Digital Twin Constructed Spatial Structure for Flexible and Efficient Task Allocation of Drones in Mobile Networks

Bo Yi, *Member, IEEE*, Jianhui Lv, *Member, IEEE*, Jiahao Chen, Xingwei Wang, and Keqin Li, *Fellow, IEEE*

***Abstract*— Applying the Multiple Drones System (MDS) to perform the repetitive and dangerous tasks for human in many complex environments has become a trend all around the world, due to the increasing capacity of mobile communication and the increasing intelligence of drone robots. However, to fulfill the target with less cost as much as possible, drones need to collaborate deeply with each other to make the optimal decision, which is now important and challenging. In this work, we focus on addressing the efficient task allocation among large-scale drones with the object of minimizing the resource waste and cost, which is proved to be NP-hard. Specifically, we first introduce the Digital Twin (DT) technology to dynamically construct the spatial structure for drones, in which a density clustering based algorithm is proposed to decompose the large-scale task allocation problem among all drones into smaller sub-problems among partial drones. Then, for each sub-problem, we propose an improved auction algorithm to allocate the sub-tasks to local drones according to the task difficulty and drone ability. The experimental results indicate that the proposed method outperforms the state-of-the-art methods in terms of the moving distance, resource utilization and task completion time, etc.**

***Index Terms*— Digital twin, drones, mobile network, spatial structure, task allocation.**

I. Introduction

WITH the rapid development of wireless communication technology (e.g., 5G/6G) and Artificial Intelligence (AI) technology, robots are becoming more and more practical and useful, since 5G/6G [1] gives robots the ability to communicate and AI gives robots the ability to think independently [2]. In this case, robots can take the place of human to perform the repetitive or heavy tasks efficiently in some complex and even dangerous environments. For example, the robots can be used to execute the search and rescue tasks in disaster areas. Compared to the manual search and rescue process, robots can finish the task safely using much less time, so as to improve the survival opportunity of disaster victims greatly [3].

Drone, as one particular kind of robots, becomes a hot topic in mobile networks [4]. Similarly, drones can be applied to execute repetitive or heavy tasks for human. The difference is that the ground robot works in a two-dimensional space, while the drone robot works in a three-dimensional space [5], which makes the problem much more complex. Nowadays, the task, as well as its execution environment, both become more and more complex. In this condition, the traditional single-drone mode is not able to handle such complex situation efficiently, due to its capacity and power limitations. Then, multiple simple drones are composed together to execute the same task in a collaborative way, which is referred to as the Multiple Drones System (MDS) [6].

By establishing the group collaborative model among drones, MDS can be used to address much more complex and diverse tasks compared to the single-drone mode. In this case, there are already a lot of research being proposed to discover the most optimal collaboration model among drones [7], [8], [9], [10]. However, when taking the actual environment into consideration, these work typically suffer from: *1) in MDS, each drone needs not only perform its own sub-tasks, but also collaborate with other drones to finish the overall task, which makes the collaboration and decision much more challenging; 2) multiple drones are used to fulfill the same task, so that how to reach the consensus among them on task allocation and how to make a trade-off when task conflict occurs between single-drone and drone-group, are both challenging; 3) the time complexity of obtaining the optimal solution for task allocation among drones is proportional to the number of drones and tasks, which makes this problem NP-hard.*

Aiming at addressing these challenges mentioned above, we first apply the technology of Digital Twin (DT) and the idea of auction theory into our design so as to propose a DT supported novel framework for MDS spatial structure construction, based on which an auction based algorithm is then proposed to address the large-scale task allocation problem among drones in MDS. Specifically, the main contributions of this work are summarized as follows:

- We establish a DT supported novel MDS framework, in which the spatial structure for drone is constructed dynamically and can be used to guide the task allocation and execution.

Manuscript received 13 November 2022; revised 16 May 2023; accepted 4 August 2023. Date of publication 2 October 2023; date of current version 26 October 2023. This work was supported in part by the National Natural Science Foundation of China under Grant 62002055, Grant 62202247, Grant 61872073, Grant 62032013, and Grant U22A2004; and in part by the Fundamental Research Funds for the Central Universities under Grant N2316006. *(Corresponding author: Jianhui Lv.)*

Bo Yi, Jiahao Chen, and Xingwei Wang are with the College of Computer Science and Engineering, Northeastern University, Shenyang 110169, China.

Jianhui Lv is with the Pengcheng Laboratory, Shenzhen 518055, China (e-mail: lvjh@pcl.ac.cn).

Keqin Li is with the Department of Computer Science, The State University of New York, Buffalo, NY 14260 USA.

Color versions of one or more figures in this article are available at https://doi.org/10.1109/JSAC.2023.3313193.

Digital Object Identifier 10.1109/JSAC.2023.3313193

- We mathematically formulate the large-scale task allocation problem among drones in MDS, which is then proved to be NP-hard.
- blueWe decompose the large-scale task allocation problem with discrete clustering objects into several non-intersecting and complete sub-problems for solving using the density clustering and collaboration mechanisms respectively. These sub-problems are to be addressed in parallel to reduce the computing and communication cost.
- We propose an improved auction based algorithm to establish the allocation relationship between drones and sub-tasks in sub-problems, which jointly considers the drone moving distance, drone capacity, task load balance and task completion time, so as to avoid the drone resource waste as much as possible.

The rest of this work are summarized as follows. Section II summarizes the related work. Section III introduces the designed system framework for spatial structure based MDS. Section IV proposes an efficient algorithm to address the task allocation among large-scale drones. Section V shows and analysis the experimental results. Section VI concludes this work.

II. Related Work

The task allocation problem in MDS has been studied by a lot of research. By analyzing the state-of-the-art work, we summarize them into three categories which are the intelligent optimization based task allocation, the auction based task allocation and the spatial structure based task allocation respectively.

A. Intelligent Optimization Based Task Allocation

As explained, the task allocation problem in MDS is NP-hard, so that it usually takes very long time solve the formulated Integer Linear Programming (ILP) [8] problem model. Hence, many intelligent optimization methods are proposed. For example, Ref. [9] proposed to address the large-scale task allocation in the drone network using the genetic algorithm, while Ref. [10] applied the variable neighborhood Simulated Annealing (SA) [11] algorithm. In addition, the particle swarm optimization is also another kind of intelligent optimization method that was leveraged by Ref. [12] to implement a simple but efficient task clustering among drones. Another work using the dynamic distributed Particle Swarm Optimization (PSO) [13] method to solve the task allocation of drones was proposed in [14]. The difference between them was that the former was carried out in a centralized manner while the latter was in a distributed manner.

The SA and PSO based methods were usually unstable, because their corresponding computational convergence speed was greatly affected by the drone environment which was very complex and dynamic. Therefore, Ref. [15] proposed a meta method which generated multiple solutions for the problem, so that the solution robustness towards achieving the optimal single-object was improved. Besides, the robot behavior was also used to help making decisions for task allocation. For example, Ref. [16] separated the robots into two kinds according to their behavior. Then, one kind focused on high-level tasks such as communication and task recognition, while the second kind focused on low-level tasks such as navigation and obstacle avoidance. Moreover, Ref. [17] proposed a threshold based clustering strategy for task allocation, in which the threshold value was calculated also based on the robot behavior to the available resource.

Generally, the SA/PSO based methods are suitable for relatively static environments, while the behavior based intelligent optimization methods are suitable for task allocation in dynamic environments [18]. Nevertheless, we should be aware that the task environment of drone is becoming more and more complex, which makes the traditional intelligent optimization methods no longer effective. Although the behavior based methods provide scalable and robust solution for drone task allocation, they also cost great energy consumption and time.

B. Auction Based Task Allocation

The relationship between tasks and drones can be easily formulated as an auction model, where tasks are the items for auction and drones are bidders. Following this idea, many work have been proposed. For example, targeting on solving the multiple robots collaboration problem, Ref. [19] proposed an auction based algorithm. In this work, the centralized auction mode was used, that is, all the tasks were allocated to bidders in order. However, it is aware that the centralized mode would easily results in performance bottleneck. Reconsidering this situation in the drone task allocation scenario, the node selected to perform auction process would also require high cost and energy to remain reliability, since the failure of this node may stop all drones from working [20].

Then, distributed auction is leveraged to ease such bottleneck. For example, Ref. [21] proposed to solve the drone flying path planning and the task allocation jointly. In particular, the task allocation was solved by a distributed auction algorithm, namely, there were not only one node that can perform the auction process. In this case, the failure of one node would not affect the other drones from working. Besides, Ref. [22] divided the region into several parts and selected one primary drone for each part to execute the task allocation process locally. Similarly, Ref. [23] also proposed a distributed algorithm to address such distributed task allocation problem. Despite this, we should be aware that distributed structure suffers from frequently communication and may not be able to well response to dynamically arriving tasks.

Therefore, the above two auction modes are combined in many other state-of-the-art studies. One typical example was proposed in Ref. [24], where a hybrid auction algorithm was proposed. This work first followed the ideas in [22] and [23]. The difference was that this work also selected a centralized controller for these separated primary nodes.

C. Spatial Structure Based Task Allocation

It has been mentioned that the number of drones is continuously increasing and the environment in which drones execute the tasks is also becoming more and more complex. All these trends lead to the poor performance of both the intelligent optimization and auction based algorithms. In this regard, Ref. [25] was proposed and it created a hierarchical structure among drones, in which some drones were leaders and some were followers. Based on this structure, the task allocation was carried out following the greedy strategy. Obviously, this algorithm reduced the time complexity, but it also suffered from the priori allocation challenge of resource [26]. On one hand, the resource allocation was difficult to obtain in advance. On the other hand, the greedy strategy would easily lead to resource waste.

Another work [27] also formulated the hierarchical structure for drones which were separated into different groups. In particular, this work intended to decompose the problem into two levels, that is, the group level and the individual level. Then, the complex task beyond the individual drone's capacity would be assigned to the group drones. Moreover, Ref. [28] studied the uncertain factors in the drone spatial structure, such as the drone risk level and the chaos drone path. Then, it introduced a task reassignment mechanism to ease the side-effects caused by these uncertain factors.

Nevertheless, it is aware that these work focused on establishing the drone space using a hierarchical structure which suffered from poor fault tolerance.

III. DT Supported MDS Framework and Problem Model

A. System Framework

Generally, the tasks in a MDS scenario can be divided into two categories which are the searching task and the rescue task. In particular, these tasks will gather as clusters in different locations. Besides, the number of drones increases with the increasing of the number of tasks, which results in exponential growth of the computing complexity towards optimal task allocation. Hence, the large-scale drone task allocation in MDS should be considered and optimized. Since drones work in the three-dimensional space, we first introduce DT into this work to construct the spatial structure for drones. Then, in this DT supported spatial structure, we mainly focus on the task cluster division to reduce the complexity when fulfilling the task allocation.

The DT supported MDS framework is designed and shown in Fig. 1, where there are two planes, that is, the real-plane and the emulated-plane. In particular, on one hand, the bottom plane is the real-plane that consists of the elements in the real world. The physical searching drones in this plane will collect the corresponding task and environment data for DT to construct the emulated-plane (i.e., the collected information will be transferred to emulated-plane through the DT channel). On the other hand, the upper plane is the emulated-plane that is constructed using the DT technology, and it focuses on emulating the practical environment on from the real world including the drones, tasks, areas, etc. After the construction of the emulated-plane, it can then be used to guide the drone task allocation in a more flexible manner. Moreover, the drones in the spatial structure are divided into different groups and responsible for tasks in different areas, thus to decompose the large-scale and complex task allocation problem into smaller sub-ones which can then be solved in parallel efficiently.

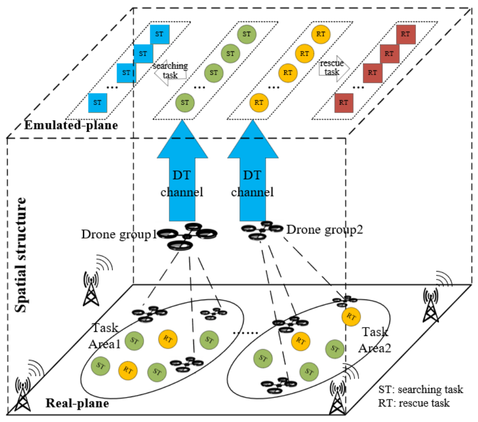

Fig. 1. DT enabled system framework.

B. Problem Model

1) Drone: There are two kinds of drones in this model, which are the searching drone and the rescue drone. The searching drones are equipped with detecting sensors and the rescue drones are equipped with resources such as medicine and food. Thus, we denote the set of drones by $D = \{D^S, D^R\}$, where $D^S = \{d_1^S, d_2^S, \ldots, d_{|D^S|}^S\}$ and $D^R = \{d_1^R, d_2^R, \ldots, d_{|D^R|}^R\}$. Given any drone $d_i \in D$, the maximum number of tasks this drone can carry is set to d_i^l, due to the power limitation of drones. Meanwhile, the capacity of d_i is denoted by d_i^c, which indicates the task difficulty level that can be processed by d_i. The position of any drone can be denoted by a triple $pos(d_i) = (x_i^d, y_i^d, z_i^d)$, where x_i^d, y_i^d, z_i^d indicate the coordinates of three dimensions.

2) Task: Similarly, there are two kinds of tasks and we denote them by $T = \{T^S, T^R\}$, where $T^S = \{t_1^S, t_2^S, \ldots, t_{|T^S|}^S\}$ and $T^R = \{t_1^R, t_2^R, \ldots, t_{|T^R|}^R\}$. In particular, T^S represents the searching tasks and T^R represents the rescue tasks. Given any task $t_j \in T$, it has a difficulty level denoted by t_j^d and the task should be allocated to the drone having close capacity in order to avoid resource waste as much as possible. Each task is also attached with the coordinates that are formulated as the triple $pos(t_j) = (x_j^t, y_j^t, z_j^t)$.

3) Mapping Between Drones and Tasks: Now, the tasks should be allocated to the drones, so that a binary variable is defined as follows:

$$X_i^j = \begin{cases} 1, & \text{if } t_j \text{ is allocated to } d_i. \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

It is noted that $X_i^j = 1$ happens only when the following constraints are satisfied,

$$(d_i \in D^S \&\& t_j \in T^S) || (d_i \in D^R \&\& t_j \in T^R) \tag{2}$$

$$t_j^d \leq d_i^c \tag{3}$$

where the first constraint makes sure that the task is allocated to the same kind of drone and the second constraint guarantees that the drone is able to process the allocated task.

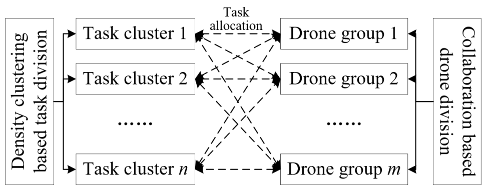

Fig. 2. Basic structure for task allocation.

4) Objective: The object of this model is to minimize the overall flying distance. Actually, it is proportional to the overall cost, because the drones need power to support its flying. In this way, the longer the flying distance, the more power consumed, which naturally leads to higher cost. In addition, we should also be aware that the smaller the distance, the lower the energy consumption and cost, and the faster the rescue speed. Taking these conditions into consideration, our object is to minimize the overall drone flying distance, as follows:

$$\begin{aligned} \text{Minimize:} \quad & \sum_{i\in[1,|D|]} \sum_{j\in[1,|T|]} dis\left(pos(d_i), pos(t_j)\right) X_i^j \\ s.t. \quad & \sum_{i\in[1,|D|]} X_i^j = 1 \\ & 1 \leq \sum_{j\in[1,|T|]} X_i^j \\ & \sum_{j\in[1,|T|]} t_j^d X_i^j \leq d_i^c \\ & (1), (2), (3) \end{aligned} \tag{4}$$

where $dis(*)$ indicates the Euclidean distance between any two points. Assuming that the spatial geographical locations of d_i and t_j are (x_i^d, y_i^d, z_i^d) and (x_j^t, y_j^t, z_j^t) respectively, then the distance between them is calculated as follows:

$$\begin{aligned} & dis\left(pos(d_i), pos(t_j)\right) \\ & \quad = \sqrt{(x_i^d - x_j^t)^2 + (y_i^d - y_j^t)^2 + (z_i^d - z_j^t)^2}. \end{aligned} \tag{5}$$

IV. DT Supported Spatial Structure Based Drone Task Allocation Algorithm

The large-scale drone task allocation problem in MDS is NP-hard and the **proof** can be found in the **Appendix**. Hence, we cannot solve it in linear time simply using the integer linear programming based methods. In this work, to reduce the computing complexity and time complexity, we first decompose the large-scale task allocation problem into several sub-problems which are then solved by a proposed improved auction based algorithm. The basic structure for task allocation is shown in Fig. 2, where the task clusters are generated by density clustering based strategy and the drone groups are created by a collaboration based strategy.

A. Spatial and Density Clustering Based Problem Decomposition

According to the actual disaster situation in real world, the rescue tasks usually exist in clusters, so that the places of task clusters can be viewed as the dense region which is divided by some low-density regions in the spatial space. In this regard, the density clustering based method is leveraged to carry out the task division in the spatial space created by DT.

As explained, the tasks are denoted by the set of T with the size $n = |T|$ and each task t_j is drawn from a three-dimensional space of the real values (x_j^t, y_j^t, z_j^t), where $(x_j^t, y_j^t, z_j^t) \in \mathbb{R}^3$. Now, given any task t_j, we set its nearest neighbor distance as ϵ, based on which two neighborhood functions are firstly defined for t_j as follows:

Definition 1: The *ϵ-nearest neighborhood of the task t_j* is defined by the set of $N_\epsilon(t_j) = \{\theta_j^i | i \in [1, |N_\epsilon(t_j)|]\}$, where the following constraints should be satisfied.

- $N_\epsilon(t_j) \subseteq T/\{t_j\}$,
- $|N_\epsilon(t_j)| = \epsilon$,
- $dis(pos(t_j), pos(t_{j'})) \leq dis(pos(t_j), pos(t_{j''}))$, $\forall t_{j'} \in N_\epsilon(t_j), t_{j''} \in T/(N_\epsilon(t_j) + \{t_j\})$.

Definition 2: The *reverse ϵ-nearest neigborhood of the task* t_j is defined by the set of $R_\epsilon(t_j) = \{\phi_j^i | i \in [1, |R_\epsilon(t_j)|]\}$, where the following constraints should be satisfied.

- $R_\epsilon(t_j) \subseteq T/\{t_j\}$,
- $t_j \in N_\epsilon(t_{j'}), \forall t_{j'} \in R_\epsilon(t_j)$.

It is commonly known that there are three types of samples in density clustering, which are observed to be core, boundary and noise. In this paper, each task is a sample and set to core, boundary or noise according to the distance between it and other tasks. Based on such mapping relationship, it is stated that: 1) a task $t_j \in T$ is a core observation iff $N_\epsilon(t_j) \geq \epsilon$; 2) t_j is a boundary or noise observation if $N_\epsilon(t_j) < \epsilon$. In particular, the core and boundary observations indicate tasks that will be finally clustered, while the noise observations are interfering samples that do not need to be clustered.

There exists the reachability relationship between any two task observations, which is an important metric to generate task clusters, so that we define it as follows:

Definition 3: A task t_j is *directly density reachable* to another task $t_{j'}$, if

- $N_\epsilon(t_j) \geq \epsilon$,
- $t_{j'} \in N_\epsilon(t_j)$.

where the first constraint indicates that t_j should be a core observation and the second constraint indicates that $t_{j'}$ should be within the ϵ-nearest neighbors of t_j. For simplicity, we use $\rightarrow$ to represent *directly density reachable*.

In fact, the directly density reachable is non-symmetric. However, according to definition 3, we can say that a task t_j is *density reachable* (represented by the notation $\rightsquigarrow$) to another task $t_{j'}$ if there exists a chain of observations like $(t_{j_1}, t_{j_2}, \ldots, t_{j_m})$, where $t_{j_1} = t_{j'}, t_{j_m} = t_j$ and t_{j_i} is *directly density reachable* to $t_{j_{i+1}}, \forall i \in (1, m)$. This concludes to the following lemma:

Lemma 1: The density reachable is transitive, that is, t_x is density reachable to t_z, if $\exists t_y$ is density reachable to both t_x and t_z.

Apparently, in definition 3, the *density reachable* is a canonical extension of the *directly density reachable* with the transitive feature. Taking this characteristic into consideration, we have the following theorem:

Theorem 1: The task observation t_j is symmetrically density connected to the task observation $t_{j'}$ if they are both density reachable from $t_{j''}$.

Proof: Let the three task observations be in the same task observation chain denoted by $(t_{j'}, t_{j''}, t_j)$ in sequence, on one hand, $t_{j'}$ is density reachable to $t_{j''}$ and $t_{j''}$ is density reachable from t_j, that is, $t_{j'} \rightsquigarrow t_{j''} \rightsquigarrow t_j$. Based on the transitive characteristic among task observations, we have $t_{j'} \rightsquigarrow t_j$. On the other hand, we denote the observation chain by $(t_j, t_{j''}, t_{j'})$ on the contrary sequence, so that similar reduction process can be applied to achieve that $t_j \rightsquigarrow t_{j''} \rightsquigarrow t_{j'}$. Combining the two cases, Theorem 1 is proved. □

Now, based on the above definitions, we give the task cluster definition as follows:

Definition 4: A cluster C consists of many tasks within the ϵ-nearest neighbors and $C(\neq \emptyset) \subseteq T$ should meet the following constraints that

- $t_i \rightsquigarrow t_j, \forall t_i, t_j \in C$,
- $t_j \in C(\subseteq T), \forall t_i \in C \&\& t_i \rightsquigarrow t_j$.

It is noted that there are soft clustering and hard clustering. The former allows one task been assigned to multiple clusters while the latter do not. In this work, we also define that one task can only be assigned to one cluster. Despite this, we should be aware that, for any boundary task $\in C$, it may also be the boundary of another cluster C', such that a selection process must be provided to fulfill this constraint. Since the boundary tasks do not have much effects on the overall clustering, so that a random approach is used to make the selection.

Given any cluster C, it includes many tasks (core or boundary). To better describe the features of C, we define that the density of C is equal to the maximum directly density reachable distances among all core task observations in C. In this way, the density of C is calculated as follows:

$$dens(C) = \max_{t_i, t_j \in C} dis(pos(t_i), pos(t_j)), \tag{6}$$

where $1) N_\epsilon(t_i) \geq \epsilon; 2) N_\epsilon(t_j) \geq \epsilon; 3) t_i \rightsquigarrow t_j; 4) t_i, t_j \in C$, should be satisfied.

Based on the above definitions, we can carry out the task clustering process in terms of the task set T and the nearest neighbor parameter ϵ. Firstly, for any initial and unclustered task t_j, we first calculate its neighbors as follows:

$$neighbors(t_j) = N_\epsilon(t_j) + \{t_{j'} \in R_\epsilon(t_j) : |R_\epsilon(t'_j)| \geq \epsilon\}. \tag{7}$$

For any neighbor of t_j, let's say $n_i \in neighbors(t_j)$, it follows two conditions that 1) n_i will be classified in a new cluster as the core if it is unclustered and $N_\epsilon(n_i) \geq \epsilon$. 2) otherwise n_i is in the cluster whose core is t_j. Following the above principle, we formulate the task division problem as a directed ϵ nearest neighbor graph denoted by $G_\epsilon(V, E)$, where V equals to the task set T and E describes the connection relations. For example, for any $(u, v) \in E$ $(u, v \in V)$, let u be the core task observation, then, v is one of the ϵ nearest neighbors of u (i.e., $v \in N_\epsilon(u)$). In this way, the core nodes in V is calculated as follows:

$$core = \{v \in V | N_\epsilon(v) \geq \epsilon\} \tag{8}$$

According to (8), we can obtain some new clusters denoted by $C = \{C_1, C_2, \ldots, C_l\}$ which is initially defined by weakly connected components (i.e., tasks) within the core observation subgraph $G_\epsilon/(V/core)$. Then, these new clusters will be expanded by the breadth first search through all unclustered reachable task observations. Specifically, given any unclustered node $v \notin C_i, \forall i \in [1, l]$, if $\exists u$, making

1) $u \in C_i$,
2) $u \in core$,
3) $v \in N_\epsilon(u)$,

satisfied, v is assigned to the cluster C_i by cluster expansion of $C_i = C_i \cup v$. Repeating the above process until all tasks are clustered.

According to the calculated task clusters, we next need to separate drones into different groups, so that one drone group can serve one task cluster. As explained, the tasks around the same area are in one cluster, so that the smaller the task cluster, the shorter the survival time may be, because this situation means that such rescue tasks occur in some remote regions with less resource. In addition, considering the urgent characteristics of the rescue task, we need to first fulfill the cluster with less tasks. Based on the above consideration, we sort the cluster set $C = C^S \cup C^R$ in ascending order of the task number in each cluster $C_i \in C, \forall i \in [1, |C|]$. In particular, C^S represents the searching task set and C^R represents the rescue task set.

Since the searching drones do not need to carry special things and perform the same function, so that the rescue task cluster should be satisfied in the first place. For all the clusters in C^R, the rescue drones are assigned according to their rate, as follows:

$$G_i^R = \frac{|C_i^R|}{\sum\limits_{i \in [1, |C^R|]} C_i^R} \times |D^R|, \tag{9}$$

where G_i^R indicates the number of rescue drones assigned to perform the tasks in C_i^R under the constraint that:

$$|C_i^R| \leq \sum_{i \in [1, |G_i^R|]} d_i^c. \tag{10}$$

Note that, if there exists any pair of C_i^R and G_i^R that violates the constraint in (10), i.e., $|C_i^R| > \sum_{i \in [1, |G_i^R|]} d_i^c$, we need to exchange G_i^R and $G_{i'}^R$ which is searched to fulfill the following linear target.

$$\begin{aligned} &Minimize : ||G_{i'}^R| - |G_{i'}^R|| \\ &s.t. \quad \sum_{i' \in [1, |G_{i'}^R|]} d_{i'}^c > |C_i^R| \\ &\qquad \sum_{i \in [1, |G_i^R|]} d_{i'}^c > |C_{i'}^R| \\ &\qquad |G_{i'}^R| > |G_i^R| \end{aligned} \tag{11}$$

Now, assuming that the groups are already classified for the rescue drones and denoted by $G = \{G_i | \forall i \in [1, |G|]\}$, then, given the group G_i, the searching drones are assigned by $\frac{|G_i|}{\sum_{i \in [1, |G|]} G_i} \times |D^S|$, where the drones in G_i are used to serve

Algorithm 1 Spatial and Density Clustering Based Task and Drone Decomposition

```
Input: Task set $T$, drone set $D$, $\epsilon$
Output: Task cluster set $C$ and drone group set $G$
1  $C, G, core \leftarrow \emptyset$;
2  $i \leftarrow 0$;
3  for $t_i \in T$ do
4      $N_\epsilon(t_i) \leftarrow$ Calculate the $\epsilon$ neighbors for $t_i$;
5      if $N_\epsilon(t_i) \geq \epsilon$ then
6          $core \leftarrow core \cup t_i$;
7  for $t_i \in core$ do
8      $C_i \leftarrow \{t_i\}$;
9      for $t_j$ that is directly reachable to $C_i$ do
10         if $t_j \in N_\epsilon(t_i)$ and unclustered then
11             $C_i \leftarrow C_i \cup t_j$;
12 Sort $C$ in ascending order based on the number of
     tasks;
13 for $C_i \in C$ do
14     Calculate the drone group $G_i$ for $C_i$ based on (9);
15     if (10) is violated between $G_i$ and $C_i$ then
16         Find $G_j \in G$ according to (11) and exchange
             $G_i$ and $G_j$;
17     $G \leftarrow G \cup G_i$;
18 return $C, G$;
```

the tasks in C_i. The corresponding pseudo-code is shown in Algorithm 1.

B. Improved Auction for Decomposed Task Allocation

Since the large-scale task allocation is decomposed into small-scale ones, we next can address the small-scale task allocation problems in parallel, so as to improve the efficiency. For one task cluster $C_i = \{t_i^j | j \in [1, |C_i|]\}$ and the corresponding drone group $G_i = \{d_i^k | k \in [1, |G_i|]\}$, we propose an improved auction algorithm to solve the task allocation between C_i and G_i. Apparently, to establish the mapping relationship, we have: 1) the tasks in C_i are auction items; 2) the drones in G_i are the bidders; 3) a virtual auction agent should be introduced so as to manage and host the auction process.

Firstly, for each auction task t_i^j, it requires a basic starting bidding price which is evaluated by its difficulty level (i.e., $t_i^{j,d}$). Secondly, for each bidder d_i^k, it needs to bid during each round of auction. Considering the special nature of drone rescue, the bid price of each drone is comprehensively determined by the distance between d_i^k and t_i^j, the gap between its capacity (i.e., $d_i^{k,c}$) and task difficulty level (i.e., $t_i^{j,d}$), and the task loads on d_i^k. In addition, Different drones may offer different prices, so that we should also normalize the three price indicators. Then, according to the previous definitions, the three aspects can be calculated and normalized as follows:

$$distance(d_i^k, t_i^j) = \frac{dis(pos(d_i^k), pos(t_i^j))}{\max\limits_{d_i^k \in G_i, t_i^j \in C_i} dis(pos(d_i^k), pos(t_i^j))} \quad (12)$$

$$gap(d_i^k, t_i^j) = \frac{|d_i^{k,c} - t_i^{j,d}|}{\max\limits_{d_i^k \in G_i, t_i^j \in C_i} |d_i^{k,c} - t_i^{j,d}|} \quad (13)$$

$$load(d_i^k) = \frac{\sum_{j \in [1,|C_i|]} X_k^j}{\max\limits_{k \in [1,|G_i|]} \sum_{j \in [1,|C_i|]} X_k^j}, \quad \forall k \in [1, |G_i|] \quad (14)$$

Apparently, the farther the distance between the task and the drone, the higher the cost will be spent to perform this task. Besides, the drone can bid the task item, only when its capacity exceeds the task difficulty level. However, the larger the gap, the higher the cost, because the extra capacity maybe wasted. Lastly, the more tasks auctioned by this drone, the higher cost will be for this drone to bid other task items, because these tasks are usually not in the same place. One task may be very far from drone and all the task geographical positions maynot be fully taken into consideration when executing the task allocation, such that the drone would need to pay more cost on the extra flying distance and power consumption from the position of one task to the position of another. Therefore, we formulate the bidding price for the bidder d_i^k, as follows:

$$\begin{aligned} price_i^{j,k} &= \omega_1 distance(d_i^k, t_i^j) \\ &\quad + \omega_2 gap(d_i^k, t_i^j) + \omega_3 load(d_i^k), \end{aligned} \quad (15)$$

where $\omega_1, \omega_2, \omega_3$ are the coefficiencies limited by the constraint that $\omega_1 + \omega_2 + \omega_3 = 1$.

Then, based on the above definitions, the problem becomes

$$\begin{aligned} Minimize: &\sum_{j \in [1,|C_i|]} \sum_{k \in [1,|G_i|]} price_i^{j,k} \times X_k^j \\ s.t. \quad &\sum_{j \in [1,|C_i|]} X_k^j \leq 1, \quad \forall k \in [1, |G_i|] \\ &\sum_{k \in [1,|G_i|]} X_k^j \leq 1, \quad \forall j \in [1, |C_i|] \\ &d_i^{k,c} \geq t_i^{j,d}, \quad \forall j, k, X_k^j = 1, \end{aligned} \quad (16)$$

where the first constraint means that one bidder can only successfully bid one task item in one round auction. The second constraint means that one task item can only be assigned to one bidder. The third constraint makes sure that the bidder has the capacity to process the task item.

Next, we need to fulfill the task allocation for searching drones and rescue drones respectively. For the searching tasks, its difficulty level reduces to zero, so that we do not need to consider the second part in (15) and the corresponding coefficiencies are set as $\omega_1 = 0.6, \omega_2 = 0, \omega_3 = 0.4$ according to their effects on the cost. Unlike the traditional auction that selects the solution which can maximize the profits. In this paper, the goal is to minimize the cost spent on performing the searching and rescue tasks.

In order to make it clear and easy to follow, we simplify the three-dimensional scenario as a two-dimensional scenario, in which the searching task allocation principle is explained. For example, as shown in Fig. 3, there are two searching drones and three searching tasks in (a). The bidding prices of d_1^S and d_2^S can be calculated according to (15) and we can see

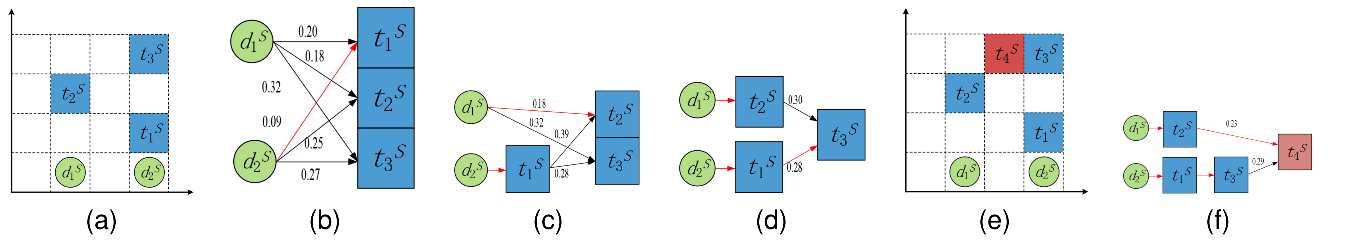

Fig. 3. Searching task allocation illustration. (a) 2 drones and 3 tasks scenario. (b) First round bidding. (c) Second round bidding. (d) Third round bidding. (e) t_4^S appear. (f) New round bidding.

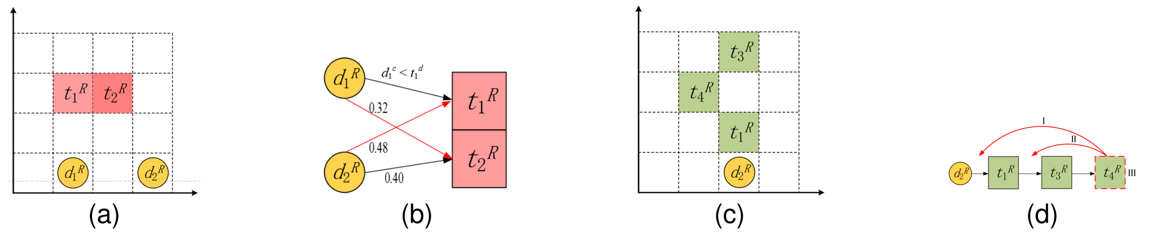

Fig. 4. Rescue task allocation illustration. (a) 2 drones and 2 tasks scenario. (b) Bidding. (c) New task arrival. (d) Task sequence adjustment.

that the normalized bidding prices in the first round are shown in (b), where d_2^S provides the lowest price for t_1^S, so that t_1^S is added to the task list of d_2^S as $List(d_2^S) = List(d_2^S) \cup \{t_1^S\}$. Note that, the searching distance will be doubled if the drone begins executing the second task from its origin, so that the ending position of the previous task will be the starting position of the next task. This design is reflected in the auction process, so that we have

$$pos(d_2^S) = pos(t_1^S). \tag{17}$$

The second round auction is shown in Fig. 3(c), where d_2^S provides a bid on the basis of t_1^S, which means that the corresponding *distance, gap* and *load* of d_2^S are changed in this round. In this way, the normalized bidding prices are 0.18 (d_1^S for t_2^S), 0.32 (d_1^S for t_3^S), 0.39 (d_2^S for t_2^S), 0.28 (d_2^S for t_3^S) respectively. Hence, d_1^S provides the lowest price for t_2^S, so that t_2^S is added to the task list of d_1^S as $List(d_1^S) = List(d_1^S) \cup \{t_2^S\}$. Similarly, the third round auction is carried out after d_1^S and d_1^S update their positions, and shown in Fig. 3(d). However, new searching tasks would appear suddenly. For example, a new searching task t_4^S appears next to t_3^S in 3(e) and the bidding process is shown in 3(f), in which we can also observe that this new round bidding is carried out based on the result of the previous auction.

On the other hand, for the rescue task, the auction process follows similar principles as in the searching task auction and the coefficiencies are set as $\omega_1 = 0.42, \omega_2 = 0.28, \omega_3 = 0.3$. Then, an example is given in Fig. 4, where 4(a) provides a scenario with 2 rescue drones and tasks and 4(b) shows the bidding process. Apparently, the bid price is calculated according to equation (15) and d_1^R offers the lower cost for t_1^R and t_2^R. However, unlike the searching task with the execution difficulty approaching to zero, the rescue task has basic requirements, that is, the rescue task difficulty exceeds zero. Hence, although d_1^R offers the lower cost for t_1^R, its capacity is lower than the requirements of t_1^R, that is, $d_1^c < t_1^d$. In this case, the auction results are that t_1^R is allocated to d_2^R and t_2^R is allocated to d_1^R.

Nevertheless, we should be aware that the *position* and the *load* of the rescue drone will be changed after every round of auction, which would result in a situation that the drone may need to go back to the origin for re-supply before it can perform the next task. Returning to the origin will increase the flying distance and leads to extra cost. In this regard, the task execution sequence should be dynamically adjusted for drones. Given the drone d_i^R and its task list $List(d_i^R)$, if new bidding t_j^R is obtained by d_i^R, then, we can 1) insert t_j^R into appropriate position in $List(d_i^R)$; 2) re-arrange all elements in $List(d_i^R) \cup \{t_j^R\}$. Regarding the two ways, we come to the following theorem.

Theorem 2: Let $|List(d_i^R)| = n$, the new task insertion operation can improve the computing efficiency by $O(n!)$ with up to twice the distance cost compared with task re-ordering.

Proof: Denoting the new bidding task by $t_{k'}^R$, then there are two cases. Firstly, we can insert $t_{k'}^R$ into $List(d_i^R)$, so that we have $|List(d_i^R)| = n+1$ with the worst searching complexity of $O(n+1)$. Secondly, if we re-order all the elements in $List(d_i^R)$ to achieve a global optimal execution order, the worst time complexity is $O((n+1)!)$ according to the permutation theory. Hence, the time complexity can be improved by about $\frac{O((n+1)!)}{O(n+1)} = O(n!)$.

In addition, assuming that $t_{k'}^R$ is inserted between t_k^R and t_{k+1}^R, then we have

$$\begin{aligned} &dis(pos(t_k^R), pos(t_{k'}^R)) \le dis(pos(t_k^R), pos(t_{k+1}^R)) \\ &dis(pos(t_{k'}^R), pos(t_{k+1}^R)) \le dis(pos(t_k^R), pos(t_{k+1}^R)). \end{aligned} \tag{18}$$

Regarding $t_k^R, t_{k'}^R, t_{k+1}^R$ as the three vertices of a triangle, then according to the length relationship between the three edges of a triangle, it follows that

$$\begin{aligned} &dis(pos(t_k^R), pos(t_{k'}^R)) + dis(pos(t_{k'}^R), pos(t_{k+1}^R)) \\ &\ge dis(pos(t_k^R), pos(t_{k+1}^R)) \end{aligned} \tag{19}$$

Now, substituting (18) into (19), we can obtain the lower and upper distance bounds, that is, $[1, 2] \times dis(pos(t_k^R), pos(t_{k+1}^R))$. In addition, the distance is directly proportional to the cost in this phase, so that the cost spent on new task insertion is up

Algorithm 2 Auction Based Task Allocation

```
Input: cluster set $C$, drone group set $G$
Output: Task allocation solution
1  $i \leftarrow 0$;
2  for $C_i \in C$ and $G_i \in G$ do
3      for $t_i^j \in C_i$ do
4          for $d_i^k \in G_i$ do
5              $price_i^{j,k} \leftarrow$ Calculate the bidding price for
               $d_i^k$ towards $t_i^j$;
6          Select the $d_i^k$ with the lowest price as winner
           of $t_i^j$;
7          Update the position and load of $d_i^k$ according
           to equations (14) and (17);
8          $pre\_cost(d_i^k) \leftarrow$ Calculate the flying cost for
           $d_i^k$ to put $t_i^j$ at the end of its task list;
9          for $\lambda \in [1 : |list(d_i^k)|]$ do
10             $new\_cost(d_i^k) \leftarrow$ Calculate the flying cost
               for $d_i^k$ to insert $t_i^j$ at the $\lambda$-th position on its
               task list;
11             if $pre\_cost(d_i^k) > new\_cost(d_i^k)$ then
12                 $index \leftarrow \lambda$;
13         Insert $t_i^j$ into the $index$-th position of $list(d_i^k)$;
14 return $list(G)$;
```

TABLE I
PARAMETERS AND TESTING CASES

Parameters						
Searching drone	speed (m/s) maximum task number	3 6				
Rescue drone	speed (m/s) maximum task number capacity	1 6 {3,6,9}				
Searching task	execution time (s) task difficulty level	5 0				
Rescue task	exeuction time (s) task difficulty level	10 [1,9]				
Testing case1 (small scale)						
Scenario ID	1	2	3	4	5	6
Drone number	6	6	6	10	10	10
Task number	12	18	24	20	30	40
Testing case2 (large scale)						
Scenario ID	1	2	3	4	5	6
Drone number	30	30	30	50	50	50
Task number	60	80	100	100	120	140

TABLE II
TASK ALLOCATION SOLUTIONS IN SMALL SCALE SCENARIOS

	task allocation (**drone number - task number**)					
drone	**6-12**	**6-18**	**6-24**	**10-20**	**10-30**	**10-40**
S-D1	2,0	0,3	2,16,23,6	9,0	0,2,12	22,29,2,3
S-D2	6,10	12,16,6,9	20,12,9,10	2,13	3,6,9	32,20,13
S-D3	9,3	10,13,2	22,19,3,0,13	3	26,10,20	6,9,12,39,23
S-D4	—	—	—	12,19,10	13,22	10,36,30,33
S-D5	—	—	—	6,16	19,16,23,29	0,19,16,26
R-D1	1,4,5	17,11	15,14,21,5	11,15	11,24	24,4,38,27
R-D2	8,11	14,8,15	17,4,7	4,5,17	5,27,8,18	35,34,1,5
R-D3	7	4,5,1,7	18,1,8,11	8	25,7,14	18,28,7,37
R-D4	—	—	—	7,14	28,21,1	14,15,21,31
R-D5	—	—	—	1,18	4,15,17	25,17,8,11

to twice that of the original solution, so that Theorem 2 is proved. □

The task insertion process is explained with the example given in Fig. 4(c), where t_1^R, t_3^R, t_4^R are successfully auctioned by d_2^R. For the last task item t_4^R, there are three insertion solutions, that is, 1) $\{t_4^R \rightarrow t_1^R \rightarrow t_3^R\}$ with cost of 6.58; 2) $\{t_1^R \rightarrow t_4^R \rightarrow t_3^R\}$ with cost of 3.83; 3) $\{t_1^R \rightarrow t_3^R \rightarrow t_4^R\}$ with cost of 4.41. In this way, the second insertion solution is selected. The corresponding pseudo-code is shown in Algorithm 2.

V. Performance Evaluation

A. Setup

The experiments are carried out based on ROS Kinetic, an open source platform for multi-robot simulation, using the python programming language. It is also a distributed platform, so as to well simulate the practical drone actions. Besides, in order to provide a simulation environment as real as possible, the searching drones are set to equip sensors for searching and the rescue drones are set to equip medicine for rescuing, which are power consuming. The parameter details are summarized and shown in Table I. Specifically, the flying speed of a searching drone is set to 3 m/s, while that of a rescue drone is set to 1 m/s, because the rescue drones are heavier than the searching drones. Meanwhile, the maximum number of tasks that can be accepted by the searching/rescue drone is 6. The capacity of rescue drone is set in {3,6,9} and the task difficulty level is set within the scope of [1,9]. Moreover, considering the characteristics of the searching and rescue tasks, we set their execution time by 5 seconds and 10 seconds respectively.

The hardware environments are Intel(R) Core(TM) i7-6700 CPU 3.40GHz with 16G memory and 1T hard disk space with the OS of Ubuntu 16.04.

B. Metrics and Benchmarks

In order to comprehensively evaluate the proposed algorithm, we select four testing metrics which are:

- Average flying distance, it reflects the average moving distance of all drones during the task execution process.
- Average utilization of drone capacity, it reflects the resource utilization of drones, so that the higher this metric, the more tasks that will be accepted.
- Task load balance, it reflects the task distribution among drones and is evaluated using the standard deviation among the number of tasks carried by drones.
- Task completion time, it reflects the total time that a drone group finishes the allocated tasks. The smaller the better.

Besides, the task load balance reflects the task distribution among drones and it is evaluated using the standard deviation among the number of tasks carried by drones.

In addition, we compare the proposed method with two novel benchmarks over two testing cases. Specifically, the three comparison methods are:

- SSI, a Sequential and Single-Item based auction algorithm that is used to address the task allocation for heterogeneous drones [29].

- CBBA, a Consensus and Binding Based Algorithm that is used to group decentralized tasks and allocate these tasks to distributed drones [30].
- DCAA, the proposed method that uses the Density Clustering and Auction based Algorithm to address the large scale task allocation among drones.

C. Results

The results are achieved under two testing cases and the details are summarized in Table I. The first one is a small scale testing case with a $100 \times 100 \times 100\ m^3$ three-dimensional space, in which there are 6 kinds of scenarios that have (6, 12), (6, 18), (6, 24), (10, 20), (10, 30), (10, 40) drones and tasks respectively. As for the large scale testing case, it is a $2000 \times 2000 \times 100\ m^3$ three-dimensional space, in which there are also 6 scenarios that have (30, 60), (30, 80), (30, 100), (50, 100), (50, 120), (50, 140) drones and tasks respectively. The above testing scenarios are set to build a perfect match relationship between the number of drones and the number of tasks by taking the capacity of drones and the task difficulty into consideration. Because on one hand, if the number of tasks is much lower than the drones' capacities, which means that a lot of drones maybe idle for a long time, so that their capacity utilization will be extremely lower. On the other hand, if the number of tasks is much higher than the drones' capacities, then all the drones will be extremely overloaded, such that there is almost no room for optimization and all methods may perform the same.

The task allocation solutions under the two cases are summarized and shown in Tables II and III respectively. It is noted that the terms "S-D" and "R-D" are short for "Searching-Drone" and "Rescue-Drone" respectively. Besides, it is aware the number of searching drone is equal to the number of rescue drone, and the number of searching task is also equal to the number of rescue task. Despite this setting, the searching and rescue tasks are randomly generated and distributed in different locations. Moreover, the values in the two tables indicate the task serial number that starts from zero, for example, S-D1 in "6-12" has two searching tasks with one numbered zero.

Firstly, for the results in Table II, we can see that 1) all the tasks are allocated to drones. For example, for the scenario with 6 drones and 12 tasks, there are 3 searching drones responsible for 6 searching tasks and 3 rescue drones responsible for 6 rescue tasks. The difference is the task allocation solution, because each searching drone is allocated with 2 searching tasks, while the three rescue drones are in charge of {3,2,1} rescue tasks respectively. 2) the task load of any drone is in a relatively balanced state. For example, for the scenario with 10 drones and 24 tasks, the maximum and minimum number of tasks carried by drones are 5 and 3 respectively. Despite this, it is note that only one drone (i.e., S-D2) has 3 tasks and one drone (i.e., S-D3) has 5 tasks. The rest drones all have 4 tasks. 3) there are some empty spaces in Table II. This is reasonable, because some drones do not exist at all. For example, for the case of 6-12 in terms of S-D4, it means that the fourth searching drone is not allocated with any task. In this case, there are 6 drones. Half of them are the searching drones, while the rest half are the rescue drones. In this case, the fourth searching drone (i.e., S-D4) does not exist at all.

Then, for the results in Table III, we can also observe similar phenomena as in Table II. The difference is that the results in Table III are obtained in the large scale testing case. Taking the first scenario as an example, there are 30 drones and 60 tasks, so that each drone needs to carry two tasks on average. Since the proposed algorithm has already taken the load balance into consideration, we can see that most drones have been allocated with two tasks in this scenario, except S-D1, S-D4, S-D6, S-D10, R-D7 and R-D9. Note that the gap between the maximum and minimum number of tasks carried by drones is only 2 in this scenario, which means that we achieve a perfect load balance among drones. Observing the other five scenarios, we can see that such gap values are 2 (for 30 drones and 80 tasks), 1 (for 30 drones and 100 tasks), 2 (for 50 drones and 100 tasks), 2 (for 50 drones and 120 tasks), 3 (for 50 drones and 140 tasks), which also reflect the similar conclusion. Moreover, we can also observe similar empty spaces in Table III, due to the same reasons explained in terms of the results in Table II.

Now, reviewing the results in Table II, we can see that the maximum number of tasks carried by one drone is 5 (only one drone has five tasks, i.e., S-D3 in "10-40"), while that in Table III is 4 (most drones have 4 tasks). Besides, the average task load balance in Table III is better than that in Table II. Hence, all these conditions reflect that the proposed algorithm is more suitable for processing the large scale task allocation problem in MDS.

Based on the task allocation solutions, the drones need to execute their tasks accordingly, during which we calculate their flying paths which are shown vividly in Fig. 5. In particular, we can see that there are 12 sub-figures, where the first 6 sub-figures display the flying paths of solutions from Table II and the rest 6 sub-figures show the flying paths of solutions from Table III. In addition, the drone is represented by the circle and the task is represented by the square. Note that in the small case scenarios (i.e., 5(a)-5(f)), there are a few tasks that are distributed randomly in the spatial structure, so that we do not form the task clusters. However, as for the large scale scenarios (i.e., 5(g)-5(l)), we can easily see that there are task clusters formed, because the number of tasks becomes large, so that they tend to gather together gradually.

Besides, let's observe the drone distribution in these sub-figures. Apparently, we can see that the initial positions of all the searching and rescue drones are along the x-axis in the small scale testing scenarios, while that in the large scale testing scenarios are along the x-axis and the y-axis. This is because the maximum number of drones is 10 in the small scale testing scenarios, while the minimum number of drones is 30 in the large scale testing scenarios. Such difference leads to the situation that if we set the initial starting positions of all drones along the same axis in large scale scenarios, these drones may frequently complete the tasks closer to them and cause extra calculation cost.

TABLE III
TASK ALLOCATION SOLUTIONS IN LARGE SCALE SCENARIOS

	task allocation (**drone number - task number**)					
drone	**30-60**	**30-80**	**30-100**	**50-100**	**50-120**	**50-140**
S-D1	0	6,16,19	13,19,6	6,16	13,2,12	3,19,2
S-D2	6,9	3,10,2,13	16,12,9,2	19,9	10,16	13,9,10,16
S-D3	10,13	0,12,9	10,3,0	10,12	6,9,19	12,0,6
S-D4	16,3,12	23,26,36,29	39,30,36	2,13	0,3	30,32,36,29
S-D5	19,2	32,30,22	32,22,23,26	0,3	33,23,36	39,26,33
S-D6	20	39,33,20	33,29,20	26,29,36	29,22	22,20,23
S-D7	32,29	50,53	66,80,69,83	20,39	26,30	40,53
S-D8	33,39	59,66	60,93,96	33,32	20,39,32	93,69,59
S-D9	20,26	60,42	86,63,62	22,30	40,43	70,83
S-D10	22,36,23	72,56,63	99,92,90	72,42	72,52,50	123,99,96
S-D11	46,43	49,79,73	46,43,82,42	63,60	42,46	66,100
S-D12	56,59	43,46	76,73,89,53	52,50	56,69	42,50,116
S-D13	52,42	52,69	79,70,72	66,69	62,60,66	119,120
S-D14	49,50	70,76	56,59,49	79,93	76,89	126,132
S-D15	40,53	40,62	52,50,40	80,49	83,93	129,133
S-D16	—	—	—	53,70	96,102	139,52,112
S-D17	—	—	—	82,62	99,103	103,43,60,49
S-D18	—	—	—	56,99	110,106	63,72,79
S-D19	—	—	—	92,59	80,73,112	86,136,80
S-D20	—	—	—	90,96	79,100,90	90,76,82
S-D21	—	—	—	86,73	53,92,113	46,130
S-D22	—	—	—	43,76	59,63	102,110,122,62
S-D23	—	—	—	40,83	86,109	109,106
S-D24	—	—	—	46	116,119	56,89,73
S-D25	—	—	—	89	82,70,49	92,113
R-D1	15,11	18,1,14	15,17,11	1	11,17,1	17,11,7
R-D2	14,8	17,15,5	4,14,5,1	15,7	8,18,14	4,15,18
R-D3	17,5	7,4,8,11	7,8,18	11,8,17	15,7	1,14,5,8
R-D4	18,4	38,28,27	37,31,28	14,4	4,5	25,34,27
R-D5	1,7	37,34,35	25,38,34	18,5	21,27	35,38,21
R-D6	28,34	31,21,24,25	21,24,35,27	35,24	37,24	37,24,28,31
R-D7	35,21,27	41,57	41,54,77	25,21	34,38	41,57,125
R-D8	25,37	51,65,44	61,44,65	38,28	25,28,35,31	55,45
R-D9	24	67,54	47,55,84,75	34,37	47,57	107,48,58
R-D10	31,38	74,77,75	67,85,95	27,31	55,68	61,67,47
R-D11	45,54	45,48	97,74,88	41,61	61,78,94	74,71
R-D12	44,51	61,55	94,87,98,45	44,48	81,88	81,77
R-D13	47,41	64,71	71,57,58,81	47,67	84,91	84,88
R-D14	57,55	68,47	64,78,68	75,45	97,98	87,91,118
R-D15	58,48	78,58	91,51,48	87,54	65,115	101,98,121
R-D16	—	—	—	85,88	44,51	105,104
R-D17	—	—	—	91,64	67,74	115,108
R-D18	—	—	—	97,94	111,85,95	127,124
R-D19	—	—	—	98,57	101,105	135,131,111
R-D20	—	—	—	51,55	45,107	85,44,51
R-D21	—	—	—	68,65	108,118	54,64,75
R-D22	—	—	—	84,77	114,87,58	65,117,134
R-D23	—	—	—	58,95	41,77,75	68,95,78
R-D24	—	—	—	78,71	54,71,104	128,94,137
R-D25	—	—	—	81,74	48,117,64	114,138,97

One more detail can be found is that the most drones start their task execution following a strict principle, that is, the closer the task, the larger probability this task will be executed in advance, which then can reduce the total flying distance, so as to reduce the overall cost. For example, as shown in Fig. 5(a), the position points of its path are $(\mathbf{x}, \mathbf{y}, \mathbf{z}) = \{(45, 16, 13), (65, 87, 27)\}$. Since the drones all begins at the x-axis, so that the second position (65,87,27) is farther from the starting point than the first position (45,16,13). As for the other subfigures, we can also observe the same flying principle which is satisfied by most drones. In this way, by carefully taking the positions of different tasks into consideration and building the relationship among them for task allocation decision making, then the overall cost can be greatly reduced, due to the reduction of the flying distance that is proportional to the cost.

Based on the above evaluation, the correctness and robustness of the proposed method are reflected in a certain extend. Hence, we next compare it with the benchmark methods. The experiments are carried out 30 times and the average results are shown in Figs. 6-9 respectively, where Fig. 6 shows the statistical results about the average flying distance, Fig. 7 shows the utilization of drone capacity, Fig. 8 shows the task load balance of drones and Fig. 9 shows the task completion time.

Firstly, for the results of the average flying distance in Fig. 6, it is a vital important metric to evaluate the effectiveness of the methods, because the longer the flying distance, the larger the

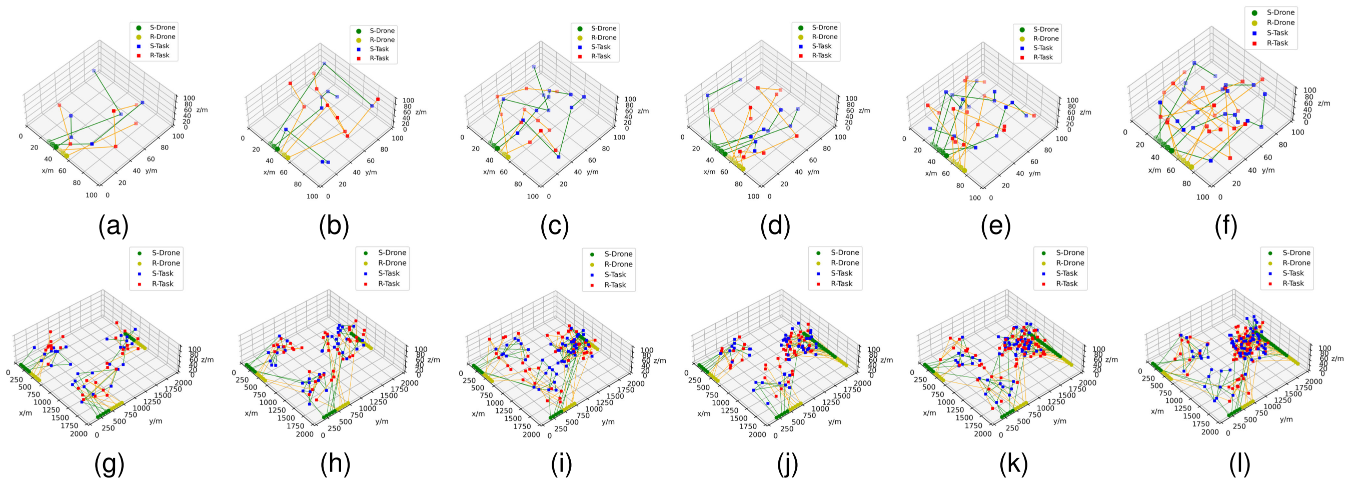

Fig. 5. Flying path. (a) 6 drones and 12 tasks. (b) 6 drones and 18 tasks. (c) 6 drones and 24 tasks. (d) 10 drones and 20 tasks. (e) 10 drones and 30 tasks. (f) 10 drones and 40 tasks. (g) 30 drones and 60 tasks. (h) 30 drones and 80 tasks. (i) 30 drones and 100 tasks. (j) 50 drones and 100 tasks. (k) 50 drones and 120 tasks. (l) 50 drones and 140 tasks.

cost. In addition, the long flying distance will also consume more power of drones, which results in the condition that this drone may not be able to make full use of its capacity before running out its power. Apparently, the average flying distance achieved in case2 is larger than that in case1, due to the fact that the number of tasks in case2 is over triple times that in case1. Besides, we can see that the proposed method achieves the second shortest distance in case1. On one hand, the proposed method jointly considers the distance, load balance and the capacity utilization when calculating the overall cost, so that we may not be able to achieve the optimal performance toward the metric of flying distance. On the other hand, implementing load balance among drones means that more drones will participate in task execution, such that the overall distance of the drone group will also increase.

Despite this, we should be aware that such non-optimal performance only happens in the small scale case, due to the limited number of drones/tasks and resource. As for the large scale case, the proposed method achieves the shortest flying path than the other two benchmarks. This is reasonable, because we evaluate the distance between task cluster and drone group, instead of establishing the mapping relationship between any drone and any task. For the other benchmarks, their flying distance increases greatly with the increasing of the number of drones/tasks. That is because they lack of dynamic adjustment strategy, so that their task allocation solution in large scale scenarios would easily lead to the drone overload which prevents them from accepting more tasks. Observing the details in Fig. 6, we can see that the decreasing rate of the average flying distance achieved by the proposed method is around [25%,45%] in terms of the large-scale case. The larger the value of flying distance decreasing rate, the less power consumed, so that more tasks can be executed.

Secondly, for the results of the utilization of drone capacity in Fig. 7, several phenomena can be observed, that is, 1) the proposed method achieves the highest utilization (i.e., about 80%) in both case1 and case2. The higher the utilization of drone capacity, the higher the resource utilization, since this metric describes the matching degree between the task

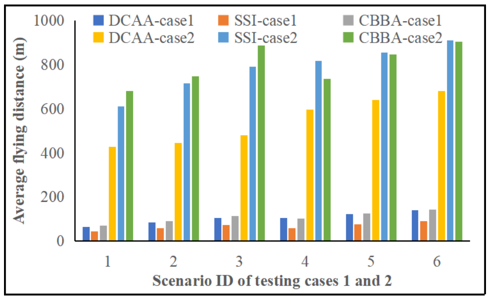

Fig. 6. Average flying distance.

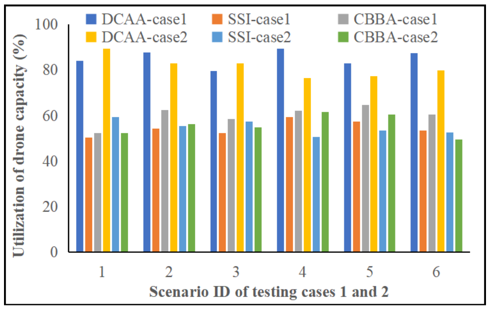

Fig. 7. Utilization of drone capacity.

difficulty level and drone capacity. Besides, SSI has the lowest utilization (about 50%) and CBBA is slightly higher (about 60%) than that of SSI, because they do not consider the matching degree between the task difficulty level and drone capacity. Then, this will lead to the situation that the drone capacity may exceed the task difficult greatly, so that a lot of resource would be wasted. 2) the drone capacity utilization achieved by the proposed method in both case1 and case2 are similar (i.e., around 80%), which means that the proposed can better adapt to the scale of the number of tasks/drones. As for SSI and CBBA, we can observe similar phenomenon. Despite this, it is aware that the utilization achieved by SSI and CBBA are about 50% and 60% respectively, which are both lower than that of the proposed method. Compared with the proposed method, such performance gap is around [20%, 30%].

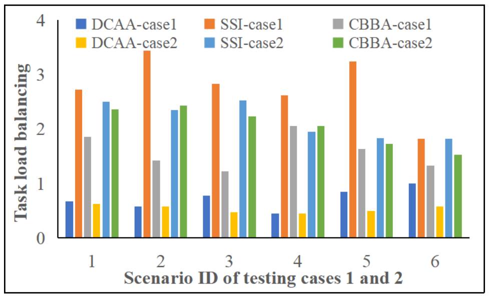

Fig. 8. Task load balance.

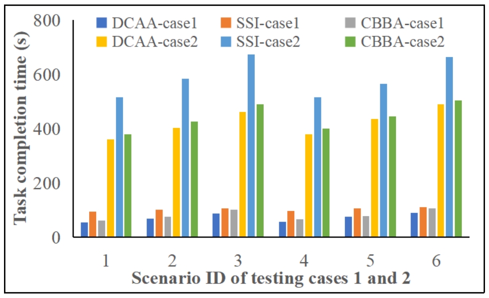

Fig. 9. Task completion time.

Thirdly, the task load balance conditions are calculated and presented in Fig. 8, where we can see several phenomena that: 1) the proposed method achieves the smallest value (about 0.8 in case1 and 0.5 in case2) of task load balance. As explained, the load balance is evaluated by the standard deviation, so that the smaller this value, the more balance the task load among drones. In addition, the load balance condition can also reflect the cooperation degree within the whole drone group. In other words, unbalanced task load means that some drones may be allocated with many tasks, while the others may not be allocated with tasks. In this case, the heavy loaded drones are always busy working, while the others are idle. Such situation would accelerate the damage of the heavy loaded drones and may even take long time to complete the tasks. 2) the load balance of the proposed method remains very stable (the gap is less than 0.5) in different scenarios, which can also be observed in Tables II and III. However, the minimum and maximum values of load balance achieved by SSI and CBBA are within [1.8, 3.5] and [1.4, 2.5] respectively. Comparing the gap values (0.5 for DCAA; 1.7 for SSI; 1,1 for CBBA), the proposed method outperforms the benchmarks. 3) the values of load balance in case1 and case2 does not increase strictly with the increasing of the number of tasks/drones. Taking case1 as the example, the load balance values of the three methods are {0.8, 0.79, 0.9, 0.75, 1, 1.2} (DCAA), {2.8, 3.4, 2.9, 2.7, 3.1, 1.9} (SSI) and {1.9, 1.6, 1.5, 2.2, 1.85, 1.55} (CBBA) respectively. That is because the ratio between the task number and drone number is different in terms of the six scenarios.

Lastly, for the results of the task completion time in Fig. 9, we can also see several phenomena. On one hand, the task completion time increases with the increasing of the number of tasks for the three methods. However, when the number of tasks exceeds four times that of drone, the task completion time increase slowly. That is because the strategy of SSI and CBBA would easily overload some drones and make the rest drones idle. Only when the number of tasks far exceed the number of drones, SSI and CBBA would consider to allocate the tasks to the idle drones, so that the task load balance is improved. As for the proposed method DCAA, its task completion time increases slowly with the increasing of the number of tasks, due to the perfect task load balance achieved by DCAA. On the other hand, the proposed method achieves the lowest task completion time, because most drones in the same group are allocated with almost the same number of tasks, so that they can execute the tasks on parallel to reduce the task completion time as much as possible. On the contrary, the task allocation provided by SSI and CBBA is unbalanced, so that the task completion time depends on the time that the last drone finishes its tasks. Moreover, looking deep into the results, we can see that the proposed method can decrease the overall task completion time by around 22.5% on average.

VI. Conclusion

The MDS are gradually used to perform the repetitive and dangerous tasks for human in many complex environments with the rapid development of drones. However, the task allocation in large scale MDS is a critical and challenging problem that need to be addressed urgently. In this work, we design a DT enabled system framework and propose a density clustering and auction based method to address the task allocation in large scale MDS. Experimental results indicate that the proposed method outperforms the state-of-the-art benchmarks. Despite this, how to apply the task allocation theory and method proposed in this work into real application still needs to be explored.

Appendix

Theorem: *The task allocation problem in MDS is NP-hard.*

Proof: We use a reduction from the Multiple Knapsack Problem (MKP) to prove that the task allocation problem in MDS is NP-hard. In an instance of MKP, we are given a set of m knapsacks with the capacity $c_i (i \in [1, m])$ and a set of n objects with value v_j and weight $w_j (j \in [1, n])$. Then, MKP is to select m disjoint subsets of objects for each knapsack under the constraint that the total weight of the objects does not exceed the capacity of this knapsack and the overall value should be maximized, as follows:

$$\begin{aligned} \text{P1:} \quad & \text{Maximize:} z = \sum_{i=1}^{m} \sum_{j=1}^{n} v_j x_{i,j} \\ & \text{s.t.} \quad \sum_{j=1}^{n} w_j x_{i,j} \le c_i, \quad \forall i \in [1, m] \\ & \qquad \sum_{i=1}^{m} x_{i,j} \le 1, \quad \forall j \in [1, n] \\ & \qquad x_{i,j} \in \{0, 1\}, \quad \forall i \in [1, m], j \in [1, n] \end{aligned} \tag{20}$$

Reviewing the model of task allocation in MDS in equation (4), we can see that they have different optimization objects. In this way, let $v'_j = -v_j$ and substituting v'_j into (20), a new model objective (P2) is formulated as

$$\text{P2:} \quad \text{Minimize:} z' = \sum_{i=1}^{m} \sum_{j=1}^{n} v'_j x_{i,j}. \tag{21}$$

On one hand, we assume that one object must be allocated to one knapsack, so that the constraint $\sum_{i=1}^{m} x_{i,j} \leq 1$ becomes $\sum_{i=1}^{m} x_{i,j} = 1$. On the other hand, for each knapsack, it is allowed to contain many objects within its capacity, so that the new constraint $\sum_{j=1}^{n} x_{i,j} \geq 1$ is applied. Based on the above definition, P2 is updated as:

$$\begin{aligned} \text{P3:} \quad & \text{Minimize:} z' = \sum_{i=1}^{m} \sum_{j=1}^{n} v'_j x_{i,j} \\ \text{s.t.} \quad & \sum_{i=1}^{m} x_{i,j} = 1, \quad \forall j \in [1,n] \\ & 1 \leq \sum_{j=1}^{n} x_{i,j} \\ & \sum_{j=1}^{n} w_j x_{i,j} \leq c_i, \quad \forall i \in [1,m] \\ & x_{i,j} \in \{0,1\}, \quad \forall i \in [1,m], j \in [1,n] \end{aligned} \tag{22}$$

Now, reviewing the model in (4), we simplify $dis(pos(d_i), pos(t_j))$ as V_j^i, so that the objective in (4) becomes:

$$\text{Minimize:} \sum_{i \in [1,|D|]} \sum_{j \in [1,|T|]} V_j^i X_i^j \tag{23}$$

Since v'_j is a fixed value and V_j^i is not, we let $V_j^i = V_j + (V_j^i - V_j)$ where the value of V_j is fixed, such that we can decompose the objective in (4) as follows:

$$\begin{aligned} \text{Minimize:} & \sum_{i \in [1,|D|]} \sum_{j \in [1,|T|]} V_j X_i^j \\ & + \sum_{i \in [1,|D|]} \sum_{j \in [1,|T|]} (V_j^i - V_j) X_i^j. \end{aligned} \tag{24}$$

Then, the overall problem of (4) can be decomposed into two sub-problems as follows:

$$\begin{aligned} \text{P4:} \quad & \text{Minimize:} \sum_{i \in [1,|D|]} \sum_{j \in [1,|T|]} V_j X_i^j \\ & s.t. \quad \sum_{i \in [1,|D|]} X_i^j = 1 \\ & \quad 1 \leq \sum_{j \in [1,|T|]} X_i^j \\ & \quad \sum_{j \in [1,|T|]} t_j^d X_i^j \leq d_i^c \\ & \quad (1), (2), (3) \end{aligned} \tag{25}$$

$$\begin{aligned} \text{P5:} \quad & \text{Minimize:} \sum_{i \in [1,|D|]} \sum_{j \in [1,|T|]} (V_j^i - V_j) X_i^j \\ & s.t. \quad \sum_{i \in [1,|D|]} X_i^j = 1 \\ & \quad 1 \leq \sum_{j \in [1,|T|]} X_i^j \\ & \quad \sum_{j \in [1,|T|]} t_j^d X_i^j \leq d_i^c \\ & \quad (1), (2), (3) \end{aligned} \tag{26}$$

Observing the forms of P3 in (22) and P4 in (25), while let the knapsack represents the drone and the item represents the task, we can conclude that they are the same model. Since P3 is reduced from P2 and P2 is reduced from P1, where P1 (MKP) has already been proved to be NP-hard, so that P4 is NP-hard naturally. It is aware that the problem of task allocation in MDS is the combination of P4 and P5, so that this theorem is proved. $\square$

References

[1] F. Wilhelmi, M. Carrascosa, C. Cano, A. Jonsson, V. Ram, and B. Bellalta, "Usage of network simulators in machine-learning-assisted 5G/6G networks," *IEEE Wireless Commun.*, vol. 28, no. 1, pp. 160–166, Feb. 2021.

[2] Y. Roh, G. Heo, and S. E. Whang, "A survey on data collection for machine learning: A big data-AI integration perspective," *IEEE Trans. Knowl. Data Eng.*, vol. 33, no. 4, pp. 1328–1347, Apr. 2021.

[3] L. Jin and S. Li, "Distributed task allocation of multiple robots: A control perspective," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 48, no. 5, pp. 693–701, May 2018.

[4] X. Shi, C. Yang, W. Xie, C. Liang, Z. Shi, and J. Chen, "Anti-drone system with multiple surveillance technologies: Architecture, implementation, and challenges," *IEEE Commun. Mag.*, vol. 56, no. 4, pp. 68–74, Apr. 2018.

[5] E. Yanmaz, S. Yahyanejad, B. Rinner, H. Hellwagner, and C. Bettstetter, "Drone networks: Communications, coordination, and sensing," *Ad Hoc Netw.*, vol. 68, pp. 1–15, Jan. 2018.

[6] V. Hassija, V. Saxena, and V. Chamola, "Scheduling drone charging for multi-drone network based on consensus time-stamp and game theory," *Comput. Commun.*, vol. 149, pp. 51–61, Jan. 2020.

[7] R. Zhang, Q. Lv, J. Li, J. Bao, T. Liu, and S. Liu, "A reinforcement learning method for human-robot collaboration in assembly tasks," *Robot. Comput.-Integr. Manuf.*, vol. 73, pp. 1–10, Feb. 2022.

[8] F. Maaroufi, H. Camus, and O. Korbaa, "A mixed integer linear programming approach to schedule the operating room," in *Proc. IEEE Int. Conf. Syst., Man, Cybern. (SMC)*, Oct. 2016, pp. 3882–3887.

[9] S. Mousavi, F. Afghah, J. D. Ashdown, and K. Turck, "Use of a quantum genetic algorithm for coalition formation in large-scale UAV networks," *Ad Hoc Netw.*, vol. 87, pp. 26–36, May 2019.

[10] O. Ozkan, "Optimization of the distance-constrained multi-based multi-UAV routing problem with simulated annealing and local search-based matheuristic to detect forest fires: The case of Turkey," *Appl. Soft Comput.*, vol. 113, pp. 1–10, Dec. 2021.

[11] K. Wang, X. Li, L. Gao, P. Li, and S. M. Gupta, "A genetic simulated annealing algorithm for parallel partial disassembly line balancing problem," *Appl. Soft Comput.*, vol. 107, pp. 10–17, Aug. 2021.

[12] N. Geng, Z. Chen, Q. A. Nguyen, and D. Gong, "Particle swarm optimization algorithm for the optimization of rescue task allocation with uncertain time constraints," *Complex Intell. Syst.*, vol. 7, no. 2, pp. 873–890, Apr. 2021.

[13] X. Xia et al., "Triple archives particle swarm optimization," *IEEE Trans. Cybern.*, vol. 50, no. 12, pp. 4862–4875, Dec. 2020.

[14] A. Asma and B. Sadok, "PSO-based dynamic distributed algorithm for automatic task clustering in a robotic swarm," *Proc. Comput. Sci.*, vol. 159, pp. 1103–1112, Jan. 2019.

[15] L. Huang, Y. Ding, M. Zhou, Y. Jin, and K. Hao, "Multiple-solution optimization strategy for multirobot task allocation," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 50, no. 11, pp. 4283–4294, Nov. 2020.

[16] P. Schillinger, M. Bürger, and D. V. Dimarogonas, "Simultaneous task allocation and planning for temporal logic goals in heterogeneous multi-robot systems," *Int. J. Robot. Res.*, vol. 37, no. 7, pp. 818–838, Jun. 2018.

[17] W. Lee and D. Kim, “Adaptive approach to regulate task distribution in swarm robotic systems,” *Swarm Evol. Comput.*, vol. 44, pp. 1108–1118, Feb. 2019.

[18] A. Yaghoubzadeh-Bavandpour, O. Bozorg-Haddad, B. Zolghadr-Asli, and A. H. Gandomi, “Improving approaches for meta-heuristic algorithms: A brief overview,” in *Computational Intelligence for Water and Environmental Sciences* (Studies in Computational Intelligence), vol. 1043, O. Bozorg-Haddad and B. Zolghadr-Asli, Eds. Singapore: Springer, 2022.

[19] D.-H. Lee, “Resource-based task allocation for multi-robot systems,” *Robot. Auton. Syst.*, vol. 103, pp. 151–161, May 2018.

[20] V. Hassija et al., “Fast, reliable, and secure drone communication: A comprehensive survey,” *IEEE Commun. Surveys Tuts.*, vol. 23, no. 4, pp. 2802–2832, 4th Quart., 2021.

[21] J. Hu and J. Yang, “Application of distributed auction to multi-UAV task assignment in agriculture,” *Int. J. Precis. Agricult. Aviation*, vol. 1, no. 1, pp. 1–10, 2018.

[22] X. Fu, P. Feng, and X. Gao, “Swarm UAVs task and resource dynamic assignment algorithm based on task sequence mechanism,” *IEEE Access*, vol. 7, pp. 41090–41100, 2019.

[23] X. Chen, P. Zhang, G. Du, and F. Li, “A distributed method for dynamic multi-robot task allocation problems with critical time constraints,” *Robot. Auton. Syst.*, vol. 118, pp. 31–46, Aug. 2019.

[24] H.-L. Choi, L. Brunet, and J. P. How, “Consensus-based decentralized auctions for robust task allocation,” *IEEE Trans. Robot.*, vol. 25, no. 4, pp. 912–926, Aug. 2009.

[25] M. Otte, M. J. Kuhlman, and D. Sofge, “Auctions for multi-robot task allocation in communication limited environments,” *Auton. Robots*, vol. 44, nos. 3–4, pp. 547–584, Mar. 2020.

[26] M. G. Bakulin, V. B. Kreyndelin, V. A. Grigoriev, V. O. Aksenov, and A. S. Schesnyak, “Bayesian estimation with successive rejection and utilization of a priori knowledge,” *J. Commun. Technol. Electron.*, vol. 65, no. 3, pp. 255–264, Mar. 2020.

[27] J. Chen and D. Sun, “Resource constrained multirobot task allocation based on leaderfollower coalition methodology,” *Int. J. Robot. Res.*, vol. 30, no. 12, pp. 1423–1434, 2011.

[28] Z. Talebpour and A. Martinoli, “Risk-based human-aware multi-robot coordination in dynamic environments shared with humans,” in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2018, pp. 3365–3372.

[29] N. Sullivan, S. Grainger, and B. Cazzolato, “Sequential single-item auction improvements for heterogeneous multi-robot routing,” *Robot. Auton. Syst.*, vol. 115, pp. 130–142, May 2019.

[30] K.-S. Kim, H.-Y. Kim, and H.-L. Choi, “A bid-based grouping method for communication-efficient decentralized multi-UAV task allocation,” *Int. J. Aeronaut. Space Sci.*, vol. 21, no. 1, pp. 290–302, Mar. 2020.

Bo Yi (Member, IEEE) is currently a Lecturer of computer science and engineering with Northeastern University, China. He has authored and coauthored more than 20 journals and conference papers on IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, IEEE TRANSACTIONS ON CLOUD COMPUTING, IEEE GlobeCom, IEEE COMMUNICATIONS LETTER, and *Computer Networks*. His current research interests include service computing, routing, virtualization, and cloud computing in SDN, NFV, and DetNet. He is currently a reviewer of IEEE COMMUNICATIONS SURVEY AND TUTORIAL, IEEE COMMUNICATIONS LETTER, and *Computer Networks*.

Jianhui Lv (Member, IEEE) received the B.S. degree in mathematics and applied mathematics from the Jilin Institute of Chemical Technology, Jilin, China, in 2012, and the M.S. and Ph.D. degrees in computer science from Northeastern University, Shenyang, China, in 2014 and 2017, respectively. He is currently an Associate Professor with the Pengcheng Laboratory, China. He has published more than 40 journals and conference papers. His current research interests include artificial intelligence, ICN, in-network caching-enabled networks, the IoT, bio-inspired networking, evaluation computation, edge computing, and smart city. He has served as the Leader Guest Editor (LGE) for several international journals, such as *Applied Soft Computing*, *Digital Communications and Networks*, *Expert Systems*, *Wireless Networks*, *International Journal on Artificial Intelligence Tools*, and *Mobile Information Systems*.

Jiahao Chen received the B.S. degree in communication engineering from the Central South University of Forestry and Technology, Changsha, China, in 2020. He is currently pursuing the M.S. degree with Northeastern University, Shenyang, China. His current research interests include network routing optimization and the application of artificial intelligence technology in networks.

Xingwei Wang received the B.S., M.S., and Ph.D. degrees in computer science from Northeastern University, Shenyang, China, in 1989, 1992, and 1998, respectively. He is currently a Professor with the College of Computer Science and Engineering, Northeastern University. He has published more than 100 journal articles, books and book chapters, and refereed conference papers. His current research interests include cloud computing and future internet. He has received several best paper awards.

Keqin Li (Fellow, IEEE) is currently a SUNY Distinguished Professor of computer science with The State University of New York. He is also a National Distinguished Professor with Hunan University, China. He has authored or coauthored more than 780 journal articles, book chapters, and refereed conference papers. He holds more than 60 patents announced or authorized by the Chinese National Intellectual Property Administration. His current research interests include cloud computing, fog computing and mobile edge computing, energy-efficient computing and communication, embedded systems, cyber-physical systems, heterogeneous computing systems, big data computing, high-performance computing, CPU-GPU hybrid and cooperative computing, computer architectures and systems, computer networking, machine learning, and intelligent and soft computing. He has received several best paper awards. He is among the world’s top 10 most influential scientists in distributed computing based on a composite indicator of Scopus citation database. He has chaired many international conferences. He is currently an Associate Editor of the *ACM Computing Surveys* and the *CCF Transactions on High-Performance Computing*. He has served on the editorial boards of the IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, the IEEE TRANSACTIONS ON COMPUTERS, the IEEE TRANSACTIONS ON CLOUD COMPUTING, the IEEE TRANSACTIONS ON SERVICES COMPUTING, and the IEEE TRANSACTIONS ON SUSTAINABLE COMPUTING.