 Single-Objective Optimization

There has been a large amount of research studying single-objective optimization (SOO) problems in the context of UAV-assisted MEC, where only one objective is considered for optimization, e.g., delay or enery consumption. Traditional methods and deep reinforcement learning (DRL) are mainstream optimization techniques.

SOO With Traditional Methods. Liu et al. [7] investigated the computation offloading and UAV trajectory planning problem, with the total energy consumption of UAVs minimized. The authors used a convex optimization method to address it. Zhang et al. [8] emphasized task offloading and UAV relay communication in an MEC system with one UAV and one BS, where the successive convex approximation technology was adopted to minimize the system's energy consumption. The same technology was also used in [9] to reduce the energy consumption of a UAV by optimizing its trajectory and offloading schedule. Tun et al. [10] proposed a successive convex method that minimized the energy consumption of IoT devices and UAVs, with the task offloading decision and UAVs' trajectories taken into account. Apostolopoulos et al. [11] presented a data offloading decision-making framework consisting of ground and UAV-assisted MEC servers and the authors applied convex optimization to maximize each user's satisfaction utility. Ye et al. [12] studied the energy-efficient flight speed scheduling problem, with the purpose of minimizing the UAV's energy consumption. The authors obtained near-optimal solutions to UAV's flight speed scheduling via heuristics. In [13], a Lyapunov-based method was developed to minimize the average energy consumption of UAVs, where the task offloading and UAV trajectory were taken into account. Ei et al. [14] investigated a multi-UAV and BS collaborative MEC system, where multi-UAV provided SDs with the computing and relaying services. An efficient algorithm based on block successive upper bound was proposed to minimize the energy consumption of SDs and UAVs.

SOO With DRL Methods. Chen et al. [15] developed a DRL-based online method to maximize the long-term computation performance, where two deep Q-networks (DQN) were adopted. Zhao et al. [16] studied the UAV trajectory planning and power allocation problem and applied deep deterministic policy gradient (DDPG) to maximize the long-term network utility. Based on double deep Q-network (DQN), Liu et al. [17] proposed a two-phase DRL offloading algorithm for multi-UAV systems, with the system's total utility maximized. To minimize the total resource consumption of SDs, Wang et al. [18] presented an intelligent resource allocation method based on multi-agent Q-learning. In [19], a hierarchical RL (HRL) algorithm was developed to minimize the average delay of tasks by jointly optimizing the movement locations of SDs and offloading decisions. To minimize the energy consumption of all SDs, Wang et al. [20] presented a trajectory control method based on DDPG with prioritized experience replay. Dai et al. [21] considered a UAV-and-BS enabled MEC system and devised a DDPG-based task association scheduling method to minimize the system's energy consumption. Seid et al. [22] designed a multi-UAV and BS hybrid MEC network, where UAVs provided computing services to different SDs in a cooperative manner. The authors proposed a multi-agent RL based method to reduce the computation cost. Samir et al. [23] proposed the proximal policy optimization (PPO) based algorithm to minimize the expected weighted sum age of information (AOI) by finding the optimal altitude and scheduling policy. Ji et al. [24] formulated a multimedia content dissemination problem, aiming at minimizing the sum content acquisition delay of all users. The BS agent took actions corresponding to the user association, while each UAV agent made the cache placement and UAV trajectory decisions. Nie et al. [25] proposed a semi-distributed multi-agent federated RL (FRL) algorithm to minimize the energy consumption of SDs and UAVs by optimizing the offloading decision, frequency resources, and transmission power.

1.1.2 Multi-Objective Optimization

In nature, multiple possibly conflicting objectives exist in UAV-assisted MEC. For example, one should consider the trade-off between delay and energy consumption in the task offloading decision-making process; one should balance the energy consumption and flying speed when planning a UAV's trajectory. Some research efforts have been dedicated to multi-objective optimization (MOO) problems.

MOO With Traditional Methods. In [5], a game-theory-based method was proposed to optimize the weighted cost of delay and energy consumption in UAV-assisted MEC with multiple SDs and single UAV, subject to the resource competition constraint. Ning et al. [6] considered the computation offloading and server deployment problem and designed two stochastic game methods to minimize the computation delay and energy consumption of each UAV. Zhan et al. [26] studied the computation offloading and resource allocation problem and designed a successive convex optimization method to minimize the energy consumption and completion delay of a UAV. Lin et al. [27] developed a Lyapunov based resource allocation method for UAV-assisted MEC systems, aiming at reducing the overall energy consumption and computation delay. Yu et al. [28] investigated a UAV and multiple edge servers to collaboratively provide SDs with computing services. The authors proposed a successive convex approximation based algorithm to minimize the weighted sum of the delay and energy consumption. Zhu et al. [29] proposed an improved fast and elitist non-dominated sorting genetic algorithm (NSGA-II) to minimize the cost and completion time, simultaneously.

MOO With DRL Methods. Chen et al. [30] considered a three-dimensional UAV-assisted MEC system, minimizing the task processing delay and energy consumption by double DQN. In [31], DQN was used to minimize the energy consumption and computation delay of MEC networks simultaneously. Sun et al. [32] studied a bi-objective optimization problem with AoI and UAV's energy-consumption as two

objectives to minimize and devised a twin-delayed DDPG (TD3) for UAV trajectory control. Wang et al. [33] proposed a multi-agent DDPG based trajectory control algorithm that took the geographical fairness among UAVs and energy consumption of SDs as two objectives for optimization. Peng et al. [34] studied the single-UAV trajectory control problem and adopted double DQN to minimize the UAV's energy consumption and maximize the amount of offloaded data, simultaneously. Sacco et al. [35] proposed a multi-agent RL algorithm to optimize the energy efficiency and task completion time. Each agent could make computation offloading decisions in real-time by combining state information from other SDs. Cheng et al. [36] proposed an FRL framework to learn the joint task offloading and energy allocation decision, aiming at maximizing the long-term reward, as well as reducing the training cost and preserving privacy.

1.1.3 Analysis and Motivation

Despite the ample research efforts dedicated, UAV-assisted MEC still faces great challenges in terms of system design and optimization. We discuss these challenges from two aspects, i.e., system modeling and optimization techniques.

System Modeling. In most existing works, see [6], [26], [33], a system only adopts one or more UAVs for task collection and local processing, where no BS is involved. Although it suffices in cases where the number of SDs is small, such a system cannot satisfy large-scale MEC deployment since UAVs usually have limited computing resources. Multiple UAVs could alleviate the computing pressure, but at the expense of extra deployment cost. To handle the issue, some works [8], [11] focus on UAV-assisted MEC systems that are integrated with BSs. With efficient collaboration between UAV and BS, various computing services can be provisioned to ground SDs. Thus, UAV-assisted MEC involving BSs is a practical scenario.

In some extreme scenarios, SDs cannot be reached by BS due to natural disasters, military attacks or simply being out of BS's coverage. In this case, a UAV has two roles to play: (1) an MEC server that runs some of the collected computation tasks from SDs and sends back results to them, or (2) a relay that forwards some computation tasks to a BS. However, this scenario has received little research attention in the literature. That is our motivation to consider a UAV-assisted MEC system without direct connection between SDs and BSs.

On the other hand, considering delay and energy consumption as optimization objectives is one of the main research streams on UAV-assisted MEC. Most existing works optimize the two individually. The fact that the conflicts between objectives are neglected easily leads to biased optimization results. Meanwhile, a few studies focus on the maximization of the number of tasks collected by UAV(s), which also reflects the benefits that an MEC system brings to us. Therefore, delay, energy consumption and number of tasks collected are three important concerns when designing UAV-assisted MEC systems. However, little research has been dedicated to a system with these three objectives taken into account. That is why we are motivated to emphasize the UAV-assisted MEC system with delay, energy consumption and number of tasks collected as three objectives for optimization.

Optimization Technique. Traditional methods, including convex optimization [7], [8], [9], [10], [11], [26], heuristics [12], Lyapunov optimization [13], [27], and game theory [5], [6], work well when dealing with various optimization issues under static scenarios, such as a UAV hovering over a fixed spot during the whole flying mission. However, these methods are hardly adapted to a dynamic environment, especially when UAVs move quickly and tasks arrive unpredictably. That is because the dynamics and uncertainty frequently trigger execution of the above methods that launch from scratch, resulting in high computational burdens and slow response. Thus, these methods are not suitable for always responding quickly to users while the MEC environment is ever-changing.

Different from the traditional methods, DRL can deal with complicated control problems with little prior information extracted from dynamic MEC scenarios. The reason is that DRL methods are able to quickly adapt their behaviors to the changes by interacting with the corresponding environment. However, all the DRLs above are single-objective RL (SORL), which defines the user utility as a linear scalarization based on preferences (i.e., weights) across objectives. These SORL methods first aggregate multiple objectives into a scalar reward via weighted sum and then optimize the reward. Nevertheless, the conflicts between objectives are ignored because weighted sum is usually biased and hardly strikes a balance between objectives.

Multi-objective RL (MORL) can well address the challenge above [37], [38]. According to the number of learned policies, MORLs can be divided into two categories, namely single-policy MORLs and multi-policy MORLs. A single-policy MORL aims to optimize one policy for a given preference. For example, the authors in [39] extended a single-objective DDPG to a single-policy MORL to optimize the data rate, total harvested energy, and UAV's energy consumption. However, a single-policy MORL cannot output multiple optimal policies after a run, each of which optimizes a certain preference.

Unlike single-policy MORLs, multi-policy MORLs can learn a set of policies that approximate the true Pareto front. These policies correspond to different trade-offs, and the decision maker can select the one that matches the current preference. With the multi-task multi-objective proximal policy optimization (PPO), the evolutionary MORL (EMORL) algorithm [38] has promising potential to find a set of high-quality policies. This algorithm has been successfully applied to continuous robotic control problems. This is why we adapt EMORL to the UAV-assisted MEC concerned in this paper.

Table 1 clearly shows the differences between similar works that consider the UAV-and-BS hybrid scenarios and ours in terms of three aspects, i.e., system modeling, optimization technique, and the number of policies. It can be seen that we adopt MORL to address the modeled MOO problem and output multiple non-dominated policies.

1.2 Contribution

This paper studies the trajectory control and task offloading (TCTO) problem in a UAV-assisted MEC system, where a UAV and a BS work together to provide SDs with computing services. We consider the scenario that SDs are not

TABLE 1
Differences Between Similar Works and Ours

Reference		[5]	[8]	[11]	[14]	[17]	[21]	[22]	[23]	[24]	[28]	[29]	[30]	[32]	[35]	[36]	Ours
System modeling	SOO		✓	✓	✓	✓	✓	✓	✓	✓							
	MOO	✓									✓	✓	✓	✓	✓	✓	✓
Optimization technique	Traditional	✓	✓	✓	✓						✓	✓					
	SORL					✓	✓	✓	✓	✓			✓	✓	✓	✓	
	MORL																✓
Number of polices	Single	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓	✓	✓	✓	
	multiple											✓					✓

directly connected by the BS and the UAV plays as an MEC server when processing a collected computation task locally or a wireless relay when forwarding the task to the BS. The UAV collects computation tasks from the SDs within its coverage and decides the proportion of these tasks to be offloaded to the BS for remote processing. Different from the existing works that either optimize a single objective or a number of objectives via weighted sum, this paper considers three conflicting objectives and aims to optimize them, simultaneously. To obtain a set of Pareto optimal policies, we adapt EMORL to the MOO problem. The main contributions are summarized as follows.

- We study a UAV-and-BS collaborative MEC system, where one UAV and one BS work together to provide SDs with computing services. The TCTO problem is formulated as an MOO problem, aiming at minimizing the task delay and UAV's energy consumption, and maximizing the number of tasks collected by the UAV, simultaneously. The MOO problem is difficult to address because the three objectives conflict with each other and to strike a balance between them is quite challenging.
- We model a multi-objective Markov decision process (MOMDP) with a vector reward of three elements for the TCTO problem, where each element corresponds to an optimization objective. Based on the MOMDP model, we propose an improved EMORL algorithm, namely EMORL-TCTO, to solve the TCTO problem. Specifically, we improve the multi-task multi-objective PPO in EMORL-TCTO by retaining all new learning tasks in the offspring population, which ensures promising learning tasks are preserved. EMORL-TCTO can output multiple policies to satisfy various preferences of users at a run. To our knowledge, this is the first work that applies a multi-policy MORL to the UAV-assisted MEC field.
- We conduct extensive experiments using six test instances. The results clearly show that the proposed EMORL-TCTO obtains a set of high-quality non-dominated policies and outperforms two state-of-the-art multi-objective evolutionary algorithms, two exclusively devised multi-policy MORLs, and the original EMORL against several evaluation criteria, including the inverted generational distance, hyper volume, average comprehensive objective indicator, and Friedman test.

The remainder of the paper is organized as follows. The system model and problem formulation are presented in Section 2. In Section 3, we briefly review the MOMDP and MOO. In Section 4, we introduce the proposed algorithm for the TCTO problem in detail. Section 5 analyzes and discusses the simulation results. Finally, Section 6 presents the conclusion and future work.

2 System Model and Problem Formulation

As shown in Fig. 1, this paper considers a UAV-assisted MEC system consisting of one UAV, one BS, and a set of SDs. These SDs are randomly scattered in a rectangular area and their computation tasks arrive dynamically. A rotary-wing UAV can hover in the air and fly at a low altitude sufficiently close to SDs. Considering the economical and scalable deployment, this paper considers a rotary-wing UAV with limited computing resources. The UAV is responsible for task collection, i.e., it flies along a planned trajectory to collect computation tasks from SDs within its coverage. It either executes all these tasks locally or offloads a proportion of them to the BS for processing when needed. The BS has abundant computing resources and acts as a complementary offloading solution to the UAV.

We consider a discrete time system, where each time slot has a time duration of τ. Suppose the entire task collection process of the UAV lasts for T time slots. Let $\mathcal{T} = \{1, \ldots, T\}$ denote the set of time slots. Let $\mathcal{K} = \{1, \ldots, K\}$ be the set of SDs, where K is the number of SDs. The main notations used in this paper are summarized in Table 2.

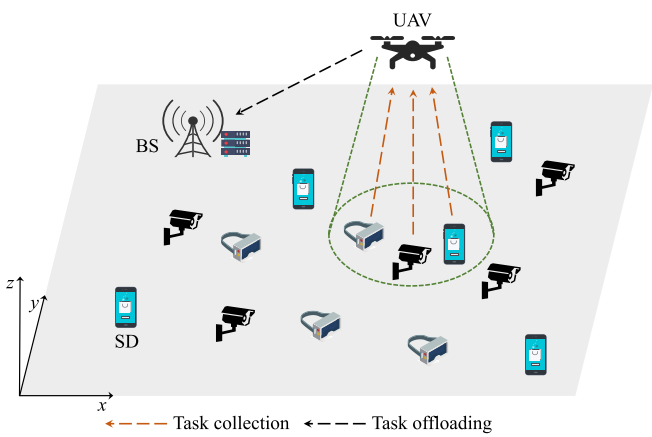


Fig. 1. UAV-assisted MEC system.

TABLE 2
Summary of Main Notations

Notation	Definition
	Notation used in system model
b_t	Offloading decision of the UAV in time slot t
$d_{\max}$	Maximal distance the UAV can move in each time slot
d_t	Horizontal distance the UAV flies in time slot t
f_{U}	Computing capability of the UAV
H	Fixed flying altitude of the UAV
k	The k-th SD
K	Number of SDs
$\mathcal{K}$	Set of SDs
$\mathcal{K}_t^{\mathrm{c}}$	Set of SDs covered by the UAV in time slot t
l_t^k	Task arrival indicator of SD k in time slot t
L_t^k	Number of tasks in the k-th SD's queue in time slot t
$N_{\max}$	Maximum number of tasks in the computing queue
N_t^{c}	Number of collected tasks from SDs in time slot t
N_t^{O}	Number of tasks offloaded to the BS in time slot t
N_t^{L}	Number of tasks executed by the UAV in time slot t
P_{U}	Transmission power of the UAV
$R_{\max}$	Maximum horizontal coverage of the UAV
T	Number of time slots
$\mathcal{T}$	Set of time slots
W	Channel bandwidth
α	Input data size of a task
β	Number of CPU cycles required to process a task
$\vartheta_{\max}$	Maximal azimuth angle of the UAV
ϑ_t	Horizontal direction the UAV flies in time slot t
ζ^k	Parameter of Bernoulli random variable of SD k
κ	Effective capacitance coefficient
μ_t	Data rate of the wireless channel in time slot t
σ^2	Background noise power
τ	Time duration of a time slot
ϕ	Number of tasks handled by the UAV within a time slot
	Notation used in reinforcement learning
a	Action
$\mathcal{A}$	Action space
$\mathbf{A}_t$	Vector-valued advantage function
$A_t^{\mathbf{w}_i}$	Extended advantage function with weight vector $\mathbf{w}_i$
$\mathbf{F}(\pi)$	Objective vector of policy π
n	Number of learning tasks
$\mathbf{r}_t$	Vector-valued reward at time step t
$\mathbf{R}_\pi$	Vector-valued return following policy π
s	State
$\mathcal{S}$	State space
$\mathbf{V}_\pi(s)$	Multi-objective value function in state s
$\mathcal{W}$	Set of evenly distributed weight vectors
λ	Parameter of general advantage estimator
γ	Discount factor
Γ_i	The i-th learning task in Ω, $i = 1, \ldots, n$
Ω	Set of learning tasks

2.1 Task Model

We assume that the computation tasks arriving at SD $k \in \mathcal{K}$ can be modeled as an independent and identically distributed sequence of Bernoulli random variables with parameter $\zeta^k \in [0, 1]$. Different SDs are associated with different parameters of Bernoulli random variables. Let l_t^k denote the task arrival indicator of SD k in time slot t. $l_t^k = 1$ if a task is generated at the beginning of t and $l_t^k = 0$, otherwise. We have $\Pr(l_t^k = 1) = 1 - \Pr(l_t^k = 0) = \zeta^k$, where $\Pr(\cdot)$ stands for the probability of an event occurring. A computation task is modeled as tuple $\langle \alpha, \beta \rangle$, where α denotes the input data size of the task and β is the number of CPU cycles required to process the task. For an arbitrary SD, a computation task generated in t is stored in its task queue. Let L_t^k be the number of tasks in the k-th SD's queue waiting to be uploaded in t, which is updated by

$$L_{t+1}^k = \min\{L_t^k + l_t^k, L_{\max}\}, \tag{1}$$

where $L_{\max}$ is the maximum number of tasks allowed to be stored in the k-th SD's queue. If the queue is full, each newly arrival task is dropped. Hence, it is of great significance for SDs to upload their computation tasks to the UAV in time. In this paper, the time division multiple access protocol is adopted for uploading computation tasks.

2.2 UAV Movement Model

We assume that the UAV flies at an altitude of H, where H is a positive constant. Let ϑ_t and d_t denote the horizontal direction and distance with which the UAV flies in time slot t, respectively, with the following constraints met

$$0 \le \vartheta_t \le 2\pi, 0 \le d_t \le d_{\max}, \tag{2}$$

where $d_{\max}$ is the maximal flying distance that the UAV can move in each time slot due to the limited power budget.

Similar to previous studies [20], [39], we adopt the Cartesian coordinate system to model the movement of the UAV. Let $\mathbf{c}_t^{\mathrm{U}} = [x_t^{\mathrm{U}}, y_t^{\mathrm{U}}]$ denote the UAV's horizontal coordinate in time slot t. Based on ϑ_t and d_t, we obtain the UAV's horizontal coordinate in time slot $t + 1$ by

$$\begin{cases} x_{t+1}^{\mathrm{U}} = x_t^{\mathrm{U}} + d_t \cdot \cos(\vartheta_t) \\ y_{t+1}^{\mathrm{U}} = y_t^{\mathrm{U}} + d_t \cdot \sin(\vartheta_t). \end{cases} \tag{3}$$

Assume that the UAV flies at a constant velocity $v_t = d_t/\tau$, limited by a pre-defined maximum flying velocity $v_{\max}$. The UAV can only move within a rectangular area whose side lengths are $x_{\max}$ and $y_{\max}$. We have

$$0 \le x_t^{\mathrm{U}} \le x_{\max}, 0 \le y_t^{\mathrm{U}} \le y_{\max}. \tag{4}$$

When a rotary-wing UAV flies, its propulsion power consumption with speed v, $P(v)$, is defined as [39]

$$P(v) = P_1\left(1 + \frac{3v^2}{U_{\mathrm{tip}}^2}\right) + P_2\left(\sqrt{1 + \frac{v^4}{4v_0^4}} - \frac{v^2}{2v_0^2}\right)^{1/2} + \frac{1}{2}d_0\rho g A v^3. \tag{5}$$

It is seen that $P(v)$ consists of three parts: the blade profile, induced power, and parasite power. P_1 and U_{tip} denote the blade profile power under hovering status and tip speed of rotor blade, respectively. P_2 and v_0 represent the induced power and mean rotor induced velocity in hovering, respectively. As for the parasite power, d_0, ρ, g, and A indicate the fuselage drag ratio, air density, rotor solidity, and rotor disc area, respectively. Note that when the UAV hovers (i.e., $v = 0$), the corresponding power consumption P_{h} is the summation of P_1 and P_2. The energy consumption when the

UAV is flying and hovering during a time duration of T, E_{fly}, is obtained by

$$E_{\text{fly}} = \int_0^T P(v_t)dt. \tag{6}$$

2.3 Computing Model

2.3.1 Local Computing

Assume the UAV maintains a computing queue that stores the computation tasks collected from SDs awaiting for further processing. As the UAV can stay at a low altitude sufficiently close to SDs, this paper ignores the delay for collecting the computation tasks in each time slot, so does the corresponding receiving power consumption at the UAV. In this paper, the delay for processing tasks locally on the UAV in time slot t consists of the local processing and queuing delays. Let $N_t^{\text{u}} \in [0, N_{\max}]$ represent the number of uncompleted tasks in the computing queue at the beginning of t, where $N_{\max}$ is the maximum number of tasks allowed. Let $b_t \in [0, 1]$ be the proportion of tasks in the computing queue to be offloaded to the BS in t, namely the UAV's offloading decision for t. Specifically, the UAV offloads $N_t^{\text{O}} = \lfloor b_t N_t^{\text{u}} \rfloor$ computation tasks to the BS for remote processing, where $\lfloor \cdot \rfloor$ denotes the floor function. The remaining $N_t^{\text{L}} = N_t^{\text{u}} - N_t^{\text{O}}$ computation tasks are locally executed on the UAV. Let $\phi = \lfloor \tau f_{\text{U}}/\beta \rfloor$ denote the number of computation tasks processed by the UAV within each time slot, where f_{U} denotes the UAV's computing capability. Based on N_t^{u} and N_t^{O}, the number of queueing tasks in the computing queue at the end of t, N_t^{q}, is defined as

$$N_t^{\text{q}} = \max\{N_t^{\text{u}} - \phi - N_t^{\text{O}}, 0\}. \tag{7}$$

Let $\mathbf{c}^k = [x^k, y^k]$ be the horizontal coordinate of SD $k \in \mathcal{K}$. The UAV can only collect the tasks within its coverage area. Let $\mathcal{K}_t^{\text{c}}$ represent the set of SDs covered by the UAV in time slot t, which is defined as

$$\mathcal{K}_t^{\text{c}} = \{k | d_t^k \le R_{\max}, k \in \mathcal{K}\}, \tag{8}$$

where $d_t^k = \sqrt{(x_t^{\text{U}} - x^k)^2 + (y_t^{\text{U}} - y^k)^2}$ is the horizontal distance between the UAV and SD k in t. $R_{\max}$ is the UAV's maximal horizontal coverage, given that it has a maximal azimuth angle $\vartheta_{\max}$ [20]. $R_{\max}$ is calculated by

$$R_{\max} = H \cdot \tan(\vartheta_{\max}). \tag{9}$$

Based on Eq. (8), the number of tasks collected by the UAV in t is obtained by

$$N_t^{\text{c}} = \sum_{k \in \mathcal{K}_t^{\text{c}}} L_t^k. \tag{10}$$

The number of uncompleted tasks to be processed in $t+1$, N_{t+1}^{u}, is updated at the end of t as

$$N_{t+1}^{\text{u}} = \min\{N_t^{\text{q}} + N_t^{\text{c}}, N_{\max}\}. \tag{11}$$

In t, the delay for completing the N_t^{L} tasks locally on the UAV can be calculated by

$$D_t^{\text{L}} = \frac{\min\{\phi, N_t^{\text{L}}\}\beta}{f_{\text{U}}} + \tau N_t^{\text{q}}. \tag{12}$$

There are two parts in Eq. (12). The first part, $\min\{\phi, N_t^{\text{L}}\}\beta/f_{\text{U}}$, is the local processing delay, and the second one, τN_t^{q}, is the queuing delay of all N_t^{q} tasks waiting in the computing queue. The corresponding energy consumption of the UAV is calculated by

$$E_t^{\text{L}} = \kappa \cdot \min\{\phi, N_t^{\text{L}}\}\beta \cdot (f_{\text{U}})^2, \tag{13}$$

where κ is the effective capacitance coefficient depending on the chip structure used.

2.3.2 Task Offloading

The UAV allows a proportion of its collected tasks to be offloaded to the BS for remote processing. According to the Shannon-Hartley theorem [4], we define the data rate of the wireless link between the UAV and BS in t as

$$\mu_t = W \cdot \log_2(1 + \Upsilon_t), \tag{14}$$

where W and Υ_t is the channel bandwidth of the wireless link and signal-to-noise ratio (SNR) between the UAV and BS, respectively. As the UAV flies at a low altitude, communication outage may occur. To maintain wireless connectivity, Υ_t is greater than or equal to the threshold SNR Υ_{thr}. In other words, if $\Upsilon_t \ge \Upsilon_{\text{thr}}$, the UAV can successfully connect to the BS; otherwise, the wireless connectivity is unavailable between the UAV and BS. The SNR in t is defined below.

$$\Upsilon_t = \frac{P_{\text{U}} \cdot 10^{\frac{PL(d_t^{\text{UB}}, \vartheta_t^{\text{UB}})}{10}}}{\sigma^2}, \tag{15}$$

where P_{U}, σ^2, and $PL(d_t^{\text{UB}}, \vartheta_t^{\text{UB}})$ are the transmission power of the UAV, background noise power, and pathloss between the UAV and BS, respectively. Referring to [4], this paper defines the pathloss between the UAV and BS in t as

$$PL(d_t^{\text{UB}}, \vartheta_t^{\text{UB}}) = 10A_0 \log(d_t^{\text{UB}}) + B_0(\vartheta_t^{\text{UB}} - \theta_0)\mathrm{e}^{\frac{\theta_0 - \vartheta_t^{\text{UB}}}{C_0}} + \eta_0, \tag{16}$$

where d_t^{UB} and ϑ_t^{UB} are the distance and vertical angle between the UAV and BS in t, respectively. d_t^{UB} and ϑ_t^{UB} in Eq. (16) are obtained based on the horizontal coordinates of the UAV and BS.

The UAV needs to complete the transmission process of the N_t^{O} computation tasks before it flies out of the BS's coverage. Thus, the time duration φ_t that the UAV has been staying in the coverage of the BS since the beginning of t is written as

$$\varphi_t = \arg\min_l \left(\sum_{i=t}^{t+l} \tau\mu_i \ge \alpha N_t^{\text{O}} \right), \tag{17}$$

where α stands for the input data size of a computation task. Let D_t^{O} denote the delay for offloading the N_t^{O} computation tasks to the BS, which is calculated by

$$D_t^{\text{O}} = \begin{cases} (\varphi_t - 1)\tau + \frac{\alpha N_t^{\text{O}} - \sum_{i=t}^{\varphi_t - 1} \tau\mu_i}{\mu_{t+\varphi_t}}, & \text{if } \alpha N_t^{\text{O}} < \sum_{i=t}^{\varphi_t} \tau\mu_i \\ \tau\varphi_t, & \text{if } \alpha N_t^{\text{O}} = \sum_{i=t}^{\varphi_t} \tau\mu_i \end{cases} \tag{18}$$

The corresponding energy consumption of the UAV is calculated as

$$E_t^{\mathrm{O}} = P_{\mathrm{U}} \cdot D_t^{\mathrm{O}}. \tag{19}$$

Assume that the BS is of rich computing resources. Thus, the delay for processing the tasks on the BS can be neglected. Further, the delay for returning the task results to an SD is also ignored because the computation result of a task is usually much smaller than its input data size.

2.4 Problem Formulation

Based on Eqs. (12) and (18), the delay for completing the $N_t^{\mathrm{L}} + N_t^{\mathrm{O}}$ computation tasks in the UAV's computing queue in t is written as

$$D_t = D_t^{\mathrm{L}} + D_t^{\mathrm{O}}. \tag{20}$$

Similarly, based on Eqs. (13) and (19), the UAV's energy consumption for local computing and transmitting tasks to the BS in t is defined as

$$E_t = E_t^{\mathrm{L}} + E_t^{\mathrm{O}}. \tag{21}$$

The total delay for completing all the collected tasks, D_{total}, and total energy consumption of the UAV, E_{total}, during T time slots are calculated as

$$D_{\mathrm{total}} = \sum_{t=1}^{T} D_t, \tag{22}$$

$$E_{\mathrm{total}} = \sum_{t=1}^{T} E_t + E_{\mathrm{fly}}. \tag{23}$$

Based on the number of collected tasks defined in Eq. (10) in each time slot, the total number of collected tasks during time duration T can be obtained by

$$N_{\mathrm{total}} = \sum_{t=1}^{T} N_t^{\mathrm{c}}. \tag{24}$$

In this work, we aim to minimize the total task delay D_{total} and total energy consumption E_{total}, and maximize the total number of tasks collected N_{total}, simultaneously, through optimizing the UAV's flying trajectory (i.e., ϑ_t and d_t) and task offloading decision (i.e., b_t), namely the TCTO problem. This problem is an MOO problem in nature, defined as

$$\max_{\vartheta_t, d_t, b_t} (-D_{\mathrm{total}}, -E_{\mathrm{total}}, N_{\mathrm{total}}) \tag{25}$$

subject to:

$$\begin{aligned}
&\mathrm{C1}: 0 \le \vartheta_t \le 2\pi, && \forall t \in \mathcal{T},\\
&\mathrm{C2}: 0 \le d_t \le d_{\max}, && \forall t \in \mathcal{T},\\
&\mathrm{C3}: b_t \in [0,1], && \forall t \in \mathcal{T},\\
&\mathrm{C4}: 0 \le x_t^{\mathrm{U}} \le x_{\max}, && \forall t \in \mathcal{T},\\
&\mathrm{C5}: 0 \le y_t^{\mathrm{U}} \le y_{\max}, && \forall t \in \mathcal{T},\\
&\mathrm{C6}: d_t^k \le R_{\max}, && \forall k \in \mathcal{K}_t^{\mathrm{c}}, t \in \mathcal{T}.
\end{aligned}$$

Constraints C1 and C2 confine the horizontal direction and distance of a flying UAV. Constraint C3 specifies that the offloading decision for time slot t is a variable between 0 and 1. Constraints C4 and C5 together specify the UAV's movement area. Constraint C6 ensures that the UAV can only collect computation tasks from SDs within its coverage.

It is easily understood that to increase N_{total}, the UAV should fly with an appropriate trajectory so that it can cover as many SDs and collect their computation tasks as possible. However, the more the computation tasks collected, the higher the energy consumption incurred on the UAV because more tasks need to be handled by the UAV. Admittedly, offloading helps to reduce the UAV's energy consumption as some tasks are processed by the BS. However, it results in additional transmission delays. So, one can easily observe that the three objectives, i.e., minimization of D_{total}, minimization of E_{total}, and maximization of N_{total}, conflict with each other.

3 Overview of MOMDP and MOO

This section first recalls the multi-objective Markov decision process (MOMDP). Then, we introduce the multi-objective optimization (MOO) problem.

3.1 MOMDP

The TCTO problem is a multi-objective control problem that can be modeled by MOMDP [38]. An MOMDP is defined by tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{Q}, \mathbf{r}, \gamma, \mathcal{D} \rangle$, where $\mathcal{S}$ is the state space. $\mathcal{A}$ is the action space and $\mathcal{Q}(s'|s,a)$ is the state transition probability. $\mathbf{r} = (r^1, \ldots, r^m)$ is the vector-valued reward function and m is the number of objectives. $\gamma \in [0,1]$ is the discount factor, and $\mathcal{D}$ is the initial state distribution.

In MOMDPs, a policy $\pi : \mathcal{S} \to \mathcal{A}$ is a state-to-action mapping associated with a vector of expected return $\mathbf{R}_\pi = (R_\pi^1, \ldots, R_\pi^m)$, where R_π^j is the expected return corresponding to the j-th objective, defined as

$$R_\pi^j = \mathbb{E}_\pi \left[\sum_{t=1}^{T} \gamma^{t-1} r^j(s_t, a_t) | s_1 \backsim \mathcal{D}, a_t \backsim \pi(s_t) \right]. \tag{26}$$

For the TCTO problem, we have $m = 3$, namely, R_π^1, R_π^2 and R_π^3 are associated with $-D_{\mathrm{total}}$, $-E_{\mathrm{total}}$, and N_{total}, respectively.

The value function $\mathbf{V}_\pi(s) : \mathcal{S} \to \mathbb{R}^m$ maps a state s to the vector of expected return under policy π, defined as

$$\mathbf{V}_\pi(s) = \mathbb{E}_\pi \left[\sum_{k=t}^{T} \gamma^{k-t} \mathbf{r}_k | s_t = s \right], \tag{27}$$

where $\mathbf{r}_k = (r_k^1, \ldots, r_k^m)$ denotes the immediate vector-valued reward at time step k. Because each element of $\mathbf{r}_k$ corresponds to a particular objective, $\mathbf{V}_\pi(s)$ is a multi-objective value function.

3.2 MOO

An MOO problem [38] can be formulated as

$$\begin{aligned}
&\max_\pi \mathbf{F}(\pi) = \max_\pi (f^1(\pi), \ldots, f^m(\pi)),\\
&\text{subject to:} \quad \pi \in \Pi.
\end{aligned} \tag{28}$$

where π is a policy in search space Π. In objective vector $\mathbf{F}(\pi)$, there are m objective functions, and they generally conflict with each other. Note that the objective value $f^j(\pi)$ is set to R^j_π, $j = 1, \ldots, m$.

Let $\pi_1, \pi_2 \in \Pi$ denote two different policies. π_1 is said to dominate π_2, denoted by $\pi_1 \succ \pi_2$, if and only if $f^j(\pi_1) \geq f^j(\pi_2)$ for all $j = 1, \ldots, m$, and $f^l(\pi_1) > f^l(\pi_2)$ for at least one index $l \in \{1, \ldots, m\}$. A policy $\pi^* \in \Pi$ is Pareto optimal if it is not dominated by any other policies in Π. All Pareto optimal policies (also called non-dominated policies) form a Pareto optimal set whose mapping in the objective space is known as the Pareto front.

There are mainly two methods to tackle an MOO problem. One is to convert it into an SOO problem by objective aggregation. In this case, the commonly used method is the weighted sum, where each objective is assigned a weight that must be set in advance. For example, the SORL methods first aggregate multiple objectives into a scalar reward via the weighted sum and then optimize the reward. However, the weighted sum based methods only output a unique optimal policy by running them once. If user preferences change, these methods need to be re-executed. Therefore, this kind of method only obtains a compromised policy that cannot reflect the conflicting features between objectives. In other words, the policy obtained is only optimal for the current preference.

The other method to handle MOO problems is to adopt multi-objective algorithms, such as the multi-objective evolutionary algorithms (MOEAs) and multi-policy MORLs. These methods can obtain multiple non-dominated policies in a single run, reflecting the Pareto-dominance relation among them. This is what a decision-maker expects to know. Although the user preferences change, the non-dominated policies obtained by a multi-objective algorithm are still valid. Thus, the ultimate aim of solving an MOO problem is to obtain a set of high-quality non-dominated policies. Each policy in the set is associated with a certain preference. In other words, for a given preference, we can find the corresponding optimal policy from the set. Therefore, we can balance multiple objectives by obtaining multiple non-dominated policies. However, MOEAs usually suffer from prematurity and local optima when handing high-dimensional MOO problems in dynamic environments, causing unacceptable non-dominated policies [40]. Compared with MOEAs, EMORL has been reported to find much better non-dominated policies [38]. That is why we are motivated to adapt EMROL to the TCTO problem concerned in this paper.

4 EMORL-TCTO for Trajectory Control and Task Offloading

This section first introduces the MOMDP model for the TCTO problem and then describes the proposed EMORL-TCTO algorithm in detail.

4.1 MOMDP Model

To address the TCTO problem by an MORL, we need an MOMDP model for the problem first. The state space, action space, and reward function are described one by one.

4.1.1 State Space

$$\mathcal{S} = \{s_t | s_t = (\mathbf{c}^{\mathrm{U}}_t, N^{\mathrm{u}}_t, N^{\mathrm{c}}_t), \forall t \in \mathcal{T}\}, \tag{29}$$

where $\mathbf{c}^{\mathrm{U}}_t = [x^{\mathrm{U}}_t, y^{\mathrm{U}}_t]$ is the horizontal coordinate of the UAV in time slot t. N^{u}_t is the number of uncompleted tasks at the beginning of t, and N^{c}_t is the number of newly collected tasks from SDs in t.

4.1.2 Action Space

$$\mathcal{A} = \{a_t | a_t = (\vartheta_t, d_t, b_t), \forall t \in \mathcal{T}\}, \tag{30}$$

where ϑ_t and d_t denote the horizontal direction and distance with which the UAV flies in t, respectively, and b_t is the UAV's offloading decision in t.

4.1.3 Reward Function

$$\mathbf{r}_t = (r^{\mathrm{D}}_t, r^{\mathrm{E}}_t, r^{\mathrm{N}}_t) = \begin{cases} (-D_t, -\frac{E_t}{100}, N^{\mathrm{c}}_t), & \text{if } \mathbb{1}_t = 1 \\ (-\varepsilon_1 D_t, -\varepsilon_2 \frac{E_t}{100}, \varepsilon_3 N^{\mathrm{c}}_t), & \text{otherwise} \end{cases} \tag{31}$$

where r^{D}_t, r^{E}_t, and r^{N}_t are the scalar rewards corresponding to D_t, E_t, and N^{c}_t in time slot t, respectively. $\mathbb{1}_t$ is an indicator variable that equals 0 if the UAV flies out of the rectangular area in t and $\mathbb{1}_t$ is equal to 1, otherwise. Coefficient $\frac{1}{100}$ in Eq. (31) is to make sure the three scalar rewards are in the same order of magnitude. This can effectively optimize three objectives simultaneously without any biases between objectives.

In addition, we should punish the three scalar rewards if the UAV flies out of the rectangular area in t. Thus, the penalty coefficients ε_1, ε_2, and ε_3 are used to reduce the values of $-D_t$, $-\frac{E_t}{100}$, and N^{c}_t, respectively. On the one hand, ε_1 and ε_2 are larger than 1 while ε_3 is smaller than 1. These settings guarantee that the values of $-D_t$, $-\frac{E_t}{100}$, and N^{c}_t can decrease. On the other hand, each scalar reward should decrease by similar size based on their original rewards. For example, if $\varepsilon_1 = 4$, $\varepsilon_2 = 4$, and $\varepsilon_3 = -2$, the decrements of the three rewards are $3D_t$, $\frac{3E_t}{100}$, and $3N^{\mathrm{c}}_t$, respectively. The purpose of doing so is to ensure the three scalar rewards are still in the same order of magnitude when the UAV flies out of the rectangular area in t.

Based on the vector-valued reward $\mathbf{r}_t$, we obtain the return which is the summation of the discounted reward generated at each time step over the long run. Let $\mathbf{R}_\pi = (R^{\mathrm{D}}_\pi, R^{\mathrm{E}}_\pi, R^{\mathrm{N}}_\pi)$ be the return of r^{D}_1, r^{E}_1, and r^{N}_1 under policy π at the first time step, defined as

$$R^{\mathrm{D}}_\pi = -\sum_{t=1}^{T} \gamma^{t-1}(\varepsilon_1 + \mathbb{1}_t - \mathbb{1}_t\varepsilon_1) D_t, \tag{32}$$

$$R^{\mathrm{E}}_\pi = -\sum_{t=1}^{T} \gamma^{t-1}(\varepsilon_2 + \mathbb{1}_t - \mathbb{1}_t\varepsilon_2) \frac{E_t}{100}, \tag{33}$$

$$R^{\mathrm{N}}_\pi = \sum_{t=1}^{T} \gamma^{t-1}(\varepsilon_3 + \mathbb{1}_t - \mathbb{1}_t\varepsilon_3) N^{\mathrm{c}}_t. \tag{34}$$

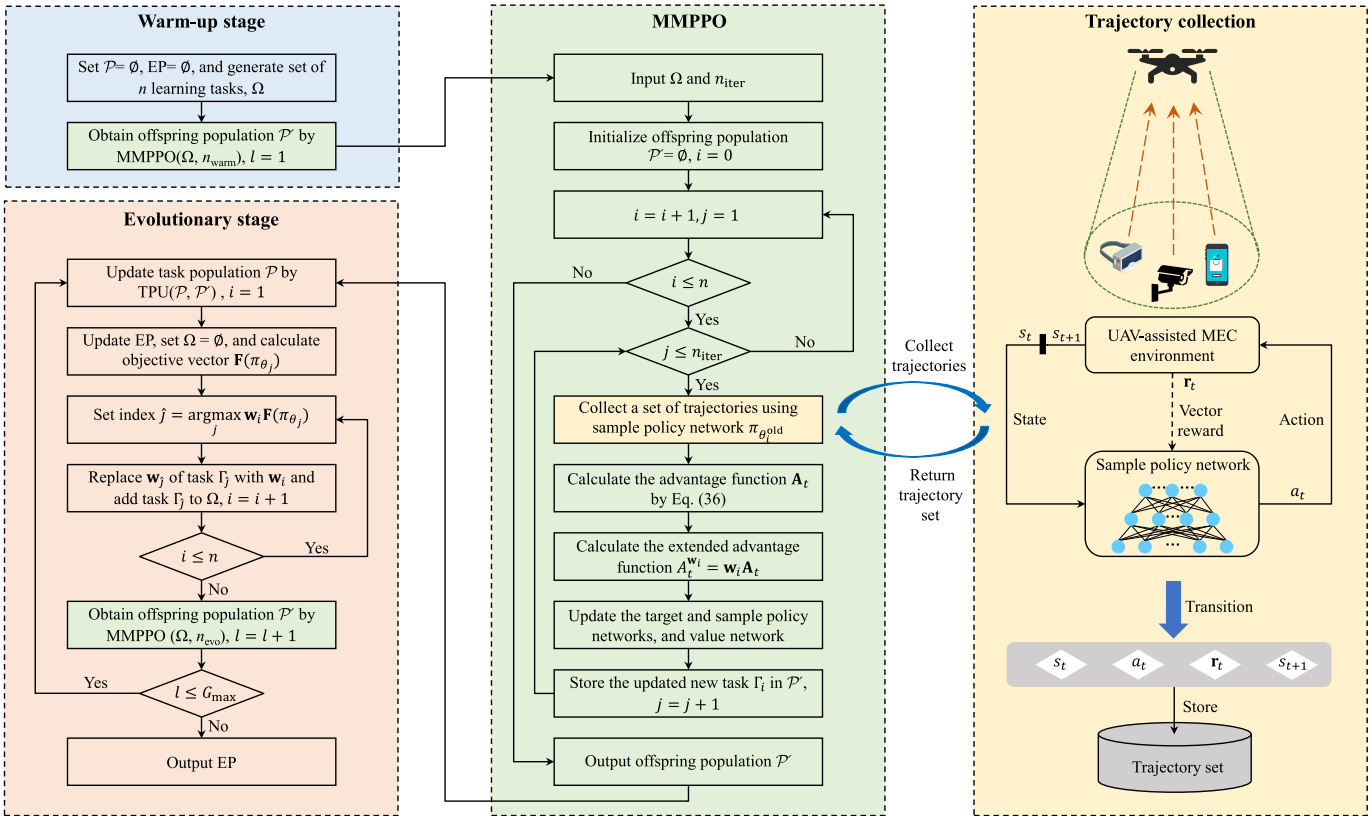


Fig. 2. Framework of EMORL-TCTO.

Maximizing the expected return $\mathbb{E}[\mathbf{R}_\pi]$ is equivalent to minimizing D_{total} and E_{total}, and maximizing N_{total}, simultaneously.

4.2 EMORL-TCTO Algorithm

This paper represents a learning task by tuple $\Gamma = \langle \mathbf{w}, \pi_\theta, \pi_{\theta^{\text{old}}}, \mathbf{V}_{\pi_\theta} \rangle$, where $\mathbf{w}(\sum_{j=1}^{m} w^j = 1)$ is the weight vector. π_θ is the target policy used to select actions and $\pi_{\theta^{\text{old}}}$ is the sample policy used to collect trajectories[1]. $\mathbf{V}_{\pi_\theta}$ is the multi-objective value function for evaluating the selected actions. Through interacting with the environment, the sample policy $\pi_{\theta^{\text{old}}}$ is used to generate the set of trajectories. The generated set is used to update the target policy π_θ for several epochs. To avoid a large update of the target policy, a clipped surrogate objective is adopted, which is defined as

$$J_\Gamma^{\text{C}}(\theta, \mathbf{w}) = \mathbb{E}\left[\sum_{t=1}^{T} \min\left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta^{\text{old}}}(a_t|s_t)} A_t^{\mathbf{w}}, \text{clip}_{1-\epsilon}^{1+\epsilon}\left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta^{\text{old}}}(a_t|s_t)}\right) A_t^{\mathbf{w}}\right)\right], \tag{35}$$

where $A_t^{\mathbf{w}} = \mathbf{w}\mathbf{A}_t$ is the extended advantage function at time step t, i.e., the weighted-sum of all elements in the vector-valued advantage function $\mathbf{A}_t$. $\mathbf{A}_t$ is obtained by the general advantage estimator (GAE) [41], defined as

$$\mathbf{A}_t = \sum_{k=0}^{T-t+1} (\gamma\lambda)^k (\mathbf{r}_{t+k} + \gamma \mathbf{V}_{\pi_\theta}(s_{t+k+1}) - \mathbf{V}_{\pi_\theta}(s_{t+k})), \tag{36}$$

where $\lambda \in [0, 1]$ is a parameter for tuning the trade-off between variance and bias. $\text{clip}_{1-\epsilon}^{1+\epsilon}(\Delta)$ is the clip function that constrains the value of Δ, removing the incentive for moving Δ outside of the interval $[1 - \epsilon, 1 + \epsilon]$.

The value function loss is defined as

$$J_\Gamma^{\text{V}}(\theta) = \mathbb{E}\left[\sum_{t=1}^{T} \|\mathbf{V}_{\pi_\theta}(s_t) - \widehat{\mathbf{V}}_{\pi_\theta}(s_t)\|^2\right], \tag{37}$$

where $\mathbf{V}_{\pi_\theta}(s_t)$ is the value function defined in Eq. (37) and $\widehat{\mathbf{V}}_{\pi_\theta}(s_t) = \mathbf{r}_t + \gamma \mathbf{V}_{\pi_\theta}(s_{t+1})$ is the target value function. Through this extension, the value function trained in the previous learning process can be directly adapted to optimize the same policy with the new weight vectors.

The proposed EMORL-TCTO aims to learn a set of Pareto optimal policies through interacting with the environment and its framework is shown in Fig. 2. EMORL-TCTO shares the same algorithm structure with the original EMORL [38]. EMORL-TCTO starts from the warm-up stage, where n learning tasks are randomly generated. The offspring population is produced by executing the multi-task multi-objective PPO (MMPPO). Note that each learning task uses its associated sample policy to collect a set of trajectories by interacting with the UAV-assisted MEC environment. After the warm-up stage, EMORL-TCTO proceeds with the evolutionary stage. Both the task population and external Pareto (EP) archive are updated based on the offspring population. Then, we select n new learning tasks from the task

1. Note that the term "trajectories" refers to a sequence of transitions in RL, each of which consists of state, action, reward, and next state. However, the term "trajectory" used in the system model represents the UAV's flying path.

population for each weight vector. These tasks are optimized by MMPPO to generate a new generation of the offspring population. The evolutionary stage terminates when a predefined number of generations are completed.

Note that EMORL-TCTO is different from the policy space response oracles (PSRO) [42], a game-theoretic multi-agent RL algorithm. Each agent in PSRO is based on SORL, ignoring the fact that the objectives conflict with each other. In addition, PSRO only obtains an optimal policy for each agent in a run. Hence, PSRO is unsuitable for addressing the TCTO problem due to the two inherent disadvantages above. On the other hand, HRL usually decomposes an optimization problem into multiple sub-problems and adopts the SORL-based methods to slove them [19]. However, the TCTO problem concerned in this paper is not decomposed, i.e., it is solved as a whole by multiple learning tasks, each of which is based on MORL.

The pseudo-code of EMORL-TCTO is shown in Algorithm 1. We elaborate the warm-up and evolutionary stages in detail.

Algorithm 1. Evolutionary Multi-Objective Reinforcement Learning for TCTO Problem (EMORL-TCTO)

Input: number of learning tasks n, number of warm-up iterations n_{warm}, number of task iterations n_{evo}, number of maximum evolution generations $G_{\max}$.

// Warm-up stage

Initialize task population $\mathcal{P} = \emptyset$ and external Pareto archive $\text{EP} = \emptyset$;
Generate n evenly distributed weight vectors $\{\mathbf{w}_1, \ldots, \mathbf{w}_n\}$;
Initialize n target policy networks $\{\pi_{\theta_1}, \ldots, \pi_{\theta_n}\}$;
Initialize the i-th sample policy network, $\pi_{\theta_i^{\text{old}}} \leftarrow \pi_{\theta_i}, i = 1, \ldots, n$;
initialize n value networks $\{\mathbf{V}_{\pi_{\theta_1}}, \ldots, \mathbf{V}_{\pi_{\theta_n}}\}$;
Denote the task set by $\Omega = \{\Gamma_1, \ldots, \Gamma_n\}$, $\Gamma_i = \langle \mathbf{w}_i, \pi_{\theta_i}, \pi_{\theta_i^{\text{old}}}, \mathbf{V}_{\pi_{\theta_i}} \rangle$;
Obtain offspring population $\mathcal{P}'$ by $\text{MMPPO}(\Omega, n_{\text{warm}})$;

// Evolutionary stage

for $l = 1, \ldots, G_{\max}$ **do**
  Update task population $\mathcal{P}$ by $\text{TPU}(\mathcal{P}, \mathcal{P}')$;
  Update EP based on $\mathcal{P}'$;
  Set $\Omega = \emptyset$;
  Calculate $\mathbf{F}(\pi_{\theta_j})$ of target policy π_{θ_j} of each task $\Gamma_j \in \mathcal{P}$;
  for $\mathbf{w}_i \in \{\mathbf{w}_1, \ldots, \mathbf{w}_n\}$ **do**
    Set index $\hat{j} = \arg\max_{j=1,\ldots,|\mathcal{P}|} \{\mathbf{w}_i \mathbf{F}(\pi_{\theta_j})\}$;
    Replace weight vector $\mathbf{w}_{\hat{j}}$ of task $\Gamma_{\hat{j}}$ with $\mathbf{w}_i$;
    Add task $\Gamma_{\hat{j}}$ to Ω;
end for
Obtain offspring population $\mathcal{P}'$ by $\text{MMPPO}(\Omega, n_{\text{evo}})$;
end for

Output: external Pareto archive EP.

4.2.1 Warm-Up Stage

In this stage, n learning tasks are randomly generated. These tasks share the same state space, action space, and reward function but their dynamics may differ. The dynamics means that each learning task will generate various new offspring tasks after running MMPPO once. In general, these offspring learning tasks generated by different tasks have great differences because they have different weight vectors and neural network parameters. The task generation procedure is described as follows.

Algorithm 2. Multi-Task Multi-Objective PPO (MMPPO)

Input: task set Ω, number of iterations n_{iter}.

Initialize offspring population $\mathcal{P}' = \emptyset$;
for $\Gamma_i = \langle \mathbf{w}_i, \pi_{\theta_i}, \pi_{\theta_i^{\text{old}}}, \mathbf{V}_{\pi_{\theta_i}} \rangle \in \Omega$ **do**
  for $j = 1, \ldots, n_{\text{iter}}$ **do**
  Collect a set of trajectories using sample policy $\pi_{\theta_i^{\text{old}}}$;
  Calculate the advantage function $\mathbf{A}_t$ by Eq. (36);
  Calculate the extended advantage function $A_t^{\mathbf{w}_i} = \mathbf{w}_i \mathbf{A}_t$;
  Update the target policy network's parameter θ_i by Eq. (35) for several epochs;
  Update the sample policy network's parameter θ_i^{old}, i.e., $\theta_i^{\text{old}} \leftarrow \theta_i$;
  Update the value network $\mathbf{V}_{\pi_{\theta_i}}$ by Eq. (37);
  Store the updated new task Γ_i in $\mathcal{P}'$;
  end for
end for

Output: Offspring population $\mathcal{P}'$.

Algorithm 3. Task Population Update (TPU)

Input: task population $\mathcal{P}$, offspring population $\mathcal{P}'$, reference point $\mathbf{Z}_{\text{ref}}$, P_{num}, and P_{size}.

Generate P_{num} evenly distributed weight vectors $\{\mathbf{w}_1, \ldots, \mathbf{w}_{P_{\text{num}}}\}$;
Set performance buffer $\mathcal{B}_i = \emptyset, i = 1, \ldots, P_{\text{num}}$;
for $\Gamma = \langle \mathbf{w}, \pi_\theta, \pi_{\theta^{\text{old}}}, \mathbf{V}_{\pi_\theta} \rangle \in \{\mathcal{P} \cup \mathcal{P}'\}$ **do**
  Calculate objective vector $\mathbf{F}(\pi_\theta)$;
  Set $\mathbf{F}_{\text{temp}} = \mathbf{F}(\pi_\theta) - \mathbf{Z}_{\text{ref}}$;
  Set index $\hat{j} = \arg\max_{j=1,\ldots,P_{\text{num}}} \{\mathbf{w}_j \mathbf{F}_{\text{temp}}\}$;
  Store task Γ in $\mathcal{B}_{\hat{j}}$;
  Calculate distance between $\mathbf{F}(\pi_\theta)$ and $\mathbf{Z}_{\text{ref}}$;
  if $|\mathcal{B}_{\hat{j}}| > P_{\text{size}}$ **then**
    Sort all tasks in $\mathcal{B}_{\hat{j}}$ in descending order of their distances;
    Retain the first P_{size} tasks in $\mathcal{B}_{\hat{j}}$;
  end if
end for
Set new task population $\mathcal{P}_{\text{new}} = \{\mathcal{B}_1 \cup, \ldots, \cup \mathcal{B}_{P_{\text{num}}}\}$;

Output: population $\mathcal{P}_{\text{new}}$.

First, the systematic method [43] is adopted to generate n evenly distributed weight vectors, $\mathcal{W} = \{\mathbf{w}_1, \ldots, \mathbf{w}_n\}$. Each weight vector is sampled from a unit simplex. $n = \binom{m+\delta-1}{m-1}$ points with a uniform spacing of $1/\delta$, are sampled on the simplex for any number of objectives, where $\delta > 0$ is the number of divisions considered along each objective axis. As [44] suggests, to obtain intermediate weight vectors within the simplex, we have $\delta > m$. For example, for the TCTO problem with three objectives ($m = 3$), if four divisions ($\delta = 4$) are considered for each objective axis, $n = \binom{3+4-1}{3-1} = 15$ evenly distributed weight vectors are generated. We plot these weights vectors in Fig. 3.

Second, n target policy networks, $\{\pi_{\theta_1}, \ldots, \pi_{\theta_n}\}$, are randomly initialized. The corresponding sample policy networks, $\{\pi_{\theta_1^{\text{old}}}, \ldots, \pi_{\theta_n^{\text{old}}}\}$, are initialized, with their parameters set the same as the target policy networks', i.e., $\theta_i^{\text{old}} = \theta_i, i =$

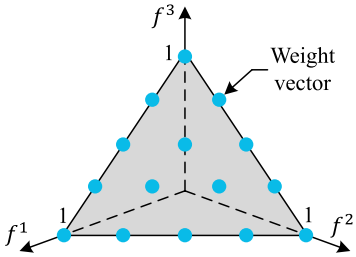


Fig. 3. Fifteen evenly distributed weight vectors for a three-objective problem with $\delta = 4$.

$1, \ldots, n$. Then, n multi-objective value networks, $\{\mathbf{V}_{\pi_{\theta_1}}, \ldots, \mathbf{V}_{\pi_{\theta_n}}\}$, are randomly initialized. In each value network, the number of neurons in the output layer is the same as that of optimization objectives, i.e., m.

Finally, we denote the set of learning tasks by $\Omega = \{\Gamma_1, \ldots, \Gamma_n\}$, where $\Gamma_i = \langle \mathbf{w}_i, \pi_{\theta_i}, \pi_{\theta^{\text{old}}}, \mathbf{V}_{\pi_{\theta_i}} \rangle$. After generating the tasks, we run MMPPO to obtain the offspring population, as shown in Algorithm 2, where each learning task $\Gamma_i \in \Omega$ is optimized by executing multi-objective PPO (steps 3-11) for a predefined number of iterations, Φ (equals to n_{warm} in this stage).

It is quite important for an evolutionary learning algorithm to design a proper operation to generate high-quality offspring learning tasks in the evolution process. This is because high-quality offspring learning tasks inherit the excellent features of parent tasks, which helps to preserve population diversity and improve global exploration. MMPPO in EMORL plays a vital role when generating offspring population $\mathcal{P}'$. However, the original MMPPO only stores the last learning task in $\mathcal{P}'$ after Φ iterations, which may throw away a significant number of promising learning tasks. To overcome this drawback, we improve the original MMPPO by storing each new learning task in $\mathcal{P}'$ after each iteration. In other words, we preserve all the learning tasks generated by MMPPO in the offspring population. Thus, running our MMPPO once can obtain $n \cdot \Phi$ new learning tasks, where n is the number of learning tasks. The improved MMPPO can generate a high-quality offspring population, thus enhancing the MOO performance of the original EMORL.

The warm-up stage can provide a set of promising learning tasks of which policies reside in high-performance region in the search space. To start with these tasks, the EMORL-TCTO's learning process is of low noise, hence more likely to achieve excellent MOO results.

4.2.2 Evolutionary Stage

In this stage, the task population $\mathcal{P}$ is first updated based on the offspring population $\mathcal{P}'$ (step 9 in Algorithm 1). The task population update procedure is shown in Algorithm 3. We adopt the performance buffer strategy in [38] to update $\mathcal{P}$. A number of performance buffers are used to store $\mathcal{P}$ for the purpose of diversity and performance preservation. Let P_{num} and P_{size} denote the number of performance buffers and their size, respectively. The performance space is evenly divided into P_{num} performance buffers, each of which stores P_{size} learning tasks at most. According to the target policy's objective value, $\mathbf{F}(\pi_\theta)$, and a reference point $\mathbf{Z}_{\text{ref}}$, we store the task associated with π_θ in the corresponding performance buffer.

For an arbitrary performance buffer, we sort the tasks in descending order according to their distances to $\mathbf{Z}_{\text{ref}}$. If the number of tasks exceeds P_{size}, we only retain the first P_{size} tasks in that buffer. Finally, the learning tasks in all performance buffers form a new task population.

An EP is employed to store non-dominated policies found during evolution. In each generation, EP is updated based on the offspring population $\mathcal{P}'$ (step 10 in Algorithm 1). For the target policy π_θ of each learning task in $\mathcal{P}'$, we remove those policies dominated by π_θ, and add π_θ to EP if no policies in EP dominates π_θ.

For each weight vector $\mathbf{w}_i \in \mathcal{W}$, We select the best learning task from $\mathcal{P}$ and update the set of learning tasks Ω with it. First, we calculate the objective vector $\mathbf{F}(\pi_{\theta_j})$ of the target policy π_{θ_j} of each learning task $\Gamma_j \in \mathcal{P}, j = 1, \ldots, |\mathcal{P}|$. To be specific, at time step t, state s_t is input to π_{θ_j} which outputs action $a_t = (\vartheta_t, d_t, b_t)$. The UAV takes the action a_t, and it receives the reward $\mathbf{r}_t$ and next state s_{t+1}. The set of immediate rewards $\{\mathbf{r}_1, \ldots, \mathbf{r}_T\}$ is obtained T time steps later. We calculate $\mathbf{F}(\pi_{\theta_j}) = \mathbf{r}_1 +, \ldots, +\mathbf{r}_T$, where "+" is the vector addition. Then, for $\mathbf{w}_i \in \mathcal{W}$, the best learning task in $\mathcal{P}$ is selected based on $\mathbf{w}_i$ and $\mathbf{F}(\pi_{\theta_j})$. Finally, the n selected learning tasks are added to Ω. We obtain $\mathcal{P}'$ by running MMPPO with Ω and n_{evo} as its input, where n_{evo} is the predefined number of task iterations in the evolutionary stage.

The evolutionary stage terminates when a predefined number of evolution generations are completed. All non-dominated policies stored in EP are output as the approximated Pareto optimal policies for the TCTO problem. These policies correspond to different trade-offs between delay, energy consumption and number of tasks, being helpful for decision makers to compromise between conflicting issues/concerns when designing complicated UAV-assisted MEC systems.

Unlike the distributed FRL executes local training and uploads/downloads model parameters [25], [36], EMORL-TCTO is a centralized RL algorithm that needs to consume massive computing resources during training. Thus, the evolutionary learning procedure of our algorithm can be deployed on the edge server with abundant computing resources. Since the UAV is equipped with a global positioning system (GPS) device, the edge server can access the position information of the UAV. Note that we ignore the communication cost between the UAV and edge server for simplicity. EMORL-TCTO outputs a set of non-dominated policies once it is converged. These policies correspond to different trade-offs between objectives, and the decision maker can select the one that matches the current preference. The edge server allocates the selected policy to the UAV, and it generates flight trajectory and task offloading decisions by simple algebraic calculations.

4.2.3 Complexity Analysis

We first analyze the time complexity of EMORL-TCTO shown in Algorithm 1 in the evolutionary process. We analyze the complexity of the outer "for" loop (i.e., steps 8-19 in Algorithm 1) in the evolutionary stage. The loop's time

complexity mainly depends on the generation of the offspring population (i.e., step 18 in Algorithm 1). Compared with step 18, the other steps (i.e., steps 9-17 in Algorithm 1) are trivial and can be ignored. As shown in Algorithm 2, MMPPO generates the offspring population, and its time complexity mainly relies on the training of neural networks. MMPPO iteratively optimizes each learning task Γ_i in task set Ω for n_{iter} times, where n_{iter} stands for the number of task iterations (i.e., steps 2-12 in Algorithm 2). Note that n_{iter} equals to n_{evo} in the evolutionary stage. Let n_{tra} denote the number of the collected trajectories. Let n_{epo} be the number of epochs for training neural network. In our implementation, the policy network and value network use the fully connected neural network. Note that the policy network shares the same neural network structure with the value network, except for the input and output layers. The policy network consists of an input, an output, and L fully connected layers. The numbers of neurons in the input and output layers are 4 and 3, respectively. Let n_l denote the number of neurons in the l-th fully connected layer. We have $n_0 = 4$ and $n_{L+1} = 3$. Thus, the time complexity of MMPPO is $O(n \times (n_{\text{evo}} \times n_{\text{epo}} \times n_{\text{tra}} \times (\sum_{l=1}^{L+1} n_{l-1} \times n_l)))$. We analyze the complexity of EMORL-TCTO. Compared with the evolutionary stage, the time complexity of the warm-up stage is trivial and can be neglected. Therefore, EMORL-TCTO is only dependent on the complexity of MMPPO and the predefined number of maximum evolution generations, $G_{\max}$, leading to a time complexity of $O(G_{\max} \times (n \times (n_{\text{evo}} \times n_{\text{epo}} \times n_{\text{tra}} \times (\sum_{l=1}^{L+1} n_{l-1} \times n_l))))$.

Once the evolutionary process is finished, the decision-maker can select a policy from EP to match the current preference. The selected policy can quickly generate a solution to the TCTO problem through simple algebraic calculations. The computation complexity for generating the solution by the policy network is $O(T \times \sum_{l=1}^{L+1} n_{l-1} \times n_l)$, where T is the number of time slots.

5 Simulation Results and Discussion

In this section, we evaluate the performance of the proposed EMORL-TCTO algorithm for the TCTO problem. A python simulator based on PyTorch 1.7 is developed for performance evaluation. All experiments are implemented in the simulator that is deployed at a computer with Ubuntu 20.04.2 OS, Intel Xeon(R) CPU E5-2667 v4 3.2 GHz, and 128 GB RAM. In the simulation, we consider a rectangular area with the side lengths of $x_{\max} = 400$ m and $y_{\max} = 400$ m. We simulate that the UAV's mission period is 5 minutes and each time slot lasts for 1 second. Therefore, there are $T = 300$ time slots. At the beginning of each mission, the UAV takes off at a random position in the rectangular area. In each time slot, the UAV's maximal flying velocity $v_{\max}$ and distance $d_{\max}$ are set to 30 m/s and 30 m, respectively. The input data size of a computation task, α, and the number of CPU cycles required to execute the task, β, are set to 5 MB and 10^9 cycles, respectively. For each SD, its parameter of Bernoulli random variable is randomly selected from set $\{0.3, 0.5, 0.7\}$. As for the parameters of pathloss, we set A_0, B_0, θ_0, C_0, and η_0 to 3.04, -23.29, -3.61, 4.14, and 20.7, respectively [4].

TABLE 3
Parameter Configurations in Experiments

Parameter	Value
Value used in system model	
Rotor disc area (A)	0.503 m^2
Fuselage drag ratio (d_0)	0.6
Maximal distance the UAV can move ($d_{\max}$)	30 m
Computing capability of the UAV (f_{U})	1 GHz
Rotor solidity (g)	0.05
Maximum number of tasks in the computing queue ($N_{\max}$)	10
Blade profile power (P_1)	79.86
Induced power (P_2)	88.63
Transmission power of the UAV (P_{U})	1 W
Tip speed of rotor blade (U_{tip})	120 m/s
Mean rotor induced velocity in hover (v_0)	4.03
Maximum flying velocity of the UAV ($v_{\max}$)	30 m/s
Channel bandwidth (W)	10 MHz
Air density (ρ)	1.225 kg/m^3
Maximal azimuth angle ($\vartheta_{\max}$)	$\pi/4$
Effective capacitance coefficient (κ)	10^{-26}
Background noise power (σ^2)	10^{-6} W
Value used in reinforcement learning	
Number of maximum evolution generations ($G_{\max}$)	100
Number of the performance buffers (P_{num})	200
Size of each performance buffer (P_{size})	2
Discount factor (γ)	0.995
Clipping parameter (ϵ)	0.2
Parameter of general advantage estimator (λ)	0.95
Number of warm-up iterations (n_{warm})	60
Number of task iterations (n_{evo})	10
Number of divisions of weight vectors (δ)	4

The number of learning tasks n is set to 15. Each task is associated with a weight vector. So, there are 15 weight vectors, as shown in Fig. 3. For each learning task, there are two fully connected layers in the target policy network. Each layer has 64 neurons, with tanh as activation function. The target policy network's output layer uses the sigmoid function to bound actions. Except for the input and output layers, the multi-objective value network shares the same structure and activation function with the target policy network. We use Adam optimizer with a learning rate of 0.0001 to update neural networks. Other parameter configurations are summarized in Table 3.

We introduce the test instances. We consider the number of SDs, K, and the UAV's flying altitude, H, as two important parameters. We specify $K \in \{60, 100, 140\}$ and $H \in \{30, 50\}$ and generate six test instances with different combinations of K and H. These test instances are listed in Table 4.

5.1 Performance Measure

We adopt four widely used evaluation metrics to evaluate the performance of EMORL-TCTO, including the inverted generational distance [45] , hyper volume [38], and comprehensive objective indicator [2], and Friedman test [46].

5.1.1 Inverted Generational Distance (IGD)

Let $\mathcal{F}_{\text{true}}$ and $\mathcal{F}_{\text{app}}$ denote the ture Pareto front and approximated Pareto front found by an MOO algorithm, respectively.

TABLE 4
Test Instance

Instance (K, H)	Number of SDs (K)	Flying altitude (H)
I-(60,30)	60	30
I-(60,50)	60	50
I-(100,30)	100	30
I-(100,50)	100	50
I-(140,30)	140	30
I-(140,50)	140	50

IGD is the average distance from each point v in $\mathcal{F}_{\text{true}}$ to its nearest counterpart in $\mathcal{F}_{\text{app}}$, which is defined as

$$IGD = \frac{\sum_{v \in \mathcal{F}_{\text{true}}} d(v, \mathcal{F}_{\text{app}})}{|\mathcal{F}_{\text{true}}|}, \tag{38}$$

where $d(v, \mathcal{F}_{\text{app}})$ is the euclidean distance between v in $\mathcal{F}_{\text{true}}$ and its nearest point in $\mathcal{F}_{\text{app}}$. IGD can reflect both the convergence and diversity of an approximated Pareto front. An algorithm with a smaller IGD has better performance.

Note that we may not know $\mathcal{F}_{\text{true}}$ when addressing highly complicated MOO problems, like the TCTO problem. In this case, we collect the best-so-far policies found by all algorithms and select those non-dominated from them to mimic the ture Pareto optimal set. We regard the corresponding Pareto front as $\mathcal{F}_{\text{true}}$. This method has been widely used when evaluating MOO algorithms in the literature [45], [46].

5.1.2 Hyper Volume (HV)

Let $\mathbf{Z}_{\text{ref}} \in \mathbb{R}^m$ be the reference point. HV is defined as

$$HV = \int_{\mathbb{R}^m} \mathbb{1}_{H(\mathcal{F}_{\text{app}})(z)dz}, \tag{39}$$

where $H(\mathcal{F}_{\text{app}}) = \{\mathbf{z} | \exists 1 \leq j \leq |\mathcal{F}_{\text{app}}| : \mathbf{Z}_{\text{ref}} \prec \mathbf{z} \prec \mathbf{Z}_j\}$. $\mathbf{Z}_j$ is the j-th point in $\mathcal{F}_{\text{app}}$, and $\mathbb{1}_{H(\mathcal{F}_{\text{app}})}$ is a Dirac delta function that equals 1 if $\mathbf{z} \in H(\mathcal{F}_{\text{app}})$ and 0, otherwise.

The HV metric can measure both the convergence and uniformity of an approximated Pareto front without the true Pareto front known in advance. A larger HV value indicates the corresponding algorithm has better performance. In this paper, we set $\mathbf{Z}_{\text{ref}}$ to the all-zero vector.

Note that before calculating IGD and HV, we normalize the approximated Pareto front via the Min-Max normalization method.

5.1.3 Comprehensive Objective Indicator (COI)

Since the TCTO problem has three objectives, we devise a comprehensive indicator to reflect an MOO algorithm's overall performance, with the task delay, energy consumption, and number of tasks collected taken into account. For each objective vector, we aggregate its objective values into a COI value using the weighted sum method.

Let $\mathbf{F}(\pi) = (f^1(\pi), f^2(\pi), f^3(\pi))$ be the objective vector of policy π in the non-dominated policy set, EP, obtained by an algorithm. Give a weight vector $\mathbf{w} = (w^1, w^2, w^3) \in \mathcal{W}$, we define the COI value of $\mathbf{F}(\pi)$ as

$$COI_{\mathbf{w}}(\mathbf{F}(\pi)) = \mathbf{w} \cdot \mathbf{F}(\pi) = \sum_{j=1}^{3} w^j \cdot f^j(\pi). \tag{40}$$

Based on the COI values, we obtain the objective vector of the best policy in EP, $\pi_{\mathbf{w}}$, associated with $\mathbf{w}$ by Eq. (41).

$$\mathbf{F}(\pi_{\mathbf{w}}) = (f^1(\pi_{\mathbf{w}}), f^2(\pi_{\mathbf{w}}), f^3(\pi_{\mathbf{w}})), \tag{41}$$

where $\pi_{\mathbf{w}} = \arg\max_{\pi \in \text{EP}} COI_{\mathbf{w}}(\mathbf{F}(\pi))$, and $f^1(\pi_{\mathbf{w}})$, $f^2(\pi_{\mathbf{w}})$, and $f^3(\pi_{\mathbf{w}})$ are the best objective values corresponding to D_{total}, E_{total}, and N_{total}, respectively. According to Eqs. (40) and (41), we obtain the best objective vector for each weight vector in $\mathcal{W}$. After that, we calculate the average task delay (ATD), average energy consumption (AEC), average task number (ATN), and average COI (ACOI), defined as

$$ATD = \frac{1}{n} \sum_{\mathbf{w} \in \mathcal{W}} f^1(\pi_{\mathbf{w}}), \tag{42}$$

$$AEC = \frac{1}{n} \sum_{\mathbf{w} \in \mathcal{W}} f^2(\pi_{\mathbf{w}}), \tag{43}$$

$$ATN = \frac{1}{n} \sum_{\mathbf{w} \in \mathcal{W}} f^3(\pi_{\mathbf{w}}), \tag{44}$$

$$ACOI = \frac{1}{n} \sum_{\mathbf{w} \in \mathcal{W}} COI_{\mathbf{w}}(\mathbf{F}(\pi_{\mathbf{w}})). \tag{42}$$

5.1.4 Friedman Test

The Friedman test, a non-parametric test [46], is adopted to measure the differences among MOO algorithms in terms of ATD, AEC, ATN, and ACOI. All algorithms for comparison are ranked, and the average rank scores assigned to them clearly reflect how well they perform.

5.2 Performance Evaluation

We take I-(60,30) in Table 4 as an example to study the UAV's trajectories. Assume the decision-maker's current preferences are represented by $\mathbf{w}_1^{\text{p}} = (1.0, 0.0, 0.0)$, $\mathbf{w}_2^{\text{p}} = (0.0, 1.0, 0.0)$, $\mathbf{w}_3^{\text{p}} = (0.0, 0.0, 1.0)$, and $\mathbf{w}_4^{\text{p}} = (\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$. Preference $\mathbf{w}_1^{\text{p}}$ indicates that one only emphasizes minimizing the total task delay D_{total} without considering objectives E_{total} and N_{total}. Similarly, preferences $\mathbf{w}_2^{\text{p}}$ and $\mathbf{w}_3^{\text{p}}$ aim at minimizing E_{total} and N_{total}, respectively. Preference $\mathbf{w}_4^{\text{p}}$ indicates the three objectives are equally important.

After running EMORL-TCTO once, we can obtain four optimal policies from EP corresponding to the above four preferences. The UAV adopts the four policies to obtain four trajectories through simple algebraic calculations, as shown in Fig. 4. Note that the UAV's take-off point is set to the origin point. The green curve is associated with $\mathbf{w}_1^{\text{p}}$ that aims at minimizing D_{total}. It is observed that the UAV moves in the sparse SD area, helping to reduce the task delay. This is because the fewer tasks collected, the lower the task delay. The magenta curve corresponds to $\mathbf{w}_2^{\text{p}}$ that focuses on minimizing E_{total}. One can observe that the UAV flies a short distance and evades SDs, decreasing its propulsion power consumption and the energy consumed by processing the tasks collected from SDs. The black curve corresponds to $\mathbf{w}_3^{\text{p}}$ that concentrates on maximizing N_{total}. We can observe that the UAV flies to the dense SD area to

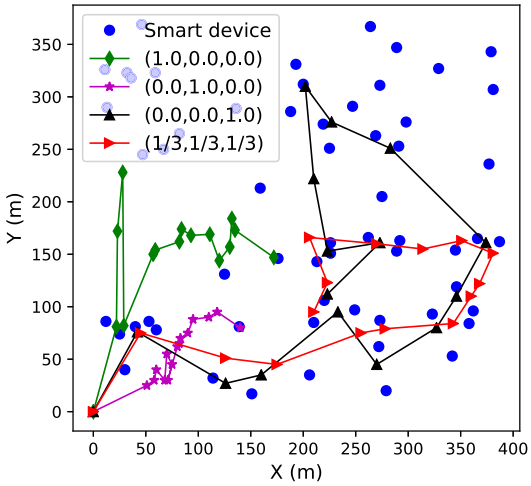


Fig. 4. Trajectories of the UAV under four different preferences.

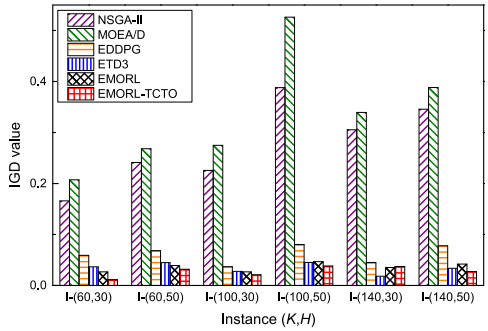


Fig. 5. Results of IGD.

collect more computation tasks without considering delay and energy consumption. The red curve is associated with $\mathbf{w}_4^{\mathrm{p}}$ that aims at minimizing D_{total}, E_{total}, and N_{total} simultaneously, and their importance is equal. It can be seen that the UAV moves in the dense SD area to collect more computation tasks from SDs. However, unlike the black trajectory, the UAV does not fly long distance to collect tasks, because long-distance travel leads to high propulsion power consumption. Based on the above analysis, the proposed EMORL-TCTO can obtain potential control policies according to different preferences in just one run, which validates the effectiveness of our algorithm.

To thoroughly study the performance of EMORL-TCTO, we implement five baseline algorithms for comparison, including two MOEAs, i.e., NSGA-II and MOEA/D, two multi-policy MORLs, i.e., EDDPG and ETD3, and the original EMORL. The compared algorithms are described below.

- NSGA-II: The fast and elitist non-dominated sorting genetic algorithm [47] adopted to minimize the average task delay and average energy consumption. The population size and number of generations are both set to 100. The crossover and mutation probabilities are set to 0.8 and 0.3, respectively.
- MOEA/D: The multi-objective evolutionary algorithm based on decomposition [46] used to minimize the average application completion time and average energy consumption. Both the population size and number of generations are set to 100. The number of neighbors for each subproblem is set to 10.
- EDDPG: The evolutionary DDPG, a variant of EMORL-TCTO that uses a multi-task multi-objective DDPG (MMDDPG) instead of MMPPO, i.e., Algorithm 2. Note that MMDDPG is extended from the single-policy DDPG [39]. We develop EDDPG for performance evaluation purpose.
- ETD3: The evolutionary TD3, another variant of EMORL-TCTO that adopts a multi-task multi-objective TD3 (MMTD3) instead of MMPPO. Note that MMTD3 is extended from the single-policy TD3 [48]. We develop ETD3 for performance evaluation purpose.
- EMORL: The original EMORL used to address continue multi-objective robotic control problems [38].
- EMORL-TCTO: The proposed algorithm in this paper.

In NSGA-II and MOEA/D, each gene in a chromosome represents a trajectory control and task offloading decision in a time slot. For fair comparison, EMORL-TCTO, EDDPG, and ETD3 use the same parameter settings.

The results of IGD and HV are shown in Figs. 5 and 6, respectively. First, one can observe that NSGA-II and MOEA/D, both widely recognized, are the two worst algorithms and cannot find a decent Pareto front in all test instances. This is because when handling high-dimensional MOO problems in dynamic environments, such as the TCTO problem, MOEAs usually spend much time in obtaining decent non-dominated policies and it is hard to converge within a short time [40], [49]. Specifically, it is time-consuming for an MOEA with a large encoding length (i.e., 900) to generate acceptable non-dominated policies. In addition, MOEAs may not have enough time to converge because the UAV-assisted MEC environment is highly dynamic and full of uncertainty. In other words, the dynamics and uncertainty frequently triggers the re-execution of MOEAs from scratch, resulting in high computational burdens and slow convergence speed. That is why NSGA-II and MOEA/D fail to achieve satisfactory performance on the TCTO problem.

Second, all MORLs outperform NSGA-II and MOEA/D in all test instances. Unlike MOEAs that make decisions for all time slots using a single chromosome, MORLs make real-time decision in each time slot according to the current environment state. Moreover, MORLs combine RL with deep neural network and can deal with sequential decision-making problems in the dynamic MEC environment. The reason is that MORLs are able to quickly adapt their behaviors to the changes by interacting with the MEC environment. Hence, MORLs can quickly converge and respond to the requirements of users. This is why an MORL is more appropriate to address the TCTO problem than an MOEA.

Third, EMORL-TCTO obtains the smallest IGD values and the largest HV values in almost all instances except I-(140,30), demonstrating its superiority over the other five algorithms. EMORL-TCTO maintains multiple learning tasks in the evolutionary process. In each generation, these learning tasks are optimized with different weight vectors by MMPPO, resulting in an offspring population that is used to update the external Pareto archive, EP (a non-

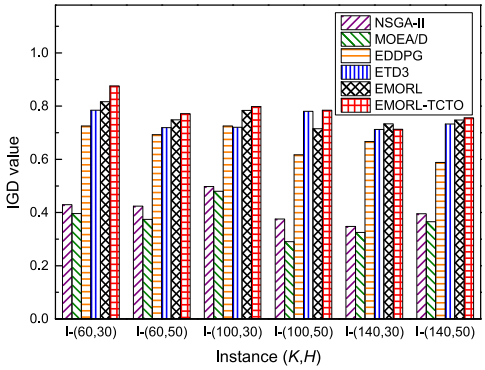


Fig. 6. Results of HV.

dominated policy set). Thus, EMORL-TCTO is able to obtain excellent non-dominated policies. Such experimental results also validate the effectiveness of our improvement in the original EMORL. This is because we improve the original MMPPO in EMORL-TCTO by storing each new learning task in the offspring population $\mathcal{P}'$ after each iteration. In other words, we preserve all the learning tasks generated by MMPPO in $\mathcal{P}'$. The improved MMPPO can generate a high-quality offspring population, thus enhancing the MOO performance of the original EMORL.

To further support our observation above, we plot the convergence curves of IGD and HV obtained by all algorithms in Figs. 7 and 8. It is obvious that EMORL-TCTO is the best among all algorithms in almost all instances except

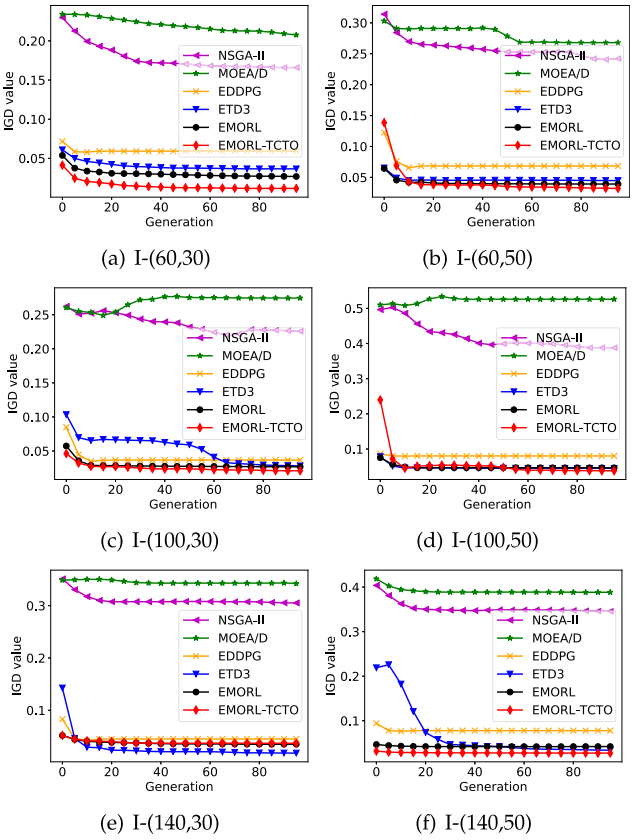


Fig. 7. Convergence curves of six algorithms in terms of IGD.

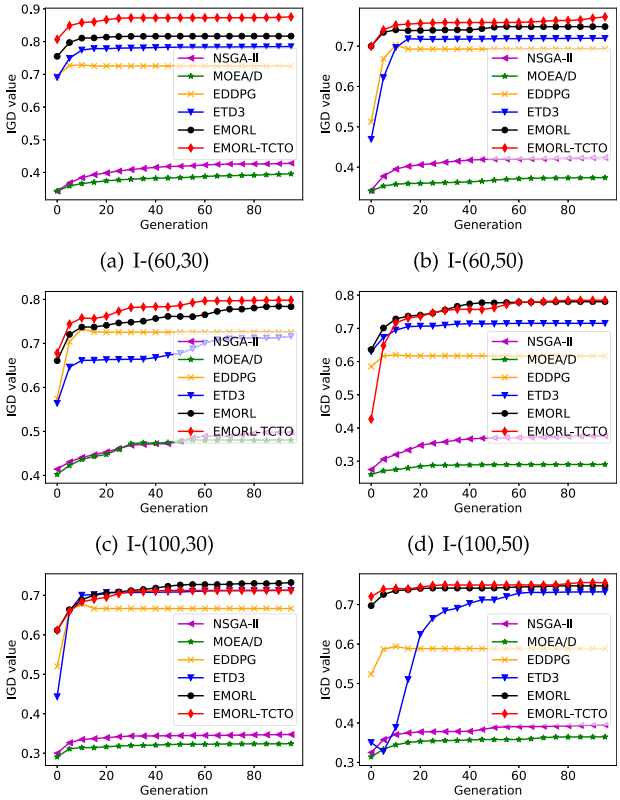


Fig. 8. Convergence curves of six algorithms in terms of HV.

I-(140,30). In addition, EMORL-TCTO converges to an approximate non-dominated policy set, EP, after about 20 generations. In other words, once the convergence of EMORL-TCTO is stable, we terminate its evolutionary learning process and output EP, which can save the running time for the algorithm and reduce the computing resource consumption of the edge server.

Tables 5, 6, and 7 show the ATD, AEC, and ATN values obtained by the six algorithms. Note that the best results are in bold. No matter which one gets fixed, K or H, the corresponding ATD, AEC, and ATN values tend to grow up as the other increases. First, the larger the number of SDs located in the rectangular area, the more the computation tasks need to be collected by the UAV. Second, given that the UAV cannot fly over its maximum allowable altitude, the higher the flying altitude, the larger the UAV's coverage, thus the more the computation tasks can be collected. However, collecting more tasks by the UAV leads to larger task processing delay and higher energy consumption because it has more tasks to handle. Tables 5, 6, and 7 well support this.

In Table 5, it is easily seen that EMORL-TCTO performs better than the other algorithms in four instances except I-(100,50) and I-(140,30). EMORL achieves the smallest ATD values in I-(100,50) and I-(140,30). However, it is worse than EMORL-TCTO in terms of AEC and ATN, with all instances considered. For example, although EMORL obtains the smallest ATD value in I-(100,50) and I-(140,30), its AEC and ATN values are both beaten by EMORL-TCTO's.

TABLE 5
Results of ATD (sec.)

Instance (K, H)	NSGA-II	MOEA/D	EDDPG	ETD3	EMORL	EMORL-TCTO
I-(60,30)	360.6705	449.3261	178.1979	207.3941	182.8770	**171.5592**
I-(60,50)	699.9270	901.6853	180.8765	663.1704	267.7958	**155.1822**
I-(100,30)	684.1330	774.9312	240.4584	258.6074	249.8071	**196.4050**
I-(100,50)	1145.2387	1252.2857	256.8315	329.8663	**208.5477**	313.1781
I-(140,30)	936.4533	1103.2919	230.6772	692.5947	**226.5553**	571.5465
I-(140,50)	1612.1876	1830.5439	448.3674	655.9662	584.5687	**407.1039**

TABLE 6
Results of AEC ($\times 100$ J)

Instance (K, H)	NSGA-II	MOEA/D	EDDPG	ETD3	EMORL	EMORL-TCTO
I-(60,30)	535.7497	607.0149	734.9652	612.6807	436.2488	**423.9196**
I-(60,50)	557.7339	621.7390	720.9539	776.6358	762.8275	**484.3969**
I-(100,30)	**575.7189**	612.6438	762.3671	578.6976	613.0563	581.5113
I-(100,50)	671.6125	578.6909	920.4442	740.0272	823.5713	**547.2583**
I-(140,30)	621.2169	636.7211	850.0938	**567.1289**	711.1420	693.9467
I-(140,50)	682.0360	**669.6477**	857.6836	914.0324	868.8991	812.7823

TABLE 7
Results of ATN

Instance (K, H)	NSGA-II	MOEA/D	EDDPG	ETD3	EMORL	EMORL-TCTO
I-(60,30)	436.5311	488.4442	629.8123	645.4441	682.8770	**692.1570**
I-(60,50)	815.9199	878.4387	1221.3755	1167.5045	1267.7958	**1327.2251**
I-(100,30)	785.3785	827.4552	1127.4746	1149.2537	1249.8071	**1301.4954**
I-(100,50)	1456.8252	1636.5010	1904.8395	1932.8283	1949.8663	**2059.4704**
I-(140,30)	1075.8181	1245.1861	1692.0124	1755.7825	1812.5947	**1982.3445**
I-(140,50)	2241.4241	2313.4791	3201.1267	3595.6318	3684.5687	**3760.8825**

TABLE 8
Results of ACOI

Instance (K, H)	NSGA-II	MOEA/D	EDDPG	ETD3	EMORL	EMORL-TCTO
I-(60,30)	21.0263	42.0819	82.5832	131.7808	173.7920	**184.7818**
I-(60,50)	84.5784	62.2835	340.6043	359.5631	389.5286	**450.1716**
I-(100,30)	64.7223	13.1348	310.7214	354.8805	397.4459	**435.1105**
I-(100,50)	272.1878	229.6503	636.6323	696.2871	704.4896	**733.6050**
I-(140,30)	192.7623	160.3794	583.9968	631.3070	702.2765	**721.8641**
I-(140,50)	564.4810	480.4235	1182.7235	1344.7802	1374.9132	**1456.0290**

As for the AEC values shown in Table 6, EMORL-TCTO outperforms the others in I-(60,30), I-(60,50), and I-(100,50). Although NSGA-II and MOEA/D achieve decent AEC results in I-(100,30) and I-(140,50), they do not perform well regarding ATD and ATN. For example, while NSGA-II obtains the smallest AEC value in I-(100,30), this algorithm causes larger ATD and smaller ATN values than EMORL-TCTO. Similar phenomenon can be observed on MOEA/D. ETD3 obtains the best AEC value in I-(140,30), but its ATD and ATN values are worse than EMORL-TCTO's.

As shown in Table 7, EMORL-TCTO is the best as it results in the largest ATN in every instance. It means EMORL-TCTO allows the UAV to collect sufficient number of computation tasks from SDs by appropriately controlling the UAV's flying trajectory, during its entire mission period.

As aforementioned, the ACOI indicator reflects an MOO algorithm's overall optimization performance. Table 8 lists the results of ACOI obtained by the six algorithms for comparison. It is easily seen that EMORL-TCTO overweighs NSGA-II, MOEA/D, EDDPG, ETD3, and EMORL in all test instances since EMORL-TCTO can better balance between objectives. In addition, the Friedman test is adopted to rank the six algorithms. Based on the ATD, AEC, ATN, and ACOI values, the average rankings and positions of algorithms are calculated and shown in Table 9. One can clearly observe that EMORL-TCTO obtains the best overall performance.

TABLE 9
Rankings of Six Algorithms

Algorithm	ATD		AEC		ATN		ACOI	
	Average rank	Position	Average rank	Position	Average rank	Position	Average rank	Position
NSGA-II	5.0000	5	2.1667	1	6.0000	6	5.1667	5
MOEA/D	6.0000	6	2.8333	2	5.0000	5	5.8333	6
EDDPG	2.0000	2	5.3333	5	3.8333	4	4.0000	4
ETD3	4.0000	4	4.0000	3	3.1667	3	3.0000	3
EMORL	2.3333	3	4.5000	4	2.0000	2	2.0000	2
EMORL-TCTO	1.6667	1	2.1667	1	1.0000	1	1.0000	1

6 Conclusion and Future Work

We model the trajectory control and task offloading (TCTO) problem by multi-objective Markov decision process (MOMDP) and propose an improved evolutionary multi-objective reinforcement learning algorithm, EMORL-TCTO, to address the problem. The proposed algorithm can output plenty of non-dominated policies for various user preferences in each run, clearly reflecting the conflicts between objectives. Compared with NSGA-II, MOEA/D, EDDPG, ETD3, and EMORL, our algorithm strikes better balance between the objectives in almost all instances regarding inverted generational distance and hyper volume. EMORL-TCTO is also the best in most instances with respect to system-related metrics, including the average task delay, average UAV's energy consumption, average number of tasks collected by the UAV, and average comprehensive objective indicator. In addition, EMORL-TCTO takes the first position in the Friedman test. Hence, the performance comparison demonstrates EMORL-TCTO's suitability to tackle the TCTO problem and its potential to be applied to multi-objective UAV-assisted MEC scenarios.

In the future, we will design a multi-UAV-assisted MEC system in which multiple UAVs move constantly and provide large-scale SDs with computation offloading services. In the MEC system, We will formulate an MOO problem, which aims to minimize the processing delay of tasks and energy consumption of UAVs by jointly optimizing offloading decisions and UAV's trajectories. However, due to the high complexity of the collision avoidance and collaboration services between UAVs, how to make computation offloading decisions for large-scale SDs and plan the trajectories of multiple UAVs is still challenging. To address the above problem, we will propose a multi-agent multi-objective reinforcement learning algorithm with the mean-field game [50], where each UAV is regarded as an agent. In this multi-agent system, each UAV considers the flight states of other UAVs to determine their flight trajectories, which can reduce the computational complexity and avoid the inter-UAVs collision.

References

[1] F. Wang, J. Xu, and S. Cui, "Optimal energy allocation and task offloading policy for wireless powered mobile edge computing systems," *IEEE Trans. Wireless Commun.*, vol. 19, no. 4, pp. 2443–2459, Apr. 2020.

[2] F. Song, H. Xing, X. Wang, S. Luo, P. Dai, and K. Li, "Offloading dependent tasks in multi-access edge computing: A multi-objective reinforcement learning approach," *Future Gener. Comput. Syst.*, vol. 128, pp. 333–348, Mar. 2022.

[3] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Commun. Surv. Tuts.*, vol. 19, no. 3, pp. 1628–1656, 3rd Quart. 2017.

[4] C. Zhou et al., "Deep reinforcement learning for delay-oriented IoT task scheduling in SAGIN," *IEEE Trans. Wireless Commun.*, vol. 20, no. 2, pp. 911–925, Feb. 2021.

[5] K. Zhang, X. Gui, D. Ren, and D. Li, "Energy-latency tradeoff for computation offloading in UAV-assisted multiaccess edge computing system," *IEEE Internet Things J.*, vol. 8, no. 8, pp. 6709–6719, Apr. 2021.

[6] Z. Ning et al., "Dynamic computation offloading and server deployment for UAV-enabled multi-access edge computing," *IEEE Trans. Mobile Comput.*, vol. 22, no. 5, pp. 2628–2644, May 2023.

[7] Y. Liu, K. Xiong, Q. Ni, P. Fan, and K. B. Letaief, "UAV-assisted wireless powered cooperative mobile edge computing: Joint offloading, CPU control, and trajectory optimization," *IEEE Internet Things J.*, vol. 7, no. 4, pp. 2777–2790, Apr. 2020.

[8] T. Zhang, Y. Xu, J. Loo, D. Yang, and L. Xiao, "Joint computation and communication design for UAV-assisted mobile edge computing in IoT," *IEEE Trans. Ind. Informat.*, vol. 16, no. 8, pp. 5505–5516, Aug. 2020.

[9] C. Sun, W. Ni, and X. Wang, "Joint computation offloading and trajectory planning for UAV-assisted edge computing," *IEEE Trans. Wireless Commun.*, vol. 20, no. 8, pp. 5343–5358, Aug. 2021.

[10] Y. K. Tun, Y. M. Park, N. H. Tran, W. Saad, S. R. Pandey, and C. S. Hong, "Energy-efficient resource management in UAV-assisted mobile edge computing," *IEEE Commun. Lett.*, vol. 25, no. 1, pp. 249–253, Jan. 2021.

[11] P. A. Apostolopoulos, G. Fragkos, E. E. Tsiropoulou, and S. Papavassiliou, "Data offloading in UAV-assisted multi-access edge computing systems under resource uncertainty," *IEEE Trans. Mobile Comput.*, vol. 22, no. 1, pp. 175–190, Jan. 2023.

[12] W. Ye, J. Luo, F. Shan, W. Wu, and M. Yang, "Offspeeding: Optimal energy-efficient flight speed scheduling for UAV-assisted edge computing," *Comput. Netw.*, vol. 183, Oct. 2020, Art. no. 107577.

[13] J. Zhang et al., "Stochastic computation offloading and trajectory scheduling for UAV-assisted mobile edge computing," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 3688–3699, Apr. 2019.

[14] N. N. Ei, M. Alsenwi, Y. K. Tun, Z. Han, and C. S. Hong, "Energy-efficient resource allocation in multi-UAV-assisted two-stage edge computing for beyond 5G networks," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 9, pp. 16421–16432, Sep. 2022.

[15] X. Chen, C. Wu, T. Chen, Z. Liu, M. Bennis, and Y. Ji, "Age of information-aware resource management in UAV-assisted mobile-edge computing systems," in *Proc. IEEE Glob. Commun. Conf.*, 2020, pp. 1–6.

[16] N. Zhao, Y. Cheng, Y. Pei, Y.-C. Liang, and D. Niyato, "Deep reinforcement learning for trajectory design and power allocation in UAV networks," in *Proc. IEEE Int. Conf. Commun.*, 2020, pp. 1–6.

[17] Y. Liu, S. Xie, and Y. Zhang, "Cooperative offloading and resource management for UAV-enabled mobile edge computing in power IoT system," *IEEE Trans. Veh. Technol.*, vol. 69, no. 10, pp. 12 229–12 239, Oct. 2020.

[18] M. Wang, S. Shi, S. Gu, N. Zhang, and X. Gu, "Intelligent resource allocation in UAV-enabled mobile edge computing networks," in *Proc. IEEE 92nd Veh. Technol. Conf.*, 2020, pp. 1–5.

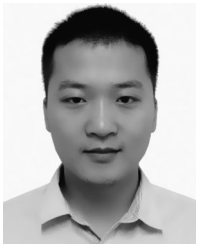
[19] T. Ren et al., "Enabling efficient scheduling in large-scale UAV-assisted mobile edge computing via hierarchical reinforcement learning," *IEEE Internet Things J.*, vol. 9, no. 10, pp. 7095–7109, May 2022.

[20] L. Wang, K. Wang, C. Pan, W. Xu, N. Aslam, and A. Nallanathan, "Deep reinforcement learning based dynamic trajectory control for UAV-assisted mobile edge computing," *IEEE Trans. Mobile Comput.*, vol. 21, no. 10, pp. 3536–3550, Oct. 2022.
[21] B. Dai, J. Niu, T. Ren, Z. Hu, and M. Atiquzzaman, "Towards energy-efficient scheduling of UAV and base station hybrid enabled mobile edge computing," *IEEE Trans. Veh. Technol.*, vol. 71, no. 1, pp. 915–930, Jan. 2022.
[22] A. M. Seid, G. O. Boateng, B. Mareri, G. Sun, and W. Jiang, "Multi-agent DRL for task offloading and resource allocation in multi-UAV enabled IoT edge network," *IEEE Trans. Netw. Service Manag.*, vol. 18, no. 4, pp. 4531–4547, Dec. 2021.
[23] M. Samir, C. Assi, S. Sharafeddine, and A. Ghrayeb, "Online altitude control and scheduling policy for minimizing AoI in UAV-assisted IoT wireless networks," *IEEE Trans. Mobile Comput.*, vol. 21, no. 7, pp. 2493–2505, Jul. 2022.
[24] J. Ji, K. Zhu, and L. Cai, "Trajectory and communication design for cache-enabled UAVs in cellular networks: A deep reinforcement learning approach," *IEEE Trans. Mobile Comput.*, early access, Jun. 13, 2022, doi: 10.1109/TMC.2022.3181308.
[25] Y. Nie, J. Zhao, F. Gao, and F. R. Yu, "Semi-distributed resource management in UAV-aided MEC systems: A multi-agent federated reinforcement learning approach," *IEEE Trans. Veh. Technol.*, vol. 70, no. 12, pp. 13 162–13 173, Dec. 2021.
[26] C. Zhan, H. Hu, X. Sui, Z. Liu, and D. Niyato, "Completion time and energy optimization in the UAV-enabled mobile-edge computing system," *IEEE Internet Things J.*, vol. 7, no. 8, pp. 7808–7822, Aug. 2020.
[27] J. Lin, L. Huang, H. Zhang, X. Yang, and P. Zhao, "A novel Lyapunov based dynamic resource allocation for UAVs-assisted edge computing," *Comput. Netw.*, vol. 205, 2022, Art. no. 108710.
[28] Z. Yu, Y. Gong, S. Gong, and Y. Guo, "Joint task offloading and resource allocation in UAV-enabled mobile edge computing," *IEEE Internet Things J.*, vol. 7, no. 4, pp. 3147–3159, Apr. 2020.
[29] J. Zhu, X. Wang, H. Huang, S. Cheng, and M. Wu, "A NSGA-II algorithm for task scheduling in UAV-enabled MEC system," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 7, pp. 9414–9429, Jul. 2022.
[30] X. Chen, T. Chen, Z. Zhao, H. Zhang, M. Bennis, and J. Yusheng, "Resource awareness in unmanned aerial vehicle-assisted mobile-edge computing systems," in *Proc. IEEE 91st Veh. Technol. Conf.*, 2020, pp. 1–6.
[31] L. Zhang et al., "Task offloading and trajectory control for UAV-assisted mobile edge computing using deep reinforcement learning," *IEEE Access*, vol. 9, pp. 53 708–53 719, 2021.
[32] M. Sun, X. Xu, X. Qin, and P. Zhang, "AoI-energy-aware UAV-assisted data collection for IoT networks: A deep reinforcement learning method," *IEEE Internet Things J.*, vol. 8, no. 24, pp. 17 275–17 289, Dec. 2021.
[33] L. Wang, K. Wang, C. Pan, W. Xu, N. Aslam, and L. Hanzo, "Multi-agent deep reinforcement learning-based trajectory planning for multi-UAV assisted mobile edge computing," *IEEE Trans. Cogn. Commun. Netw.*, vol. 7, no. 1, pp. 73–84, Mar. 2021.
[34] Y. Peng, Y. Liu, and H. Zhang, "Deep reinforcement learning based path planning for UAV-assisted edge computing networks," in *Proc. IEEE Wireless Commun. Netw. Conf.*, 2021, pp. 1–6.
[35] A. Sacco, F. Esposito, G. Marchetto, and P. Montuschi, "Sustainable task offloading in UAV networks via multi-agent reinforcement learning," *IEEE Trans. Veh. Technol.*, vol. 70, no. 5, pp. 5003–5015, May 2021.
[36] Z. Cheng, Z. Gao, M. Liwang, L. Huang, X. Du, and M. Guizani, "Intelligent task offloading and energy allocation in the UAV-aided mobile edge-cloud continuum," *IEEE Netw.*, vol. 35, no. 5, pp. 42–49, Sep./Oct. 2021.
[37] A. Abels, D. Roijers, T. Lenaerts, A. Nowé, and D. Steckelmacher, "Dynamic weights in multi-objective deep reinforcement learning," in *Proc. ACM Int. Conf. Mach. Learn.*, 2019, pp. 11–20.
[38] J. Xu, Y. Tian, P. Ma, D. Rus, S. Sueda, and W. Matusik, "Prediction-guided multi-objective reinforcement learning for continuous robot control," in *Proc. ACM Int. Conf. Mach. Learn.*, 2020, pp. 10 607–10 616.
[39] 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 Trans. Commun.*, vol. 69, no. 9, pp. 6361–6374, Sep. 2021.
[40] T. Sonoda and M. Nakata, "Multiple classifiers-assisted evolutionary algorithm based on decomposition for high-dimensional multi-objective problems," *IEEE Trans. Evol. Comput.*, vol. 26, no. 6, pp. 1581–1595, Dec. 2022.
[41] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," in *Proc. Int. Conf. Learn. Representations*, 2016, pp. 1–14.
[42] M. Lanctot et al., "A unified game-theoretic approach to multiagent reinforcement learning," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 1–14.
[43] K. Li, K. Deb, Q. Zhang, and S. Kwong, "An evolutionary many-objective optimization algorithm based on dominance and decomposition," *IEEE Trans. Evol. Comput.*, vol. 19, no. 5, pp. 694–716, Oct. 2015.
[44] K. Deb and H. Jain, "An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, Part I: Solving problems with box constraints," *IEEE Trans. Evol. Comput.*, vol. 18, no. 4, pp. 577–601, Aug. 2014.
[45] W. Xu, C. Chen, S. Ding, and P. M. Pardalos, "A bi-objective dynamic collaborative task assignment under uncertainty using modified MOEA/D with heuristic initialization," *Expert Syst. Appl.*, vol. 140, Feb. 2020, Art. no. 112844.
[46] F. Song, H. Xing, S. Luo, D. Zhan, P. Dai, and R. Qu, "A multiobjective computation offloading algorithm for mobile-edge computing," *IEEE Internet Things J.*, vol. 7, no. 9, pp. 8780–8799, Sep. 2020.
[47] L. Cui et al., "Joint optimization of energy consumption and latency in mobile edge computing for Internet of Things," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4791–4803, Jun. 2019.
[48] S. Fujimoto, H. Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *Proc. ACM Int. Conf. Mach. Learn.*, 2018, pp. 1587–1596.
[49] Q. Zhang and H. Li, "MOEA/D: A multiobjective evolutionary algorithm based on decomposition," *IEEE Trans. Evol. Comput.*, vol. 11, no. 6, pp. 712–731, Dec. 2007.
[50] L. Li, Q. Cheng, K. Xue, C. Yang, and Z. Han, "Downlink transmit power control in ultra-dense UAV network based on mean field game and deep reinforcement learning," *IEEE Trans. Veh. Technol.*, vol. 69, no. 12, pp. 15 594–15 605, Dec. 2020.

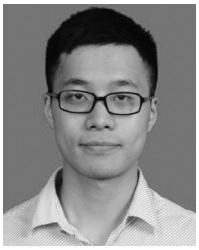
Fuhong Song received the MEng degree in computer technology from Southwest Jiaotong University, Chengdu, China, in 2018. He is currently working toward the PhD degree in computer science and technology with Southwest Jiaotong University. His research interests include edge computing, multi-objective optimization, and reinforcement learning.

Huanlai Xing (Member, IEEE) received the BEng degree in communications engineering from Southwest Jiaotong University, Chengdu, China, in 2006, the MSc degree in electromagnetic fields and wavelength technology from the Beijing University of Posts and Telecommunications, Beijing, China, in 2009, and the PhD degree in computer science from the University of Nottingham, Nottingham, U.K., in 2013. He is an associate professor with the School of Computing and Artificial Intelligence, Southwest Jiaotong University. His research interests include edge computing, network function virtualization, software defined networks, evolutionary computation, multi-objective optimization, and machine learning. He has authored and co-authored more than 60 peer-reviewed journal and conference papers.

Xinhan Wang received the BEng degree in computer science and technology from Southwest University, Chongqing, China, in 2016. He is currently working toward the PhD degree with the School of Computing and Artificial Intelligence, Southwest Jiaotong University, Chengdu, China. His research interests include evolutionary computation, software defined networks, and network function virtualization.

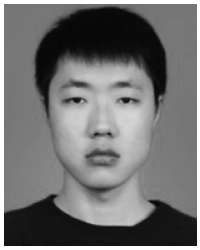


Shouxi Luo (Member, IEEE) received the bachelor's degree in communication engineering and the PhD degree in communication and information systems from the University of Electronic Science and Technology of China, in 2011 and 2016, respectively. He is currently an associate professor with the School of Computing and Artificial Intelligence, Southwest Jiaotong University. His research interests include data center networks, software-defined networking, and networked systems.



Penglin Dai (Member, IEEE) received the BS degree in mathematics and applied mathematics and the PhD degree in computer science from Chongqing University, Chongqing, China, in 2012 and 2017, respectively. He is currently an associate professor with the School of Computing and Artificial Intelligence, Southwest Jiaotong University, Chengdu, China. His research interests include intelligent transportation systems and vehicular cyber-physical systems.

Zhiwen Xiao (Member, IEEE) received his BEng degree in network engineering from Chengdu University of Information Technology in 2019. He is currently working toward the PhD degree in computer science with Southwest Jiaotong University. His research interests are deep learning, federated learning (FL), representation learning, data mining, and computer vision.



Bowen Zhao received the BEng degree in computer science and technology from Southwest Jiaotong University, Chengdu, China, in 2020. He is currently working toward the PhD degree with the School of Computing and Artificial Intelligence, Southwest Jiaotong University, Chengdu, China. His research interests include deep reinforcement learning, time series classification, and mobile edge computing.