# TMA-DPSO: Towards Efficient Multi-Task Allocation With Time Constraints for Next Generation Multiple Access

Mingfeng Huang, Victor C. M. Leung, *Life Fellow, IEEE*, Anfeng Liu, and Neal N. Xiong, *Senior Member, IEEE*

***Abstract*— Future heterogeneous services and applications require the provisioning of unprecedented massive user access, heterogeneous data traffic, high bandwidth efficiency, and low latency services in next generation multiple access. In response to the requests from these services and applications, a large number of workers with scattered computing power need to be managed uniformly and scheduled in an efficient manner to perform various tasks. Therefore, task allocation has become a crucial issue to determining whether next generation multiple access can support future heterogeneous services and applications. In this paper, we propose a novel Two-stage Multi-task Allocation method based on Discrete Particle Swarm Optimization (TMA-DPSO). TMA-DPSO is easy to implement and has good search efficiency, which is suitable for large-scale task allocation in next generation networks. Under TMA-DPSO, we redefine the particles in discrete coding form, iteratively update the position and velocity based on the individual optimal particles and the global optimal particle, and finally obtain a corrected optimal solution. Unlike previous methods that only focused on the first-stage task allocation, we make full use of workers' remaining time to perform second-stage redundant task allocation, which can not only increase workers' income, but also potentially improve fault tolerance and security. As far as we know, this is the first attempt to utilize the remaining time after first-stage allocation. Finally, we evaluate TMA-DPSO extensively using the synthetic and real-life datasets. The results demonstrate that whether in a compactly or uniformly distributed scene, TMA-DPSO outperforms three benchmark methods by increasing 2.15%-42.24% platform revenue and 6.1%-46.63% workers income.**

***Index Terms*— Next generation multiple access, multi-task allocation, discrete particle swarm optimization, platform revenue.**

Manuscript received July 16, 2021; revised November 12, 2021; accepted December 17, 2021. Date of publication January 18, 2022; date of current version April 18, 2022. This work was supported in part by the National Natural Science Foundation of China under Grant 62072475 and Grant 61772554 and in part by the Hunan Provincial Innovation Foundation for Postgraduate under Grant CX20200212 and Grant 2020zzts139. *(Corresponding author: Neal N. Xiong.)*

Mingfeng Huang and Anfeng Liu are with the School of Computer Science and Engineering, Central South University, Changsha 410083, China (e-mail: mingfenghuang@csu.edu.cn; afengliu@csu.edu.cn).

Victor C. M. Leung is with the College of Computer Science and Software Engineering, Shenzhen University, Shenzhen, Guangdong 518060, China, and also with the Department of Electrical and Computer Engineering, The University of British Columbia, Vancouver, BC V6T 1Z4, Canada (e-mail: vleung@ieee.org).

Neal N. Xiong is with the Department of Computer Science and Mathematics, Sul Ross State University, Alpine, TX 79830 USA (e-mail: xiongnaixue@gmail.com).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/JSAC.2022.3143205.

Digital Object Identifier 10.1109/JSAC.2022.3143205

## I. Introduction

With the explosive growth of Internet-of-Things (IoT) [1]–[3], mobile applications are expected to continue their current exponential growth in next generation wireless networks [4]–[7]. Meanwhile, future heterogeneous services and applications, such as Extended Reality (XR), holographic telepresence, robotics, and Brain Computer Interfaces (BCI) require the provisioning of unprecedented massive user access, sub-ms latency, 3D ubiquitous coverage, cm-level localization in next generation wireless networks [1], [4], [6], [8]. Multiple access allows multiple users to be accommodated in the allocated resource block in most efficient manner, such as time slot, frequency band, spread spectrum code and power level [9]–[12]. Therefore, the aforementioned services and applications must be supported by the novel Next Generation Multiple Access (NGMA) technology to maximize their effectiveness. To accomplish this, NGMA solutions are combined with new technologies such as NGMA communication technology [13], [14], NGMA resource allocation [15], [16], NGMA security, NGMA hardware implementation, etc. to achieve its full potential in practical scenarios. Among them, task allocation and resource utilization are important issues in NGMA [15], [16]. To timely respond to the requests from services and applications, it is necessary to manage a large number of workers with scattered computing power, schedule them in an effective manner to perform massive computation, collection, and analysis tasks [17]. Therefore, efficient task allocation is inevitably playing an important role in next generation multiple access and become the backbone for supporting future heterogeneous services and applications.

As a crucial part of four-stage life cycle (i.e., task creation, task allocation, task execution, data integration [18], [19]), task allocation has become a fundamental research issue in mobile crowdsourcing [19]–[21], robot/drone scheduling [15], [22], [23], edge computing [24], etc. Recently, many previous studies explore task allocation about multiple access edge computing [24]–[26]. For our paper, we consider how to utilize multiple access workers with scattered computing power to carry out better large-scale task allocation in NGMA. For the task allocation in conventional multiple access environment, most are single task-oriented allocation, which assumes that the tasks are isolated and each task is assigned independently [18], [27], [28]. In this single task allocation scenario, a worker

can only take on one task at a time. If he wants to take on multiple tasks, he has to interact with the platform many times and wait for the tasks to be assigned. Therefore, a worker who undertakes multiple tasks may have large accumulated traveling distances because he needs to start from the initial location each time [18], [20]. Especially with the significant increase in the number of workers and tasks in recent years, the assumption of task independence no longer holds, because tasks must compete in a shared and limited resource pool (i.e., the full set of workers). In response to these, some efforts [16], [22], [29] are made for multi-task allocation. For example, in [18] and [30], multi-task allocation for maximizing overall utility with sensing and coverage capability constraints was studied. In [17], [20], [28], several solutions for mobile crowdsourcing are proposed. Although many task allocation methods have been proposed, as far as we know, the existing methods are not suitable for NGMA due to the following reasons:

First, the scale of workers and tasks is growing explosively in NGMA [1], [5], [6], and this large-scale allocation has a huge solution space, which makes the previous methods inefficient in finding the optimal solution. It is predicting that new applications (such as XR and BCI) may require a task allocation increase by 1000x in the future [23]. Assuming there are $m$ tasks and $n$ workers, if adopts the single task allocation, there are at most $m \cdot 2^n$ combinations. However, if the multi-task allocation is adopted, there is at most $m! \cdot (n+1)^m$ combinations, which is a much larger search space. If the traditional greedy or random methods are adopted, huge computing power is first needed to establish the overhead matrix between workers and tasks, which is costly. Even if the optimized genetic algorithm is used, complex coding and adjustment parameters are required. Therefore, with the dramatic increase in the number of workers and tasks, the previous methods have been unable to cope with this large-scale multi-task allocation.

Second, task allocation in NGMA has strict time constraints for both workers and tasks, which is a complex NP-complete problem [19]. Future applications such as interactive games, augmented reality/virtual reality, etc. require sub-ms or even lower latency [9], [23], so tasks have strict time constraints. While workers who undertake tasks are limited by battery, distance, etc., so their working time is also limited. In this situation where both tasks and workers have time constraints, task allocation has become unprecedentedly complicated. It not only concerns the task scheduling sequence of a single worker, but also involves the competition between workers and tasks. As shown in Fig. 1, assuming that the revenue of each task is the same, we aim to maximize the total revenue of the platform. For a single worker $W_3$, the tasks that can be assigned to it are $\{T_4\}$, $\{T_4, T_3\}$ or $\{T_3, T_4\}$. Obviously, if only assign $\{T_4\}$ to $W_3$, then $T_3$ is not allocated, which will reduce the platform revenue. And if it is allocated in the order of $\{T_4, T_3\}$, it will lead to the time to reach $T_3$ exceeds the time constraint of the task. Therefore, how to ensure the maximum overall utility in strict constraints in NGMA is a critical issue.

Third, the resource utilization has become extremely important in NGMA due to scarce resources and fierce competition, while the previous methods tend to focus only on the allocation process, does not consider the maximum utilization of workers' time and computing resources. For workers with an initial working time of 2-10 minutes, according to our statistics in the experiment, after the first-stage task allocation, there is an average of 20.08% time remaining in the compactly distributed scene, in scene where tasks are evenly distributed, the remaining time of workers is with an average of 31.89%, even some workers who have not yet assigned tasks can reach up to 100%. But the remaining time is wasted in the previous.

To address the above issues in NGMA, we propose a Two-stage Multi-task Allocation method based on novel Discrete Particle Swarm Optimization (TMA-DPSO). TMA-DPSO has the characteristics of simple implementation, few adjustment parameters, and good search efficiency. All these characteristics can well support large-scale task allocation in future heterogeneous services and applications. What's more, we utilize the remaining time of workers to conduct the second-stage redundant allocation, thus further improving revenue. Specifically, the contributions of this article are as follows:

1) For the multi-task allocation with time constraints for both workers and tasks in next generation multiple access, we propose a TMA-DPSO method based on novel discrete particle swarm optimization. In TMA-DPSO, we first model it as a single-objective optimization problem with the goal of maximizing platform revenue, and then redefine the particle's velocity and position in discrete encoding. Based on the randomly generated initial particles, we iteratively update their velocity and position based on individual optimal particles and the global optimal particle, and repair the validity of the updated particles according to the constraints, finally obtain the best allocation result. Compared with previous methods, TMA-DPSO has the advantages of good search efficiency, low complexity, and easy implementation. It is suitable for large-scale task allocation with time constraints in next generation multiple access.
2) Different from the previous methods that only focused on task allocation in the first stage, under TMA-DPSO, we make full use of the remaining time of workers for the second-stage redundant task allocation. In redundant task allocation, each worker will take on new tasks assigned to other workers in the first stage but not assigned to it. With redundant allocation, not only the worker's income and participation enthusiasm can be effectively increased, but also the platform security and data quality can be improved. To the best of our knowledge, this is the first attempt to utilize the remaining time after first-stage allocation to improve revenue and security.
3) Extensive experiments conduct on the synthetic and real-life datasets demonstrate that TMA-DPSO has better search efficiency and stable optimization results whether in densely or evenly distributed scenarios. Compared

with baseline methods, it increases platform revenue by 2.15%-42.24%, increases average income of workers by 6.1%-46.63%.

The rest of this paper is organized as follows. Section II introduces related works. The system model is presented in Section III. In Section IV, we propose the TMA-DPSO method. Then, Section V provides performance analysis. Finally, conclusion and future works are given in Section VI.

## II. Related Work

Task allocation can be divided into pull mode and push mode [17], [31]–[33]. In the pull mode, the platform releases various tasks, and then workers proactively decide which tasks to undertaken. Since workers select tasks based on their own preference, it is difficult to achieve the global optimized utility. In the push mode, the platform centrally allocates tasks to workers according to goals. This mode has become a research focus in recent years. In this article, we mainly discuss the task allocation based on push mode.

In the past, researchers mostly focus on single-task allocation. In single-task oriented allocation, each task is independent, and a worker only undertakes one task at a time. In [17], Wang *et al.* provide a comprehensive review for feature analysis, problem modeling and implementation algorithms in task allocation of mobile crowdsensing. For Piggyback Crowdsensing, Xiong *et al.* [28] propose a generic task allocation framework iCrowd. iCrowd can achieve dual optimization goals, one is to maximize overall $k$-depth coverage under a fixed budget, and the other is to minimize incentive payments under a predefined $k$-depth coverage constraint. Cardone *et al.* [34] propose a matching algorithm to deliver sensing tasks to users in smart city. First, they propose a geo-social model to build time-variant resource maps. Then, according to specific urban crowdsensing goals to find the best matching set. Finally, a crowdsensing platform is established for task distribution. In [20], Cheung *et al.* consider time, moving cost, reputation factors, propose a distributed allocation method. This method can achieve performance comparable to centralized methods.

Obviously, this single-task oriented allocation assumes that the ability of workers is homogeneous and unlimited, which is unrealistic. So, researchers have proposed various multi-task allocation solutions. Zhang *et al.* [22] propose a task distribution algorithm, which allows multiple mobile agents to collaboratively undertake multiple sensing tasks. First, authors establish a correlation model to consider the score and feature factors of mobile agents, and then give a mobility model compiled by an exponential distribution to predict the movement trajectory of the agent. Finally, authors integrate the correlation model and mobility model to propose a task distribution algorithm with the intent to improve the efficiency and accuracy. Choi *et al.* [35] propose two decentralized algorithms CBAA and CBBA to solve the single-task and multi-task allocation problems of a fleet of autonomous vehicles. The algorithm characterizes task selection as auction with greedy heuristics, and uses a market-based decision principle for task selection. In previous studies, more attention has been paid to task allocation in scenarios where there are more participants than tasks, while in [21], Liu *et al.* look at it from another perspective, classifying multi task allocation according to the number relation between participants and tasks, and provide corresponding solutions

In recent years, some efforts [18], [19], [30] are made for multi-task allocation with constraints. For participating sensing with sensing capability constraints, Wang *et al.* [18] propose a framework called PSAllocator. Different from previous approaches, the authors hope to coordinate the allocation of multiple tasks to maximize system utility. First, it predicts participant connections based on historical data, and then models the multi-task allocation into a bipartite graph, and uses an iterative greedy to find the optimal solution. Khalil *et al.* [16] address energy-constrained task allocation issue in the IoT environment. First, the authors model the problem as a single-objective optimization, and then propose two protocols to minimize energy expenditures. It adopts meta-heuristic methods with task groups and virtual objects dependencies. In [31], Goel *et al.* propose a multi-task allocation framework TM-UNIFORM, which can realize near-optimal utility while guarantying budget feasibility. This method uses bipartite graph to describe the feasibility between participants and tasks, and each participant has certain expertise and interests for tasks. Assuming that workers are subject to certain incentives for completing tasks, Zhang *et al.* [30] choose a near-minimal number of participants to cover a predefined percentage of subareas to minimize the total incentive payment with a greedy algorithm.

However, in the real world, not only the time and energy of workers are limited, but tasks may also be constrained. This task assignment with constraints on both workers and tasks is much more complicated than traditional task allocation, and can be proved to be an NP-complete problem. In [33], authors studied a destination-aware task allocation to maximize the number of allocated tasks. Assume that workers have deadlines to reach the destinations when completing tasks. Based on the observation of task distribution dependencies, the authors use tree-decomposition technology to divide workers into independent clusters, and propose an efficient depth-first search algorithm with progressive bounds to prune non-promising allocation. In [19], the authors propose a task allocation method MATC-IGA based on genetic algorithm. First, the initial chromosomes are generated based on the random-greedy algorithm, and then the elite preservation strategy and the competition strategy are used for chromosome selection for retaining the excellent genes to the next round. After that, crossover and mutation operations are performed on the chromosomes, and finally the chromosomes are corrected according to the constraints of the workers and tasks. On the basis of MATC-IGA, the authors introduce the vaccine generation and infusion operator and propose the MATC-IGA algorithm to avoid premature population maturity.

However, the previous method is complex to implement, too many adjustment parameters, and little consideration is given to resource maximization, which is not suitable for large-scale multi task allocation in next-generation networks. Unlike the previous methods, in this paper we solve the multi task allocation with constraints on both workers and tasks based on the discrete particle swarm algorithm. We redefine coding,

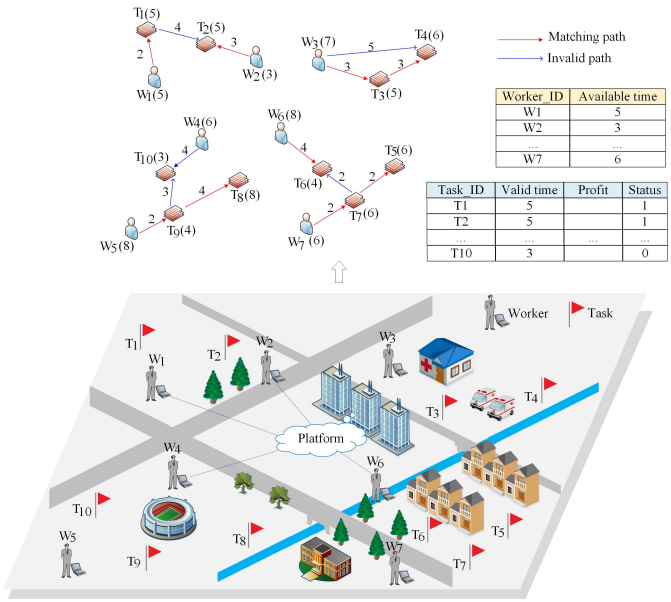



Fig. 1. An example of multi-task allocation with time constraints.

particle update rules, and introduce adjustment functions and random quantities to avoid premature and fall into local optimization. Our method has advantages of fast search, few adjustment parameters and is easy to realize. Most importantly, we make full use of the remaining time of workers to further improve performance [23].

## III. System Model and Problem Statement

### A. System Model

In this paper, we consider location-based multi-task allocation with time constraints on both workers and tasks. As shown in Fig. 1, $n$ workers and $m$ tasks are randomly distributed in a limited geographic area. Denote the set of workers as $\mathbb{W} = \{W_1, W_2, \ldots, W_n\}$, the set of tasks as $\mathbb{T} = \{T_1, T_2, \ldots, T_m\}$, the platform centrally controls workers and tasks, and their responsibilities are as follows:

1) Platform. Restricted by the geographic location and time constraints of tasks and workers, the platform determines the appropriate task allocation to maximize revenue. For a task $T_i$ with a profit $\gamma$, assign it to the worker $W_k$ with a reward $c$, the platform revenue is $\beta$, $\beta = \gamma - c$.
2) Workers. Workers are employed by the platform to undertake tasks. Motivated by task rewards, a worker $W_k$ can be charged with multiple tasks, and its information include $\left(W_k, t_{av}^{W_k}\right)$, $W_k$ is the unique identifier of the worker, $t_{av}^{k}$ is the available working time.
3) Task. A task can only be assigned to one worker at most, and its information include $\left(T_i, t_v^{T_i}, \gamma_i, \chi_i\right)$. $T_i$ is the task unique identifier, $t_v^{T_i}$ is the valid time, $T_i$ must be executed by the worker within $t_v^{T_i}$ duration from the current time. $\gamma_i$ is the task profit, and $\chi_i$ is the assignment status. For the $\chi_i$ of task $T_i$, there are:

$$\chi_{i,k} = \begin{cases} 1, & if\ T_i\ is\ assigned\ to\ W_k, \\ 0, & otherwise. \end{cases} \tag{1}$$

Since we are considering multi-task allocation with time constraints, suppose $|d_{W_k \to T_i}|$ is the distance from the current location of $W_k$ to task $T_i$, and $\varsigma_k$ is the travel speed of $W_k$, then the time cost required for the worker $W_k$ from its current location to $T_i$ is calculated as:

$$t_{W_k \to T_i} = \frac{|d_{W_k \to T_i}|}{\varsigma_k}. \tag{2}$$

### B. Research Objective

We hope to find the best match between workers and tasks, so as to maximize the revenue of the platform, so the research objective can be expressed as:

$$\begin{aligned} Max\ & \sum_{i=1}^{m}\sum_{k=1}^{n} \beta_i \cdot \chi_{i,k}. \\ \text{subject to: } & C1: \chi_{i,k} \in \{0,1\}, \quad \forall W_k \in \mathbb{W},\ T_i \in \mathbb{T}, \\ & C2: \sum_{k=1}^{n} \chi_{i,k} \le 1, \quad \forall T_i \in \mathbb{T}, \\ & C3: \sum_{i=1}^{\vartheta} t_{W_k \to T_i} \le t_{av}^{W_k}, \quad \forall W_k \in \mathbb{W}, \\ & C4: t_{W_k \to T_i} \le t_v^{T_i}, \quad \forall \chi_{i,k} = 1, T_i \in \mathbb{T}. \end{aligned} \tag{3}$$

In Formula (3), $\beta_i$ is the revenue obtained by the platform after task $T_i$ is successfully assigned. $\chi_{i,k}$ is the status of whether $T_i$ is assigned to worker $W_k$. $t_{W_k \to T_i}$ is the time required from $W_k$ to $T_i$. $t_{av}^{W_k}$ is the available working time of $W_k$, $t_v^{T_i}$ is the time constraint of $T_i$. Constraints C1 and C2 ensure that a task can only be assigned to one worker at most. C3 guarantees that the accumulated time from $W_k$ to the $\vartheta$ tasks must be less than its available working time, thereby satisfying the time constraint of $W_k$. C4 guarantees that the time from $W_k$ to $T_i$ must meet the time constraint of $T_i$. In this paper, we stipulate that a valid allocation must satisfy the constraints C1-C4 simultaneously. The task allocation with red path shown in Fig. 1 is a valid allocation (also called a valid particle).

For clarity, we have classified and summarized the symbols in this paper. The numerical parameters and their descriptions are listed in Table I, and the vector symbols and their interpretation are listed in Table II.

## IV. Our Proposed TMA-DPSO Scheme

In the next generation multiple access, the scale of task allocation is larger and the time constraint will be stricter, which is a complex combinatorial optimization problem. As proved by [19], its solution space is very large, and is difficult to obtain the exact solution in a polynomial time by traditional methods. Particle swarm optimization has been proven to be one of the most popular optimization techniques due to its advantages of fast search, high efficiency, few adjustment parameters and easy to realize [36]–[38]. However, the traditional particle swarm algorithm is based on continuous space search. While we are solving a discrete allocation in this paper, so we adopt discrete particle swarm optimization algorithm to map the discrete space to the continuous particle motion space [39], [40]. We still retain the advantages of the classic

TABLE I
NUMERICAL PARAMETERS AND THEIR DESCRIPTIONS

| Parameters | Description |
|---|---|
| $n$ | Number of workers |
| $m$ | Number of tasks |
| $\gamma$ | Task profit |
| $c$ | Task reward |
| $\beta$ | Platform revenue |
| $t_{av}^{W_k}$ | Available working time of $W_k$ |
| $t_v^{T_i}$ | Valid time of $T_i$ |
| $\chi_i$ | Allocation status of $T_i$ |
| $t_{W_k \to T_i}$ | Time required from $W_k$ to $T_i$ |
| $d_{W_k \to T_i}$ | Distance from $W_k$ to $T_i$ |
| $\varsigma_k$ | Travel speed of $W_k$ |
| $\vartheta$ | Number of tasks assigned to $W_k$ |
| $\mu$ | Number of particles |
| $i_{Max}$ | The maximum number of iterations |

TABLE II
VECTOR SYMBOLS AND THEIR INTERPRETATION

| Symbols | Interpretation |
|---|---|
| $\mathbb{W}$ | Set of workers |
| $W_k$ | The *k-th* worker |
| $\mathbb{T}$ | Set of tasks |
| $T_i$ | The *i-th* task |
| $\mathbb{Q}/\mathbb{SQ}$ | Particle swarm in the first/second stage |
| $\mathcal{Q}_j/S\mathcal{Q}_j$ | The *j-th* particle (position) in the first/second stage |
| $\mathcal{F}\left(\mathcal{Q}_j^{(i)}\right)$ | The fitness of $\mathcal{Q}_j$ in the *i-th* iteration |
| $\mathcal{Q}_{Pbest}^{j(i)}$ | Individual optimal particle of $\mathcal{Q}_j$ in the *i-th* iteration |
| $\mathcal{Q}_{Gbest}^{(i)}$ | Global optimal particle in the *i-th* iteration |
| $\mathcal{V}_j^{(i)}$ | Velocity of $\mathcal{Q}_j$ in the *i-th* iteration |
| ${\mathcal{Q}_{Frag}}_j^k$ | Particle (position) fragment of $W_k$ in $\mathcal{Q}_j$ |
| $\mathcal{V}_j^k$ | Velocity fragment of $W_k$ in $\mathcal{V}_j$ |
| ${Max\_\mathcal{Q}_{Frag}}_j^{k(i)}$ | Particle (position) fragment with maximum fitness |
| $\mathbb{C}_{\mathcal{Q}_{Frag}}^{T_i}$ | Set of all particle (position) fragments containing $T_i$ |
| $\mathbb{C}_{\mathcal{Q}_{Frag}}^{Max\_T_i}$ | Set of fragment(s) with maximum fitness in $\mathbb{C}_{\mathcal{Q}_{Frag}}^{T_i}$ |
| $\mathbb{T}_{UA}$ | Set of unassigned tasks |
| $\mathbb{W}_{RT}$ | Set of workers with remaining time |

particle swarm algorithm, and introduce random variables to balance the convergence speed and avoid premature.

Specifically, we propose a Two-stage Multi-task Allocation scheme based on Discrete Particle Swarm Optimization (TMA-DPSO). It mainly includes two phases of task allocation. In the first-stage task allocation, the initial particles and their velocity are generated randomly based on the initial workers and tasks information. Then, the particle velocity and position are updated continuously based on individual and global optimal values, and the validity of particles is repaired based on constraints. Finally, the best task allocation (particle) which makes the platform revenue maximum is found. On this basis, we further improve and make full use of the remaining time to allocate redundant tasks in the second stage. As far as we know, this is the first time to use the remaining time for redundant allocation to enhance fault tolerance and security. In the second-stage task allocation, each worker takes on some tasks that have been allocated to other workers in the first stage but have not been allocated to itself. We call this redundant allocation.

Overall, the TMA-DPSO addresses two issues. The first is to solve the multi-task allocation with time constraints by improving the discrete particle swarm algorithm, and obtain the best worker-task matching result. The second is to make full use of the remaining time of workers to allocate redundant tasks, thus further improving performance without incurring other costs. Next, we explain these two issues in detail.

### A. First-Stage Task Allocation of TMA-DPSO

The implementation of first-stage task allocation based on discrete particle swarm optimization is given in Algorithm 1. It includes the following steps: (1) Determine the particle position and velocity representation. In this article, the position of particles is encoded with a varying length, and the velocity is a vector of fixed length with a value of 0 or 1. (2) Initialize the position and velocity of particles. To maintain the diversity of particles, Algorithm 2 is used to generate $\mu$ particles based on the random and greedy principle, thus forming an initial particle swarm $\mathbb{Q} = \{\mathcal{Q}_1, \mathcal{Q}_2, \ldots \mathcal{Q}_\mu\}$, and the $\mathbb{Q}$ is used for the first iteration (Lines 1-5). (3) Compute the fitness of each particle, and find the individual optimal particle and the global optimal particle. The fitness of each particle is computed with Formula (4), Then compare the fitness in multiple iterations to find the individual optimal particle, compare the individual optimal particles of all particles to obtain the global optimal particle (Lines 7-11). (4) Update the velocity of particles in the next iteration based on the history velocity, individual optimal particle, and global optimal particle (Line 13). (5) Update the position of particles based on the new velocity, and repair the validity of particles using Algorithm 3. Then, the particle swarm in the next iteration is obtained (Lines 14-16). Repeat the above (3)-(5) steps until the Algorithm 1 converges or reaches the maximum number of iterations. Finally, the best task allocation result and its fitness is computed.

*1) Particle Representation:* Particles have two important parameters: position and velocity. Taking the task allocation in Fig. 1 as an example, the position and velocity representation is illustrated in Fig. 2.

*a) Position:* A particle's position represents a allocation solution, and a position fragment represents the task allocation of a worker in the particle. Suppose there are $n$ workers, the position is represented as $\mathcal{Q}_j = \left\{{\mathcal{Q}_{Frag}}_j^1, {\mathcal{Q}_{Frag}}_j^2, \ldots {\mathcal{Q}_{Frag}}_j^n\right\}$, it is coded in order according to the identifier of workers, and each particle position fragment ${\mathcal{Q}_{Frag}}_j^k$ corresponds to the task assignment of the worker $W_k$. The length of the position fragment of each worker is not fixed. When the worker is not assigned or only assigned one task, its length is 1, as workers $W_1, W_2, W_4$, and $W_6$ shown in Fig. 2. Otherwise, it is the number of assigned tasks, such as $W_3, W_5$, and $W_7$. The position not only indicates the match between workers and tasks, but also reflects the sequential order of the multiple tasks assigned to a worker. For example, the first task allocated to $W_3$ is $T_3$, then $T_4$. If the order of $T_3$ and $T_4$ is exchanged

**Algorithm 1** First-Stage Task Allocation Based on DPSO

Input: $\mathbb{W}$ set of workers, $\mathbb{T}$ set of tasks, $\mu$ number of particles, $i_{Max}$ number of iterations
Initialize: i = 1, $\mathbb{Q}$ = null
1 **While** $|\mathbb{Q}| < \mu$ **do**
2 Generate a new particle $\mathcal{Q}_j$ using Algorithm 2
3 $\mathbb{Q} \leftarrow \mathbb{Q} + \{\mathcal{Q}_j\}$
4 **End while**
5 Let $\mathbb{Q}^{(i)} \leftarrow \mathbb{Q}$
6 **While** $i < i_{Max}$ **do**
7 **For** each particle $\mathcal{Q}_j^{(i)} \in \mathbb{Q}^{(i)}$ **do**
8 Compute its fitness $\mathcal{F}(\mathcal{Q}_j^{(i)})$ using Formula (4)
9 Compute its individual optimal particle $\mathcal{Q}_{Pbest}^{j(i)}$
10 **End for**
11 Compute global optimal particle $\mathcal{Q}_{Gbest}^{(i)}$ of all particles
12 **For** each particle $\mathcal{Q}_j^{(i)} \in \mathbb{Q}^{(i)}$ **do**
13 Update its velocity $\mathcal{V}_j^{(i+1)}$ using Formulas (5) and (6)
14 Update its position $\mathcal{Q}_j^{(i+1)}$ using Formula (7)
15 Repair validity of $\mathcal{Q}_j^{(i+1)}$ using Algorithm 3
16 Let $\mathbb{Q}^{(i+1)} \leftarrow \mathbb{Q}^{(i+1)} + \left\{\mathcal{Q}_j^{(i+1)}\right\}$
17 **End for**
18 i++
19 **End while**
Output: The optimal particle and its fitness

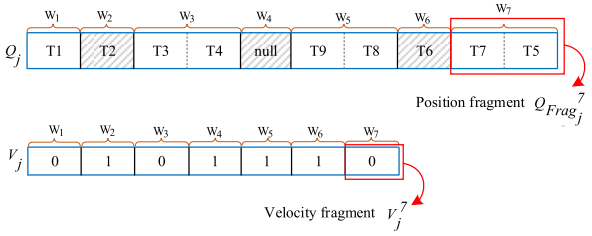



Fig. 2. Particle position and velocity representation.

in the representation, it represents another task assignment. A valid position representation needs to satisfy the following constraints: 1) The length of particle position encoding is at least $n$ bits; 2) The value of each bit is null or $T_i, T_i \in \mathbb{T}$; 3) A task $T_i$ can only appear in the particle at most once.

*b) Velocity:* Velocity represents the change state of the particle position. It is a binary-coded vector with a fixed length of $n$, and the *k-th* bit represents the velocity of the worker $W_k$. The velocity corresponding to the particle $\mathcal{Q}_j$ can be expressed as $\mathcal{V}_j = \{\mathcal{V}_j^1, \mathcal{V}_j^2, \ldots \mathcal{V}_j^n\}$. If $\mathcal{V}_j^k = 1$, it means $W_k$ adjusts its position in the next iteration. If $\mathcal{V}_j^k = 0$, $W_k$ keeps its original state. A good velocity guides the particle to keep getting closer to the optimized value. Randomly generate the velocity, we get the velocity representation of $\mathcal{Q}_j$, which is depicted in Fig. 2.

*2) Particle Initialization:* The particle initialization is given in Algorithm 2. Suppose the set of $n$ workers is $\mathbb{W} = \{W_1, W_2, \ldots W_n\}$, the set of $m$ tasks is $\mathbb{T} = \{T_1, T_2, \ldots T_m\}$. We create the workers candidate set $\mathbb{W}'$, the unallocated tasks set $\mathbb{T}_{UA}$, and the task traversal candidate set $T'$ (Line 1). First, a worker $W_k$ is randomly selected from the workers candidate set $\mathbb{W}'$, and a task $T_i$ is randomly selected from the unallocated tasks set. Then check whether it is valid to assign the task $T_i$ to $W_k$, that is, check whether the C1-C4 constraints of Formula (3) are met. If it is a valid particle position fragment, assign task $T_i$ to $W_k$, and append $T_i$ to the end of task assignment sequence of $W_k$. At the same time, task $T_i$ is removed from the unallocated tasks set $\mathbb{T}_{UA}$, and the geographic location and remaining working time of $W_k$ are also updated (Lines 2-13). Then continue to find the next task until all tasks are traversed, and the task allocation of $W_k$ is completed. Next, randomly generate a value with 0 or 1 as the initial velocity of $W_k$. When the position and velocity of $W_k$ are initialized, it is deleted from the workers candidate set $\mathbb{W}'$, and a new worker is randomly selected for task allocation until all workers are traversed (Lines 14-17). Finally, it is sorted according to the identifier of workers. When sorting, the order of tasks within a worker is kept unchanged (Line 18). Run the Algorithm 2 $\mu$ times to get the initial particle swarm $\mathbb{Q}$.

*3) Fitness Evaluation:* Fitness is used to evaluate the quality of particles. Our fitness function is like Formula (3). For a particle $\mathcal{Q}_j$, if $T_i$ appears in the position of $\mathcal{Q}_j$, it means that the task $T_i$ is allocated, $\chi_i^j = 1$, otherwise $\chi_i^j = 0$. Suppose $\beta_i$ is the revenue obtained by the platform after $T_i$ is successfully assigned, then the fitness of particle $\mathcal{Q}_j$ is calculated as follows:

$$\mathcal{F}(\mathcal{Q}_j) = \sum_{i=1}^{m} \beta_i \cdot \chi_i^j. \tag{4}$$

**Algorithm 2** Particle Position and Velocity Initialization

Input: $\mathbb{W}$ set of workers, $\mathbb{T}$ set of tasks
1 Let $\mathbb{W}' \leftarrow \mathbb{W}$, $\mathbb{T}_{UA} \leftarrow \mathbb{T}$
2 **While** $\mathbb{W}' \neq \emptyset$ **do**
3 Randomly select a worker $W_k$ from $\mathbb{W}'$
4 Let $\mathbb{T}' \leftarrow \mathbb{T}_{UA}$
5 **While** $\mathbb{T}' \neq \emptyset$ **do**
6 Randomly select a task $T_i$ from $\mathbb{T}'$
7 $\mathbb{T}' \leftarrow \mathbb{T}' - \{T_i\}$
8 **If** $\mathcal{Q}_{Frag(W_k \leftarrow T_i)}$ is a valid position fragment for $W_k$ **do**
9 Assign $T_i$ to $W_k$ and append it to task sequence of $W_k$
10 $\mathbb{T}_{UA} \leftarrow \mathbb{T}_{UA} - \{T_i\}$
11 Update geographic location and working time of $W_k$
12 **End if**
13 **End while**
14 Let $\mathcal{V}_k$ be a random value in 0 or 1
15 $\mathcal{V} \leftarrow \mathcal{V} + \{\mathcal{V}_k\}$
16 $\mathbb{W}' \leftarrow \mathbb{W}' - \{W_k\}$
17 **End while**
18 Reorganize the task sequence of workers to obtain a particle
Output: A new particle including position and velocity

For a particle swarm $\mathbb{Q} = \{\mathcal{Q}_1, \mathcal{Q}_2, \ldots \mathcal{Q}_\mu\}$, if its each particle is continuously updated and iterated for $i$ rounds, then we define the individual optimal particle $\mathcal{Q}^j_{Pbest}$ and the global optimal particle $\mathcal{Q}_{Gbest}$ as follows: $\mathcal{Q}^j_{Pbest}$ is the position of the particle $\mathcal{Q}_j$ with the largest fitness in all iterations; $\mathcal{Q}_{Gbest}$ is the position with the largest fitness among all individual optimal particles.

*4) Particle Velocity Update:* The velocity $\mathcal{V}_j = \{\mathcal{V}^1_j, \mathcal{V}^2_j, \ldots \mathcal{V}^n_j\}$ of particle $\mathcal{Q}_j$ reflects its position change in each iteration. If $\mathcal{V}^k_j = 0$, the fragment ${\mathcal{Q}_{Frag}}^k_j$ of worker $W_k$ in the particle remains unchanged, otherwise the position is adjusted in the next iteration. For a particle $\mathcal{Q}_j$, suppose its position in the *i-th* iteration is $\mathcal{Q}^{(i)}_j$, its individual optimal particle is $\mathcal{Q}^{j(i)}_{Pbest}$, and the global optimal particle is $\mathcal{Q}^{(i)}_{Gbest}$, then its velocity $\mathcal{V}^{(i+1)}_j$ in the *(i+1)-th* iteration is updated as follows:

$$\mathcal{V}^{(i+1)}_j = S\left(\omega\mathcal{V}^{(i)}_j + c_1 r_1\left(\mathcal{Q}^{j(i)}_{Pbest} \oplus \mathcal{Q}^{(i)}_j\right) + c_2 r_2\left(\mathcal{Q}^{(i)}_{Gbest} \oplus \mathcal{Q}^{(i)}_j\right)\right). \tag{5}$$

In Formula (5), $\oplus$ is an XOR operation. The particle velocity is adjusted by three factors. One is its own velocity inertia $\omega\mathcal{V}^{(i)}_j$, where $\omega$ is inertia weight, which is used to record the current velocity and adjust the search range of the solution space. It is a non-negative number. The second is the influence of its historical experience $c_1 r_1\left(\mathcal{Q}^{j(i)}_{Pbest} \oplus \mathcal{Q}^{(i)}_j\right)$, $c_1$ is acceleration, which can adjust the maximum step length of learning. When $c_1 = 0$, its own historical experience is not considered, which will lead to the loss of particle diversity. The third is the historical experience influence of all particles $c_2 r_2\left(\mathcal{Q}^{(i)}_{Gbest} \oplus \mathcal{Q}^{(i)}_j\right)$, $c_2$ is acceleration, which can adjust the maximum step length of learning and the search space of the solution. When $c_2 = 0$, the experience of other particles is not considered. At this time, the convergence may slow down due to the lack of information sharing. Previous studies [36], [39] have shown that when $c_1$ and $c_2$ are set to 1.494 or 2, the algorithm works well for most applications. $r_1$ and $r_2$ are two random numbers with values between 0-1. If there is no special description, we set $\omega = 1$, $c_1$, $c_2$ to 1.494.

To ensure $\mathcal{V}^{(i+1)}_j = \{\mathcal{V}^{1(i+1)}_j, \mathcal{V}^{2(i+1)}_j, \ldots \mathcal{V}^{n(i+1)}_j\}$ computed by Formula (5) is binary, let $Y = \{Y_1, Y_2, \ldots Y_n\}$, $Y_k = \omega\mathcal{V}^{k(i)}_j + c_1 r_1({\mathcal{Q}_{Frag}}^{j,k(i)}_{Pbest} \oplus {\mathcal{Q}_{Frag}}^{k(i)}_j) + c_2 r_2({\mathcal{Q}_{Frag}}^{k(i)}_{Gbest} \oplus {\mathcal{Q}_{Frag}}^{k(i)}_j)$, then for each $\mathcal{V}^{k(i+1)}_j \in \mathcal{V}^{(i+1)}_j$, we define:

$$\mathcal{V}^{k(i+1)}_j = \begin{cases} 1, & if\ \mathfrak{r}_d < S(Y_k), \\ 0, & if\ \mathfrak{r}_d \geq S(Y_k), \end{cases}$$
$$and\ \ \mathfrak{r}_d = rand(0,1),\ S(Y_k) = \frac{1}{1+e^{-Y_k}}. \tag{6}$$

In Formula (6), $S(Y_k)$ is a Sigmoid function. It can map values of $Y_k$ to the interval of (0,1) without being restricted by the domain of definition, and is centrosymmetric at 0.5, thus helping us realize the binary value of $\mathcal{V}^{(i+1)}_j$. In the particle velocity update mechanism, we introduce random variables of $\mathfrak{r}_d$ to balance the convergence rate and avoid falling into local optimal.

*5) Particle Position Update and Repair:* Based on the velocity $\mathcal{V}^{(i+1)}_j = \left\{\mathcal{V}^{1(i+1)}_j, \mathcal{V}^{2(i+1)}_j, \ldots \mathcal{V}^{n(i+1)}_j\right\}$, for each ${\mathcal{Q}_{Frag}}^k_j$ of $\mathcal{Q}_j = \left\{{\mathcal{Q}_{Frag}}^1_j, {\mathcal{Q}_{Frag}}^2_j, \ldots {\mathcal{Q}_{Frag}}^n_j\right\}$ in the *(i+1)-th* iteration, it is updated as follows:

$${\mathcal{Q}_{Frag}}^{k(i+1)}_j = \begin{cases} {\mathcal{Q}_{Frag}}^{k(i)}_j, & if\ \mathcal{V}^{k(i+1)}_j = 0, \\ Max\_{\mathcal{Q}_{Frag}}^{k(i)}_j, & if\ \mathcal{V}^{k(i+1)}_j = 1. \end{cases} \tag{7}$$

Here, $Max\_{\mathcal{Q}_{Frag}}^{k(i)}_j$ refers to the position fragment with the largest fitness among ${\mathcal{Q}_{Frag}}^{k(i)}_j$, ${\mathcal{Q}_{Frag}}^{j,k(i)}_{Pbest}$ and ${\mathcal{Q}_{Frag}}^{k(i)}_{Gbest}$. Obviously, after the above position update, the new particle is not valid, because a task may be assigned to several workers. Then, we repair the particle validity. Since the position fragment of the particle is obtained based on the particle itself, the individual optimal particle or the global optimal particle, the time constraints of C3 and C4 are satisfied. Therefore, the particle validity repair process is as follows:

First, check whether constraints C1 and C2 are satisfied, that is, ensure a task is allocated to one worker at most. Assuming that a task is assigned to multiple workers, first calculate the fragment fitness of each worker, and then keep the fragment with the highest fitness unchanged (if the fragments of several workers are the maximum, one of them is randomly selected), then adjust the fragments of other workers. When adjusting, first delete this repetitive task from other fragments, and then find the largest subset that satisfies the constraints from the original task allocation. There are two checks to be done in finding the largest subset, one is to check whether the remaining tasks can be finished within the constrained time after the task is deleted; the other is to check whether it conflicts with the fragments of other workers. Repeat the above process until all tasks no longer conflict.

Next, further improve the existing match between tasks and workers, update the remaining time of workers, and traverse all unassigned tasks to see if new tasks can be added to workers with enough remaining time.

The above particle validity repair process can be represented by Algorithm 3.

*6) An Illustrative Example for Particle Evolution:* Next, we show a particle's velocity update, position update and validity repair process with a specific example.

1) Suppose there are 10 tasks and 7 workers, the profits of $T_1 - T_{10}$ are $\{2,5,3,4,1,2,3,5,4,5\}$. At the *i-th* iteration, $\mathcal{Q}^{(i)}_j = \{\{T_1\},\{T_2\},\{T_3,T_4\}.,$ null, $.\{T_9,T_8\},\{T_6\},\{T_7,T_5\}\}$, and the velocity is $\mathcal{V}^{(i)}_j = [0,0,1,1,0,1,1]$. The individual optimal particle of $\mathcal{Q}^{(i)}_j$ is $Q^{j(i)}_{Pbest} = \{\{T_1\},\{T_2\},\{T_5\},\{T_6\},\{T_9,T_8\},\{T_4,T_3\},\{T_7\}\}$. The global optimal particle is $\mathcal{Q}^{(i)}_{Gbest} = \{\{T_4,T_6\},\{T_2\},\{T_5\}.,$ null, $.\{T_9,T_8\},\{T_1,T_3\},\{T_7,T_{10}\}\}$. According to Formula (5) we can get the velocity of $Q^{(i)}_j$ in the next iteration, its calculation is shown in Fig. 3. We first set parameters $\omega = 1, c_1 = c_2 = 1.494$, generate two random values $r_1 = 0.52$, $r_2 = 0.86$, then

**Algorithm 3** Particle Validity Repair

Input: $Max\_\mathcal{Q}_{Frag_j}^{(i)}$ an invalid particle
**For** each task $T_i$ in $Max\_\mathcal{Q}_{Frag_j}^{(i)}$ **do**
  $\mathbb{C}_{\mathcal{Q}_{Frag}}^{T_i}$ ←Set of particle position fragments containing $T_i$
  $\mathbb{C}_{\mathcal{Q}_{Frag}}^{Max\_T_i}$ ←Set of fragment(s) with maximum fitness in $\mathbb{C}_{\mathcal{Q}_{Frag}}^{T_i}$
  **If** $\left|\mathbb{C}_{\mathcal{Q}_{Frag}}^{Max\_T_i}\right| > 1$ **do**
    Randomly select a fragment $\mathcal{Q}_{Frag_j}^{k(i)}$ from $\mathbb{C}_{\mathcal{Q}_{Frag}}^{Max\_T_i}$
    Let $\mathbb{C}_{\mathcal{Q}_{Frag}}^{Max\_T_i} \leftarrow \left\{\mathcal{Q}_{Frag_j}^{k(i)}\right\}$
  **End if**
  **For** each fragment $\mathcal{Q}_{Frag_j}^{k(i)}$ in $\mathbb{C}_{\mathcal{Q}_{Frag}}^{T_i} - \mathbb{C}_{\mathcal{Q}_{Frag}}^{Max\_T_i}$ **do**
    Remove task $T_i$
    Find maximum valid fragment $\mathcal{Q}_{Frag_j}^{k(i)^*}$ satisfying C1-C4
    Let $\mathcal{Q}_{Frag_j}^{k(i)} \leftarrow \mathcal{Q}_{Frag_j}^{k(i)^*}$
    Update the location and remaining working time of $W_k$
  **End for**
**End for**
Update the set of unassigned tasks set $\mathbb{T}_{UA}$
Update the set of workers with remaining time $\mathbb{W}_{RT}$
**While** $\mathbb{T}_{UA} \neq \emptyset$ **do**
  Randomly select a task $T_i$ from $\mathbb{T}_{UA}$
  Let $\mathbb{W}'_{RT} \leftarrow \mathbb{W}_{RT}$
  **While** $\mathbb{W}_{RT} \neq \emptyset$ **do**
    Randomly select a worker $W_k$ from $\mathbb{W}_{RT}$
    $\mathbb{W}'_{RT} \leftarrow \mathbb{W}'_{RT} - \{W_k\}$
    **If** $\mathcal{Q}_{Frag(W_k \leftarrow T_i)}$ is a valid particle fragment for $W_k$ **do**
      Assign $T_i$ to $W_k$ and append it to task sequence of $W_k$
      $\mathbb{T}_{UA} \leftarrow \mathbb{T}_{UA} - \{T_i\}$
      Update geographic location and working time of $W_k$
      **Goto** step 30
    **End if**
  **End while**
  Update the set of workers with remaining time $\mathbb{W}_{RT}$
**End while**
Let $\mathcal{Q}_j^{(i+1)} \leftarrow Max\_\mathcal{Q}_{Frag_j}^{(i)}$
Output: The repaired valid particle $\mathcal{Q}_j^{(i+1)}$

the velocity is calculated as follows:

$$\begin{aligned} Y &= \omega\mathcal{V}_j^{(i)} + c_1 r_1\left(\mathcal{Q}_{Pbest}^{j(i)} \oplus \mathcal{Q}_j^{(i)}\right) + c_2 r_2\left(\mathcal{Q}_{Gbest}^{(i)} \oplus \mathcal{Q}_j^{(i)}\right) \\ &= 1 * [0,0,1,1,0,1,1] + 1.494 * 0.52 * [0,0,1,1,0,1,1] \\ &\quad + 1.494 * 0.86 * [1,0,1,0,0,1,1] \\ &= [1.28,1,2.06,1.78,1,3.06,2.06] \end{aligned}$$

In the above, $\oplus$ means to perform XOR operation on two objects, and the result is 1 only when the values of the two objects are the same, otherwise it is 0. Then we map the value of Y to [0-1] through the Sigmoid function $\frac{1}{1+e^{-Y}}$, and

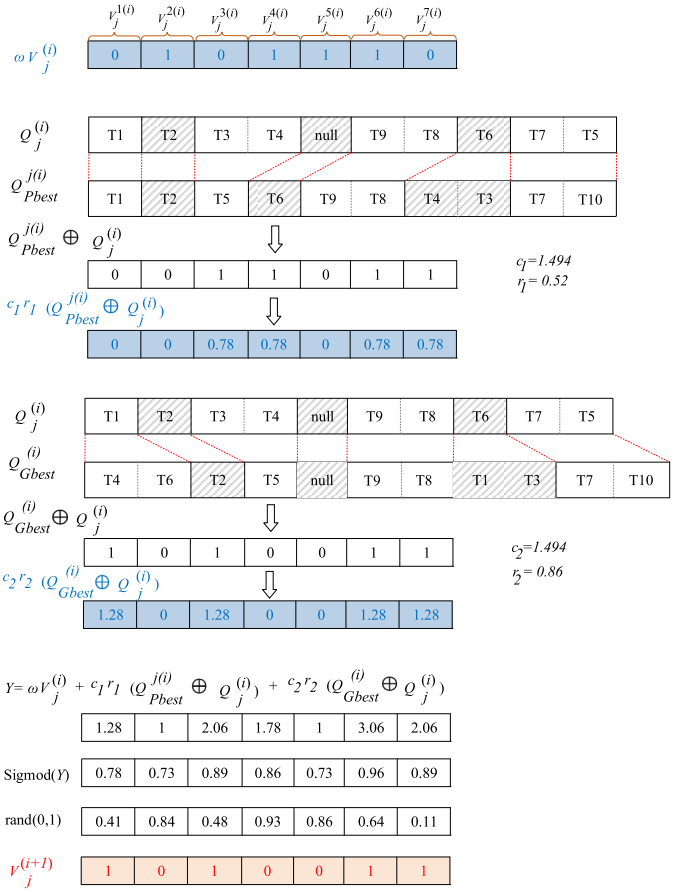



Fig. 3. Particle velocity update.

obtain Sigmoid(Y) = $[0.78, 0.73, 0.89, 0.86, 0.73, 0.96, 0.89]$. Next, we generate random values $[0.41, 0.84, 0.48, 0.93, 0.86, 0.64, 0.11]$. Compare Sigmoid(Y) and random value one-to-one. If it is greater than the random value, $\mathcal{V}_j^{(i)} = 1$, otherwise $\mathcal{V}_j^{(i)} = 0$. Finally, the velocity of the particles in the *(i+1)-th* iteration is updated to $\mathcal{V}_j^{(i+1)} = [1,0,1,0,0,1,1]$.

2) As shown in Fig. 4, according to the velocity $\mathcal{V}_j^{(i+1)} = [1,0,1,0,0,1,1]$, update the particle position at the *(i+1)-th* iteration. First, adjust the position fragment of the first worker $W_1$, which is $\{T_1\}$. Because $\mathcal{V}_j^{1(i+1)} = 1$, according to Formula (7), the task allocation (position fragment) of $W_1$ needs to be adjusted to $Max\_\mathcal{Q}_{Frag_j}^{1(i)}$. And the calculation of $Max\_\mathcal{Q}_{Frag_j}^{1(i)}$ is as follows: First calculate the fitness of position fragment of $W_1$ in the particle $\mathcal{Q}_j^{(i)}$, the individual optimal particle $\mathcal{Q}_{Pbest}^{j(i)}$ and the global optimal particle $\mathcal{Q}_{Gbest}^{(i)}$, which are $\mathcal{F}(\mathcal{Q}_{Frag_j^{1(i)}}) = 2$, $\mathcal{F}(\mathcal{Q}_{Frag_{Pbest}^{j,1(i)}}) = 2$, $\mathcal{F}(\mathcal{Q}_{Frag_{Gbest}^{1(i)}}) = 6$. According to the definition of $Max\_\mathcal{Q}_{Frag_j}^{1(i)}$, it is the fragment with maximum fitness, namely $Max\_\mathcal{Q}_{Frag_j}^{1(i)} = \mathcal{Q}_{Frag_{Gbest}}^{1(i)} = \{T_4, T_6\}$. That is, adjust the task allocation (position fragment) of $W_1$ to $\{T_4, T_6\}$. Next, we adjust the position fragment of $W_2$. Since $\mathcal{V}_j^{2(i+1)} = 0$, we keep $W_2$ unchanged, $Max\_\mathcal{Q}_{Frag_j}^{2(i)} = \mathcal{Q}_{Frag_j}^{2(i)} = \{T_2\}$.

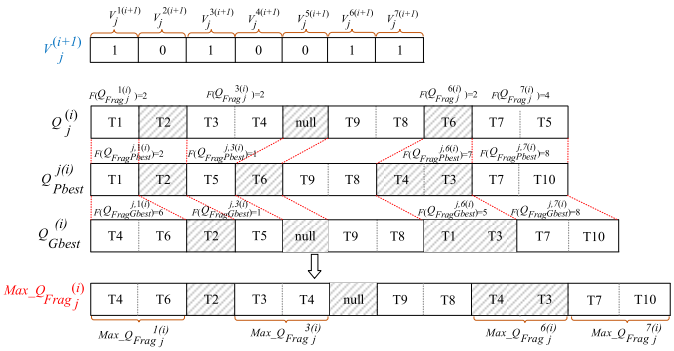



Fig. 4. Particle position update.

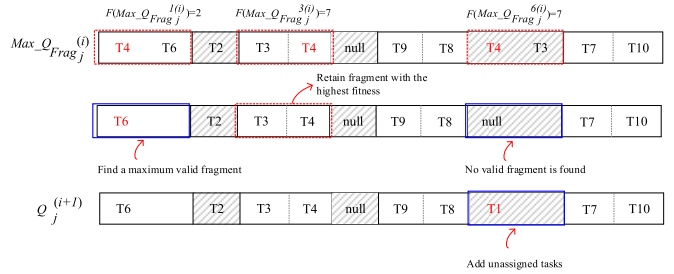



Fig. 5. Particle validity repair.

Repeat the above process until all workers' fragments are updated, $Max_{\mathcal{Q}_{Frag_j}}^{(i)} = \{\{T_4, T_6\}, \{T_2\}, \{T_3, T_4\}, \text{null}, \{T_9, T_8\}, \{T_4, T_3\}, \{T_7, T_{10}\}\}$. It is obvious that $Max\_\mathcal{Q}_{Frag_j}^{(i)}$ is not a legal particle.

3) As shown in Fig.5, we repair the validity of $Max\_\mathcal{Q}_{Frag_j}^{(i)}$. First check the position fragment of $W_1$, it can be seen that task $T_4$ is assigned to workers $W_1$, $W_3$, $W_6$. So, we compute the fragment fitness of each worker, $\mathcal{F}(Max\_\mathcal{Q}_{Frag_j}^{1(i)}) = 2$, $\mathcal{F}(Max\_\mathcal{Q}_{Frag_j}^{3(i)}) = 7$, $\mathcal{F}(Max\_\mathcal{Q}_{Frag_j}^{6(i)}) = 7$. Keep the position fragment of $W_3$ unchanged (Randomly select one from $W_3$ and $W_6$ with the highest fitness). And delete task $T_4$ from the fragments of $W_1$ and $W_6$, and search for new valid fragments. First repair the position fragment of $W_1$, select the valid task allocation fragment with maximum fitness to replace the original task allocation fragment from $\emptyset, \{T_4\}, \{T_6\}, \{T_4, T_6\}$, here we find $T_6$ meets the requirements, and repair the fragment of $W_1$ to $\{T_6\}$. Next, repair the fragment of $W_6$. It is found that after deleting task $T_4$, $W_6$ cannot undertake $T_3$ due to the constraints. Then the fragment of $W_6$ is updated to null. Repeat the above process to check each worker's position fragment until all tasks no longer conflict. After repairing the particle, update the working time of each worker, and then traverse the unassigned task. We find that task $T_1$ assigned to worker $W_6$ is still legal, so update and add the match of $T_1$ into the particle. Repeat the above until all unassigned tasks are traversed. Finally, the legal particle $Q_j^{(i+1)}$ is obtained.

As shown in Fig.4 and Fig. 5, the fitness of the particle $Q_j$ in the *i-th* iteration is $\mathcal{F}(\mathcal{Q}_j^{(i)}) = 29$, and its fitness in the *(i+1)-th* iteration is $\mathcal{F}(\mathcal{Q}_j^{(i+1)}) = 33$, which shows that our method has the ability to gradually guide particles to the optimal value.

### B. Second-Stage Redundant Task Allocation of TMA-DPSO

Many workers still have remaining time after the first-stage task assignment, especially in the scenario with more workers and few tasks. Unlike previous solutions, in this article we make full use of the remaining time of workers to perform the second-stage redundant task allocation. The so-called redundant allocation means that for each worker $W_k$, we search for other tasks near its current location and not yet assigned to it in the first stage based on the remaining time.

The second-stage redundant task allocation is still implemented by discrete particle swarm algorithm. The implementation process also uses Algorithm 1 (particle swarm initialization adopts Algorithm 4), which includes the following steps:

*Step 1:* Update the geographic location, remaining working time, and assigned task information of each worker according to the result in the first stage. Worker and task information is the same as in the first stage, $\mathbb{W} = \{W_1, W_2, \ldots, W_n\}$, $\mathbb{T} = \{T_1, T_2, \ldots T_m\}$. The updated information of a worker $W_k$ includes $\left(W_k, t_{av}^{W_k}, T_{W_k}\right)$, where $T_{W_k}$ is the set of tasks assigned to $W_k$ in the first stage.

*Step 2:* According to the workers and tasks information, combined with the remaining time of each worker, using the Algorithm 4 to randomly and greedily generate the initial particle swarm $\mathbb{SQ} = \{S\mathcal{Q}_1, S\mathcal{Q}_2, \ldots S\mathcal{Q}_\delta\}$. Since the remaining time of workers in the second stage is small, we set the number of particles $\delta$ to a small value.

*Step 3:* Compute the fitness of each particle $S\mathcal{Q}_j$. Different from the first stage, the redundant task allocation in the second stage is to achieve the maximum allocation of the number of tasks. Because the more tasks are assigned, a wider range of trust evaluation can be achieved compared with the results of the first stage through the trust mechanism of [23]. For a particle $S\mathcal{Q}_j$, if task $T_i$ appears in the position of it, it means task $T_i$ is assigned, $\chi_i^j = 1$, otherwise $\chi_i^j = 1$, then the fitness of the particle $S\mathcal{Q}_j$ is calculated as follows:

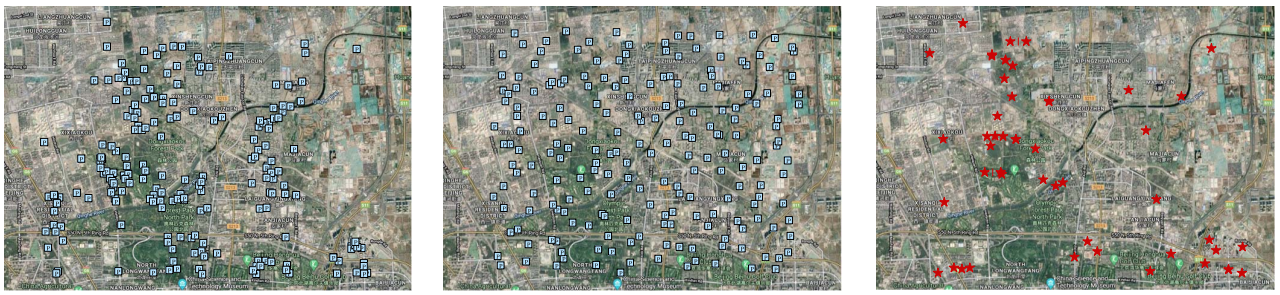
$$F\left(S\mathcal{Q}_j\right) = \sum_{i=1}^{m} \chi_i^j. \tag{8}$$

*Step 4:* According to the fitness of all particles, we define $S\mathcal{Q}_{Pbest}^j$ as the individual optimal particle of the particle $S\mathcal{Q}_j$ in all iterations, and $S\mathcal{Q}_{Gbest}$ as the global optimal particle.

*Step 5:* According to particle fitness, individual optimal particle and global optimal particle, update the velocity of each particle $S\mathcal{Q}_j$ in the next iteration according to Formula (5) and Formula (6). Adjust the position of each particle based on the updated velocity according to Formula (7), and repair the validity of the particles according to Algorithm 3, finally obtain the particle swarm for the next iteration.

Repeat the above process until the algorithm converges or reaches the maximum number of iterations.

Compared with the previous methods, the significance of the second-stage redundant allocation is to improve the fault tolerance and security. Considering the application scenario



(a) Distribution of tasks in scene 1 (b) Distribution of tasks in scene 2 (c) Distribution of workers

Fig. 6. The distribution of tasks and workers, where the mixed task distribution in (a) comes from the dataset "parking places of Beijing" [41], the uniform task distribution in (b) is randomly generated, and the workers distribution in (c) comes from the dataset "T-drive" [42].

with computation tasks, if sudden failures occur during task execution, then the result of the second-stage task allocation can be called to improve the fault tolerance and reliability of the system. In addition, in the scenario of crowdsourcing [20], [21] or data collection [2], [23], we can compare the authenticity and accuracy of the data by comparing the same task information provided by the workers in the first and second stages, so as to conduct trust evaluation in the workers, such as in the study in [23]. All this shows that our research not only affects task allocation and resource utilization, but also has potential impacts on other related fields in the future.

## V. Performance Analysis

### A. Experiment Setup

In this section, we evaluate effectiveness of TMA-DPSO based on real-life datasets. We regard vehicles as workers who are constantly moving in the system to undertake multiple tasks, and regard parking places as tasks. The distribution of tasks and workers is given in Fig. 6. And Fig. 6(a) shows a scenario where tasks are mixed and compactly distributed. The task data in Fig. 6(a) comes from a real-world dataset called parking places of Beijing [41]. To avoid accidental optimal results caused by scene characteristics, we synthesize a uniformly distributed task scene as shown in Fig. 6(b). The distribution of workers in Fig. 7(c) comes from the dataset T-drive [42], which contains the GPS trajectories of 10,357 taxis during the period of Feb. 2 to Feb. 8 within Beijing.

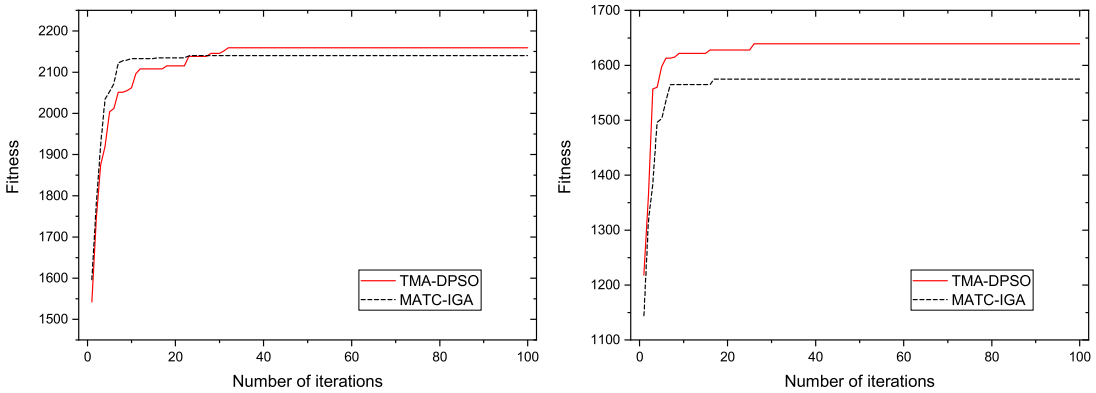
Other experimental parameters are set as follows: (a) we randomly select 50 workers and 200 tasks from the datasets, they are located within 25 kilometers centered on (116.41667, 39.91667); (b) the travel speed of workers is 20 km/h, and the working time is randomly generated in [2, 10] minutes; (c) the valid time of each task is between [5, 15] minutes, and the allocation status of all tasks is 0 in initialization; (d) the profit of tasks is [5, 35] currencies. In the first-stage task allocation, the task payment to workers is 2 currencies. And in the second stage, the payment paid by the platform to the worker is 0.5.

For comparison, we choose three typical methods. The first is Random Allocation method (the baseline method in [18]), referred to as RA, the second is NearestFirst Greedy allocation method (the baseline method in [19]), referred to as NFG, and the third is the MATC-IGA algorithm [19].

1) RA. The RA algorithm randomly assigns tasks to workers as long as the time constraints of both the task and the worker are met. Since the order of random selection affects the overall utility of the platform, we repeatedly and independently run the algorithm multiple times, and take the most beneficial result in the iterations as the final result.
2) NFG. In the NFG, each worker chooses the task closest to it each time. If it is a valid assignment that satisfies the time constraint, assign the task to the worker. Repeat this process until workers have no time to undertake tasks. NFG is also an unstable algorithm, so we also

**Algorithm 4** Second-Stage Particle Initialization

Input: $\mathbb{W}$ set of workers, $\mathbb{T}$ set of tasks
**For** each worker $W_k \in \mathbb{W}$ **do**
  Let $\mathcal{V}_k$ be a random value in 0 or 1
  $\mathcal{V} \leftarrow \mathcal{V} + \{\mathcal{V}_k\}$
  **If** remaining time $t_{av}^{W_k} > 0$ **do**
    Let $\mathbb{W}_{RT} \leftarrow \mathbb{W}_{RT} + \{W_k\}$
  **End if**
  **Else**
    Particle fragment $S\mathcal{Q}_{Frag} = null$
**End for**
Let $\mathbb{T}' \leftarrow \mathbb{T}$
**While** $\mathbb{W}_{RT} \neq \emptyset$ **do**
  Randomly select a worker $W_k$ from $\mathbb{W}_{RT}$
  Let $\mathbb{T}'' \leftarrow \mathbb{T}' - \mathbb{T}_{W_k}$
  **While** $\mathbb{T}'' \neq \emptyset$ **do**
    Randomly select a task $T_i$ from $\mathbb{T}''$
    $\mathbb{T}'' \leftarrow \mathbb{T}'' - \{T_i\}$
    **If** $S\mathcal{Q}_{Frag(W_k \leftarrow T_i)}$ is a valid particle fragment for $W_k$ **do**
      Assign $T_i$ to $W_k$ and append it to task sequence of $W_k$
      $\mathbb{T}' \leftarrow \mathbb{T}\prime - \{T_i\}$
      Update geographic location and working time of $W_k$
    **End if**
  **End while**
  $\mathbb{W}_{RT} \leftarrow \mathbb{W}_{RT} - \{W_k\}$
**End while**
Reorganize the task sequence of all workers to obtain a particle
Output: A new particle including position and velocity




(a) Convergence in scene 1 (b) Convergence in scene 2

Fig. 7. Convergence of TMA-DPSO and MATC-IGA in different task distribution scenarios.

choose the optimal result in iterations as the allocation result.

3) MATC-IGA. This method is an iterative optimization algorithm. First, a set of initial chromosomes are generated, and then through selection, vaccine production, infusion, crossover, mutation, and correction operations, iteratively update and obtain the optimal task allocation result.

### B. Theoretical Analysis

We first analyze the complexity, convergence, and stability of the four methods theoretically.

*1) Complexity:* Assuming there are $n$ workers and $m$ tasks, the complexity of four methods is shown in Table III. First, the TMA-DPSO includes particle initialization, fitness evaluation, velocity update, position update, and particle validity correction five steps. The particle initialization is implemented by Algorithm 2, and the complexity is $O(mn)$; the fitness evaluation is achieved by traversing the task allocation state, the complexity is $O(m)$; velocity and position update need to traverse the assigned task sequence of each worker, the particle validity correction is implemented by Algorithm 3. In the worst case, their complexity is $O(mn)$. Therefore, the complexity of TMA-DPSO is $O(mn)$. In RA, workers randomly select tasks with consideration of the time constraint. In the worst case, a worker has to traverse $m$ tasks to determine its task sequence, so its complexity is less than $O(mn)$. In NFG, it is required to calculate the distance matrix between workers and tasks, then each worker queries the distance matrix when selecting the nearest task, so the complexity is $O(mn)$. Finally, according to the analysis in [19], the complexity of MATC-IGA is $O(mn)$.

*2) Convergence:* Since RA and NFG are random algorithms and have no convergence characteristics, we only compare the convergence of TMA-DPSO and MATC-IGA. Fig. 7(a) is the convergence of scene 1 running on the real-life dataset, and Fig. 7(b) is the convergence of scene 2 running on evenly distributed tasks. Combining Table III and the figure, MATC-IGA converges faster than TMA-DPSO, but the optimal value reached when it converges is significantly lower than TMA-DPSO. This shows that TMA-DPSO can jump out of local optimum and seeks global optimum.

TABLE III
ANALYSIS OF COMPLEXITY, CONVERGENCE, AND STABILITY

| | TMA-DPSO | RA | NFG | MATC-IGA |
|---|---|---|---|---|
| Complexity | $O(mn)$ | $<O(mn)$ | $O(mn)$ | $O(mn)$ |
| Convergence | Start: 32nd iteration<br>Fitness: 2159 | - | - | 22th iteration<br>2140 |
| | Start: 27th iteration<br>Fitness: 1639 | - | - | 17st iteration<br>1575 |
| Stability | Range: 2103-2212<br>Average: 2147.2<br>Variance: 630.93 | 1623-1725<br>1679.8<br>1458.1 | 2157-2223<br>2193.2<br>322.76 | 2034-2128<br>2080.0<br>847.6 |
| | Range: 1578-1679<br>Average: 1621.6<br>Variance: 727.44 | 1134-1289<br>1237.1<br>1339.93 | 1451-1524<br>1492.6<br>503.64 | 1526-1650<br>1567.1<br>985.41 |

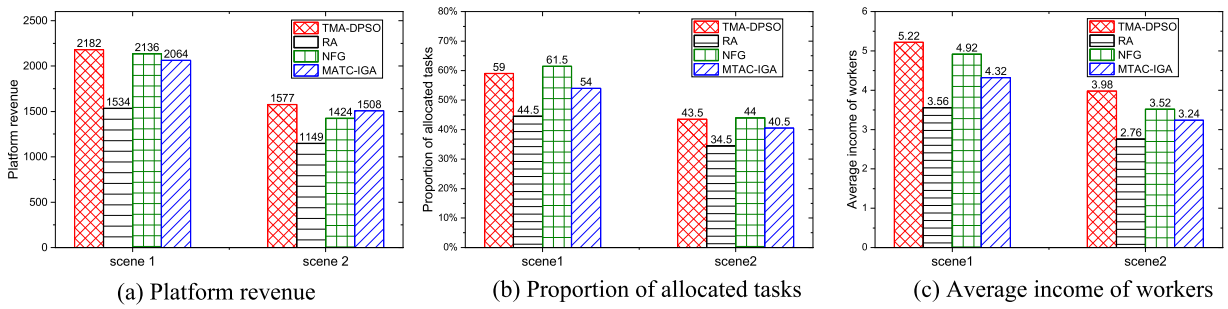
*3) Stability:* Given the same experimental environment, run the four algorithms independently for 10 times, the running results is summarized in Table III. As can be seen, no matter in scene 1 with mixed compact distribution or in scene 2 with uniform distribution, the TMA-DPSO proposed in this paper has stable searching efficiency, higher fitness value, and the variance between each running result is small. The stability of RA algorithm is the worst. The searching efficiency of the NFG algorithm is easily affected by the characteristics of the scene.

### C. Experimental Result

In the experiment, we compare the performance in platform revenue, task completion, workers payment, and running time. Here, we set the population number $\mu = 20$ in TMA-DPSO and MATC-IGA, and the maximum iterations $i_{Max} = 100$.

First, we analyze the performance of the four methods in a fixed number of workers and tasks, namely $n = 50, m = 200$.

Fig. 8(a) is comparison of platform revenue. In scene 1, the TMA-DPSO proposed in this paper has a slightly higher revenue than the NFG algorithm by 2.15%; compared to RA, TMA-DPSO increases platform revenue by 42.24%; compared to MATC-IGA, it increases platform revenue by 5.72%. In scene 2, TMA-DPSO has increased revenue by 37.25%,




(a) Platform revenue (b) Proportion of allocated tasks (c) Average income of workers

Fig. 8. When $n = 50, m = 200$, the performance comparison of TMA-DPSO, RA, NFG, and MATC-IGA in two scenes.

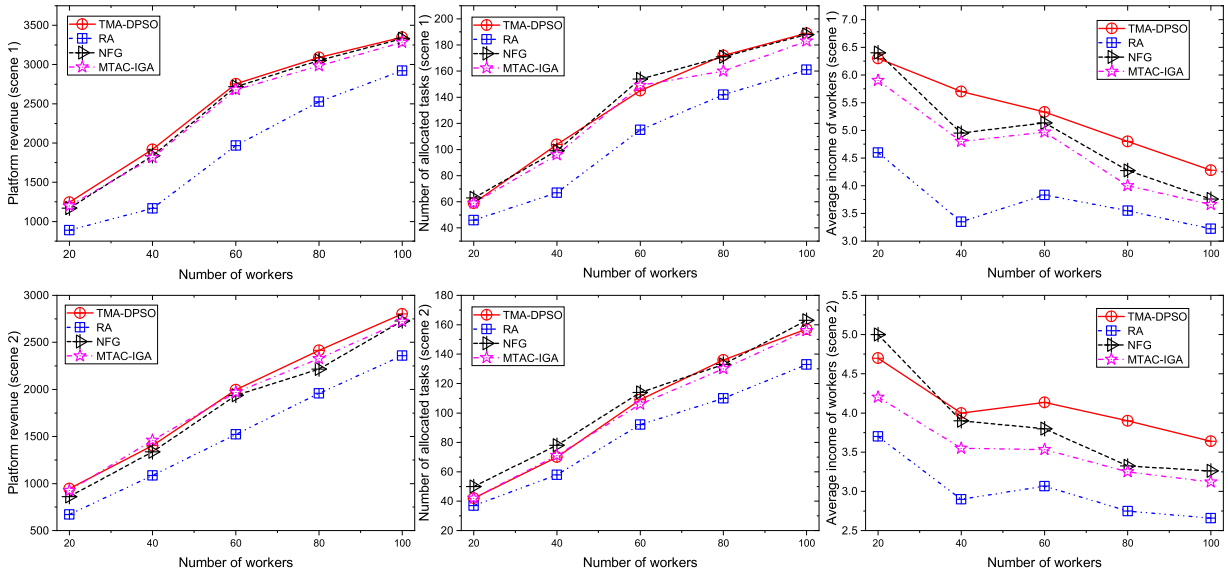



Fig. 9. When $m = 200$, the effect of different number of workers on task allocation of TMA-DPSO, RA, NFG, and MATC-IGA.

10.74%, and 4.58% compared to RA, NFG, and MATC-IGA. Fig. 8(b) is the proportion of task allocation. It can be seen that the proportion of assigned tasks under the NFG is the highest, but its platform revenue is not the highest. This is because NFG adopts the nearest task first allocation strategy, and the tasks are matched with workers without considering the overall platform revenue, while TMA-DPSO allocates tasks based on the platform utility. In scene 2 where tasks are evenly distributed, there is little difference in the proportion of tasks allocated between TMA-DPSO and NFG. Fig. 8(c) is the average income of workers. Generally speaking, the greater the income, the more motivate workers to actively participate. In this paper, workers use the remaining time to undertake the redundant tasks in the second stage. Although task payment in the second stage is much less than the first stage, the overall income of workers is increased. In scene 1, compared to RA, NFG, and MATC-IGA, the TMA-DPSO increases the average income of workers by 46.63%, 6.1%, and 20.84%. In scene 2, compared to RA, NFG, and MATC-IGA, it increases the average income of workers by 44.2%, 13.07%, and 22.84%.

Next, we analyze the performance of the four methods with the increase number of workers and tasks.

Assuming that the number of tasks is $m = 200$, and randomly select different number of workers from the dataset, where $n = \{20, 40, 60, 80, 100\}$. As shown in Fig. 9, on the whole, the platform revenue and the number of allocated tasks continue to grow with the increase in the number of workers, and the increasing number of workers will share the task profits, so the average income of workers decreases. Comparing the four methods, it can be seen that no matter which parameter setting, the platform revenue and workers income of TMA-DPSO are relatively high.

Keep the number of workers at $n = 50$, and constantly adjust the number of tasks, $m = \{50, 100, 150, 200, 250\}$. The tasks in scene 1 of Fig. 10 are randomly selected from the dataset parking places of Beijing, and these tasks are mixed and compactly distributed in the plane; while the tasks in scene 2 are evenly distributed, and the task information is randomly generated. As the number of tasks increases, the platform revenue and the number of allocated tasks increase, and the average income of workers also continues to increase.

Combining Fig. 9- Fig. 10, the following conclusions can be drawn: when the difference between the number of workers and tasks is small, such as $n = 50, m = 50$, at this time, since most of the time constraints can be satisfied, the platform revenue between the four methods is close; when the quantitative relationship between workers and tasks is quite different, such as $n = 50, m = 250$, the difference in platform

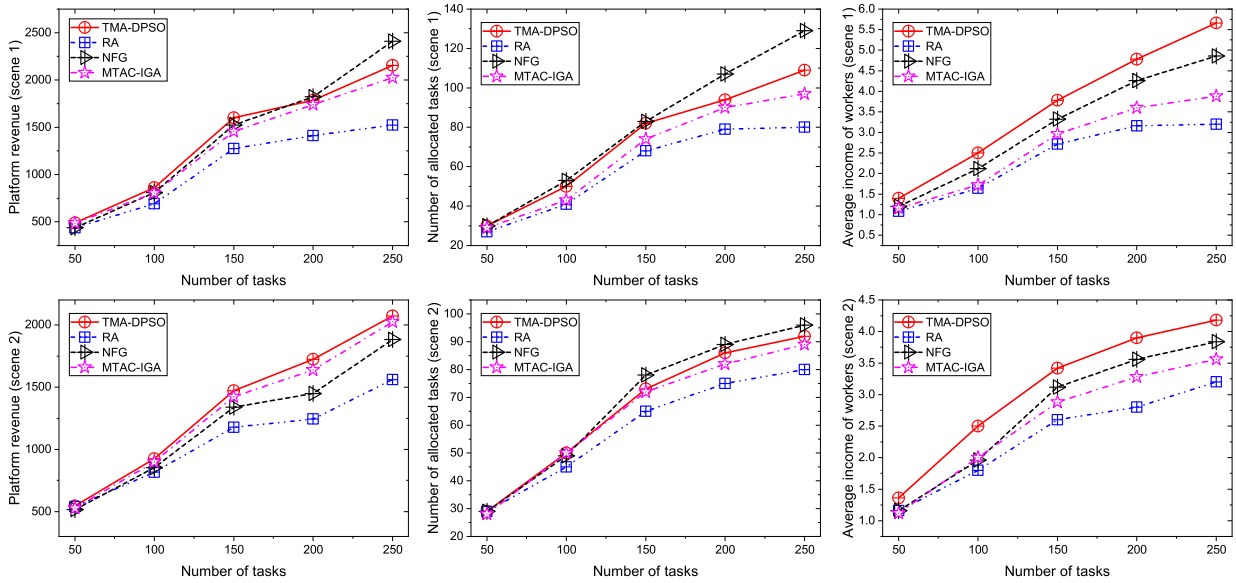



Fig. 10. When $n = 50$, the effect of different number of tasks on task allocation of TMA-DPSO, RA, NFG, and MATC-IGA.

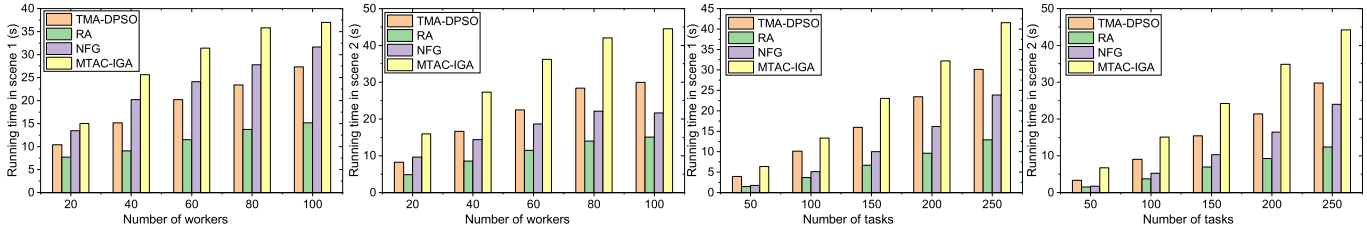



Fig. 11. The running time of TMA-DPSO, RA, NFG, and MATC-IGA required for 100 iterations with the increase of workers and tasks.

revenue between different methods and different scenarios is significant. Therefore, the purpose of task allocation is not only to meet the optimal allocation in special scenarios, but also to find the best task allocation algorithm, which can make the maximum utility and stability in differentiated scenarios, which is the advantage of TMA-DPSO.

Suppose the algorithm iterates 100 times in the same experimental environment, Fig. 11 is the running time under the four methods. Among them, RA has simple logic and low complexity. Therefore, its running time is very small. When the number of tasks is fixed at $m = 200$, the running time of TMA-DPSO is 18.64% less than that of NFG and 33.52% less than that of MATC-IG under scene 1. In scene 2, the running time of NFG is lower because the solution space of NFG for uniformly distributed tasks is small. TMA-DPSO reduces the running time by 38.03% compared with MATC-IGA. When the number of workers is fixed at $n = 50$, TMA-DPSO decreases by 29.48%-39.7% compared with MATC-IGA.

## VI. Conclusion and Future Work

To serve future heterogeneous services and applications supported by generation multiple access, numerous scattered workers with computing power need to be efficiently scheduled to complete tasks. Therefore, a two-stage multi-task allocation method TMA-DPSO by improving the discrete particle swarm optimization is proposed in this paper for next generation multiple access. First, we randomly and greedily generate multiple initial particles, and constantly update the velocity and position of these particles based on the individual optimal particles and the global optimal particle. And we introduce random variables during the adjustment process to avoid premature maturity. After that, we repair the validity of the particles, and iterate continuously to obtain the optimal solution. Different from the previous methods, after the first-stage task allocation, we make full use of the remaining time of the workers to carry out the second-stage redundant task assignment, thereby further improving the system performance. Finally, we conduct extensive experiments based on synthetic and real-life datasets. Experimental results demonstrate that TMA-DPSO has better search efficiency and stable optimization results regardless of densely distributed or evenly distributed scenarios. For the future work, we will expand the content of the second-stage task allocation and discuss in depth how to use the two-stage allocation results to assist in evaluating network performance, thereby enhancing network security, data quality, and fault tolerance.

## References

[1] I. Ahmad *et al.*, "The challenges of artificial intelligence in wireless networks for the Internet of Things: Exploring opportunities for growth," *IEEE Ind. Electron. Mag.*, vol. 15, no. 1, pp. 16–29, Mar. 2021.

[2] H. Tran-Dang, N. Krommenacker, P. Charpentier, and D.-S. Kim, "Toward the Internet of Things for physical internet: Perspectives and challenges," *IEEE Internet Things J.*, vol. 7, no. 6, pp. 4711–4736, Jun. 2020.

[3] M. B. Shahab, R. Abbas, M. Shirvanimoghaddam, and S. J. Johnson, "Grant-free non-orthogonal multiple access for IoT: A survey," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 3, pp. 1805–1838, 1st Quart., 2020.
[4] Y. Dai, D. Xu, S. Maharjan, Z. Chen, Q. He, and Y. Zhang, "Blockchain and deep reinforcement learning empowered intelligent 5G beyond," *IEEE Netw.*, vol. 33, no. 3, pp. 10–17, May 2019.
[5] M. Mohammadi *et al.*, "Full-duplex non-orthogonal multiple access for next generation wireless systems," *IEEE Commun. Mag.*, vol. 57, no. 5, pp. 110–116, May 2019.
[6] W. Shi *et al.*, "Two-level soft RAN slicing for customized services in 5G-and-beyond wireless communications," *IEEE Trans. Ind. Informat.*, early access, May 25, 2021, doi: 10.1109/TII.2021.3083579.
[7] Z. Ding, X. Lei, G. K. Karagiannidis, R. Schober, J. Yuan, and V. Bhargava, "A survey on non-orthogonal multiple access for 5G networks: Research challenges and future trends," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 10, pp. 2181–2195, Jul. 2017.
[8] Q. Wu *et al.*, "A comprehensive overview on 5G-and-beyond networks with UAVs: From communications to sensing and intelligence," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 10, pp. 2912–2945, Oct. 2021.
[9] M. Vaezi, R. Schober, Z. Ding, and H. V. Poor, "Non-orthogonal multiple access: Common myths and critical questions," *IEEE Wireless Commun.*, vol. 26, no. 5, pp. 174–180, Oct. 2019.
[10] Q. Wu, S. Zhang, B. Zheng, C. You, and R. Zhang, "Intelligent reflecting surface aided wireless communications: A tutorial," *IEEE Trans. Commun.*, vol. 69, no. 5, pp. 3313–3351, May 2021.
[11] X. Liu, Y. Liu, Y. Chen, and H. V. Poor, "RIS enhanced massive non-orthogonal multiple access networks: Deployment and passive beamforming design," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 4, pp. 1057–1071, Apr. 2021.
[12] W. K. New, C. Y. Leow, K. Navaie, Y. Sun, and Z. Ding, "Interference-aware NOMA for cellular-connected UAVs: Stochastic geometry analysis," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 10, pp. 3067–3080, Oct. 2021.
[13] Y. Cai, Z. Qin, F. Cui, G. Y. Li, and J. A. McCann, "Modulation and multiple access for 5G networks," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 1, pp. 629–646, 1st Quart., 2018.
[14] Y. Chen *et al.*, "Toward the standardization of non-orthogonal multiple access for next generation wireless networks," *IEEE Commun. Mag.*, vol. 56, no. 3, pp. 19–27, Mar. 2018.
[15] M. Afrin, J. Jin, A. Rahman, A. Rahman, J. Wan, and E. Hossain, "Resource allocation and service provisioning in multi-agent cloud robotics: A comprehensive survey," *IEEE Commun. Surveys Tuts.*, vol. 23, no. 2, pp. 842–870, 2nd Quart., 2021.
[16] E. A. Khalil, S. Ozdemir, and S. Tosun, "Evolutionary task allocation in Internet of Things-based application domains," *Future Gener. Comput. Syst.*, vol. 86, pp. 121–133, Sep. 2018.
[17] J. Wang, L. Wang, Y. Wang, D. Zhang, and L. Kong, "Task allocation in mobile crowd sensing: State-of-the-art and future opportunities," *IEEE Internet Things J.*, vol. 5, no. 5, pp. 3747–3757, Oct. 2018.
[18] J. Wang, Y. Wang, D. Zhang, F. Wang, Y. He, and L. Ma, "PSAllocator: Multi-task allocation for participatory sensing with sensing capability constraints," in *Proc. ACM Conf. Comput. Supported Cooperat. Work Social Comput.*, Feb. 2017, pp. 1139–1151.
[19] X. Li and X. Zhang, "Multi-task allocation under time constraints in mobile crowdsensing," *IEEE Trans. Mobile Comput.*, vol. 20, no. 4, pp. 1494–1510, Apr. 2021.
[20] M. H. Cheung, F. Hou, J. Huang, and R. Southwell, "Distributed time-sensitive task selection in mobile crowdsensing," *IEEE Trans. Mobile Comput.*, vol. 20, no. 6, pp. 2172–2185, Jun. 2021.
[21] Y. Liu, B. Guo, Y. Wang, W. Wu, Z. Yu, and D. Zhang, "TaskMe: Multi-task allocation in mobile crowd sensing," in *Proc. ACM Int. Joint Conf. Pervasive Ubiquitous Comput.*, Sep. 2016, pp. 403–414.
[22] Y. Zhang, Y. Tao, S. Zhang, L. Zhang, and H. Long, "Optimal sensing task distribution algorithm for mobile sensor networks with agent cooperation relationship," *IEEE Internet Things J.*, vol. 8, no. 10, pp. 8223–8233, May 2021.
[23] M. Huang, A. Liu, N. N. Xiong, and J. Wu, "A UAV-assisted ubiquitous trust communication system in 5G and beyond networks," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 11, pp. 3444–3458, Nov. 2021.
[24] P. A. Apostolopoulos, G. Fragkos, E. E. Tsiropoulou, and S. Papavassiliou, "Data offloading in UAV-assisted multi-access edge computing systems under resource uncertainty," *IEEE Trans. Mobile Comput.*, early access, Mar. 31, 2021, doi: 10.1109/TMC.2021.3069911.
[25] M. Salmani and T. N. Davidson, "Uplink resource allocation for multiple access computational offloading," *Signal Process.*, vol. 168, Mar. 2020, Art. no. 107322.
[26] B. Cao, L. Zhang, Y. Li, D. Feng, and W. Cao, "Intelligent offloading in multi-access edge computing: A state-of-the-art review and framework," *IEEE Commun. Mag.*, vol. 57, no. 3, pp. 56–62, Mar. 2019.
[27] S. Reddy, D. Estrin, and M. Srivastava, "Recruitment framework for participatory sensing data collections," in *Proc. Int. Conf. Pervasive Comput.*, 2010, pp. 138–155.
[28] H. Xiong, D. Zhang, G. Chen, L. Wang, V. Gauthier, and L. E. Barnes, "iCrowd: Near-optimal task allocation for piggyback crowdsensing," *IEEE Trans. Mobile Comput.*, vol. 15, no. 8, pp. 2010–2022, Aug. 2016.
[29] D. Smith, J. Wetherallm, S. Woodhead, and A. Adekunle, "A cluster-based approach to consensus based distributed task allocation," in *Proc. 22nd Euromicro Int. Conf. Parallel, Distrib., Netw.-Based Process.*, Feb. 2014, pp. 428–431.
[30] D. Zhang, H. Xiong, L. Wang, and G. Chen, "Crowdrecruiter: Selecting participants for piggyback crowdsensing under probabilistic coverage constraint," in *Proc. ACM Int. Joint Conf. Pervasive Ubiquitous Comput.*, Sep. 2014, pp. 703–714.
[31] G. Goel, A. Nikzad, and A. Singla, "Mechanism design for crowdsourcing markets with heterogeneous tasks," in *Proc. 2nd AAAI Conf. Hum. Comput. Crowdsourcing*, 2014, pp. 1–10.
[32] S. S. Ponda, L. B. Johnson, A. N. Kopeikin, H. L. Choi, and J. P. How, "Distributed planning strategies to ensure network connectivity for dynamic heterogeneous teams," *IEEE J. Sel. Areas Commun.*, vol. 30, no. 5, pp. 861–869, Jun. 2012.
[33] Y. Zhao, K. Zheng, Y. Li, H. Su, J. Liu, and X. Zhou, "Destination-aware task assignment in spatial crowdsourcing: A worker decomposition approach," *IEEE Trans. Knowl. Data Eng.*, vol. 32, no. 12, pp. 2336–2350, Dec. 2020.
[34] G. Cardone *et al.*, "Fostering participAction in smart cities: A geo-social crowdsensing platform," *IEEE Commun. Mag.*, vol. 51, no. 6, pp. 112–119, Jun. 2013.
[35] 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.
[36] M. Gong, Q. Cai, X. Chen, and L. Ma, "Complex network clustering by multiobjective discrete particle swarm optimization based on decomposition," *IEEE Trans. Evol. Comput.*, vol. 18, no. 1, pp. 82–97, Feb. 2014.
[37] X. Li, X. Wu, S. Xu, S. Qing, and P.-C. Chang, "A novel complex network community detection approach using discrete particle swarm optimization with particle diversity and mutation," *Appl. Soft Comput.*, vol. 81, Aug. 2019, Art. no. 105476.
[38] J. Sánchez-García, D. G. Reina, and S. L. Toral, "A distributed PSO-based exploration algorithm for a UAV network assisting a disaster scenario," *Future Gener. Comput. Syst.*, vol. 90, pp. 129–148, Jan. 2019.
[39] T. Djemai, P. Stolf, T. Monteil, and J.-M. Pierson, "A discrete particle swarm optimization approach for energy-efficient IoT services placement over fog infrastructures," in *Proc. 18th Int. Symp. Parallel Distrib. Comput. (ISPDC)*, Jun. 2019, pp. 32–40.
[40] S. Rahimi, A. Abdollahpouri, and P. Moradi, "A multi-objective particle swarm optimization algorithm for community detection in complex networks," *Swarm Evol. Comput.*, vol. 39, pp. 297–309, Apr. 2018.
[41] (2014). *Parking Places of Beijing*. [Online]. Available: https://www.beijingcitylab.com/data-released-1/
[42] J. Yuan *et al.*, "T-drive: Driving directions based on taxi trajectories," in *Proc. 18th SIGSPATIAL Int. Conf. Adv. Geograph. Inf. Syst. (GIS)*, Nov. 2010, pp. 99–108.

**Mingfeng Huang** received the B.E. degree from the School of Engineering and Design, Hunan Normal University, China, in 2017. She is currently pursuing the Ph.D. degree with the School of Computer Science and Engineering, Central South University, China. She has so far been SCI-cited more than 100 times and published journals include IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS, IEEE INTERNET OF THINGS JOURNAL, and IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS. Her research interests include trusted computing and wireless sensor networks.

**Victor C. M. Leung** (Life Fellow, IEEE) received the B.A.Sc. degree (Hons.) in electrical engineering from The University of British Columbia (UBC) in 1977 and the Ph.D. degree in electrical engineering from the Graduate School, UBC, in 1982, under the Natural Sciences and Engineering Research Council Postgraduate Scholarship.

Dr. Leung is a Registered Member of Engineers and Geoscientists BC, Canada. He is a fellow of the Academy of Science, Royal Society of Canada, a fellow of the Engineering Institute of Canada, a fellow of the Canadian Academy of Engineering, and a Voting Member of ACM. He was a recipient of the 2011 UBC Killam Research Prize, the IEEE Vancouver Section Centennial Award, the 2017 Canadian Award for Telecommunications Research, and the 2018 IEEE ComSoc TGCC Distinguished Technical Achievement Recognition Award. He has coauthored papers won the 2017 IEEE ComSoc Fred W. Ellersick Prize, the 2017 IEEE SYSTEMS JOURNAL Best Paper Award, and the 2018 IEEE ComSoc CSIM Best Journal Paper Award. He was awarded the APEBC Gold Medal as the Head of graduating class at the Faculty of Applied Science for his B.A.Sc. degree in 1977. He is listed among the Clarivate Analytics Highly Cited Researchers. He is serving on the editorial boards for the IEEE TRANSACTIONS ON GREEN COMMUNICATIONS AND NETWORKING, IEEE TRANSACTIONS ON CLOUD COMPUTING, and *IEEE Network*. He has previously served on the editorial boards for the IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS (Wireless Communications Series and Series on Green Communications and Networking), IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS, IEEE TRANSACTIONS ON COMPUTERS, IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, IEEE WIRELESS COMMUNICATIONS LETTERS, and *Journal of Communications and Networks*.

**Anfeng Liu** received the M.Sc. and Ph.D. degrees in computer science from Central South University, China, 2002 and 2005, respectively. He was a Visiting Scholar with BBCR Group, University of Waterloo, from 2011 to 2012. He is currently a Professor at the School of Computer Science and Engineering, Central South University. His research interest is wireless sensor networks. He is a member (E200012141 M) of the China Computer Federation (CCF).

**Neal N. Xiong** (Senior Member, IEEE) received the Ph.D. degrees from Wuhan University in 2007 and the Japan Advanced Institute of Science and Technology in 2008.

Before he attended Northeastern State University, he worked at Georgia State University, Wentworth Technology Institution, and Colorado Technical University (a Full Professor about five years) about ten years. He is currently an Associate Professor at the Department of Computer Science and Mathematics, Sul Ross State University, Alpine, TX, USA. He has published over 200 international journal articles and over 100 conference papers. His research interests include cloud computing, security and dependability, parallel and distributed computing, networks, and optimization theory.

Dr. Xiong has been a Senior Member of IEEE Computer Society since 2012. He has received the Best Paper Award in the Tenth IEEE International Conference on High Performance Computing and Communications (HPCC-08) and the Best Student Paper Award in the 28th North American Fuzzy Information Processing Society Annual Conference (NAFIPS2009). He is the Chair of "Trusted Cloud Computing" Task Force, IEEE Computational Intelligence Society (CIS). He is serving as the Editor-in-Chief, an Associate Editor, or an Editor Member for over ten international journals, including an Associate Editor for IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS: SYSTEMS, IEEE TRANSACTIONS ON NETWORK SCIENCE AND ENGINEERING, and *Information Sciences*, and the Editor-in-Chief for *Journal of Internet Technology* (JIT) and *Parallel and Cloud Computing* (PCC).