Competition-Congestion-Aware Stable Worker-Task Matching in Mobile Crowd Sensing

Guisong Yang, Buye Wang, Xingyu He, Jiangtao Wang, and Haris Pervaiz

DOI 10.1109/TNSM.2021.3072638
ISSN 1932-4537

Publisher: IEEE

Competition-Congestion-Aware Stable Worker-Task Matching in Mobile Crowd Sensing

Guisong Yang, *Member, IEEE*, Buye Wang, Xingyu He, Jiangtao Wang, *Member, IEEE*, Haris Pervaiz, *Member, IEEE*

***Abstract*—Mobile Crowd Sensing is an emerging sensing paradigm that employs massive number of workers' mobile devices to realize data collection. Unlike most task allocation mechanisms that aim at optimizing the global system performance, stable matching considers workers are selfish and rational individuals, which has become a hotspot in MCS. However, existing stable matching mechanisms lack deep consideration regarding the effects of workers' competition phenomena and complex behaviors. To address the above issues, this paper investigates the competition-congestion-aware stable matching problem as a multi-objective optimization task allocation problem considering the competition of workers for tasks. First, a worker decision game based on congestion game theory is designed to assist workers in making decisions, which avoids fierce competition and improves worker satisfaction. On this basis, a stable matching algorithm based on extended deferred acceptance algorithm is designed to make workers and tasks mapping stable, and to construct a shortest task execution route for each worker. Simulation results show that the designed model and algorithm are effective in terms of worker satisfaction and platform benefit.**

***Index Terms*—Mobile crowd sensing, stable matching, task allocation, congestion game, deferred acceptance.**

I. Introduction

THE Internet of Things (IoT) utilizes sensor-based embedded systems to interact with others, providing a wide range of applications and services to upper-level users [1]. Mobile Crowd Sensing (MCS) is an emerging pervasive sensing paradigm in IoT uses massive number of workers' smart mobile terminals (smart phones, tablet computers, sensors, etc.), due to their availability at large scale, wide distribution, and high sensing capabilities to collect sensing data with high correlation and strong analyzability [2]. Therefore, MCS is considered as an economic, effective, and reliable sensing scheme [3] providing great support in pervasive computing domain such as intelligent transportation [4], environmental monitoring [5], healthcare application [6], urban public management [7], and so forth.

In MCS, task allocation is a fundamental and important research issue, which has become a hot topic in the fields of social computing, cooperative computing, and intelligent computing in recent years [8]. Existing works on task allocation mechanism mostly aim at optimizing the global system performance, while ignoring the mutual preferences between workers and crowdsensing tasks. Since workers and platform are owned by different entities with the possibility of having diverse needs resulting in conflict among the maximization of worker satisfaction or platform benefit. For example, workers prefer tasks according to their own satisfaction, while the platform wants to allocate tasks to workers with better service qualities and lower sensing costs to maximize platform benefit. In addition, the platform generally does not change the formulated allocation strategy in practical applications. However, since workers are selfish and rational individuals, they are not necessarily aligned with platform optimization [9]. Workers may violate the current task allocation scheme when they are not satisfied with the allocated tasks, which results in system instability. Here, the system instability means that the established allocation strategy cannot be completed due to the dissatisfaction of workers.

In order to solve the conflict of interest between platform and workers and avoid system instability, some previous works [10]–[14] from the recent literatures utilize stable matching theory to transform task allocation problem into two-sided matching problem. However, there are still some deficiencies in these studies. They assume that workers are only concerned about their personal benefit and select tasks based on their preferences in contrast to the case where workers need to compete for the tasks in the resource pool. As the workers are not fully aware of the status of other workers' applications for tasks, they often apply for tasks based on their own preferences and historical experience, resulting in fierce competition for some tasks among the workers. Since the number of workers required for a specific task is constrained, massive workers picking the same task will result in partly workers matching the task while remaining workers not. Although stable matching

This work was supported in part by the National Natural Science Foundation of China under Grant 61802257 and 61602305; and by the Natural Science Foundation of Shanghai under Grant 18ZR1426000 and 19ZR1477600. *(Corresponding author: Xingyu He.)*

G. Yang is with the Department of Computer Science and Engineering at University of Shanghai for Science and Technology, Shanghai 200093, China, and also with Shanghai Key Lab of Modern Optical system, Shanghai 200093, China (e-mail: gsyang@usst.edu.cn).

B. Wang is with the Department of Computer Science and Engineering at University of Shanghai for Science and Technology, Shanghai 200093, China (e-mail: 182590511@st.usst.edu.cn).

X. He is with College of Communication and Art Design at University of Shanghai for Science and Technology, Shanghai 200093, China (e-mail: xy_he@usst.edu.cn).

J. Wang is with the Center for Intelligent Healthcare, Coventry University, Coventry CV1 5RW, United Kingdom (email: jiangtao.wang@coventry.ac.uk).

H. Pervaiz is with School of Computing and Communication, Lancaster University, Lancaster, United Kingdom (email: h.b.pervaiz@lancaster.ac.uk).

can map the suboptimal task options for the remaining workers which may reduce their interest for participation in sensing activities.

In order to address the aforementioned issues, this paper investigates the competition-congestion-aware stable matching problem as a multi-objective optimization task allocation problem considering the competition of workers for tasks obtaining a unique and optimal solution. To accomplish this problem, there are at least two challenges. Firstly, the competition and complex behaviors of workers pose a challenge in designing an appropriate model to assist workers in making decisions. Secondly, different from the traditional unilateral optimization problem, the conflict between the maximization of platform benefit and the worker's desire to receive satisfactory tasks results in another critical challenge.

In this work, in order to study the competition on workers' decision-making, competition congestion metric is proposed to measure the competition of workers for tasks. On this basis, a worker decision model based on congestion game theory is designed by jointly considering worker benefit, worker preference, and competition congestion metric. The model assists workers in making decisions by publishing their application status for tasks, which avoids fierce competition and improves worker satisfaction. Here, avoiding fierce competition does not mean ignoring the fact that workers compete for tasks in practical applications. When the number of workers applying for a task exceeds the number of workers required for the task, a stable matching algorithm based on extended deferred acceptance algorithm is designed to allocate a set of tasks to each worker, and to construct shortest task execution routes for them. The algorithm can not only satisfy the workers as much as possible, but also improve platform benefit by allocating tasks to the most beneficial workers when the workers compete for task resources.

The main contributions are summarized as follows:

(1) We investigate the drawback of existing solutions for stable matching in MCS, and study a novel problem by introducing competition congestion metric for each task. Keeping the metric in mind, this paper aims to avoid the fierce competition among workers for tasks, and to allocate an appropriate set of tasks to each worker through many-to-many stable matching to maximize worker satisfaction, as well as maximize platform benefit. To the best of our knowledge, we are the first to consider the influence of competition on worker behavior for stable matching in MCS.

(2) We design a worker decision game based on congestion game theory to generate utility list for each worker, which jointly considers the influence of worker benefit, worker preference, and competition congestion metric on worker behavior. Then, we design a stable matching algorithm based on extended deferred acceptance algorithm. The algorithm incorporates both workers' utility lists and tasks' benefit lists to make workers and tasks mapping stable according to three designed rules, and then constructs a shortest task execution route for each worker.

(3) We evaluate the performance of the proposed mechanism by comparing it with traditional benchmark stable matching algorithm and five baseline task allocation algorithms from the recent state-of-the-art. The results verify that, under various settings, the proposed mechanism is effective in terms of worker satisfaction and platform benefit.

The rest of the paper is organized as follows. In Section II, the related works are introduced. In Section III, the system models are introduced. Accordingly, the competition-congestion-aware stable matching algorithm (CCASM) is designed to realize the system models in Section IV. The performance evaluation and discussion are given in Section V. Finally, Section VI discusses the future work directions and Section VII concludes this paper.

II. Related Works

In this section, the relevant works from the recent state of the art on unstable task allocation, stable matching, and game theory in MCS are presented. Due to the few researches on using game theory to solve task allocation, the main presentation of relevant works on game theory is the research of improving MCS incentive mechanism utilizing game theory.

A. Unstable Task Allocation in Mobile Crowd Sensing

The earlier research works on MCS focused on single task allocation wherein a task is allocated to only one worker for execution. Zhang et al. proposed a novel worker selection framework named CrowdRecruiter [15], which can find a near-minimal set of workers for a task that meets its coverage ratio requirement. On similar trend, Xiong et al. defined a novel spatial-temporal coverage metric named k-depth coverage [16] by selecting a group of workers for the task according to different optimization goals to participate in the sensing activities. In [17], Liu et al. proposed an energy-efficient participant selection scheme that could meet the high quality-of-information requirements for the services incorporating the energy efficiency requirements of the workers.

With the increasing number of tasks in MCS given the limited sensing resources (e.g., the number of workers, the power of mobile devices, the type of sensors mounted on mobile devices), the multi-task allocation has become a hot research topic, wherein multiple tasks are allocated to multiple workers for execution. In [18], Wang et al. proposed a novel multi-task allocation framework named Mtasker to allocate appropriate tasks to each worker to maximize the overall system utility by introducing task-specific minimal sensing quality thresholds. In [19], Hu et al. studied the Quality of Service (QoS) based sensitive task allocation problem for MCS involving variable tasks and flexible rewards with the aid of a greedy algorithm to solve the aforementioned problem. Due to the diverse time-sensitivity and delay tolerant requirements of the tasks, a worker selection framework named ActiveCrowd was proposed to select suitable workers to minimize the cost [20]. Another work considered two situations depending on the number of tasks and workers, such as the case of more tasks with less workers and less tasks with more workers [21].

TABLE I
SYMBOLS USED

Symbols	Descriptions
T_i, t_i, T_N	A task T_i, number of samples required for task T_i, and task set containing all tasks.
W_j, w_j, W_K	A worker W_j, maximum workload of worker W_j, and worker set containing all workers.
$c_{i,j}$, $x_{i,j}$	The status of whether worker W_j applies for task T_i, the status of whether task T_i is allocated to worker W_j.
$H_{i,j}$, $B_{i,j}$	The utility/benefit that is achieved by allocating task T_i to worker W_j to be accomplished.
$P_{i,j}$, $C_{i,j}$, $F_{i,j}$	The reward/cost/benefit for worker W_j to accomplish task T_i.
s_j, S, S_{-j}	Worker W_j's strategy profile, set of strategy profiles for all workers, and set of strategy profiles for all workers except worker W_j.
$I^i(S)$	Competition congestion metric of task T_i under the strategy profile S.
$\delta_{i,j}$	Worker W_j's preference for task T_i.
$U_{i,j}$	The utility for worker W_j to accomplish task T_i.
$E_{i,j}$	Worker W_j's goal for the utility of completing task T_i.
$G_{i,j}(S)$, $G_j(S)$, $\Phi(S)$	Worker W_j's score for completing task T_i under S, worker W_j's score under S, and the score of all workers under S.
L^i, L^j	Task T_i's benefit list, worker W_j's utility list.
V_j	Worker W_j's satisfaction.
H_j, B_j, P_j	The utility/benefit that worker W_j brings to the platform, the total reward to worker W_j.
$BRS_j(S)$	Better response set of worker W_j under the strategy profile S.
$T_{i\text{-}pair}$, $T_{i\text{-}re}$	Number of pairs that task T_i has matched, remaining number of pairs that task T_i can be matched.
$W_{j\text{-}pair}$, $W_{j\text{-}re}$	Number of pairs that worker W_j has matched, remaining number of pairs that worker W_j can be matched.

Most of the above-mentioned research works focused on optimizing the global system performance assuming that workers are selfless, i.e., completing each task allocated by the platform unconditionally. In reality, workers are selfish and rational individuals, which is not necessarily aligned with platform optimization causing unstable consequences. In order to tackle this issue, this paper formulates the multi-task allocation problem into a stable matching problem.

B. *Stable Matching in Mobile Crowd Sensing*

Stable matching algorithms have been used in many two-sided matching problems in various fields. For example, the authors in [22] proposed a modified swap matching algorithm based on stable matching theory to allocate the spectrum resource rationally when both macro-cellular and femto-cellular users coexist in the same network. Baïou and Balinski proposed a stable allocation (or ordinal transportation) problem [23], which is a many-to-many generalization of the classical stable matching problem.

In contrast to the existing task allocation mechanisms that may cause instability resulting in the application of stable matching in MCS. Initially the stable matching was mainly used to study the one-to-one matching problems. For example, Zhou et al. [10] solved the route planning and task allocation problem of UAV-aided MCS by studying the influence of benefit on matching, with the goal of minimizing energy consumption. Chen [11] analyzed the existing task allocation types, and proposed a stable task allocation algorithm to match workers and tasks by considering worker preference. Later on, some researches have investigated the stable matching for the many-to-one and many-to-many matching problems which is more suitable for MCS scenarios. Zhang et al. [9] studied a matching problem between workers and task requesters by considering workers' preferences and different types of tasks. The authors in [12] and [13] studied the stable matching between workers and task requesters under the constraint of budget in many-to-one and many-to-many scenarios, respectively. Abououf et al. [14] studied a framework based on many-to-many stable matching, which can maximize the level of worker satisfaction, QoS and the completion confidence of the tasks.

In this paper, we investigate a competition-congestion-aware stable matching problem and extend the deferred acceptance algorithm to many-to-many by considering the multi-task allocation scenario. Different from existing studies on workers matching task requesters [9], [12], [13] and one-to-one worker-task matching [11], a pervasive model of multiple workers matching multiple tasks is formulated. In contrast to the study using stable matching to achieve multi-task allocation [14], our designed model aims to avoid the fierce competition among workers for tasks and solve the conflict of interest between platform and workers. Our proposed work investigates the influence of worker benefit, worker preference, and competition congestion metric on worker's behavior in matching process, which is more align with the real situation in MCS. In addition, the concept of stable matching in many-to-many scenarios is redefined in our proposed work, and the corresponding analysis of stability, optimization and complexity is presented.

C. *Game Theory*

Game theory is a theoretical framework that studies strategic interactions, which is applied to formulate and tackle the phenomena of struggle and competition. Till now, some MCS studies have utilized game theory to develop optimization strategies for incentive mechanism. For example, the most common approaches mentioned in [24]–[26] to establish models based on game theory are used to motivate the participations of mobile sensing workers. In addition, reverse auction approach [27] in game theory is also a common model in the study of incentive mechanism.

Due to the difference between the behavior pattern of nodes in game theory and that of workers in MCS, the existing game theory concepts cannot be directly applied to task allocation. Therefore, we study the factors that influence the decision-making of mobile workers, and combine them with game theory.

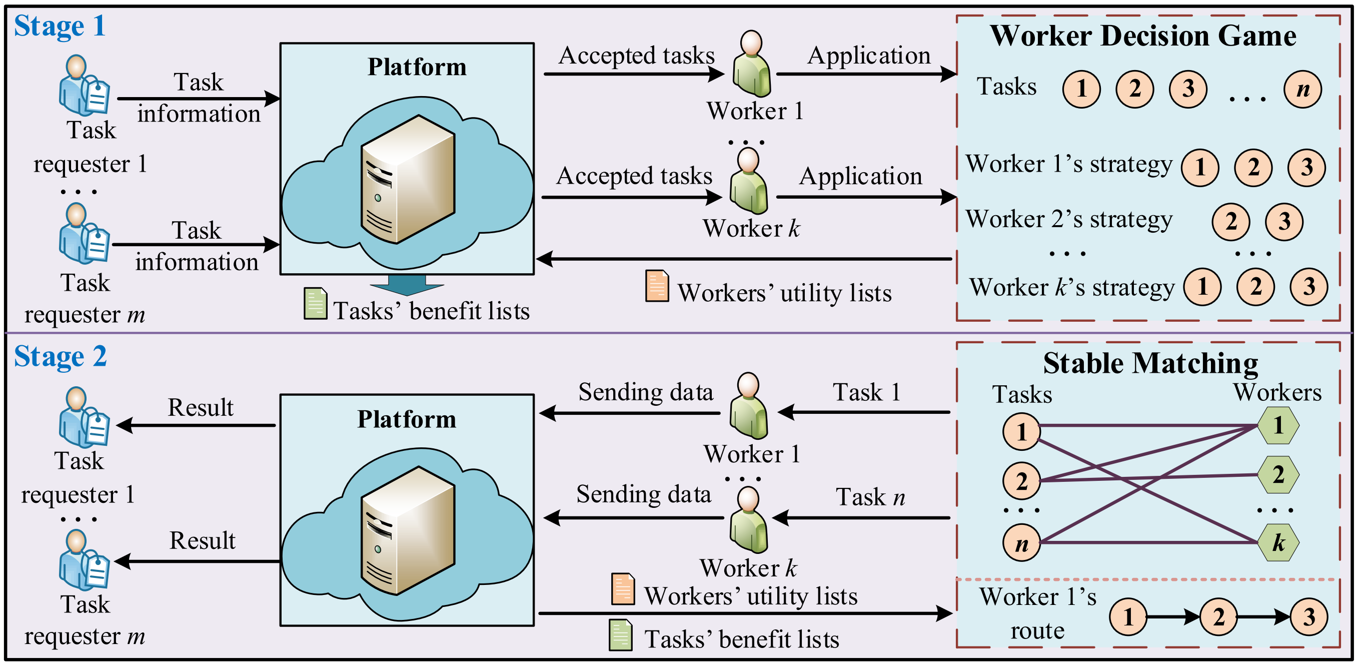


Fig. 1. The system model.

III. System Models

In this work, a multi-task allocation scenario is considered where the platform wants to allocate multiple tasks to multiple workers. In this scenario, each task can be allocated to multiple workers while each worker can receive multiple tasks. The system model, as shown in Fig. 1, is divided into two stages. To ease the presentation, the main symbols used in this paper are summarized in Table I.

Stage 1: Task requesters delegate various sensing tasks to the platform and provide detailed task information (e.g. task location, sensing content, number of samples required for the task, etc.). Then, the platform publishes accepted tasks to workers at an appropriate time (newly arrived tasks will be postponed to the next task allocation), and provides workers a certain time session to submit and modify their applications for tasks that they are satisfied with through worker decision game. After that, the platform generates a utility list for each worker by ranking the relevant tasks from high to low according to the utilities to the worker. In addition, the platform generates a benefit list for each task by ranking workers in descending order according to the benefit to the task.

Stage 2: By incorporating both workers' utility lists and tasks' benefit lists, the platform realizes a many-to-many stable matching for worker-task pairs, based on which, a shortest task execution route is designed for each worker. When the tasks are sensed completely, the worker uploads the sensing data to the platform, and the platform sends the sensing results to task requesters.

In this scenario, we make the following assumptions. Firstly, similar to existing recent research works (e.g. [18][28]), to avoid workers accepting more tasks than they can handle, each worker is allowed to set a maximum workload, that is, the maximum number of tasks that the worker can accept. Secondly, the physical parameters of each mobile device can be obtained when the worker registers onto the platform. Finally, we assume that the worker's location is available during task allocation through GPS, Wi-Fi, and cellular networks [29].

With the aforementioned system model and assumptions, first of all, it is necessary to generate the worker's utility list and the task's benefit list for each worker and task. Then, based on the generated lists, workers and tasks can be stably matched under the constraints of the maximum workload of workers and the number of samples required for tasks.

A. List Generation

Denote the set of k workers as $W_K = \{W_1, \ldots, W_j, \ldots, W_k\}$, and the set of n tasks as $T_N = \{T_1, \ldots, T_i, \ldots, T_n\}$. In MCS, the number of tasks that workers can execute over a period of time is constrained, and the maximum workload of worker W_j is denoted as w_j. In addition, the number of samples required for task T_i is denoted as t_i. We assume that the worker can only provide one sample for a task in task allocation to ensure the quality of the sample.

When the platform publishes task information, workers can apply for the tasks they want to perform. The status of whether worker W_j applies for task T_i is denoted as $c_{i,j}$, which is a binary value, where $c_{i,j} = 1$ represents worker W_j applying for task T_i, and otherwise $c_{i,j} = 0$. Workers can add tasks they apply for to their strategy profiles, and each worker's strategy profile is denoted as $s_j = \{T_i \in T_N \mid c_{i,j} = 1\}$. The set of strategy profiles for all k workers can be denoted as $S = \{s_1, \ldots, s_j, \ldots, s_k\}$. After that, the platform will try to allocate tasks to the worker according to worker's strategy profile, and the allocation state between task T_i and worker W_j is denoted as $x_{i,j}$, where $x_{i,j} = 1$ represents that T_i is allocated to W_j, and otherwise $x_{i,j} = 0$.

1) Generate the task's benefit list

For a task, the main factors affecting its sensing utility include the sensing quality and the sensing time. Hence, the sensing utility that is achieved by allocating task T_i to worker W_j to be accomplished is calculated as

$$H_{i,j} = \alpha \cdot Q_j \cdot ST_{i,j}, \tag{1}$$

where α is the price coefficient, which is used to transform time units into monetary units, e.g., dollars. Q_j represents the sensing quality of worker W_j, with a value between 0 and 1, which is determined by the platform according to the physical parameters of the worker's mobile device. $ST_{i,j}$ represents the sensing time that worker W_j takes to execute task T_i.

After a task is completed by a worker, the platform needs to pay the worker a certain reward, which is positively correlated to sensing quality, sensing time, and movement time. Hence, the reward for worker W_j to accomplish task T_i is calculated as

$$P_{i,j} = \beta \cdot Q_j \cdot (MT_{i,j} + ST_{i,j}), \tag{2}$$

where β is the price coefficient, which plays a similar role to α. $MT_{i,j}$ represents the movement time, the time worker W_j takes to move to the destination of task T_i, which is calculated as

$$MT_{i,j} = d_{i,j} / v, \tag{3}$$

where $d_{i,j}$ represents the Euclidean distance between task T_i and worker W_j. v represents the average movement speed of workers, which is a fixed value for all workers and is introduced to transform distance into time.

Therefore, for task T_i, the benefit that is achieved by allocating task T_i to worker W_j to be accomplished is calculated as

$$B_{i,j} = H_{i,j} - P_{i,j}. \tag{4}$$

Considering that the platform wants to allocate tasks to workers with better service qualities and lower sensing costs to maximize task benefit, for each task $T_i \in T_N$, its benefit list L^i containing all workers is generated, in which workers are sorted from high to low according to the benefit $B_{i,j}$ they bring to task T_i.

2) Generate the worker's utility list based on worker decision game

Different from tasks, worker behavior is influenced by many factors, so a worker decision game based on congestion game is designed to assist workers in decision-making.

Firstly, the worker usually attaches importance to its benefit, which can be calculated by reward minus cost. The cost for a worker to accomplish a task mainly comes from the movement cost and sensing cost, hence the cost for worker W_j to accomplish task T_i is calculated as

$$C_{i,j} = \gamma \cdot (MT_{i,j} + ST_{i,j}), \tag{5}$$

where γ is the price coefficient, which plays a similar role to α.

Therefore, by calculating the reward $P_{i,j}$ minus the cost $C_{i,j}$, the benefit for worker W_j to accomplish task T_i is calculated as

$$F_{i,j} = P_{i,j} - C_{i,j}. \tag{6}$$

Secondly, workers not only care about their benefit, but also have preferences for different tasks. The preference of worker W_j for task T_i is denoted as $\delta_{i,j}$, with a value between 0 and 1. Specifically, 1 indicates that the worker likes the task most, and 0 indicates that the worker least likes the task.

Thirdly, workers' competition for task resources will also affect workers' decisions. The fiercer the competition among workers for a task, the more difficult it is for workers to match the task successfully. To measure the competition of workers for a task, a competition congestion metric is proposed. And the competition congestion metric of task T_i under the strategy profile S is calculated as

$$I^i(S) = \frac{sig(\sum_{j=1}^{j=k} c_{i,j} - t_i)}{\sum_{j=1}^{j=k} c_{i,j}}, \tag{7}$$

where $c_{i,j}$ represents the status of whether worker W_j applies for task T_i and t_i represents the number of samples required for task T_i.

$$sig(x) = \begin{cases} x, x > 0 \\ 0, x \le 0 \end{cases}. \tag{8}$$

Based on the above three indicators of worker benefit, worker preference and competition congestion metric, a set of non-negative and non-increasing functions is used to represent the worker utility. The worker utility is proportional to worker benefit and worker preference, and is inversely proportional to competition congestion metric, which is calculated as

$$U_{i,j} = \left(\frac{F_{i,j} - min((F_{i,j})_{i\in(1,n)})}{max((F_{i,j})_{i\in(1,n)}) - min((F_{i,j})_{i\in(1,n)})} \cdot \delta_{i,j} \cdot (1 - I^i(S))\right)^{\frac{1}{3}}. \tag{9}$$

where $min((F_{i,j})_{i\in(1,n)})$ and $max((F_{i,j})_{i\in(1,n)})$ represent the minimum and maximum benefit for worker W_j to accomplish tasks in task set T_N, respectively.

Although the worker utility is a good measure of the value a task brings to the worker, different workers may have different expectations of utility which are entirely determined by themselves. Therefore, we simplify the model by defining worker goal and worker score. In detail, worker W_j's goal for the utility of completing task T_i is expressed as $E_{i,j}$, and worker W_j's score for completing task T_i is expressed as $G_{i,j}$. In the worker decision game model, the state of worker score can be divided into the following categories:

- A worker who applies for task T_i and the utility $U_{i,j}$ is not smaller than the worker's goal $E_{i,j}$ is satisfied. In this case, the worker will not change its strategy profile, and the score $G_{i,j} = 1$.
- A worker who applies for task T_i and the utility $U_{i,j}$ is smaller than the worker goal $E_{i,j}$ is unsatisfied, and the score $G_{i,j} = -1$. In this case, the worker will change its strategy profile to not apply for task T_i.
- A worker who does not apply for task T_i is neutral, and the score $G_{i,j} = 0$.

Based on the above analysis, the worker W_j's score for completing task T_i under the strategy profile S can be formally represented as:

$$G_{i,j}(S) = \begin{cases} 1, \text{if } T_i \in s_j \text{ and } U_{i,j} \geqslant E_{i,j}, \\ 0, \text{if } T_i \notin s_j, \\ -1, \text{if } T_i \in s_j \text{ and } U_{i,j} < E_{i,j}. \end{cases} \quad (10)$$

Hence, the worker W_j's score under the strategy profile S is expressed as:

$$G_j(S) = \sum_{i=1}^{i=n} G_{i,j}(S). \quad (11)$$

To sum up, the mathematical formulation of worker benefit, worker preference, competition congestion, worker utility and worker score are given in the worker decision game. The goal of the game is to make workers as satisfied as possible by maximizing their scores. The problem is formalized as follows:

$$\begin{aligned} & max\ \Phi(S) = \sum_{j=1}^{j=k} G_j(S) \\ & \text{s.t. } s_j \subseteq T_N, \quad \forall j \in (1, k). \end{aligned} \quad (12)$$

On this basis, for each worker $W_j \in W_K$, the worker's utility list L^j containing all tasks is generated, in which tasks are sorted from high to low according to the worker's utility $U_{i,j}$ for each task.

B. Stable Matching

On the basis of generated lists, the platform can achieve a stable matching between tasks and workers under the constraints of the number of samples required for tasks and the maximum workload of workers.

In order to measure the degree of worker's satisfaction with the matching result, the worker satisfaction with a value between 0 and 1 is proposed. To calculate it, a sequence of natural numbers starting at 1 is used to evaluate the rating of each task from high to low according to the utility to the worker. For example, suppose there is a worker W_1 and three tasks T_1, T_2, and T_3, whose utility to worker W_1 is 0.5, 0.8, and 0.2, respectively. By sorting the utility values, the ratings of these tasks are marked as 2, 1, and 3. Obviously, the best result for worker W_1 is matched to T_2, while the worst result is matched to T_3. The worker satisfaction is calculated as

$$V_j = (S^j_{worst} - S^j_{match}) \,/\, (S^j_{worst} - S^j_{best}), \quad (13)$$

where S^j_{best} represents the rating the worker gets when selecting the best result, S^j_{worst} represents the rating the worker gets when selecting the worst result, and S^j_{match} represents the rating the worker gets in a final matching.

Therefore, the original problem is divided into two sub-problems namely as $P1$ and $P2$. In the first formulated subproblem $P1$, the aim of the worker is to match the task at the top of its utility list to maximize its satisfaction, which is formalized as follows:

$$\begin{aligned} & (P1)\ max\ \sum_{j=1}^{j=k} V_j \\ & \text{s.t. } \sum_{j=1}^{j=k} x_{i,j} \leqslant t_i,\ \forall T_i \in T_N, \\ & \sum_{i=1}^{i=n} x_{i,j} \leqslant w_j,\ \forall W_j \in W_K, \\ & x_{i,j} = \{0, 1\},\ \forall T_i \in T_N,\ \forall W_j \in W_K. \end{aligned} \quad (14)$$

In the second formulated subproblem $P2$, the aim of the platform is to maximize task benefit, and the problem can be formalized as follows:

$$\begin{aligned} & (P2)\ max\ \sum_{i=1}^{i=n} \sum_{j=1}^{j=k} x_{i,j} B_{i,j} \\ & \text{s.t. } \sum_{j=1}^{j=k} x_{i,j} \leqslant t_i,\ \forall T_i \in T_N, \\ & \sum_{i=1}^{i=n} x_{i,j} \leqslant w_j,\ \forall W_j \in W_K, \\ & x_{i,j} = \{0, 1\},\ \forall T_i \in T_N,\ \forall W_j \in W_K. \end{aligned} \quad (15)$$

By solving these two sub-problems, we can obtain worker-task pairs with maximum worker satisfaction and maximum task benefit. In addition, different task execution order leads to different movement costs in practical applications. The total reward to each worker should be relevant to their task execution routes, which is calculated as

$$P_j(R_j) = \beta \cdot Q_j \cdot (MT_j + ST_j), \quad (16)$$

where R_j is the task execution route of worker W_j. MT_j and ST_j are the total movement time and total sensing time the worker W_j takes to execute tasks according to route R_j, respectively.

The benefit that worker W_j brings to the platform is calculated as

$$B_j = H_j - P_j(R_j) = \sum_{i=1}^{i=n} x_{i,j} H_{i,j} - \beta \cdot Q_j \cdot (MT_j + ST_j). \quad (17)$$

The platform benefit is the sum of the benefit each worker brings to it, which is calculated as

$$Platform\ Benefit = \sum_{j=1}^{j=k} B_j. \quad (18)$$

In order to maximize the platform benefit, the movement cost each worker spends on task execution should be minimized. Therefore, how to compute the shortest route between the worker and the tasks it matches becomes important.

IV. Competition-Congestion-Aware Stable Matching

In this section, a competition-congestion-aware stable matching algorithm (CCASM) is designed to implement the aforementioned system models. The designed algorithm includes a list generation algorithm and a stable matching algorithm.

A. The List Generation Algorithm

The concept of the congestion game was first proposed by Rosenthal [30], which is usually used to solve the problem of multiple players competing for resources. Different players can complete for different resources, but the number of simultaneous players choosing the same resource affect each player's revenue. In general, the more players choose the same resource, the less revenue each player gets.

In order to better show the process of the list generation algorithm, we introduce the preliminary concepts of game as follows:

Definition 1. (Better Response Action) The set of strategy profiles for all workers can be written as $S = (s_j, S_{-j})$, with $S_{-j} = (s_1, s_2, s_3, \ldots, s_{j-1}, s_{j+1} \ldots, s_k)$ represents the set of strategy profiles for all workers except worker W_j. The event where a worker changes its strategy profile from s_j to $s_{j'}$ is a better response action if and only if $G_j(s_{j'}, S_{-j}) > G_j(s_j, S_{-j})$.

Definition 2. (Better Response Set) The set of tasks that enable the worker to make better response action is called the better response set, which is defined as

$$BRS_j(S) = \{T_i | \; T_i \in s_{j'} \text{ and } G_j(s_{j'}, S_{-j}) > G_j(S)\}. \quad (19)$$

Definition 3. (Pure Nash Equilibrium) When any worker under strategy profile S is unable to make a better response action, S is called pure Nash equilibrium strategy.

Since the function $\Phi(S)$ defined as Eq. (12) is a non-negative and non-increasing potential function, the list generation algorithm can achieve Nash equilibrium in finite steps [31]. In this case, all workers are satisfied with their strategy profiles and will not change them. The specific process of algorithm is shown in Algorithm 1.

Algorithm 1 List Generation Algorithm Based on Congestion Game

```
Input: the set of tasks $T_N$, the set of workers $W_K$, and the
       initial strategy profile $S = 0$.
Output: The strategy profile $S$, the workers' utility lists, and
        the tasks' benefit lists.
 1: while for any worker $W_j$ and the strategy profile $S$,
    $BRS_j(S) \neq \varnothing$ do
 2:   The platform publicizes the task congestion $(I^i(S))_{i \in (1, n)}$ to all the workers.
 3:   for $W_j \in W_K$ do
 4:     if $BRS_j(S) \neq \varnothing$ then
 5:       The worker updates his/her strategy profile $s_j$ to
          all tasks in $BRS_j(S)$ and reports the updated
          strategy profile to the platform.
 6:       The platform updates the strategy profile $S$ and
          the task congestion $(I^i(S))_{i \in (1, n)}$.
 7:     end if
 8:   end for
 9: end while
10: Construct the strategy profile $S$.
11: for $W_j \in W_K$ do
12:   Sort the tasks in the worker's strategy profile in
      descending order of utility.
13: end for
15: for $T_i \in T_N$ do
16:   Calculate the benefit that is achieved by allocating the
      task to each worker as Eq. (4).
17:   Sort the workers in descending order of benefit.
18: end for
19: Generate workers' utility lists and tasks' benefit lists.
```

B. The Stable Matching Algorithm

David Gale and Lloyd Shapley first formally came up with the stable marriage problem [32], and proposed the deferred acceptance algorithm to find a stable matching for each man and each woman through the rules of proposal and rejection. Due to the deferred acceptance algorithm is designed to solve one-to-one matching problem, it is not suitable to many-to-many task allocation problem in MCS. In this section, the deferred acceptance algorithm is extended to many-to-many by considering the maximum workload of workers and number of samples required for tasks. And the rules and concepts of stable matching are redefined.

In this work, a matching pair is denoted as (W_j, T_i), which means worker W_j matches task T_i. For task T_i, the number of samples it requires is denoted as t_i, and the number of pairs that have been matched is denoted as $T_{i\bullet pair}$, so the remaining number of pairs that can be matched is denoted as $T_{i\bullet re} = t_i - T_{i\bullet pair}$. Similarly, for worker W_j, its maximum workload is denoted as w_j, and the number of pairs that have been matched

Algorithm 2 Stable Matching Algorithm Based on Extended Deferred Acceptance

```
Input: the set of tasks $T_N$, the set of workers $W_K$, the
       number $t_i$ of samples required for task $T_i$, the
       maximum workload $w_j$ of worker $W_j$, the set of
       workers that have not been fully matched $\Omega$, the
       task's benefit list $L^i$, and the worker's utility list $L^j$.
Output: The constructed routes for all workers.
 1: while $\Omega \neq \varnothing$ do
 2:   for $W_j \in W_K$ do
 3:     $W_j$ sends matching invitations to tasks in $L^j$ in
        order. /* Invitation Rule */
 4:     Calculate $T_{i\bullet re} = t_i - T_{i\bullet pair}$.
 5:     if $T_{i\bullet re} > 0$ then
 6:       Form a candidate pair $(W_j, T_i)$ and put it into $\Psi$.
          /* Acceptance Rule */
 7:     else
 8:       Compare the rankings of inviters and workers that
          have been matched to $T_i$ in its benefit list $L^i$.
 9:       Match the top $t_i$ ranked workers with $T_i$ to form
          candidate pairs, put them into $\Psi$; meanwhile,
          delete other pairs of $T_i$ from $\Psi$. /* Rejection Rule
          */
10:     end if
11:     Calculate $W_{j\bullet re} = w_j - W_{j\bullet pair}$.
12:     Put the $W_j$ which $W_{j\bullet re} \neq 0$ and have not been
        rejected by all the tasks he applied for into $\Omega$,
        otherwise delete the $W_j$ from $\Omega$.
13:   end for
14: end while
15: Construct the candidate pairs set $\Psi$.
16: for $W_j \in W_K$ do
17:   Construct a shortest task execution route by greedy-
      edge method.
18: end for
```

is denoted as $W_{j\bullet pair}$, so the remaining number of pairs that can be matched is denoted as $W_{j\bullet re} = w_j - W_{j\bullet pair}$. The sets of workers that have not been fully matched are denoted as Ω, i.e., $\Omega = W_K$ at the beginning of the Algorithm 2. The set of candidate pairs is denoted as Ψ, i.e., $\Psi = \emptyset$ at the beginning of the Algorithm 2.

Considering the number of samples required for tasks and the maximum workload of workers, three rules are designed for the stable worker-task matching algorithm which are different from the deferred acceptance algorithm and shown below:

Definition 4. (Invitation Rule) Each worker $W_j \in W_K$ first sends matching invitations to tasks in its utility list L^j in order until $W_{j\bullet re} = 0$.

Definition 5. (Acceptance Rule) For each task $T_i \in T_N$, if $T_{i\bullet re} > 0$, the matching invitation is accepted to form a candidate pair (W_j, T_i). Accordingly, the values of $W_{j\bullet re}$ and $T_{i\bullet re}$ decrease by 1.

Definition 6. (Rejection Rule) For each task $T_i \in T_N$, if $T_{i\bullet re} = 0$ and there is an underlying pair such as ($W_{j'}$, T_i) that makes $F_{i,j'} > min(F_{i,j})$, the new candidate pair ($W_{j'}$, T_i) is formed and the previous candidate pair (W_j, T_i) is broken. Otherwise, the underlying pair ($W_{j'}$, T_i) is rejected.

According to the above three rules, tasks and workers can be matched stably. Then, the set Ψ that contains all candidate pairs is constructed, based on which, a greedy-edge method is used to construct a shortest task execution route for each worker. Specifically, a worker and its matched tasks are regarded as the nodes of an undirected graph, and the distance between nodes is regarded as the edge of the undirected graph. The method adds a shortest edge to the task execution route each time, until the constructed task execution route passes through all nodes once and only once, that is, a complete Hamilton loop is obtained. The demonstration of these steps is described in detail in Algorithm 2.

C. *The Instance*

An instance of competition-congestion-aware stable worker-task matching is depicted in Fig. 2. During the task allocation process, since worker's benefit and preference for a task are fixed, the worker's score for a task is only related to competition congestion metric. Therefore, to clearly reflect the impact of competition congestion on workers' decision-making, we use $E^I_{i,j}$ representing the goal of worker W_j for the competition congestion of task T_i. Accordingly, Eq. (10) can be rewritten as:

$$G_{i,j}(S) = \begin{cases} 1, \text{if } T_i \in s_j \text{ and } I^i(S) \leqslant E^I_{i,j}, \\ 0, \text{if } T_i \notin s_j, \\ -1, \text{if } T_i \in s_j \text{ and } I^i(S) > E^I_{i,j}. \end{cases} \quad (20)$$

For the instance given in Fig. 2, the platform publicizes task information to workers. Then, the worker W_1 determines its strategy profile $S_1 = (T_1, T_2, T_3)$ by comparing the competition congestion $I^i(S)$ and the worker's goal $E^I_{i,j}$ for each task, and returns the strategy profile to platform. After that, the platform updates the competition congestion to all workers. After some steps, since the competition congestion of task T_2 does not meet worker W_3's goal, worker W_3 returns its updated strategy profile $S_3 = (T_1, T_3)$. After some steps, as workers' applications for task T_3 increases, the competition congestion of task T_3 is higher than worker W_1's goal, so worker W_1 updates its strategy profile $S_1 = (T_1, T_2)$. Repeating the above process, workers finally form stable strategy profiles, on which the platform generates workers' utility lists and tasks' benefit lists for stable matching process. At the beginning, worker W_1 sends an invitation to task T_2, which is the first task in its utility list, and forms the candidate pair (W_1, T_2) and now $W_{1\bullet re} = 0$. Then, worker W_2 sends an invitation to task T_3, and forms the candidate pairs (W_2, T_3). Worker W_2 sends an invitation to task T_2, since now $T_{2\bullet re} = 0$ and worker W_2 has a higher ranking than worker W_1 in task T_2's benefit list, the candidate pair (W_1, T_2) is broken and a new candidate pair (W_2, T_2) is formed. Next, worker W_3 sends an invitation to task T_3. Since now $T_{3\bullet re} = 0$ and worker W_3 has a lower ranking than worker W_2 in task T_3's benefit list, worker W_3's invitation is rejected. Therefore, worker W_3 sends a new invitation to task T_1 and forms (W_3, T_1). Since the candidate pair (W_1, T_2) is broken, the worker W_1 has not been matched fully and now $W_{1\bullet re} = 1$. Therefore, worker W_1 sends an invitation to task T_1 to form the candidate pair (W_1, T_1). At this point, all three workers are matched completely and fully. The final matching result is $\Psi = \{(W_1, T_1), (W_2, T_2), (W_2, T_3), (W_3, T_1)\}$.

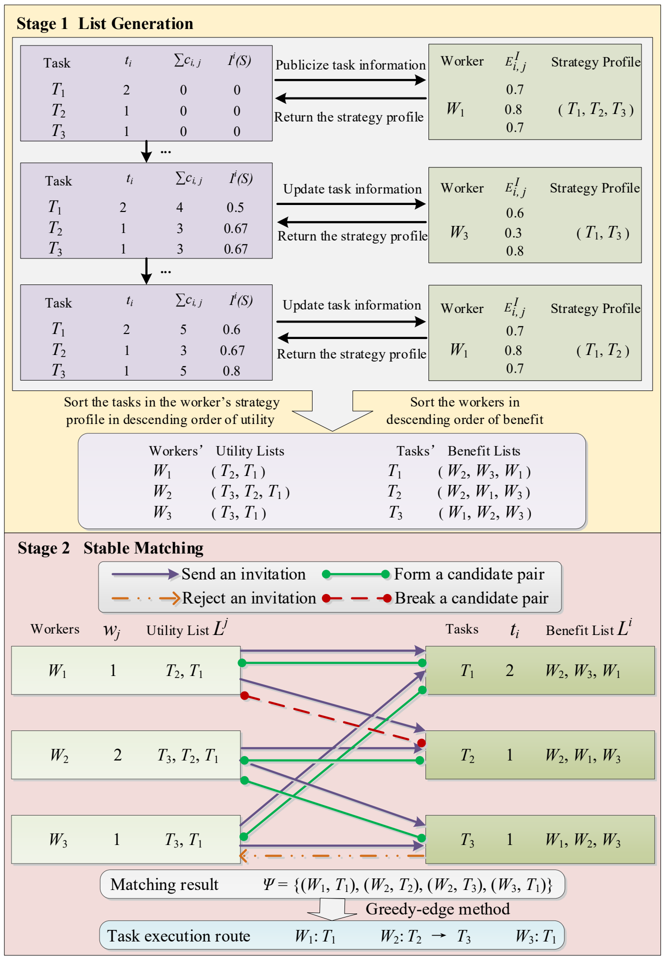

Stage 1 List Generation
Task t_i $\sum c_{i,j}$ $I^i(S)$
T_1 2 0 0
T_2 1 0 0
T_3 1 0 0
Publicize task information
Return the strategy profile
Worker $E^I_{i,j}$ Strategy Profile
W_1 0.7 0.8 0.7 (T_1, T_2, T_3)
...
Task t_i $\sum c_{i,j}$ $I^i(S)$
T_1 2 4 0.5
T_2 1 3 0.67
T_3 1 3 0.67
Update task information
Return the strategy profile
Worker $E^I_{i,j}$ Strategy Profile
W_3 0.6 0.3 0.8 (T_1, T_3)
...
Task t_i $\sum c_{i,j}$ $I^i(S)$
T_1 2 5 0.6
T_2 1 3 0.67
T_3 1 5 0.8
Update task information
Return the strategy profile
Worker $E^I_{i,j}$ Strategy Profile
W_1 0.7 0.8 0.7 (T_1, T_2)
Sort the tasks in the worker's strategy profile in descending order of utility
Sort the workers in descending order of benefit
Workers' Utility Lists
W_1 (T_2, T_1)
W_2 (T_3, T_2, T_1)
W_3 (T_3, T_1)
Tasks' Benefit Lists
T_1 (W_2, W_3, W_1)
T_2 (W_2, W_1, W_3)
T_3 (W_1, W_2, W_3)
Stage 2 Stable Matching
Send an invitation
Reject an invitation
Form a candidate pair
Break a candidate pair
Workers w_j Utility List L^j
Tasks t_i Benefit List L^i
W_1 1 T_2, T_1
W_2 2 T_3, T_2, T_1
W_3 1 T_3, T_1
T_1 2 W_2, W_3, W_1
T_2 1 W_2, W_1, W_3
T_3 1 W_1, W_2, W_3
Matching result $\Psi = \{(W_1, T_1), (W_2, T_2), (W_2, T_3), (W_3, T_1)\}$
Greedy-edge method
Task execution route W_1: T_1 W_2: T_2 → T_3 W_3: T_1

Fig. 2. An instance of competition-congestion-aware stable worker-task matching.

Finally, through the greedy-edge method, the shortest task execution route of each worker can be obtained.

D. Stability, Optimality, and Complexity Analysis

The stability, optimality, and complexity of the designed CCASM algorithm are analyzed as follows:

1) Stability Analysis

Before stability analysis, we introduce the concept of blocking pair.

Definition 7. (Blocking Pair) A pair (W_j, T_i) is a blocking pair which can block a match if W_j prefers T_i to at least one of the tasks it has matched and T_i prefers W_j to at least one of the workers it has matched.

When Algorithm 1 finishes, all workers are satisfied and unable to make a better response action under strategy profile S, so the game result is stable.

Then, due to the number of workers and tasks is not necessarily equal, the stable matching for many-to-many matching problem should be ensured to be both complete and stable. Hence, the concept of stable matching is redefined in definition 8.

Definition 8. (Stable Matching) (i) The match is complete, which means that all workers or all tasks have matched fully and completely. (ii) The match is stable, which means that for each candidate pair, there is no blocking pair that can be used as a better underlying choice to break it.

When Algorithm 2 finishes, the matching result is stable.

Proof: Since each worker sends invitations to tasks in its utility list in order, as the algorithm runs, there is always a moment that all workers or all tasks are matched fully and completely. Next, in order to prove that there is no blocking pair for the matching result, we adopt the reduction to absurdity. We assume that there is at least one blocking pair (W_p, T_q) after matching. We define a function $f_j(x)$ which maps the ranking of each task in the worker W_j's utility list to a real integer. For example, suppose the utility list of worker W_1 is $\{T_2, T_3, T_1\}$, then $f_1(T_2) = 1$, $f_1(T_3) = 2$, and $f_1(T_1) = 3$. Similarly, we define a function $g_i(x)$ which maps the ranking of each worker in the task T_i's benefit list to a real integer. The set of tasks that worker W_j has matched is represented as $\mathcal{W}_j$, the set of workers that task T_i has matched is represented as $\mathcal{T}_i$. According to definition 7, it can be deduced that:

Inference 1. $\exists\ T_m \in \mathcal{W}_p, f_p(T_m) > f_p(T_q)$ (i.e., in the set $\mathcal{W}_p$, there is at least one task T_m which is ranked lower than task T_q in worker W_p's utility list).

Inference 2. $\exists\ W_n \in \mathcal{T}_q, g_q(W_n) > g_q(W_p)$ (i.e., in the set $\mathcal{T}_q$, there is at least one worker W_n which is ranked lower than worker W_p in task T_q's benefit list).

The inference 1 shows that task T_m is ranked lower than task T_q. Since worker W_p sends invitations to tasks in its utility list in order, it can be inferred that the worker W_p has invited the task T_q. However, W_p is not matched with T_q in the end, which means the invitation was rejected or the candidate pair was broken by other workers in $\mathcal{T}_q$. So, it can be deduced that:

Inference 3. $\forall\ W_j \in \mathcal{T}_q, g_q(W_j) < g_q(W_p)$ (i.e., in the set $\mathcal{T}_q$, all workers are ranked higher than worker W_p in task T_q's benefit list).

Obviously, inference 3 is contrary to inference 2. Therefore, the assumption is invalid, that is, there is no blocking pair for the matching result. Based on the above analysis, the matching result is stable.

2) Optimality Analysis

For the algorithm 1, since each worker can add all satisfied tasks to its strategy profile, the game result is optimal.

For the algorithm 2, when the matching is finished, both workers and tasks cannot find a better solution to enhance worker satisfaction or task benefit besides the matching result. Therefore, each worker and task under the matching result are at least as well as it would be under other stable matching results, so the matching result is not only stable but also Pareto optimal.

3) Complexity Analysis

For list generation, the computation complexity of congestion game is $O(k^2)$, the computation complexity of sorting n tasks and k workers in descending order are $O(nlogn)$ and $O(klogk)$ respectively. For stable matching, the computation complexity of the iterative matching process is $O(nk)$, the computation complexity of constructing the shortest task execution routes of k workers is $O(nklogn)$. Therefore, the maximum computation complexity of the designed algorithm is $max\{O(k^2), O(nklogn)\}$.

V. Performance Evaluation

A. Experimental Setting

1) Data Sets and Model Settings

The following three data sets are employed in the simulation to validate the designed model and algorithm.

- *Berlin52* [33]. The data set contains the coordinate information of 52 locations in Berlin. The locations of tasks and workers are randomly generated from these coordinates.
- *NRW1379* [33]. Similarly, the data set contains the coordinate information of 1,379 locations in Nordrhein-Westfalen. The locations of tasks and workers are randomly generated from these coordinates.
- *GeoLife* [34]. The data set was gathered in the Geolife project (Microsoft Research Asia) by 182 participants in a period of over three years (from April 2007 to August 2012). The GPS trajectories of this dataset are represented by time-stamped points, each of which contains the latitude and longitude. This data set contains 17,621 trajectories with a total distance of about 1.2 million kilometers and a total duration of 48,203+ hours. In the simulation, we select workers in the area which is in the northern latitude from 39.975 to 40.025, and eastern longitude from 116.31 to 116.35. In addition, the locations of tasks are randomly generated in the area.

For each data set, the simulation parameters are summarized in Table II.

TABLE II
SIMULATION PARAMETERS

Symbol	Description	Value
Q_j	The sensing quality of worker W_j.	0 ~ 1
$ST_{i,j}$	The sensing time that worker W_j takes to execute task T_i.	10 ~ 20 min
α	Price coefficient.	12
β	Price coefficient.	3
γ	Price coefficient.	1
v	Average movement speed of workers.	60 m/min
$\delta_{i,j}$	Worker W_j's preference for task T_i.	0 ~ 1
t_i	Number of samples required for task T_i.	1 ~ 4
w_j	Maximum workload of worker W_j.	1 ~ 4
$E_{i,j}$	Worker W_j's goal for the utility of completing task T_i.	0 ~ 1

2) Benchmark Algorithms

In order to emphasize the advantages of worker decision game, we compare CCASM with the algorithm without considering competition congestion:

- Traditional Stable Matching (TSM): This algorithm does not consider the impact of competition congestion on workers' decision-making in the MCS system. In the algorithm, the worker's utility list is generated based on worker benefit and worker preference, and then the algorithm 2 is adopted for many-to-many stable worker-task matching. On this basis, tasks are allocated to workers according to the results of stable matching.

Moreover, in order to emphasize the advantages of the designed model and algorithm, we utilize the following baseline task allocation algorithms for comparative studies:

- Random Allocation (RA): This algorithm randomly allocates tasks to workers based on the number of samples required for tasks and the maximum workload of workers constraints.
- Asynchronous Task Selection (ATS): This algorithm simulates the process of workers' asynchronous and distributed task selection, which is a common way [35] for workers to participate in MCS activities through mobile phones. Workers apply for tasks they are satisfied with in the order they arrive at the mobile app, and the platform allocates tasks to them based on their requests. In order to avoid the influence of workers' arrival order, we conducted multiple experiments to get the average value.
- Greedy for Worker Satisfaction (GWS): This algorithm adopts greedy algorithm to maximize worker satisfaction in MCS system, and the optimization target is shown in Eq. (14). The algorithm selects worker-task pairs from high to low according to the utilities in Eq. (9).
- Greedy-enhanced Genetic Algorithm (GGA): This algorithm [20] combines greedy algorithm and genetic algorithm, and uses the result of GWS as the input of population initiation to obtain a better solution. In detail, the three parameters of generation number, population size and mutation rate are 100, 200 and 0.05 respectively.
- Greedy for Platform Benefit (GPB): This algorithm adopts greedy algorithm to maximize platform benefit in MCS system, and the optimization target is shown in Eq. (15). It selects worker-task pairs from high to low according to the benefit in Eq. (4), thus enabling the platform to obtain as much benefit as possible.

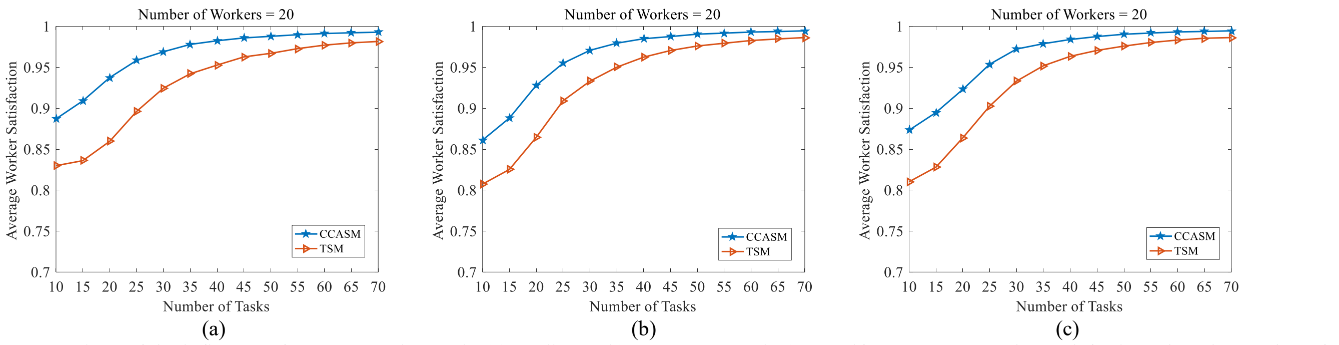


Fig. 3. Comparison of the influence of CCASM and TSM in (a) Berlin52, (b) NRW1379, and (c) GeoLife on average worker satisfaction when the number of tasks changes.

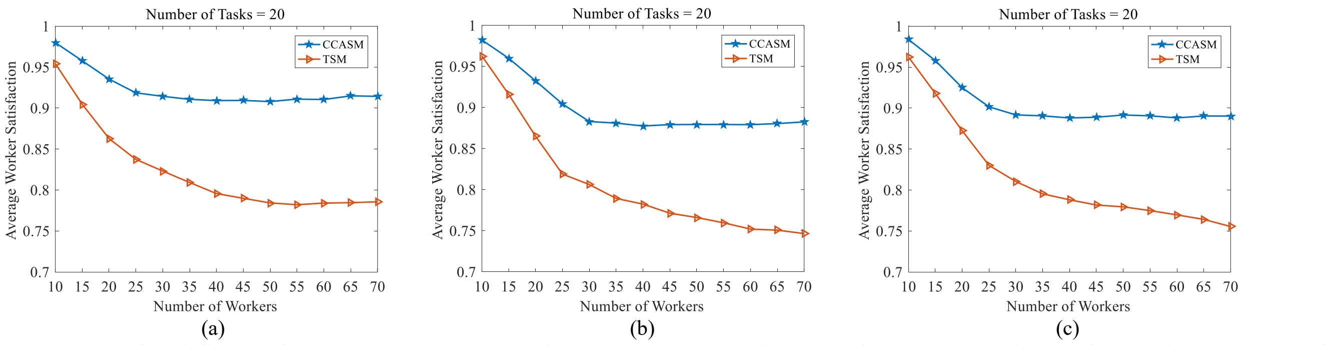


Fig. 4. Comparison of the influence of CCASM and TSM in (a) Berlin52, (b) NRW1379, and (c) GeoLife on average worker satisfaction when the number of workers changes.

By the way, the greedy-edge method is adopted to construct shortest task execution routes for workers after getting the task allocation results from the above five algorithms.

3) *Evaluation Metrics*

- *Average Worker Satisfaction*

Average worker satisfaction is characterized by the average satisfaction of all workers who have received tasks, which is used to evaluate the worker's satisfaction with task allocation.

- *Platform Benefit*

The platform benefit is the sum of the benefit each worker brings to it, which is calculated as Eq. (18).

B. *Performance Evaluation and Discussion*

To illustrate the effectiveness and superiority of the designed model and algorithm, we will show their results in task allocation and compare them with aforementioned benchmark algorithms.

1) *Superiority of Model with Congestion*

Fig. 3 and Fig. 4 show CCASM's superiority over TSM in terms of average worker satisfaction when the number of tasks and workers changes respectively. In detail, as shown in Fig. 3, (i) when the number of tasks is small, CCASM is significantly better than TSM. (ii) When the number of tasks increases, the average worker satisfaction in CCASM and TSM increases. (iii) When the number of tasks is large, the performance of CCASM and TSM are similar. The reasons are that (i) when the number of workers is fixed and the number of tasks is small, the competition congestion is high, so many workers may not be able to match the tasks they are satisfied with. However, CCASM can make the overall decision of workers more stable and assist them in making favorable decisions. (ii) The increase in the number of tasks means less competition congestion. In this case, workers are more likely to match the tasks they are satisfied with, so the average worker satisfaction increases. (iii) When the number of tasks is significantly larger than the number of workers, there is almost no competition congestion, so the performance of CCASM and TSM is similar. The results in Fig. 4 can also be interpreted in the same way. It is worth mentioning that CCASM still performs well when the number of tasks is fixed and the number of workers increases. This is because when the competition for a task is too fierce, some workers may abandon the application for the task and apply for other tasks they are satisfied with. In summary, the worker decision game can significantly improve workers' satisfaction.

2) *Average Worker Satisfaction*

Fig. 5 shows CCASM versus five baselines in terms of average worker satisfaction when the number of tasks changes. Obviously, CCASM performs best in terms of average worker satisfaction and GPB performs the worst. This is because CCASM tries to match workers with tasks they are satisfied with, while GPB is an optimization algorithm aimed at maximizing platform benefit which ignores worker satisfaction in task allocation. Moreover, as the number of tasks increases, the CCASM, GGA, GWS, and ATS perform better. The reason is that when the number of workers is fixed, as the number of tasks increases, workers are more likely to receive satisfactory tasks. Fig. 6 shows the performance of CCASM is better than ATS, RA, and GPB when the number of workers changes. In addition, CCASM performs similarly to GGA and GWS in Fig. 6(a), and worse than GGA and GWS in Fig. 6(b) and Fig. 6(c). This is because as the number of workers increases, the competition among workers becomes more fierce, resulting in a decrease of average worker satisfaction.

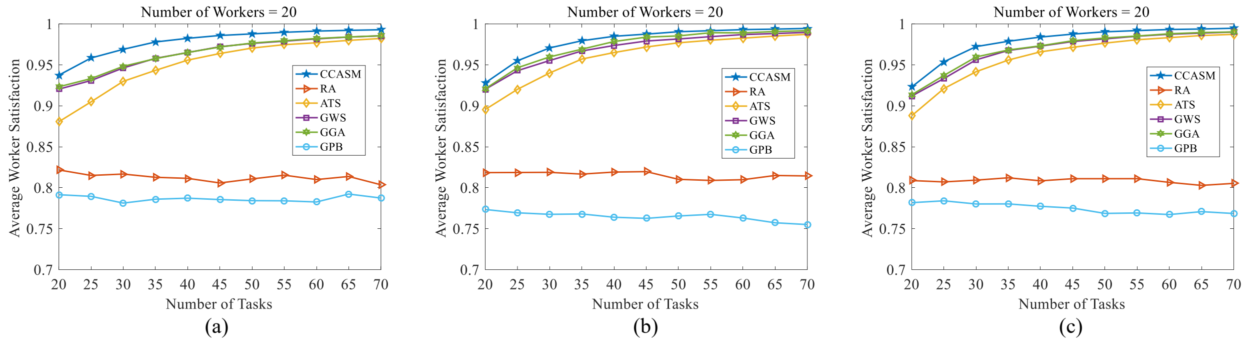


Fig. 5. Comparison of the influence of CCASM and baselines in (a) Berlin52, (b) NRW1379, and (c) GeoLife on average worker satisfaction when the number of tasks changes.

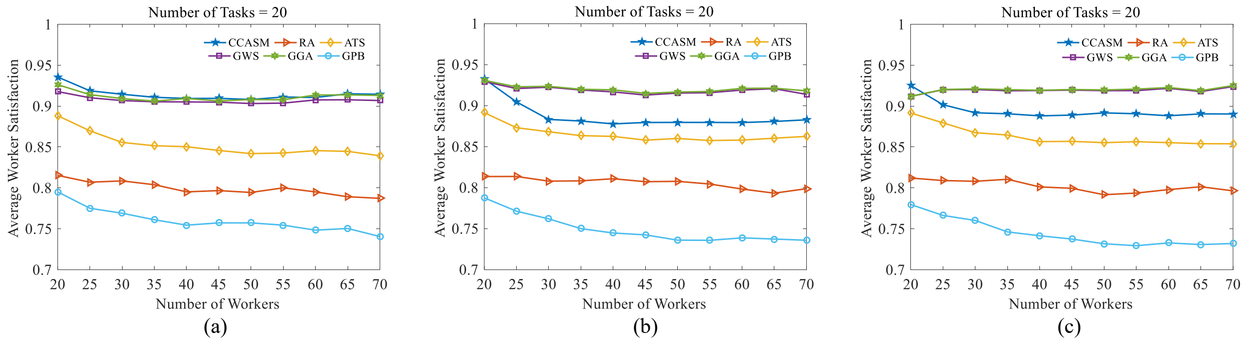


Fig. 6. Comparison of the influence of CCASM and baselines in (a) Berlin52, (b) NRW1379, and (c) GeoLife on average worker satisfaction when the number of workers changes.

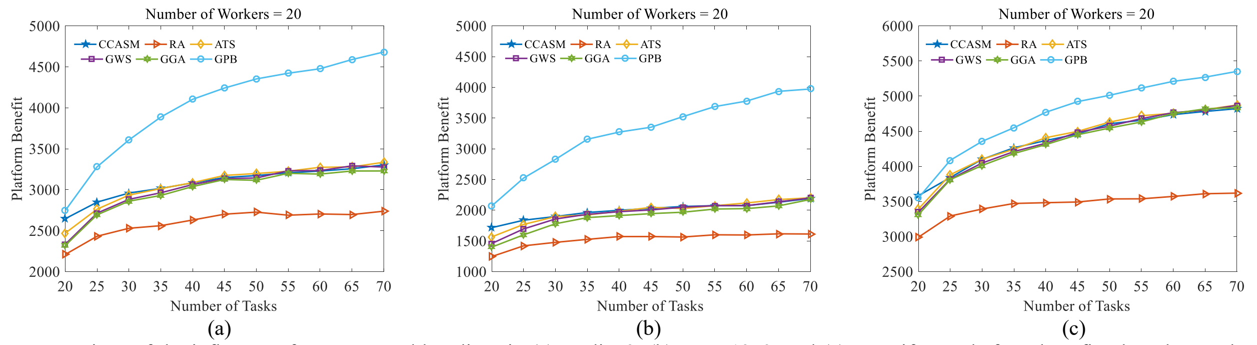


Fig. 7. Comparison of the influence of CCASM and baselines in (a) Berlin52, (b) NRW1379, and (c) GeoLife on platform benefit when the number of tasks changes.

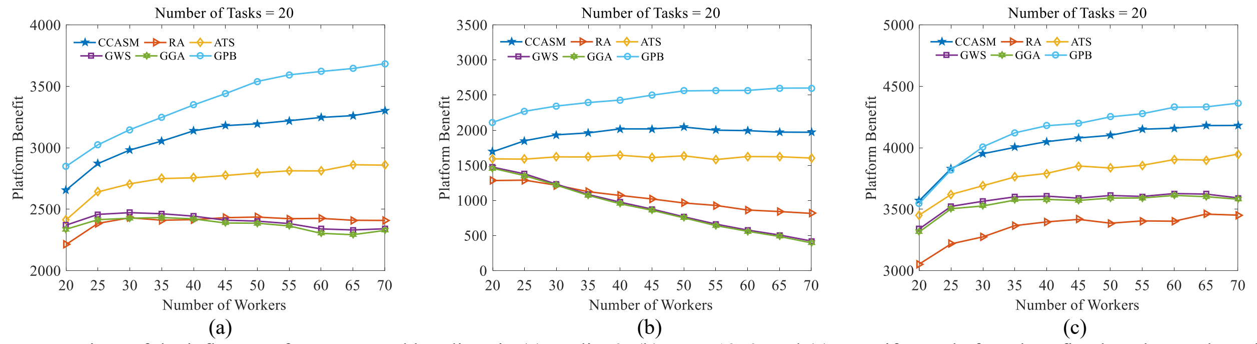


Fig. 8. Comparison of the influence of CCASM and baselines in (a) Berlin52, (b) NRW1379, and (c) GeoLife on platform benefit when the number of workers changes.

3) Platform Benefit

Fig. 7 and Fig. 8 show the difference between CCASM and above-mentioned baselines in terms of platform benefit when the number of tasks and workers changes respectively. Obviously, since the optimization goal of GPB is to maximize platform benefit, it has the best performance. In addition, when the number of workers is small and the number of tasks is large, CCASM performs similar to ATS, GWS, and GGA. Conversely, when the number of workers is large and the number of tasks is small, CCASM performs significantly better than RA, ATS, GWS, and GGA. In this case, the competition among workers is relatively fierce, and CCASM matches tasks with workers who can bring the most benefit to the task, thereby improving platform benefit.

In summary, when the competition for tasks is relatively mild, CCASM performs better than baselines in terms of average worker satisfaction. While in the case of fierce competition for tasks, CCASM not only performs well in average worker satisfaction, but also improves platform benefit significantly. According to the above discussions, it can be concluded that the designed model and algorithm are effective and meaningful.

VI. Discussion

In this section, we discuss issues that are not reported or addressed in this work due to space constraints, which are the directions of our future work. We are mainly concerned with task benefit, worker benefit, worker preference, and competition congestion metric when formulating our stable matching problem. Other types of factors may need to be considered in stable matching, such as worker reputation, task urgency, etc. Besides, this paper only considers two constraints when solving the stable matching problem, one is the number of samples required for the task, and the other is the maximum workload of workers. We plan to explore more fine-grained constraints in the future work, such as the maximum working hours of workers.

VII. Conclusion

The existing stable matching studies lack deep consideration regarding the effects of workers' competition phenomena and complex behaviors, which may reduce workers' satisfaction and enthusiasm for participation in sensing activities. In this paper, we investigated a competition-congestion-aware stable matching problem by considering the competition of workers for tasks. Due to the competition and complex behaviors of workers, a worker decision game based on congestion game theory is designed to improve worker satisfaction by jointly considering worker benefit, worker preference, and competition congestion metric. On this basis, a stable matching algorithm based on extended deferred acceptance algorithm is designed to make workers and tasks mapping stable, and to construct the shortest task execution route for each worker. Finally, the effectiveness of the proposed mechanism is verified by comparing it with traditional benchmark stable matching algorithm and five baseline task allocation algorithms. Simulation results show that the designed algorithm performs well in worker satisfaction and platform benefit.

References

[1] T. Wang, L. Qiu, A. K. Sangaiah, et. al, "Energy-efficient and Trustworthy Data Collection Protocol Based on Mobile Fog Computing in Internet of Things," IEEE Transactions on Industrial Informatics, vol. 16, no.5, pp. 3531–3539, May. 2020.

[2] R. K. Ganti, F. Ye, and H. Lei, "Mobile crowdsensing: current state and future challenges," IEEE Communications Magazine, vol. 49, no. 11, pp. 32–39, Nov. 2011.

[3] G. Yang, T. Liang, X. He, et. al, "Global and Local Reliability Based on Routing Protocol for Wireless Sensor Network," IEEE Internet Things, vol. 6, no. 2, pp. 3620–3632, Apr. 2019.

[4] S. Xu, X. Chen, X. Pi, et. al, "Poster Abstract: Vehicle Dispatching for Sensing Coverage Optimization in Mobile Crowdsensing Systems," in *Proc.* IPSN, Montreal, QC, Canada, 2019, pp. 311–312.

[5] M. Zappatore, A. Longo, and M. A. Bochicchio, "Using mobile crowd sensing for noise monitoring in smart cities," in *Proc.* 2016 International Multidisciplinary Conference on Computer and Energy Science. Split, Croatia, 2016, pp. 1–6.

[6] M. Mehdi, G. Muhlmeier, K. Agrawal, et al, "Referenceable mobile crowdsensing architecture: A healthcare use case," Procedia Computer Science, pp. 445–451, July. 2018.

[7] F. Bock, S. D. Martino, and A. Origlia, "Smart Parking: Using a Crowd of Taxis to Sense On-Street Parking Space Availability," IEEE Transactions on Intelligent Transportation Systems, vol. 21, no. 2, pp. 496–508, Feb. 2020.

[8] J. Wang, L. Wang, Y. Wang, et. al, "Task Allocation in Mobile Crowd Sensing: State of the Art and Future Opportunities," IEEE Internet of Things, vol. 5, no. 5, pp. 3747–3757, Oct. 2018.

[9] L. Zhang, M. Xiao, H. Zhao et. al, "Combined Crowdsourcing Task Auction Mechanism Based on Stable Matching," in *Proc.* IEEE 21st International Conference on High Performance Computing and Communications, Zhangjiajie, China, 2019, pp. 1908–1913.

[10] Z. Zhou, J. Feng, B. Gu, et. al, "When Mobile Crowd Sensing Meets UAV: Energy-Efficient Task Assignment and Route Planning," IEEE Transactions on Communications, vol. 66, no. 11, pp. 5526–5538, Nov. 2018.

[11] X. Chen, "A Stable Task Assignment Scheme in Crowdsourcing," in *Proc.* IEEE CSE and IEEE EUC, New York, NY, USA, 2019, pp. 489–494.

[12] F. Yucel, M. Yuksel, and E. Bulut, "QoS-based Budget Constrained Stable Task Assignment in Mobile Crowdsensing," IEEE Transactions on Mobile Computing, to be published.

[13] C. Dai, X. Wang, K. Liu, et. al, "Stable Task Assignment for Mobile Crowdsensing with Budget Constraint," IEEE Transactions on Mobile Computing, to be published.

[14] M. Abououf, S. Singh, H. Otrok, et. al, "Gale-Shapley Matching Game Selection – A Framework for User Satisfaction," IEEE Access, vol. 7, pp. 3694–3703, 2019.

[15] D. Zhang, H. Xiong, L. Wang, et. al, "CrowdRecruiter: selecting participants for piggyback crowdsensing under probabilistic coverage constraint," in *Proc*. 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing, NY, USA, 2014, pp. 703–714.

[16] H. Xiong, D. Zhang, G. Chen, et. al, "iCrowd: Near-Optimal Task Allocation for Piggyback Crowdsensing," IEEE Transactions on Mobile Computing, vol. 15, no. 8, pp. 2010–2022, Aug. 2016.

[17] C. H. Liu, B. Zhang, X. Su, et. al, "Energy-Aware Participant Selection for Smartphone-Enabled Mobile Crowd Sensing," IEEE Systems Journal, vol. 11, no. 3, pp. 1435–1446, Sept. 2017.

[18] J. Wang, Y. Wang, D. Zhang, et. al., "Multi-Task Allocation in Mobile Crowd Sensing with Individual Task Quality Assurance," IEEE Transactions on Mobile Computing, vol. 17, no. 9, pp. 2101–2113, Sept. 2018.

[19] T. Hu, M. Xiao, C. Hu, et. al, "A QoS-sensitive task assignment algorithm for mobile crowdsensing," Pervasive and Mobile Computing, pp. 333–342, Jan. 2017.

[20] B. Guo, Y. Liu, W. Wu, et. al, "ActiveCrowd: A Framework for Optimized Multi-Task Allocation in Mobile Crowdsensing Systems," IEEE Transactions on Human-Machine Systems, vol. 47, no. 3, pp. 392–403, Aug. 2016.

[21] Y. Liu, B. Guo, Y. Wang, et. al, "TaskMe: multi-task allocation in mobile crowd sensing," in *Proc.* 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing, New York, NY, USA, 2016, pp. 403–414.

[22] G. Liu, H. Zhao, and D. Li, "Resource allocation in heterogeneous networks: A Modified many-to-one swap matching," in *Proc.* IEEE 17th International Conference on Communication Technology. Chengdu, China, 2017, pp. 508–512.

[23] M. Baïou and M. Balinski, "Erratum: The Stable Allocation (or Ordinal Transportation) Problem," Mathematics of Operations Research, vol. 27, no. 4, pp. 662–680, Nov. 2002.

[24] X. Li, and Q. Zhu, "Social Incentive Mechanism Based Multi-User Sensing Time Optimization in Co-Operative Spectrum Sensing with Mobile Crowd Sensing," Sensors, vol. 18, no. 1, pp. 250, 2018.

[25] X. Liu, K. Ota, A. Liu, et. al, "An incentive game based evolutionary model for crowd sensing networks," Peer-to-Peer Networking and Applications, vol. 9, no. 4, pp. 692–711, July. 2016.

[26] Y. Sun, Y. Zhu, Z. Feng, et. al, "Sensing processes participation game of smartphones in participatory sensing systems," in *Proc.* SECON, Singapore, 2014, pp. 239–247.

[27] Y. Liu, H. Li, G. Zhao, et. al, "Reverse Auction Based Incentive Mechanism for Location-Aware Sensing in Mobile Crowd Sensing," in *Proc.* IEEE ICC, Kansas City, MO, 2018, pp. 1–6.

[28] J. Wang, F. Wang, Y. Wang, et. al, "Allocating Heterogeneous Tasks in Participatory Sensing with Diverse Participant-Side Factors," IEEE Transactions on Mobile Computing, vol. 18, no. 9, pp. 1979–1991, Sept. 2019.

[29] H. Vahdat-Nejad, E. Asani, Z. Mahmudian, et. al, "Context-aware computing for mobile crowd sensing," Future Generation Computer Systems, vol. 99, pp. 321–332, Oct. 2019.

[30] R. W. Rosenthal, "A class of games possessing pure-strategy nash equilibria," International Journal of Game Theory, vol. 2, no. 1, pp. 65–67, 1973.

[31] D. Monderer and L. S. Shapley, "Potential games," Games and Economic Behavior, vol. 14, no. 1, pp.124–143, May 1996.

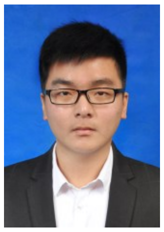
[32] D. Gale and L. S. Shapley, "College Admissions and the Stability of Marriage," American Mathematical Monthly, vol. 69, no. 1, pp. 9–15, Jan. 1962.

[33] http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/tsp/.

[34] http://research.microsoft.com/en-us/downloads/b16d359d-d164-469e-9fd4-daa38f2b2e13/.

[35] M. H. Cheung, R. Southwell, F. Hou, et. al, "Distributed Time-Sensitive Task Selection in Mobile Crowdsensing," in *proc*. 16th ACM International Symposium on Mobile Ad Hoc Networking and Computing, Hangzhou, China, 2015, pp.157–166.

Guisong Yang received his Ph.D. degree in Control Theory and Control Engineering from TongJi University in 2013, and worked as a research scholar at Michigan State University from 2009 to 2011. Currently, He is an associate professor with the Department of Computer Science and Engineering at University of Shanghai for Science and Technology. His research interests include Internet of Things and pervasive computing, delay tolerant and opportunistic networks, and mobile crowd sensing. He is a member of IEEE and ACM.



Buye Wang received the B.S. degree in information management and information system from Tianjin University of Technology, Tianjin, China, in 2017. He is currently pursuing the master's degree in computer science and technology at the University of Shanghai for Science and Technology, shanghai, China. His current research interest is mobile crowd sensing.

Xingyu He received her Ph.D. degree in Control Theory and Control Engineering from TongJi University in 2017. She is an assistant professor with College of Communication and Art Design at University of Shanghai for Science and Technology. Her research interests include wireless sensor networks and pervasive computing, delay tolerant networks, incentive scheme, swarm intelligence and mobile crowd sensing.

Jiangtao Wang received the PhD degree from Peking University, China, in 2015. He is currently an Associate Professor with Tenure in the Intelligent Healthcare Center, Coventry University, UK. Before that, he was a lecturer with the School of Computing and Communications at University, UK. His research interest includes mobile and pervasive computing, crowdsensing/crowdsourcing, and IoT.

Haris Pervaiz (S'09–M'09) received the M.Sc. degree in information security from the Royal Holloway University of London, Egham, U.K., in 2005, and the Ph.D. degree from the School of Computing and Communication, Lancaster University, Lancaster, U.K., in 2016. He is currently working as an Assistant Professor (Lecturer) with School of Computing and Communication, Lancaster University, Lancaster, U.K. He was a Research Fellow with the 5G Innovation Centre, University of Surrey, Guildford, U.K., from 2017 to 2018 and an EPSRC Doctoral Prize Fellow with the School of Computing and Communication, Lancaster University from 2016 to 2017. His current research interests include green heterogeneous wireless communications and networking, 5G and beyond, millimeter wave communication, and energy and spectral efficiency. He has been actively involved in projects, such as Sustainable and Low Emission Internet of Things for Climate Smart Agriculture, CROWN, CogGREEN, TWEETHER, Energy proportional EnodeB for LTE-Advanced and Beyond and the EPSRC GCRF funded DARE project. He is an Associate Editor of IEEE ACCESS, an editorial board member of Emerging Telecommunications Technologies (Wiley), and an Associate Editor of Internet Technology Letters (Wiley).