Peer-to-Peer Networking and Applications
https://doi.org/10.1007/s12083-021-01245-9

Joint computation offloading and deployment optimization in multi-UAV-enabled MEC systems

Zheyi Chen[1] · **Hongqiang Zheng**[2] · **Jianshan Zhang**[2] · **Xianghan Zheng**[2] · **Chunming Rong**[3]

Received: 3 July 2021 / Accepted: 9 September 2021

Abstract
The combination of unmanned aerial vehicles (UAVs) and mobile edge computing (MEC) technology breaks through the limitations of traditional terrestrial communications. The effective line-of-sight channel provided by UAVs can greatly improve the communication quality between edge servers and mobile devices (MDs). To further enhance the Quality-of-Service (QoS) of MEC systems, a multi-UAV-enabled MEC system model is designed. In the proposed model, UAVs are regarded as edge servers to offer computing services for MDs, aiming to minimize the average task response time by jointly optimizing UAV deployment and computation offloading. Based on the problem definition, a two-layer joint optimization method (PSO-GA-G) is proposed. First, the outer layer utilizes a Particle Swarm Optimization algorithm combined with Genetic Algorithm operators (PSO-GA) to optimize UAV deployment. Next, the inner layer adopts a greedy algorithm to optimize computation offloading. The extensive simulation experiments verify the feasibility and effectiveness of the proposed PSO-GA-G. The results show that the PSO-GA-G can achieve a lower average task response time than the other three baselines.

Keywords Mobile edge computing · Unmanned aerial vehicle deployment · Computation offloading · Particle swarm optimization · Genetic algorithm · Greedy algorithm

1 Introduction

With the rapid development of the Internet-of-Things (IoT) and the fifth-generation (5G) communication technology, numerous emerging mobile applications, such as unmanned driving, virtual reality, and face recognition, have been incorporated into people's daily life [1]. Commonly, such computation-intensive applications are sensitive to delay, and thus they put high demands on the computational capacity of mobile devices (MDs) during execution [2]. However, with the consideration of the existing hardware technology and device portability, the computational capacity of MDs is usually limited, which greatly reduces the performance of the above-mentioned applications when they are executed on MDs. In recent years, mobile edge computing (MEC) has been regarded as a forward-looking technology to solve this issue [3]. Through deploying edge servers at the network edge (e.g., cellular-network base stations and WiFi access points), MEC overcomes the problem of network congestion caused by the traditional cloud-based centralized processing [4]. Therefore, MEC can offer computing services with less time overhead, thereby improving the Quality-of-Service (QoS).

Traditional mobile communications rely on ground communication infrastructures [5]. However, in some isolated areas (e.g., mountains and oceans) or some emergencies (e.g., disaster relief and military exercises), the restricted ground communication infrastructures may have a huge

✉ Zheyi Chen
zc300@exeter.ac.uk

Hongqiang Zheng
zhq921579984@163.com

Jianshan Zhang
zhangjs0512@163.com

Xianghan Zheng
xianghan.zheng@fzu.edu.cn

Chunming Rong
chunming.rong@uis.no

[1] Department of Computer Science, College of Engineering, Mathematics and Physical Sciences, University of Exeter, Exeter EX4 4QF, United Kingdom

[2] College of Mathematics and Computer Science, Fuzhou University, Fuzhou 350116, China

[3] Department of Electronic Engineering and Computer Science, University of Stavanger, Stavanger 4036, Norway

Published online: 25 September 2021

impact on the efficiency of mobile communications [6]. Due to the high flexibility, strong mobility, and low deployment cost, unmanned aerial vehicles (UAVs) are gradually applied to the field of emergency communications [7]. UAVs can be regarded as mobile base stations to provide communication services for MDs on the ground, thereby building an integrated ground-air communication network [8]. There are two advantages when using UAVs as base stations (or edge servers). First, the effective Line-of-Sight (LOS) channel formed by UAVs and MDs can avoid the problems of signal attenuation and penetration loss caused by encountering obstacles. Second, UAVs can flexibly change their deployment positions in the network. When MD or network status changes, UAVs can be quickly adjusted at a lower cost.

However, there are many challenges in applying UAVs to MEC systems. On one hand, the deployment of edge servers in traditional MEC systems usually depends on ground infrastructures. When UAVs are used as edge servers, more flexible UAV deployment strategies are required, and thus the traditional deployment strategies for edge servers might not be applicable. The positions of UAVs directly affect the communication delay and energy consumption in MEC systems, and thus an effective UAV deployment strategy has become critical. On the other hand, the traditional computation offloading strategies usually offload all computing tasks from MDs to edge servers for execution. However, UAVs are not equipped with the equivalent computational resources as traditional edge servers. Therefore, UAVs may not be able to provide computing services for all MDs in their coverage area. The computation offloading strategies directly affect the computing efficiency of UAVs, and thus an effective computation offloading strategy is indispensable.

To address the above challenges, we conduct research work on the optimization problem of UAV deployment and computation offloading in a multi-UAV-enabled MEC system. The goal of this work is to minimize the average task response time by optimizing the positions of UAVs and the offloading strategy in a multi-UAV-enabled MEC system. The main contributions of this paper are summarized as follows.

- A multi-UAV-enabled MEC system is designed. Based on the proposed model, the optimization problem of average task response time with service constraints is well defined.
- Based on the defined optimization problem, a two-layer joint optimization method (PSO-GA-G) is proposed. First, the outer layer utilizes a Particle Swarm Optimization algorithm combined with Genetic Algorithm operators (PSO-GA) to optimize UAV deployment. Next, the inner layer adopts a greedy algorithm to optimize computation offloading. Finally, the optimization problem is well solved by the loop-iteration.
- The extensive simulation experiments are conducted to demonstrate the feasibility and effectiveness of the proposed PSO-GA-G. The results show that the PSO-GA-G outperforms other three baseline methods in terms of average task response time.

The rest of this paper is organized as follows. Section 2 reviews the related work. Section 3 introduces the system model and the problem definition. Section 4 describes the proposed PSO-GA-G in detail. Section 5 presents the evaluation results. Section 6 concludes the work of this paper.

2 Related work

In recent years, MEC has received widespread attention in both industry and academia. MEC can effectively reduce the communication delay between MDs and remote servers by sinking the computational resources from the remote cloud to the network edge, thereby improving the user experience.

Computation offloading is one of the key technologies of MEC. Through reasonable offloading decisions and resource allocation, the computing tasks running on MDs can be offloaded to edge servers, and then these tasks are completed by using sufficient computational and storage resources. Traditional computation offloading technology allows an end device to use a coarse-granularity manner to offload an entire computing task to an edge server [9] or to partition the task via a fine-granularity way and partly offload it to the edge server [10]. For example, Chen et al. [11] supported mobile applications with a novel offloading capability at the granularity of objects for MEC. With the development of 5G networks, a cooperative offloading technology has also emerged by integrating MEC and D2D [12]. Related research shows that the computation offloading technology can effectively shorten the task response time [13], reduce the energy consumption of devices [14], and cut down the communication cost [15].

It is expected that the combination of UAVs and MEC can provide more reliable, lower-latency, and larger-coverage network services [16], which is regarded as a new trend in MEC. Current researches on UAV-enabled MEC systems mainly focus on the problem of computation offloading. Chen et al. [17] introduced a first-come-first-served (FCFS) queuing model, defined the optimization problem as an offloading game, and then designed a distributed offloading algorithm based on the optimal concurrent response. Zhang et al. [18] designed a UAV-enabled MEC system considering offloading decisions and resource competition constraints and proposed a game-theory-based solution to prove the existence of Nash equilibrium. Kim et al. [19] developed an offloading method based on machine learning to optimize system energy consumption and delay. By using

reinforcement learning (RL), Wang et al. [20] proposed an optimization algorithm for task offloading and UAV resource allocation, which showed good optimization effects in large-scale scenarios. Seid et al. [21] designed a collaborative offloading and resource allocation scheme based on model-free deep RL, which checked the UAV status through the Jain fairness index (JFI).

The optimization of UAV deployment is a research hotspot in the UAV field. Yao et al. [22] analyzed the distribution characteristics of UAVs and proposed a random optimal response algorithm to solve the problem of UAV deployment, aiming to minimize the system energy consumption. Yang et al. [23] designed an IoT architecture based on multi-UAV-assisted MEC and proposed a UAV deployment strategy based on a differential evolutionary algorithm, which achieved the load balancing among multiple UAVs. Zhang et al. [24] developed a UAV deployment method based on a discrete PSO algorithm to maximize the UAV coverage. Huang et al. [25] adopted an evolutionary algorithm to handle the problem of UAV deployment, which reduced the system energy consumption by optimizing the number and positions of UAV stops. Fu et al. [26] optimized the UAV deployment and transmission power consumption by using a dual decomposition method. Wang et al. [27] designed an adaptive deployment strategy for UAV-aided communication networks, aiming to enhance the average throughput and the transmission success rate.

The existing researches on UAV-enabled MEC systems mainly focused on reducing the system energy consumption while rarely optimizing the system response time. For example, Sun et al. [28] designed a learning-based cooperative particle swarm optimization (LCPSO) method with a Markov random field (MRF)-based decomposition strategy, in order to reduce the response time of forest fire monitoring in UAV-enabled MEC systems. Peng and Shen [29] proposed a multi-agent deep deterministic policy gradient (MADDPG)-based method to enhance delay/QoS satisfaction ratios in MEC-UAV-Assisted vehicular networks. Zhang and Ansari [30] developed an approximation algorithm to improve the average user latency in UAV-aided MEC networks. Different from these researches, our work aims to minimize the average task response time in a multi-UAV-enabled MEC system by jointly optimizing UAV deployment and computation offloading.

3 System model and problem definition

As shown in Fig. 1, a multi-UAV-enabled MEC system is considered. In this system, a set of MDs are unevenly distributed on the ground, donated by $M = \{MD_1, MD_2, ..., MD_m\}$, and a set of UAVs are deployed in the air for offering computing services to MDs, donated by $N = \{UAV_1, UAV_2, ..., UAV_n\}$, where m and n indicate the number of MDs and UAVs, respectively. Assume that MD_i $(1 \le i \le m)$ has a computing task to be performed, denoted by $Task_i = (D_i, Si)$, where D_i represents the size of the input data (bits) of $Task_i$ and S_i represents the amount of computing resources (CPU cycles) required for processing one bit of data. Moreover, MDs and UAVs exchange data via wireless channels and MDs can offload all their computing tasks to UAVs for execution.

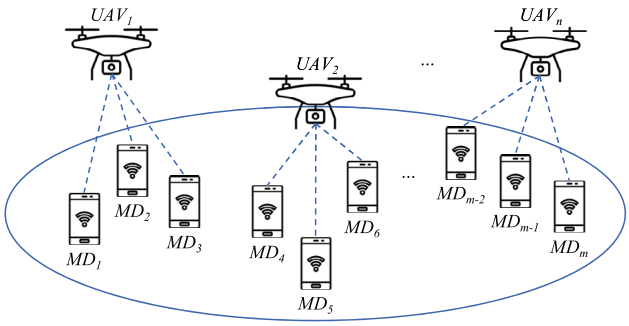

Fig. 1 A multi-UAV-enabled MEC system

3.1 Communication model

The positions of MDs and UAVs are expressed in the form of three-dimensional Cartesian coordinates. For example, the positions of MD_i $(1 \le i \le m)$ and UAV_j $(1 \le j \le n)$ are respectively defined as $p_i^{MD} = (x_i^{MD}, y_i^{MD}, 0)$ and $p_i^{UAV} = (x_j^{UAV}, y_j^{UAV}, H)$, where H represents the flying height of UAVs. To simplify the model, it is assumed that all UAVs are deployed at the same height and their positions remain unchanged after deployment in a certain time slot [31].

Due to the high positions of UAVs, the LOS channel loss of the communication link between UAVs and MDs becomes significant. Therefore, the wireless channel between MD_i and UAV_j can be simulated by using the free-space-path-loss (FSPL) model [32], where the power gain of the wireless channel link is calculated by

$$h_{i,j} = g_0 d_{i,j}^{-2} = \frac{g_0}{H^2 + ||p_i^{MD} - p_i^{UAV}||}, \quad (1)$$

where g_0 represents the channel power gain of spatial distance per unit and $d_{i,j}$ represents the spatial distance between MD_i and UAV_j.

To reduce the transmission delay, MD_i always uses its maximum transmission power P_i for data transmission, in order to improve the signal-to-noise ratio (SNR) of the communication link. Therefore, the data transmission rate between MD_i and UAV_j is

$$R_{i,j} = B\log_2(1 + \frac{P_i h_{i,j}}{\sigma^2}), \tag{2}$$

where B and σ^2 represent the channel bandwidth and the white Gaussian noise (WGN) power in the communication link, respectively.

3.2 Computing model

Considering that all UAVs and MDs can provide computing services, the computing tasks on MDs can be offloaded to UAVs and also executed locally. The offloading matrix between MDs and UAVs is defined as

$$A = \begin{bmatrix} \alpha_{1,1} & \cdots & \alpha_{1,j} & \cdots & \alpha_{1,n} & \alpha_{1,n+1} \\ \vdots & \ddots & \vdots & \ddots & \vdots & \vdots \\ \alpha_{i,1} & \cdots & \alpha_{i,j} & \cdots & \alpha_{i,n} & \alpha_{i,n+1} \\ \vdots & \ddots & \vdots & \ddots & \vdots & \vdots \\ \alpha_{m,1} & \cdots & \alpha_{m,j} & \cdots & \alpha_{m,n} & \alpha_{m,n+1} \end{bmatrix}, \tag{3}$$

where $\alpha_{i,j} \in \{0, 1\}$. When the computing tasks are offloaded from MD_i to UAV_j for execution, $\alpha_{i,j} = 1$; otherwise, $\alpha_{i,j} = 0$. Especially, when $\alpha_{i,n+1} = 1$, the computing tasks are executed on MD_i.

Therefore, there are two optional execution manners for the computing tasks on each MD as follows.

- Local execution. When $\alpha_{i,n+1} = 1$, the computing tasks are executed on MD_i. According to the definition of $Task_i$, the total number of CPU cycles required to execute the task is $S_i \times D_i$. Thus, the time required to execute a task on MD_i is calculated by

$$T_i^{loc} = \frac{S_i \times D_i}{f_i^{MD}}, \tag{4}$$

where f_i^{MD} indicates the CPU frequency of MD_i.
- Offloading execution. When $\alpha_{i,j} = 1$, the computing tasks are offloaded from MD_i to UAV_j for execution. Thus, the completion time of a task consists of three parts, including the data transmission time (denoted by $T_{i,j}^{tran}$), the task execution time (denoted by $T_{i,j}^{UAV}$), and the back time of results (denoted by $T_{i,j}^{back}$). Specifically, $T_{i,j}^{tran}$ and $T_{i,j}^{UAV}$ are calculated by

$$T_{i,j}^{tran} = \frac{D_i}{R_{i,j}}, \tag{5}$$

$$T_{i,j}^{UAV} = \frac{S_i \times D_i}{f_j^{UAV}}, \tag{6}$$

where f_j^{UAV} indicates the CPU frequency of UAV_j. It is noticed that each UAV can execute multiple tasks in parallel, and thus the computational resources of a UAV would be equally allocated by the offloaded tasks. When tasks are completed, the execution results would be sent back to the corresponding MDs. Since the amount of the back data is usually small, $T_{i,j}^{back}$ is negligible. Thus, when a task is offloaded from MD_i to UAV_j for execution, the task completion time is calculated by

$$T_{i,j}^{off} = T_{i,j}^{tran} + T_{i,j}^{UAV}. \tag{7}$$

3.3 Problem definition

Based on the above analysis, to minimize the average task response time of a multi-UAV-enabled MEC system, the optimization problem P1 is defined as

$$\begin{aligned} \text{P1}: & \min_{P^{UAV}, A} \frac{1}{m}\sum_{i=1}^{m}(\alpha_{i,n+1}T_i^{loc} + \sum_{j=1}^{n}\alpha_{i,j}T_{i,j}^{off}), \\ s.t.\ & C1: 0 \le x_j^{UAV} \le X_{\max}, \\ & C2: 0 \le y_j^{UAV} \le Y_{\max}, \\ & C3: \alpha_{i,j} \in \{0, 1\}, \\ & C4: \sum_{j=1}^{n}\alpha_{i,j} = 1, \\ & C5: 0 \le \sum_{i=1}^{m}\alpha_{i,j} \le N_{\max}, \end{aligned} \tag{8}$$

where $\forall i \in \{1, 2, ..., m\}$ and $\forall j \in \{1, 2, ..., n\}$. $C1$ and $C2$ indicate the deployment scope constraints of UAVs, $C3$ represents offloading decisions, $C4$ indicates that each MD can only offload tasks to one UAV or execute them locally, and $C5$ means that the number of tasks that each UAV can execute can not exceed the maximum number of concurrent tasks.

4 PSO-GA-G based optimization for UAV deployment and computation offloading

4.1 Problem analysis

The proposed optimization problem is a hybrid nonlinear programming problem, whose objective function and the constraint C3 are non-convex, which makes it hard to be directly solved. As a swarm intelligence search algorithm, the PSO algorithm does not use the gradient information of functions. Therefore, the PSO algorithm has no requirement for the continuity and derivability of functions and shows great potentials to handle the above optimization problem. However, there exist the following two issues when using the

traditional PSO algorithm to deal with the proposed optimization problem.

- The proposed optimization problem involves UAV deployment and computational offloading, but it is difficult for the traditional PSO algorithm to solve this multi-parameter coupling optimization problem.
- The traditional PSO algorithm may easily fall into the local optimum, and thus it might not be able to get the optimal/near-optimal solution.

To address these issues, a two-layer joint optimization method (PSO-GA-G) is proposed by integrating the PSO-GA and the greedy algorithms. In each iteration, the outer layer of the PSO-GA-G utilizes the PSO-GA algorithm to optimize the deployment positions of UAVs, while the inner layer of the PSO-GA-G adopts the greedy algorithm to optimize offloading decisions.

4.2 PSO-GA based optimization for UAV deployment

The PSO-GA algorithm inherits the features of the traditional PSO algorithm, including easy implementation, high accuracy, and fast convergence. Furthermore, by using the update operators of the GA, the PSO-GA algorithm can well avoid the local optimum and obtain a more optimized solution.

4.2.1 Particle coding

The coding manners determine the execution efficiency of algorithms, but it is are difficult to use some traditional coding manners (e.g., binary coding and integer coding) to describe the feasible solutions of P1. In the proposed model, the deployment positions of UAVs are represented by their x-axis and y-axis coordinates, and thus the following mechanism of particle coding is adopted: each particle indicates a deployment plan of all UAVs, and each quantile of the particle is encoded by a two-dimensional vector, which describes the horizontal deployment position of a UAV. At the t-th iteration, the position of the particle k is defined as

$$X_k^t = (x_{k1}^t, x_{k2}^t, ..., x_{kj}^t, ..., x_{kn}^t), \tag{9}$$

where $x_{kj}^t = (X_{kj}^t, Y_{kj}^t)$ represents the horizontal deployment position of UAV_j. X_{kj}^t and Y_{kj}^t indicate the x-axis and y-axis coordinates, respectively.

Figure 2 illustrates an example of particle coding. The particle consists of five quantiles, which describe the deployment positions of five UAVs. Each quantile is composed of a two-dimensional vector, which represents the horizontal coordinates of a UAV. For example, the coding of the 1st quantile is (114, 517), which indicates that the deployment position of the 1st UAV is (114, 517, H).

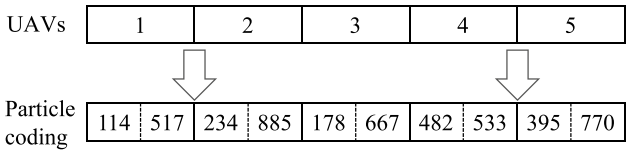

Fig. 2 Particle coding

4.2.2 Fitness function

The fitness function evaluates the quality of particles. When solving the proposed optimization problem, the objective function in Eq. (8) is regarded as the fitness function of the PSO-GA algorithm. Therefore, by inputting UAV deployment and offloading plans into the objective function, the corresponding values of the fitness function of particles can be obtained. Since our optimization goal is to minimize the average task response time, the particles with smaller values of the fitness function exhibit better performance.

4.2.3 Particle updating

In the traditional PSO algorithm, each particle moves in the problem search space with a certain direction and velocity. The particles iteratively adjust their positions and velocities in the problem search space according to the surrounding particles and their own experience. In this process, the velocity and position of a particle are updated as

$$V_k^{t+1} = wV_k^t + c_1 r_1 (pBest_k - X_k^t) + c_2 r_2 (gBest - X_k^t), \tag{10}$$

$$X_k^{t+1} = V_k^{t+1} + X_k^t, \tag{11}$$

where V_k^t and X_k^t represent the velocity and position of the k-th particle at the t-th iteration, respectively. $pBest_k$ and $gBest$ indicate the optimal position of the k-th particle in history and the entire population after t iterations, respectively. w is the inertia factor that determines the influence of the velocity of the last iteration on the current velocity. c_1 and c_2 are individual and social learning factors, which respectively reflect the learning abilities for the optimal value in the individual and population histories. r_1 and r_2 are two random values in the interval [0, 1] that are used to supplement the randomness into the iterative search process.

Based on the PSO algorithm, the PSO-GA algorithm introduces the mutation and crossover operators of the GA. Correspondingly, the update manner of a particle is given as

$$X_k^{t+1} = c_2 \oplus C_g(c_1 \oplus C_p(w \oplus M_u(X_k^t), pBest_k), gBest), \tag{12}$$

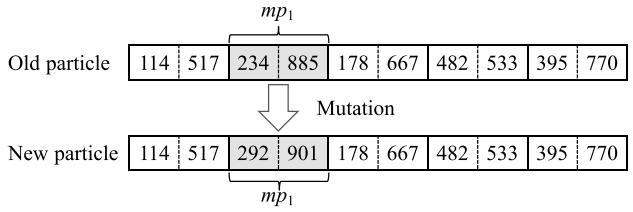

Fig. 3 Mutation operation

where $M_u()$ is the mutation operator. $C_p()$ and $C_g()$ are crossover operators.

Next, through combining with the mutation operator, the inertial part of a particle is updated by

$$A_k^t = w \oplus M_u(X_k^{t-1}) = \begin{cases} M_u(X_k^{t-1}), & r_1 < w \\ X_k^{t-1}, & \text{other cases} \end{cases}, \tag{13}$$

where r_1 is a random value in the interval [0, 1]. $M_u(X_k^{t-1})$ is to randomly select a quantile on the particle X_k^{t-1} and perform the mutation operation within the pre-defined threshold. Figure 3 illustrates an example of mutation operation. In this process, the mutation operator randomly selects a quantile mp_1 and mutates the code on this quantile from (234, 885) to (292, 901).

Moreover, through combining with the crossover operator, the parts of individual and social learning are respectively updated by

$$B_k^t = w \oplus C_p(A_k^t, pBest_k) = \begin{cases} C_p(A_k^t, pBest_k), & r_2 < c_1 \\ A_k^t, & \text{other cases} \end{cases}, \tag{14}$$

$$C_k^t = w \oplus C_g(B_k^t, gBest) = \begin{cases} C_g(B_k^t, gBest), & r_3 < c_2 \\ B_k^t, & \text{other cases} \end{cases}, \tag{15}$$

where r_2 and r_3 are two random values in the interval [0, 1]. The crossover operator is to randomly select two quantiles on the particle and replace the codes between these two quantiles by using the codes between the corresponding quantiles on $pBest_k$ or $gBest$. Figure 4 illustrates an example of crossover operation. In this process, the crossover operator randomly selects two quantiles (i.e., cp_1 and cp_2) and replaces the codes between these two quantiles (i.e., (234, 885), (178, 667), and (482, 533)) by using the codes between the corresponding quantiles on $pBest_k$ or $gBest$ (i.e., (288, 624), (167, 582), and (451, 764)).

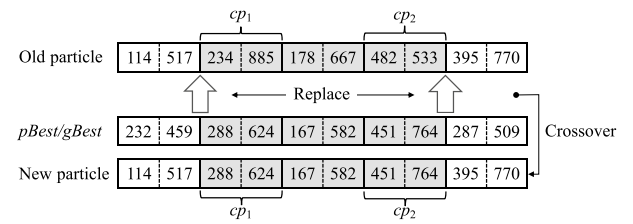

Fig. 4 Crossover operation

4.2.4 Parameter adjustment

The inertia factor w determines the convergence speed and search capability of the PSO-GA algorithm. When the value of w is large, the particles will be mutated with a greater probability, and thus the algorithm owns a stronger global search capability. On the contrary, the particles will be mutated with a lower probability, which leads to a better local search capability.

In the early stage of execution, the PSO-GA algorithm pays more attention to the diversity of the problem search space. As the search deepens, the PSO-GA algorithm will focus on the ability of the local search. Therefore, the value of w should decrease with the increasing number of iterations, whose adjustment manner is defined as

$$w = w_{\max} - iters_{cur} * \frac{w_{\max} - w_{\min}}{iters_{\max}}, \tag{16}$$

where $w_{\max}$ and $w_{\min}$ are the maximum and minimum values of w. $iters_{\max}$ and $iters_{cur}$ are the maximum and current numbers of iterations. Thus, the value of w will decrease as the number of iterations increases.

Moreover, similar adjustment manners are adopted for the individual and social learning factors (i.e., c_1 and c_2) as follows.

$$c_1 = c_{1\_start} - iters_{cur} * \frac{c_{1\_start} - c_{1\_end}}{iters_{\max}}, \tag{17}$$

$$c_2 = c_{2\_start} - iters_{cur} * \frac{c_{2\_start} - c_{2\_end}}{iters_{\max}}, \tag{18}$$

where c_{1_start} and c_{2_start} are the initial values of c_1 and c_2 before the iterations start. c_{1_end} and c_{2_end} are the final values of c_1 and c_2 when the iterations end.

4.2.5 Algorithm implementation

According to Algorithm 1, the main steps of the PSO-GA based optimization for UAV deployment are described as follows.

Step 1: (Lines 1∼11) Initialize the original population (denoted by Pop^0), including the population size (denoted by *size*), *pBest*, and *gBest*.

Step 2: (Lines 12∼25) Execute mutation and crossover operations on Pop^0, calculate the values of the fitness function (denoted by *Fitness*), and iteratively update $pBest_i$ and *gBest* according to *Fitness*.

Step 3: (Line 26) Complete iterations, and output the optimal solution (denoted by P^{UAV}) and average task response time (denoted by T_{avg}).

Algorithm 1 PSO-GA based optimization for UAV deployment

Input: $(P^{MD}, S_i, D_i, f_i^{MD}, f_j^{UAV}, N)$
Output: (P^{UAV}, T_{avg})
Procedure
1: Initialize the original population Pop^0;
2: Call Algorithm 2 to get the offloading matrix A;
3: Calculate $Pop^0.Fitness$ according to the fitness function;
4: $min = Pop_0^0.Fitness$;
5: **for** $i = 0$ **to** $Size$ **do**
6: $pBest_i = Pop_i^0$;
7: **if** $pBest_i.Fitness < min$ **then**
8: $min = pBest_i.Fitness$;
9: $gBest = pBest_i$;
10: **end if**
11: **end for**
12: **for** $iters = 0$ **to** $iters_{\max}$ **do**
13: Update w, c_1, and c_2;
14: **for** $i = 0$ **to** $Size$ **do**
15: Execute mutation and crossover operations on Pop_i^{iters};
16: Call Algorithm 2 to get the offloading matrix A;
17: Calculate $Pop_i^{iters}.Fitness$;
18: **if** $Pop_i^{iters}.Fitness < pBest_i.Fitness$ **then**
19: $pBest_i = Pop_i^{iters}$;
20: **end if**
21: **if** $Pop_i^{iters}.Fitness < gBest.Fitness$ **then**
22: $gBest = Pop_i^{iters}$;
23: **end if**
24: **end for**
25: **end for**
26: Output $P^{UAV} = gBest$ and $T_{avg} = gBest.Fitness$;
End Procedure

4.3 Greedy-based optimization for computation offloading

To obtain a complete solution to the optimization problem P1, reasonable offloading decisions should also be made for the computing tasks from MDs. To address this problem, a greedy algorithm is adopted to quickly realize the optimization of computation offloading.

Basically, the greedy algorithm always makes the current best choice. When the UAVs and MDs are closer, the power gain of the wireless channel link is greater, and thus the time required for offloading will become less. Therefore, the task response time can be effectively reduced by offloading the computing tasks from an MD to the nearest UAV.

However, due to the limited parallel computational capacity of a UAV, it can only offer offloading services for a certain number of MDs. Moreover, for some scattered MDs, even if the tasks are offloaded to the corresponding nearest UAV, the time overhead will be large. In response to these issues, a greedy-based optimization for computation offloading is proposed. According to Algorithm 2, the main steps are described as follows.

Step 1: (Line 1) Initialize each element (denoted by $\alpha_{i,j}$) in the offloading matrix (denoted by A).

Step 2: (Lines 2∼11) Calculate the distance (denoted by $d_{i,j}$) between MD_i and UAV_j, and record the UAV nearest to MD_i (denoted by UAV_{index}).

Step 3: (Lines 12∼17) Calculate the time required to execute a task on MD_i (denoted by T_i^{loc}) and on the nearest UAV (denoted by $T_{i,index}^{off}$), and judge whether the task is executed on the MD or on the UAV.

Step 4: (Lines 18∼22) If the UAV has not exceeded its service constraints (denoted by $N_{\max}$), it can still offer computing services for MDs. Otherwise, the MD with the farthest distance from UAV_{index} will execute the task locally.

Algorithm 2 Greedy-based optimization for computation offloading

Input: $(P^{UAV}, P^{MD}, S_i, D_i, f_i^{MD}, f_j^{UAV}, N, M)$
Output: the offloading matrix A
Procedure
1: Initialize the offloading matrix A with $\alpha_{i,j} = 0$;
2: Calculate the distance $d_{i,j}$ between MD_i and UAV_j;
3: **for** $i = 0$ **to** m **do**
4: $d_i =$ inf;
5: $index = 0$;
6: **for** $j = 0$ **to** n **do**
7: **if** $d_{i,j} < d_i$ **then**
8: $d_i = d_{i,j}$;
9: $index = j$;
10: **end if**
11: **end for**
12: Calculate T_i^{loc} according to Eq. (4);
13: Calculate $T_{i,index}^{off}$ according to Eq. (7);
14: **if** $T_i^{loc} < T_{i,index}^{off}$ **then**
15: $\alpha_{i,n+1} = 1$;
16: **else**
17: $\alpha_{i,index} = 1$;
18: **if** $Sum(index.sum()) > N_{\max}$ **then**
19: Search MD_k with the farthest distance from UAV_{index} and $\alpha_{k,index} = 1$;
20: $\alpha_{k,index} = 0$;
21: $\alpha_{k,n+1} = 1$;
22: **end if**
23: **end if**
24: **end for**
End Procedure

5 Performance evaluation

In this section, we evaluate the performance of the proposed PSO-GA-G for optimizing UAV deployment and computation offloading and make comparisons with the other three baseline methods.

5.1 Experimental setup

The simulation environment is established in an area of 1000×1000 m^2 , which contains 100 MDs. The input data size of computing tasks is randomly distributed in the

Table 1 Parameter settings

Parameter	Value
H	20 m
B	10 MHz
P	1 W
σ^2	-1×10^{-5} dBm
g_0	-20 dB
D	100 cycle
$N_{\max}$	10

interval [10, 20] Mbit, and the CPU frequency of an MD is 1 GHz. In this area, 10 UAVs are deployed, where each UAV can offer computing services for multiple MDs in parallel. The CPU frequency of UAVs is randomly distributed in the interval [2.5, 3.5] GHz. Other parameter settings are shown in Table 1.

Based on the parameter settings, the following four scenarios are considered, where MDs are distributed via different manners.

Scenario 1: This is a typical scenario, and there is a local area (e.g., supermarkets and hospitals) where a large number of MDs are assembled. Specifically, about 90% of MDs are assembled in this local area, and the rest are randomly distributed in other areas. The distribution of MDs in this scenario is shown in Fig. 5a.

Scenario 2: There is also a local area where MDs are assembled in this scenario, but only 50% of MDs are assembled in this area, and the rest are randomly distributed in other areas. Compared with Scenario 1, the assemble density of MDs in a local area is lower. The distribution of MDs in this scenario is shown in Fig. 5b.

Scenario 3: There are two local areas where MDs are assembled (e.g., two workshops in a factory) in this scenario. Specifically, 50% of MDs are assembled in one area, 35% of MDs are assembled in another one, and the rest are randomly distributed in other areas. The distribution of MDs in this scenario is shown in Fig. 5c.

Scenario 4: There is no local area where MDs are assembled in this scenario, and all MDs are randomly distributed (e.g., parks and streets). The distribution of MDs in this scenario is shown in Fig. 5d.

5.2 Baseline methods

To verify the feasibility and effectiveness of the proposed PSO-GA-G, we modify the following three baseline methods

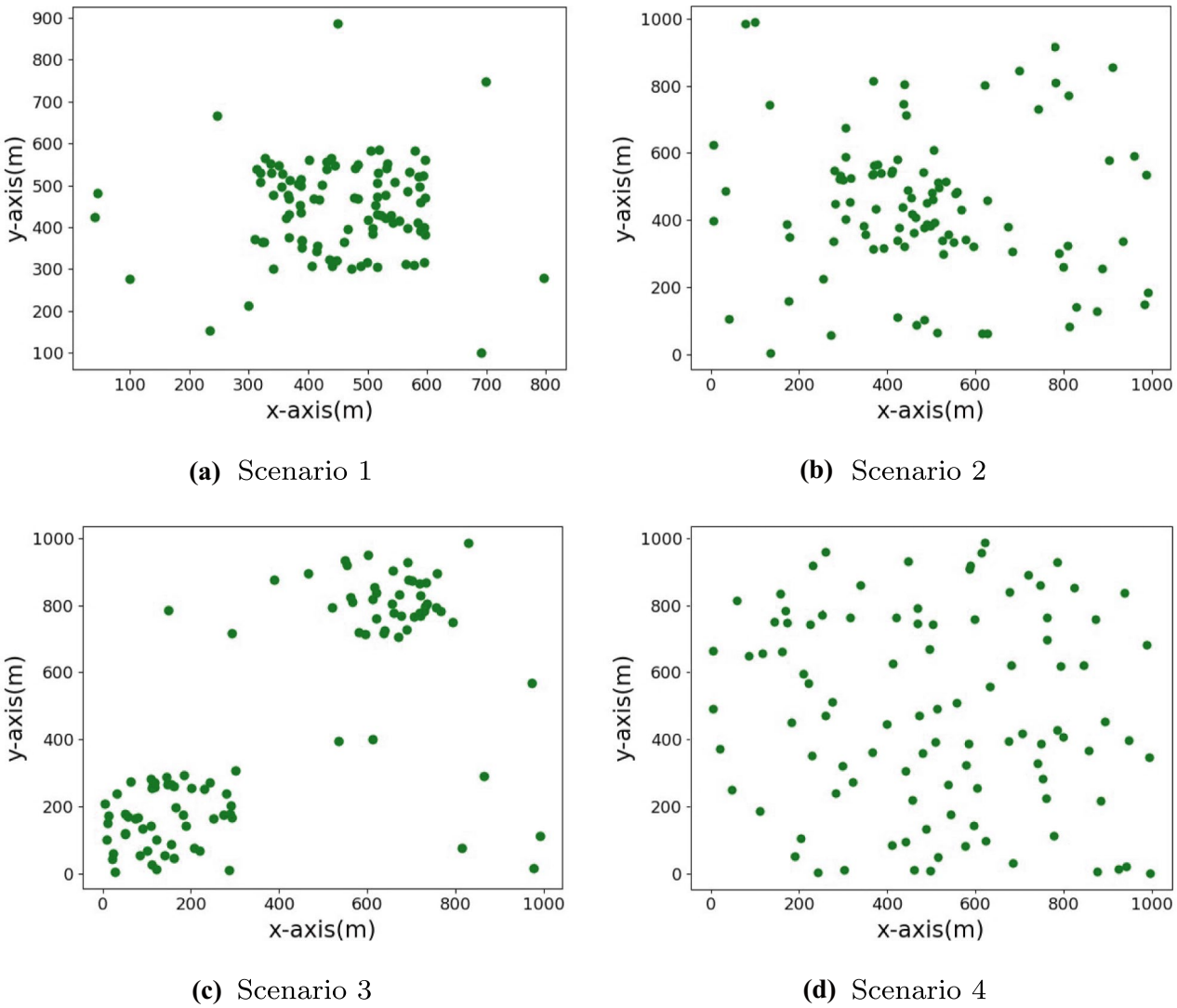

Fig. 5 Different scenarios with various MD distributions

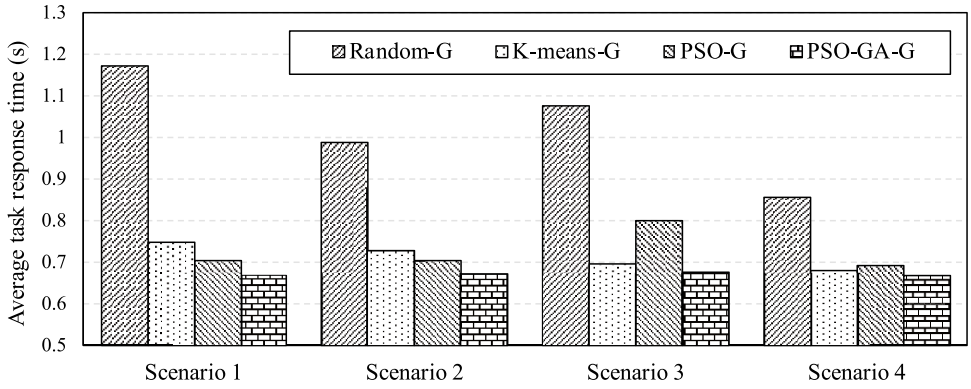

Fig. 6 Average task response time (s) by various methods in different scenarios

to make them fit the proposed problem and then conduct comparative experiments. In the experiments, when making the comparison among various methods, the simulation settings (e.g., the input data size of computing tasks and the CPU frequency of UAVs) are kept constant in each scenario. Based on the above settings, a fair environment is established for making the comparison among various methods.

Random-G. The random strategy and greedy algorithm are adopted to handle the optimization problem of UAV deployment and computation offloading, respectively. In the Random-G, UAV deployment and computation offloading are regarded as two completely independent processes, and the impact of MD positions and resource demands of computing tasks on optimization results is not considered when deploying UAVs.

K-means-G. The K-means [33] and greedy algorithms are adopted to handle the optimization problem of UAV deployment and computation offloading, respectively. In the K-means-G, UAV deployment and computation offloading are regarded as two completely independent processes, and the impact of resource demands of computing tasks on optimization results is not considered when deploying UAVs.

PSO-G. The PSO [24] and greedy algorithms are adopted to handle the optimization problem of UAV deployment and computation offloading, respectively. In the PSO-G, UAV deployment and computation offloading are jointly considered. In each iteration, the particles of UAV deployment and offloading plans are updated for continuously optimizing results.

5.3 Result analysis

To evaluate the performance of the PSO-GA-G, the Random-G, the K-means-G, and the PSO-G, we conduct 50 experiments on each scenario and make averages to increase the stability of the experimental results. The results are shown in Fig. 6. In all these four scenarios, the proposed PSO-GA-G owns the best optimization effect in terms of average task response time while the Random-G performs the worst.

In Scenario 1, the PSO-GA-G is significantly better than the other three baseline methods. The K-means-G always first deploys UAVs according to the distribution of MDs, and then makes offloading decisions based on the UAV deployment results. Therefore, during the process of deploying UAVs, the K-means-G does not consider the impact of resource demands of computing tasks on response time, which makes it impossible to evaluate the UAV deployment results on time. In contrast, the PSO-G and the PSO-GA-G jointly consider the impact of UAV

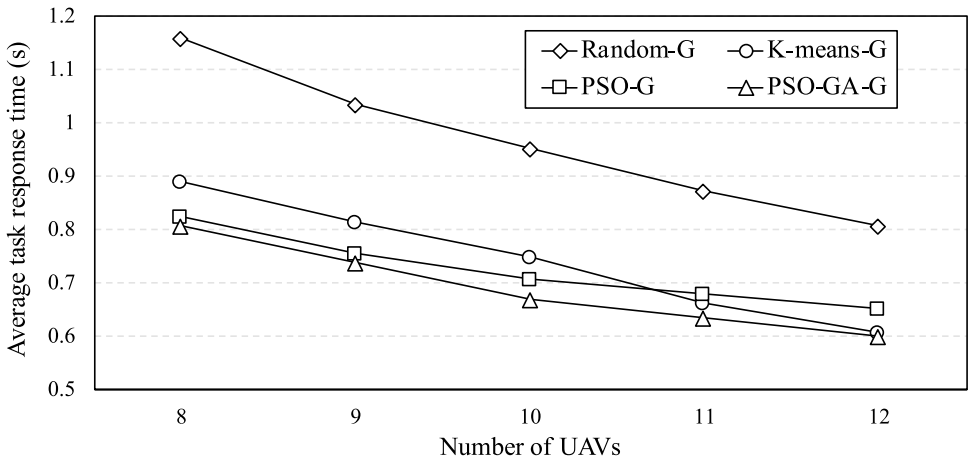

Fig. 7 Average task response time (s) by various methods with different numbers of UAVs

deployment and computation offloading on optimization results in each iteration, and thus they have better performance. By introducing the update operators of the GA, the PSO-GA-G avoids the local optimum and thus obtains better results than the PSO-G.

Compared with Scenario 1, the performance gap between the K-means-G and the PSO-GA-G becomes smaller in Scenario 2. The K-means-G divides MDs into clusters and deploys UAVs at their corresponding centroids. However, the assemble density of MDs in Scenario 1 is higher than in Scenario 2, the K-means-G would still deploy UAVs in positions with scattered MDs, which results in the serious uneven resource utilization of UAVs in different areas. In contrast, the PSO-GA-G avoids deploying UAVs in areas with scattered MDs, which improves the resource utilization of UAVs. In Scenario 2, since the assemble density of MDs is declined, the UAV deployment plan obtained by the K-means-G leads to higher resource utilization of UAVs, and thus the performance of this method can be improved.

Compared with Scenarios 1 and 2, the average task response time achieved by the PSO-GA-G basically remained unchanged, while that of the PSO-G increases greatly. This is because there are two local areas where MDs are assembled in Scenario 1, which easily causes the PSO-G to fall into the local optimum. The results fully demonstrate that the PSO-GA-G, which introduces the update operators of the GA, owns a stronger global search capability.

In Scenario 4, all MDs are randomly distributed. Although the performance gap between the PSO-GA-G and the other three baseline methods is reduced, the PSO-GA-G can still achieve the best optimization results.

Next, based on Scenario 1, we test the performance of the proposed PSO-GA-G by changing the number of UAVs. As shown in Fig. 7, as the number of UAVs increases, the average task response time of all these methods decreases. This is because, when more UAVs are deployed, computing tasks are more likely to be offloaded from MDs to UAVs for execution, and the distance between MDs and UAVs also decreases. Compared to the other three methods, the PSO-GA-G can always achieve better performance. Moreover, the decrease of the average task response time has a nonlinear relationship with the increasing number of UAVs. When the number of UAVs is less than 10, the computational resources of UAVs are insufficient, and thus the system performance can be significantly improved by increasing the number of UAVs. However, when the number of UAVs is greater than 10, the computational resources of UAVs are already sufficient, and thus the system performance cannot be obviously improved by continuing to increase the number of UAVs.

6 Conclusion

In multi-UAV-enabled MEC systems, the response time of the computing tasks from MDs can be greatly reduced by deploying UAVs to provide computation offloading services. In this paper, we propose a two-layer joint optimization method (PSO-GA-G) to jointly optimize UAV deployment and computation offloading. The extensive simulation experiments are conducted to verify the feasibility and effectiveness of the proposed method. The results show that the proposed method can achieve a lower average task response time than other three baseline methods in different scenarios.

The PSO-GA-G can effectively optimize the average task response time, but its training efficiency is not high enough and it may fall into the local optimum when combining the greedy algorithm. In our future work, we will continue to improve the performance of the proposed method according to its limitations. Moreover, we will jointly consider response time and system energy consumption to further research on the optimization problem of resource allocation.

Acknowledgements This work was partly supported by the Key-Area Research and Development Program of Guangdong Province under Grant No. 2020B0101090005.

Author contributions Zheyi Chen and Hongqiang Zheng developed the model and performed experiments. Zheyi Chen wrote the main part of the manuscript, while Hongqiang Zheng provided the support for writing materials. Zheyi Chen and Xianghan Zheng reviewed the manuscript. All the authors read and approved the final manuscript.

Declarations

Ethics approval This work does not contain any studies with human participants or animals performed by any of the authors.

Consent to participate Not applicable.

Consent for publication Not applicable.

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Chettri L, Bera R (2019) A comprehensive survey on Internet of Things (IoT) toward 5G wireless systems. IEEE Internet Things J 7(1):16–32
2. Liu X, Yu M, Ma Y et al (2018) i-Jacob: An internetware-oriented approach to optimizing computation-intensive mobile web browsing. ACM Trans Internet Technol 18(2):1–23
3. Mao Y, You C, Zhang J et al (2017) A survey on mobile edge computing: The communication perspective. IEEE Commun Surv Tutorials 19(4):2322–2358
4. Chen Z, Hu J, Min G et al (2020) Towards accurate prediction for high-dimensional and highly-variable cloud workloads with deep learning. IEEE Trans Parallel Distrib Syst 31(4):923–934
5. Wu H, Wen Y, Zhang J et al (2020) Energy-efficient and secure air-to-ground communication with jittering UAV. IEEE Trans Veh Technol 69(4):3954–396
6. Mu X, Liu Y, Guo L et al (2020) Non-orthogonal multiple access for air-to-ground communication. IEEE Trans Commun 68(5):2934–2949
7. Lin Y, Wang T, Wang S (2019) UAV-assisted emergency communications: An extended multi-armed bandit perspective. IEEE Commun Lett 23(5):938–941
8. Cheng N, Quan W, Shi W et al (2020) A comprehensive simulation platform for space-air-ground integrated network. IEEE Wirel Commun 27(1):178–185
9. Bi S, Zhang Y (2018) Computation rate maximization for wireless powered mobile-edge computing with binary computation offloading. IEEE Wirel Commun 17(6):4177–4190
10. Ning Z, Dong P, Kong X et al (2018) A cooperative partial computation offloading scheme for mobile edge computing enabled Internet of Things. IEEE Internet Things J 6(3):4804–4814
11. Chen X, Chen S, Ma Y et al (2019) An adaptive offloading framework for Android applications in mobile edge computing. Sci China Inf Sci 62(8):82102
12. Saleem U, Liu Y, Jangsher S et al (2020) Latency minimization for D2D-enabled partial computation offloading in mobile edge computing. IEEE Trans Veh Technol 69(4):4472–4486
13. Ren J, Yu G, Cai Y et al (2018) Latency optimization for resource allocation in mobile-edge computation offloading. IEEE Trans Wirel Commun 17(8):5506–5519
14. Guo S, Liu J, Yang Y et al (2018) Energy-efficient dynamic computation offloading and cooperative task scheduling in mobile cloud computing. IEEE Trans Mob Comput 5(4):18(2):319–333
15. Chen MH, Liang B, Dong M (2017) Joint offloading and resource allocation for computation and communication in mobile cloud with computing access point. IEEE Conf Comput Commun pp 1–9
16. Zhou F, Wu Y, Hu RQ et al (2018) Computation rate maximization in UAV-enabled wireless-powered mobile-edge computing systems. IEEE Journal on Selected Areas in Communications 36(9):1927–1941
17. Chen R, Cui L, Zhang Y et al (2020) Delay optimization with FCFS queuing model in mobile edge computing-assisted UAV swarms: a game-theoretic learning approach. IEEE Int Conf Wirel Commun Signal Process. pp 245–250
18. Zhang K, Gui X, Ren D et al (2020) Energy-latency tradeoff for computation offloading in UAV-assisted multi-access edge computing system. IEEE Internet Things J 8(8):6709–6719
19. Kim K, Park YM, Hong CS (2020) Machine learning based edge-assisted UAV computation offloading for data analyzing. IEEE Int Conf Inf Netw. pp 117–120
20. Wang L, Huang P, Wang K et al (2019) RL-based user association and resource allocation for multi-UAV enabled MEC. IEEE Int Wirel Commun Mob Comput Conf. pp 741–746
21. Seid AM, Boateng GO, Anokye S et al (2021) Collaborative computation offloading and resource allocation in multi-UAV assisted IoT networks: A deep reinforcement learning approach. IEEE Internet Things J (Early Access)
22. Yao K, Xu Y, Chen J et al (2020) Distributed joint optimization of deployment, computation offloading and resource allocation in coalition-based UAV swarms. IEEE Int Conf Wirel Commun Signal Process. pp 207–212
23. Yang L, Yao H, Wang J et al (2020) Multi-UAV-enabled load-balance mobile-edge computing for IoT networks. IEEE Internet of Things Journal 7(8):6898–6908
24. Zhang Y, Zhang L, Liu C (2019) 3-D deployment optimization of UAVs based on particle swarm algorithm. IEEE Int Conf Commun Technol. pp 954–957
25. Huang P, Wang Y, Wang K et al (2019) Differential evolution with a variable population size for deployment optimization in a UAV-assisted IoT data collection system. IEEE Trans Emerg Top Comput Intell 4(3):324–335
26. Fu S, Tang Y, Zhang N et al (2020) Joint unmanned aerial vehicle (UAV) deployment and power control for Internet of Things networks. IEEE Trans Veh Technol 69(4):4367–4378
27. Wang Z, Duan L, Zhang R (2019) Adaptive deployment for UAV-aided communication networks. IEEE Trans Wirel Commun 18(9):4531–4543
28. Sun L, Wan L, Wang X (2020) Learning-based resource allocation strategy for industrial IoT in UAV-enabled MEC systems. IEEE Trans Ind Inf 17(7):5031–5040
29. Peng H, Shen X (2020) Multi-agent reinforcement learning based resource management in MEC-and UAV-assisted vehicular networks. IEEE J Sel Areas Commun 39(1):131–141
30. Zhang L, Ansari N (2020) Latency-aware IoT service provisioning in UAV-aided mobile-edge computing networks. IEEE Internet Things J 7(10):10573–10580
31. Zhang X, Zhang J, Xiong J et al (2020) Energy-efficient multi-UAV-enabled multiaccess edge computing incorporating NOMA. IEEE Internet Things J 7(6):5613–5627
32. Xu J, Zeng Y, Zhang R (2018) UAV-enabled wireless power transfer: Trajectory design and energy optimization. IEEE Trans Wirel Commun 17(8):5092–5106
33. Qu H, Zhang W, Zhao J et al (2020) Rapid deployment of UAVs based on bandwidth resources in emergency scenarios. IEEE Inf Commun Technol Conf pp 86–90

Zheyi Chen is currently a Ph.D. candidate in Computer Science at the University of Exeter, United Kingdom. He received his M.Sc. degree in Computer Science from Tsinghua University, China, in 2017, and B.Sc. degree in Computer Science from Shanxi University, China, in 2014. His research interests include cloud computing, mobile edge computing, deep learning, reinforcement learning, and resource optimization.

Hongqiang Zheng is currently working toward the M.S. degree in Computer Software and Theory in the College of Mathematics and Computer Science at the Fuzhou University. He received the B.S. degree in Mechanical Engineering from University of Science and Technology Beijing, Beijing, China, in 2017. His current research interests include system software, edge computing, and cloud computing.

Jianshan Zhang received his M.S. degree in Materials Engineering from Fujian Normal University, China, in 2018. He is currently working toward the Ph.D. degree at the College of Mathematics and Computer Science, Fuzhou University. He has also been a part of the Fujian Key Laboratory of Network Computing and Intelligent Information Processing at Fuzhou University since September 2019. His current research interests include edge computing, computational intelligence and cloud computing.

Xianghan Zheng is currently a Professor with the College of Mathematics and Computer Sciences, Fuzhou University, China. He received the M.Sc. degree in distributed system and the Ph.D. degree in information communication technology from the University of Agder, Norway, in 2007 and 2011, respectively. His current research interests include new generation network with a special focus on cloud computing services and applications, and big data processing and security.

Chunming Rong is a Professor and the head of the Center for IP-based Service Innovation (CIPSI) at the University of Stavanger (UiS), Norway. He is the chair of IEEE Cloud Computing and an executive member of Technical Consortium on High Performance Computing (TCHPC) and the chair of STC on Blockchain in IEEE Computer Society, and served as global co-chair of IEEE Blockchain in 2018. Prof. Rong is alsoadvisor of the StandICT.EU to support European scandalization activities in ICT. He is also co-founder of two start-ups bitYoga and Dataunitor in Norway, both received EU Seal of Excellence Award in 2018. He was adjunct Senior Scientist leading Big-Data Initiative at NORCE (2016-2019), the vice president of CSA Norway Chapter (2016-2017). His research work focuses on cloud computing, data analytics, cyber security and blockchain.

Prof. Rong is an IEEE senior member and is honoured as member of the Norwegian Academy of Technological Sciences (NTVA) since 2011. He has extensive contact network and projects in both the industry and academic. He is also founder and Steering Chair of IEEE CloudCom conference and workshop series. He is co-Editors-in-Chief of the Journal of Cloud Computing (ISSN: 2192-113X) by Springer, has served as the steering chair (2016-2019), steering member and associate editor of the IEEE Transactions on Cloud Computing (TCC) since 2016. Prof. Rong has supervised 26 PhDs, 9 PostDocs and more than 60 master projects. He has extensive experience in managing large-scale R&D projects, both in Norway and EU.