Win-Win Approaches for Cross Dynamic Task Assignment in Spatial Crowdsourcing

Tianyue Ren, Zhibang Yang, Yan Ding, *Member, IEEE*, Xu Zhou, Kenli Li, *Senior Member, IEEE*, Yunjun Gao, *Senior Member, IEEE*, and Keqin Li, *Fellow, IEEE*

***Abstract*—Spatial crowdsourcing (SC) is becoming increasingly popular recently. As a critical issue in SC, task assignment currently faces challenges due to the imbalanced spatiotemporal distribution of tasks. Hence, many related studies and applications focusing on cross-platform task allocation in SC have emerged. Existing work primarily focuses on the maximization of total revenue for inner platform in cross task assignment. In this work, we formulate a SC problem called Cross Dynamic Task Assignment (CDTA) to maximize the overall utility and propose improved solutions aiming at creating a win-win situation for inner platform, task requesters, and outer workers. We first design a hybrid batch processing framework and a novel cross-platform incentive mechanism. Then, with the purpose of allocating tasks to both inner and outer workers, we present a KM-based algorithm that gets the accurate assignment result in each batch and a density-aware greedy algorithm with high efficiency. To maximize the revenue of inner platform and outer workers simultaneously, we model the competition among outer workers as a potential game that is shown to have at least one pure Nash equilibrium and develop a game-theoretic method. Additionally, a simulated annealing-based improved algorithm is proposed to avoid falling into local optima. Last but not least, since random thresholds lead to unstable results when picking tasks that are preferentially assigned to inner workers, we devise an adaptive threshold selection algorithm based on multi-armed bandit to further improve the overall utility. Extensive experiments demonstrate the effectiveness and efficiency of our proposed algorithms on both real and synthetic datasets.**

***Index Terms*—Game theory, multi-armed bandit, simulated annealing, spatial crowdsourcing (SC), task assignment.**

Received 23 August 2024; revised 24 July 2025; accepted 24 November 2025. Date of publication 3 December 2025; date of current version 30 December 2025. This work was supported in part by the Creative Research Groups Program of the National Natural Science Foundation of China under Grant 62321003, in part by the National Natural Science Foundation of China under Grant 62576051, Grant U23A20317, Grant 62572182, Grant 62402481, and Grant U23A20317, in part by the Natural Science Foundation of Hunan Province under Grant 2023JJ10016, and in part by China Scholarship Council under Grant CSC202306130014. An earlier version of this paper was presented at the part of 2023 IEEE 39th International Conference on Data Engineering (ICDE), Anaheim, CA, USA, [DOI: 10.1109/ICDE55515.2023.00113]. Recommended for acceptance by M. Zhang. *(Corresponding authors: Xu Zhou; Zhibang Yang.)*

Tianyue Ren, Yan Ding, Xu Zhou, and Kenli Li are with the College of Computer Science and Electronic Engineering, Hunan University, Changsha 410082, China (e-mail: hnurty@hnu.edu.cn; ding@hnu.edu.cn; zhxu@hnu.edu.cn; lkl@hnu.edu.cn).

Zhibang Yang is with the Hunan Province Key Laboratory of Industrial Internet Technology and Security, Changsha University, Changsha 410022, China (e-mail: yangzb@ccsu.edu.cn).

Yunjun Gao is with the College of Computer Science, Zhejiang University, Hangzhou 310027, China (e-mail: gaoyj@zju.edu.cn).

Keqin Li is with the Department of Computer Science, State University of New York, New Paltz, NY 12561 USA (e-mail: lik@newpaltz.edu).

Digital Object Identifier 10.1109/TKDE.2025.3639413

I. Introduction

SPATIAL crowdsourcing (SC) has become a widely recognized new economic paradigm in recent years. In traditional SC, a task requester sends the spatial task to the crowdsourcing platform, the platform matches a suitable worker with the task, and then the worker moves to a specified location to perform the task. Existing SC applications, such as Didi[1], Uber[2] and GrubHub[3], bring convenience to the daily lives of people while also generating huge economic benefits.

Most of the existing research on spatial crowdsourcing perform task assignment on a single platform, which gradually reveals some drawbacks. On the one hand, there is a shortage of workers when the orders increase dramatically. On the other hand, it is difficult to address the issue of imbalanced distribution of workers. Fig. 1 presents an example of drivers from two ride-hailing service companies (Uber and Lyft[4]) showing an uneven spatial distribution in New York City between 5:00 p.m. and 8:00 p.m.

During the order peak period, users often hail rides faster by placing orders on several different platforms. However, duplicate orders from users not only add costs to the platforms, but may also cost users additional money. In this case, the cross-platform collaborative task assignment mode can meet the needs of multiple parties. Cross-platform spatial crowdsourcing has emerged in a wide range of real-world scenarios. For instance, when we use Didi for ride-hailing, the platform not only sends its own workers to provide the service, but also collaborates with third-party platforms with the same type of service to address the imbalance between supply and demand.

Prior Work: Existing studies on task assignment in cross-platform spatial crowdsourcing [2], [3], [4] ignore the interests of outer workers as well as the competition among outer workers. In addition, some existing studies adopt a consistent allocation strategy for both inner and outer workers, which may lead to the loss of inner workers. Other studies use fixed or random reward thresholds to filter tasks prioritized for allocation to inner workers, resulting in poor or unstable results. In this work, we investigate a task assignment problem of SC, called Cross Dynamic Task Assignment (CDTA). Several win-win approaches

[1]https://web.didiglobal.com/
[2]https://www.uber.com/
[3]https://www.grubhub.com/
[4]https://www.lyft.com/

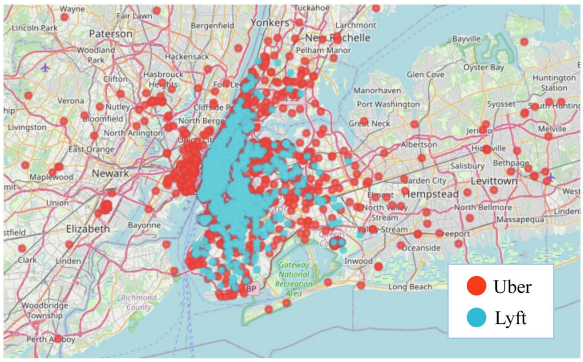


Fig. 1. Distribution of drivers in New York City.

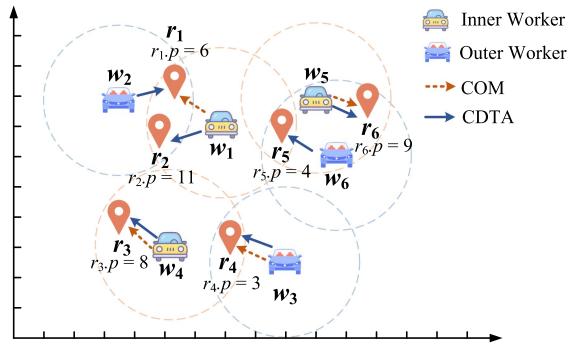


Fig. 2. Running example.

TABLE I
ARRIVAL TIME OF WORKERS AND TASKS

Time	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}	t_{11}	t_{12}
Order	w_1	r_1	w_2	r_2	w_3	w_4	r_3	r_4	w_5	r_5	w_6	r_6

are proposed to obtain more revenue for inner platform and outer workers while reducing the response time of task requesters.

Example 1 (Motivation Example): As shown in Fig. 2, there are six task requests (from the inner platform) and six workers in 2D space, where inner workers and outer workers are represented by yellow cars and blue cars, respectively. The dotted lines indicate the service area of the workers. In addition, $r.p$ denotes the payoff for task r. The arrival order of workers and tasks is presented in Table I. We set the value threshold to 5. Tasks with payoff greater than 5 are allocated to inner workers, otherwise they are allocated to outer workers. Similar to [2], an outer worker who completes a task receives 50% of the payoff for the task from the inner platform.

With the RamCOM algorithm [2], each arriving task is immediately assigned. The final matching result is $\{(r_1, w_1), (r_3, w_4), (r_4, w_3), (r_6, w_5)\}$. Thus, the revenue of the inner platform is $6{+}8{+}3{\times}50\%{+}9{=}24.5$, the number of assigned tasks is 4, and the total revenue of the outer workers is 1.5. In the CDTA problem, we use a batch processing framework which takes more information into account. Therefore, we process six tasks in the same batch. The Greedy algorithm presented in Section IV gives priority to assigning high-reward tasks to inner workers in a greedy manner. However, for unassigned high-value tasks, we will also reassign them to outer workers. We can get the matching result $\{(r_2, w_1), (r_6, w_5), (r_3, w_4), (r_5, w_6), (r_4, w_3), (r_1, w_2)\}$. Therefore, the obtained revenue of the inner platform is $11{+}9{+}8{+}4{\times}50\%{+}3{\times}50\%{+}6{\times}50\%{=}34.5$, the number of assigned tasks is 6, and the revenue of outer workers is 6.5.

Obviously, the inner platform and outer workers get better results through cross dynamic task assignment. It is worth noting that it also creates a win-win situation by allocating the task with high value to outer worker.

Challenges: There are four main challenges we need to address in this paper. First, an incentive mechanism is required to stimulate outer workers to complete tasks from inner platform. Second, the time-sensitive batching strategy is limited when processing tasks with uneven spatial distribution and calls for a more efficient batch processing strategy. Third, it is hard to devise effective solutions for the CDTA problem that simultaneously benefit the inner platform, task requesters, and outer workers. Finally, fixed-threshold or random-threshold approaches are commonly employed to determine which tasks should be prioritized for allocation to inner workers, which are not adaptable to changes in the environment.

Contributions: Existing studies [5], [6] merely design incentive mechanisms for SC workers in a single platform scenario, therefore, we propose a cross-platform incentive mechanism that introduces a novel payment pricing model for outer workers. In addition, we design a hybrid batch-based processing strategy to address the issue of imbalanced task temporal distribution. Then, four task assignment algorithms that consider the interests of the inner platform, outer workers, and users are developed. Last but not least, an adaptive threshold selction method is proposed to determine the task reward threshold flexibly.

Compared to our preliminary study [1], we mainly make the following extensions: 1) There may exist more than one Nash equilibrium (NE) solution to a strategic game, and thus existing game-theoretic methods may fall into local optima. A simulated annealing-based optimization strategy is designed to obtain a better NE. 2) To avoid unstable results, we propose an adaptive threshold selection method based on Multi-Armed Bandit (MAB). Each round of threshold selection is considered as a pulling arm action to get better global results through a continuous process of exploration and exploitation. 3) We add experiments of these two algorithms on real and synthetic datasets and compare them with prior methods. The contributions of this paper are summarized as follows:

- We investigate the problem of cross dynamic task assignment in SC and prove that it is NP-hard.
- We design a hybrid batch processing framework that handles tasks with uneven time distribution and a cross-platform incentive mechanism that encourages outer workers to complete tasks.
- We propose two algorithms for global task assignment, where the KM-based algorithm yields accurate results within each batch and the density-aware greedy algorithm improves allocation efficiency.

TABLE II
MAIN NOTATIONS

Symbol	Definition
w_{in}/W_{in}	Spatial inner worker/worker set
w_{out}/W_{out}	Spatial outer worker/worker set
ρ_w	Reputation score of worker w
r/R	Spatial task/task set
p_r	Task reward of r
$p'(w_{out}, r)$	Payment to w_{out} for completing r
SD	Shortage degree of workers
α, β	Pricing parameter for outer payment
$Rev(w_{out}, r)$	Outer revenue of w_{out} for completing r
c_w	Unit cost of w to perform a task
b_t	Time threshold for batch mode
b_n	Size threshold for batch mode

- The cross-platform task assignment is modeled as a potential game. We explore a game-theoretic algorithm and a simulated annealing-based algorithm to optimize the revenue received by outer workers.
- We design an MAB-based adaptive threshold selection approach to determine which tasks are assigned in priority to inner workers and further improve the overall utility.
- We conduct extensive experiments on real datasets and synthetic datasets, and validate the effectiveness and efficiency of our proposed approaches.

The rest of this paper is organized as follows. The definitions of key concepts are introduced in Section II. We then design a solution framework in Section III. Global task assignment algorithms are presented in Section IV and game theory based algorithms are proposed in Section V. The adaptive threshold selection method is investigated in Section VI. We report the experimental results in Section VII. Sections VIII and IX discuss related work and summarize our work, respectively.

II. PROBLEM STATEMENT

We first introduce some important concepts and then formalize the CDTA problem in this section. The frequently used annotations are summarised in Table II.

In cross dynamic SC, after the SC platform receives a task request, it can choose to assign the task to the inner worker or to the outer worker shared by a collaborative platform.

Definition 1. (Spatial Task): A spatial task, denoted by $r{=}\langle t_r, l_r, d_r, p_r\rangle$, has a location l_r in the 2D spatial map, a submission time t_r, an expiration time $t_r{+}d_r$. In addition, p_r denotes the payoff for the task r.

Definition 2. (Inner Worker): An inner worker, denoted as $w_{in}{=}\langle t_{w_{in}}, l_{w_{in}}, d_{w_{in}}, rad_{w_{in}}, \rho_{w_{in}}\rangle$, is owned by the inner platform. The inner worker has a location $l_{w_{in}}$, a maximum service radius $rad_{w_{in}}$, an arrival time $t_{w_{in}}$, and an expiration time $t_{w_{in}}{+}d_{w_{in}}$. Moreover, the reputation score $\rho_{w_{in}}{\in}[0,1]$ of w_{in} is determined by his/her historical task completion record, which depends on the number of completed tasks and the scores given by task requestors.

Definition 3. (Outer Worker): An outer worker, denoted as $w_{out}{=}\langle t_{w_{out}}, l_{w_{out}}, d_{w_{out}}, rad_{w_{out}}, \rho_{w_{out}}\rangle$, is shared by an outer platform. Each outer worker has an available time $t_{w_{out}}$, an expiration time $t_{w_{out}}{+}d_{w_{out}}$, a location $l_{w_{out}}$, a reachable service radius $rad_{w_{out}}$ and a reputation score $\rho_{w_{out}} \in [0,1]$ (same meaning as $\rho_{w_{in}}$ in Definition 2).

Definition 4. (Outer Payment): After completing a spatial task r, the outer worker w_{out} can receive an outer payment $p'(w_{out}, r){\in}(0, p_r]$.

Definition 5. (Outer Revenue): The outer revenue of outer worker w_{out} who performs the spatial task r is expressed as $Rev(w_{out}, r){=}p'(w_{out}, r){-}c_{w_{out}}{\times}d(w_{out}, r)$, where $c_{w_{out}}$ denotes the unit cost of w_{out} to perform a task and $d(w_{out}, r)$ is the euclidean distance between w_{out} and r.

Outer workers only focus on their own profit (outer revenue). However, the inner platform needs to consider both profit and user satisfaction since reputation is also important to a platform. Tong et al. [7] maximize the profit on the platform and guarantee the reliability of allocated workers by designing a utility function. Accordingly, we introduce the following utility function which considers the revenue of the inner platform and the reputation scores of the assigned workers to evaluate the task assignment result.

Definition 6. (Inner Utility): The revenue that the inner worker w_{in} brings to the inner platform after completing the task r is p_r. After the outer worker w_{out} performs a task r, the platform gets a revenue of $p_r{-}p'_r$ where p'_r represents the outer payment. Given a task assignment set M, the inner utility is defined as

$$U_M = \sum_{(r,w_{in})\in M} (p_r \times \rho_{w_{in}}) + \sum_{(r,w_{out})\in M} ((p_r - p'_r) \times \rho_{w_{out}}), \tag{1}$$

where task $r{\in}R$, inner worker $w_{in}{\in}W_{in}$ and out worker $w_{out}{\in}W_{out}$.

The CDTA problem is formally defined as follows.

Definition 7. (CDTA Problem): Given a task request set R, an inner worker set W_{in} and an outer worker set W_{out}, with workers and tasks arriving dynamically, the CDTA problem aims at finding a matching result M with the maximal inner utility, satisfying the following constraints:

- *One-on-one Constraint:* Every task is served by only one worker.
- *Unchangeable Constraint:* Once a matching between a task and a worker is formed at the end of a batch, it cannot be changed thereafter.
- *Time Constraints:* Workers are eligible to perform tasks that appear before they leave the platform. Tasks are eligible to be assigned to workers who arrive on the platform before their deadline.
- *Range Constraint:* Workers are limited to providing services for tasks within their service radius.

Lemma 1: The CDTA problem is NP-hard.

Proof: We first introduce the 0-1 knapsack problem, which is known to be NP-hard [8]. Given a set H with n items, where each item $h_i{\in}H$ has a weight a_i and a value v_i. The objective of the 0-1 knapsack problem is to find a subset H^* of H with maximum $\sum_{h_i\in H^*} v_i$ subjected to $\sum_{h_i\in H^*} a_i{\leq}M$, where M represents the maximum weight capacity.

Then, we consider an instance of the CDTA problem where the set of tasks is R and the number of tasks is n (corresponding

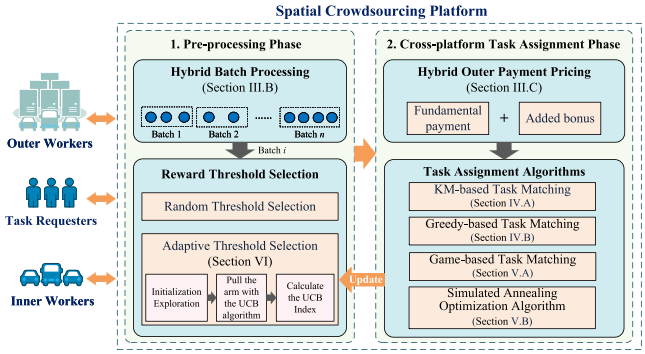


Fig. 3. Framework overview.

to the set H of the 0-1 knapsack problem). Each task $r_i \in R$ is associated with an inner or outer worker and the number of workers is sufficiently large (corresponding to the weight $a_i = 1$). Each task r_i is labeled with a value v_i, which is computed based on the function we defined. The computation is at least as difficult as v_i (a constant) in the 0-1 knapsack problem, so the difference will not make our problem any easier. Moreover, given a set of workers W and the worker number is M, the total number of workers assigned to all tasks will not exceed M (the same constraint as for the 0-1 knapsack problem). It can be seen that the goal of the CDTA problem is the same as that of the 0-1 backpack problem, i.e., to decide a task subset $R^* \subseteq R$ with maximum $\sum_{r_i \in R^*} v_i$ subjected to $\sum_{r_i \in R^*} a_i \leq M$.

In summary, based on the above mapping relationship, the 0-1 knapsack problem can be reduced to the CDTA problem. The 0-1 knapsack problem is NP-hard, thus the CDTA problem is also NP-hard. □

Remark: Information sharing among SC platforms may raise privacy concerns. Existing privacy-preserving approaches [9], [10], [11] can be applied to the CDTA problem, therefore, privacy security issues are not explored in depth in this study.

III. Solution Framework

In this section, we first illustrate the overall framework of this paper, then give the details of our proposed hybrid batch strategy and cross-platform incentive mechanism.

A. Overview

As shown in Fig. 3, our framework consists of two phases: the pre-processing phase and the cross-platform task assignment phase. In the pre-processing phase, the hybrid batch strategy is used to appropriately divide dynamically arriving tasks into different batches, addressing the issue of uneven temporal distribution of tasks. In each batch, a reward threshold is first chosen to determine which tasks are prioritized for assignment to inner workers. The random selection strategy generates a random threshold for each batch. The adaptive threshold selection strategy employs an MAB-based selection process, making full use of the information obtained from previous rounds. In the cross-platform task assignment phase, we design a hybrid outer payment pricing mechanism to incentivize the participation of outer workers. To ensure mutually beneficial outcomes for the inner platform, inner workers, and outer workers, several task assignment algorithms are developed to solve the CDTA problem. The allocation results will be fed back to the adaptive threshold selection module to dynamically optimize the choice of thresholds in each iteration.

Algorithm 1: Batch-Based Framework.

Input: Batch time threshold b_t, batch size threshold b_n
Output: Task assignment results of every batch
1 $R \leftarrow \emptyset, W_{in} \leftarrow \emptyset, W_{out} \leftarrow \emptyset$;
2 **while** *current time is within* b_t *and current batch size is less than* b_n **do**
3 $R \leftarrow$ Search for all the available tasks;
4 $W_{in} \leftarrow$ Search for all the available inner workers;
5 $W_{out} \leftarrow$ Search for all the available outer workers;
6 Use our proposed algorithms in Section IV or V to obtain the allocation result M;
7 **foreach** *task-worker pair* $(r_i, w_j) \in M$ **do**
8 Notify worker w_j to conduct task r_i;
9 Delete r_i and w_j;
10 Delete timeout tasks and workers from the next batch;
11 **if** $|R| = b_n$ **then**
12 Delete partial earliest appearing tasks from R;

B. Hybrid Batch-Based Processing Strategy

In dynamic task scenarios, batch processing based on the fixed time window size [12] is a common method. However, this strategy fails to address the issue of uneven task time distribution. During rush hours, there may be too many tasks in a batch, while during idle hours, there are very few tasks in a batch. Therefore, to efficiently handle dynamically arriving tasks, we propose a hybrid batch processing strategy to divide tasks into batches with suitable sizes.

Our hybrid batch processing strategy takes into account the task number and the time window. During idle time, it selects tasks within the batch based on the time window size, otherwise each batch needs to wait for a longer time. Considering the peak period, if the task number within a batch is not controlled, it will result in too many tasks being processed in a batch, affecting the efficiency of task allocation. To solve this problem, when the number of tasks in a batch reaches a threshold, the task assignment is executed without further waiting. This strategy fully improves the efficiency of task processing and reduces the user waiting time.

The batch-based framework aims to provide an overall solution to the CDTA problem.

In Algorithm 1, the dynamically arriving tasks are batched under the hybrid batching strategy (lines 2-12). Task assignment is performed for tasks and workers in a batch at the same time. For each task of the batch, available workers are filtered and a task assignment algorithm is selected to compute the assignment result. Workers or tasks are removed from the set in two cases: 1) they have already been assigned. 2) they will time out in the next batch. In the experiments, a phenomenon occurs where the latest arriving tasks are unable to be inserted into the task set due

to too many tasks that failed to be assigned in the current batch. This can also happen in strategies with fixed batch sizes and has a significant impact on efficiency. To address this problem, when $|R|{=}b_n$, we update the available task set R by removing some earliest arriving tasks.

C. Cross-Platform Incentive Mechanism

Existing pricing approaches [13] ignore the moving cost of outer workers to perform tasks, leading to an increased possibility of task rejections. Thus it is challenging to design a suitable incentive mechanism to encourage outer workers to perform tasks from other platforms. We first propose a metric for evaluating the shortage degree of workers in a given area, and then design a novel pricing model for outer workers.

Definition 8. (Shortage Degree of Workers): Given a spatial region Re and a time slot T, the shortage degree of workers is defined as

$$SD_{Re}^{T}{=}\begin{cases}0, & \text{if } n_w{>}k{\cdot}n_r \text{ or } n_r=0\\ 1, & \text{if } n_w{=}0\\ -\tanh\left(\ln(n_w/k{\cdot}n_r)\right), & \text{otherwise,}\end{cases} \tag{2}$$

where n_w and n_r represent the number of workers and tasks within the region Re, respectively, and k is the worker-task ratio customized by the platform.

In a given region, the shortage degree of workers (SD) can be calculated based on the ratio between the number of workers and tasks. The value of SD lies in the range of [0,1]. The parameter k is responsible for regulating the requirements on the relative number of workers and tasks. Consider a case where $k{=}1$: if the number of workers is smaller than the number of tasks, SD increases as the number of workers decreases; if $n_w{=}0$, SD will reach its maximum value of 1; if the number of workers is not less than number of tasks, SD is constant at 0, suggesting that the worker number is saturated.

To motivate outer workers to perform their allocated tasks with good performance and accept tasks in regions that lack workers, we present a reasonable incentive mechanism for calculating the outer payment.

Definition 9. (Outer Payment Pricing): Given an outer worker w who performs the task request r, the outer payment is computed as

$$p'(w,r)=\alpha{\cdot}p_r{\cdot}\rho_w+\beta{\cdot}\sum_{r'\in\mathbb{R}}p_{r'}{\cdot}\frac{SD_r}{\sum_{r'\in\mathbb{R}}SD_{r'}} \tag{3}$$

where $\mathbb{R}$ is the set of tasks which are assigned to outer workers and belong to the same batch as r. In addition, the parameters $\alpha{\in}[0,1]$ and $\beta{\in}[0,1]$ are used to control the dominance of the corresponding portions, and $\alpha{+}\beta{\leq}1$ is required.

Our incentive mechanism have two basic goals. On the one hand, the value of the task and the service quality of the worker are considered. On the other hand, to address the issue of imbalanced spatial distribution of tasks and workers, we consider the density of workers in the region where the task is located. Specifically, our pricing model is composed of two components. *(1) Fundamental payment.* It is determined by a combination of the reward from assigned tasks and the reputation score of the outer worker. Hence workers with better completion qualities receive higher payment. *(2) Added bonus.* The calculation is based on the worker shortage degree (SD) of the region where the outer worker is located. Specifically, we extract a portion of the total reward obtained by outer workers and allocate it to each outer worker according to SD. The higher the SD, the more extra bonus the outer worker receives. In this way, more outer workers within low worker density regions are incentivized to complete tasks, and pricing fairness is ensured.

Algorithm 2: KM-Based Algorithm.

Input: R, W_{in}, W_{out}
Output: Assignment result M
1 Let $M{\leftarrow}\emptyset$ and $U_M{=}0$;
2 Construct the weighted bipartite graph $G{=}(L,R,E)$;
3 **foreach** *edge* $e{\in}E$ **do**
4 **if** *worker* $w_e{\in}W_{in}$ **then**
5 Calculate the utility value $U_e{=}p_{r_e}{\times}w_e{.}\rho$;
6 **else if** *worker* $w_e{\in}W_{out}$ **then**
7 Calculate the utility value $U_e{=}(p_{r_e}{-}p'_{r_e}){\times}w_e{.}\rho$;
8 $M{\leftarrow}$The matching result with maximum utility $U_M{=}\sum_{e\in M}U_e$ is obtained using the KM algorithm [14];
9 **return** M.

IV. Global Task Assignment Algorithm

In this section, we propose a KM-based algorithm to get the optimal allocation results for inner and outer workers in each batch. To further improve the efficiency and address the problem of uneven spatio-temporal distribution of workers and tasks, we design a density-aware greedy algorithm.

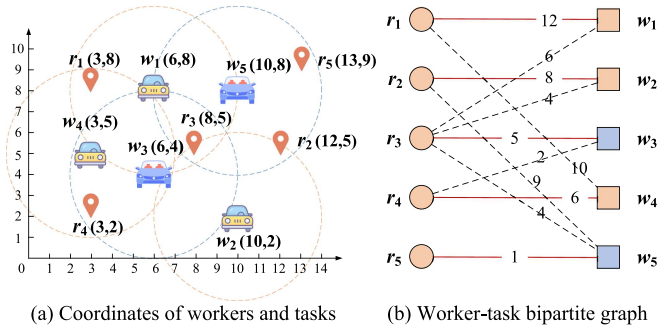
A. KM-Based Algorithm

Since the information of workers and tasks is unknown and the CDTA problem is NP-hard, the global optimal task assignment result cannot be obtained. However, we process the tasks in batches, so we can transform the CDTA problem to a Bipartite Maximum Weight Matching problem and use the Kuhn-Munkres (KM) Algorithm [14] to get the optimal allocation result for each batch.

In practice, many SC platforms offer outer workers with a fixed proportion of task payments as rewards (i.e., $\beta=0$ in Definition 9). Therefore, the outer payment of a worker w for performing a task r can be calculated as $p'(w,r)=\alpha\cdot p_r\cdot\rho_w$.

Algorithm Details: As shown in Algorithm 2, we construct a weighted bipartite graph $G{=}(R,W,E)$ based on the information of workers and tasks in the current batch, where R denotes tasks, W denotes workers, and the weights of the edges are utility values calculated according to Definition 6 (lines 2-7). An edge exists between a worker and a task only if the constraints in Definition 7 are satisfied. Then, a maximum weight bipartite graph matching is computed using the KM Algorithm [14] and we can get the matching result (line 8).

Example 2 (KM-based): In Fig. 4, orange coordinates indicate tasks, yellow cars indicate inner workers, and blue cars indicate outer workers. Thus, we can construct a



(a) Coordinates of workers and tasks (b) Worker-task bipartite graph

Fig. 4. An example of KM-based.

weighted bipartite graph in Fig. 4(b). The matching result is $M=\{(r_1,w_1),(r_2,w_2),(r_3,w_3),(r_4,w_4),(r_5,w_5)\}$ and the total utility is $U_M=12+8+5+6+1=32$.

Time Complexity: The KM-based algorithm has time complexity $O(\max(|R|^3,|W|^3)$, where $|R|$ denotes the task number and $|W|$ denotes the worker number.

B. Density-Aware Greedy Algorithm

The KM-based algorithm obtains the optimal solution within the batch, but has a high time complexity. In order to allocate tasks efficiently and overcome the issue of non-uniform distribution of tasks and workers in space and time, we design a density-aware greedy algorithm in this subsection.

It is important to give priority to inner workers when assigning tasks, otherwise it will lead to dissatisfaction of inner workers and worker turnover. First, some high-reward tasks are selected and allocated to inner workers to ensure the interests of the inner platform and inner workers. Then, to avoid the situation where no tasks are available in the areas of the unassigned inner workers in later batches, we further consider the task density in their regions. Finally, we assign all remaining tasks to outer workers with a greedy strategy.

Algorithm Details: The greedy algorithm includes three main stages. *1) Filter high-reward tasks for inner workers* (lines 2-7). First, we randomly select the threshold from a threshold set calculated from the maximum reward of tasks. Then, the task $r_{\max}$ with maximum reward $\max(p_r)$ and greater than the threshold are greedily selected over several iterations. In each iteration, the worker who satisfies four constraints in Definition 7 and has the highest reputation $\rho_{\widetilde{w}}$ is assigned to $r_{\max}$. *2) Reassignment of idle inner workers located in regions with low task density* (lines 9-14). We calculate the shortage degree $SD_{\overline{w}}$ based on Definition 8 for each idle inner worker $\overline{w}$ and a candidate task set $Cand_{\overline{w}}$. If $SD_{\overline{w}}=0$, it indicates that $\overline{w}$ is located at a region where the task number is low and the worker number is high. Hence $\overline{w}$ is assigned to the task $r_{\max}\in Cand_{\overline{w}}$ with the maximum reward. *3) Assign remaining tasks to outer workers* (lines 16-24). We choose the outer worker $\widetilde{w}$ who has the highest reputation $\rho_{\widetilde{w}}$ for the task $r_{\max}$ in every round. The outer revenue Rev_w of each matched outer worker is computed

Algorithm 3: Density-Aware Greedy (Greedy) Algorithm.

Input: R, W_{in}, W_{out}
Output: Assignment result M
1 **// Filter high-reward tasks for** W_{in}
2 $\phi\leftarrow\lceil\ln(max_{r\in R}(p_r)+1)\rceil$;
3 $g\leftarrow$Randomly select an integer from $\{0,...,\phi-1\}$;
4 $r_{max}\leftarrow \text{argmax}_{r\in R}\ p_r$;
5 **while** $max(p_r)\geq e^g$ *and* $W_{in}\neq\emptyset$ **do**
6 Greedily assign $\widetilde{w}\in W_{in}$ to r_{max} that can serve it with maximum $\rho_{\widetilde{w}}$ and remove them from the sets;
7 $r_{max}\leftarrow \text{argmax}_{r\in R}\ p_r$;
8 **//Filter highly competitive tasks for** W_{in}
9 **if** *idle inner worker set* $\overline{W}_{in}\neq\emptyset$ **then**
10 **foreach** $\overline{w}\in\overline{W}_{in}$ **do**
11 $SD(\overline{w}.l)$ is calculated according to Equation 2;
12 Calculate a candidate task set $Cand_{\overline{w}}$;
13 **if** $SD(\overline{w}.l)=0$ *and* $Cand_{\overline{w}}\neq\emptyset$ **then**
14 Assign $\overline{w}$ to the task with the highest reward and remove them from the sets;
15 **//Assign** W_{out} **to remaining tasks**
16 $r_{max}\leftarrow \text{argmax}_{r\in R}\ p_r$;
17 **while** $R\neq\emptyset$ *and* $W_{out}\neq\emptyset$ **do**
18 $A_{out}\leftarrow(\widetilde{w},r_{max})$ where $\widetilde{w}\in W_{out}$ can serve r_{max} with maximum $\rho_{\widetilde{w}}$;
19 $r_{max}\leftarrow \text{argmax}_{r\in R}\ p_r$;
20 **foreach** $(w,r)\in A_{out}$ **do**
21 Outer payment p' to w is calculated;
22 **if** $Rev_w>0$ **then**
23 Assign w to r and remove them from the sets;
24 **return** M

by Definitions 5 and 9. If $Rev_w>0$, then we assign w to r. After finishing the three stages, the allocation result M is obtained.

Example 3 (Greedy): Back to the running example in Fig. 4(a). The task rewards for r_1-r_5 are 12, 5, 10, 6, and 9, respectively. The reputation scores of workers w_1-w_5 are 0.8, 0.9, 0.7, 0.6, and 0.8, respectively. As $\max(p_r)=12$, we can calculate that $\phi=3$, and threshold set is $\{0,1,2\}$. Assume $g=2$, then tasks with rewards p_r more than $e^g\approx7.39$ are selected and allocated to inner workers. In Phase 1, task r_1 with the largest reward has two candidate outer workers w_1 and w_4. As $\rho_{w_1}>\rho_{w_4}$, r_1 is assigned to w_1. Likewise, we assign r_3 to w_2. In Phase 2, the parameter is $k=1$ and $SD_{w_4}=0$ holds (according to (2)), so r_4 is allocated to w_4. Finally, r_2 and r_5 are assigned to outer workers. Because r_5 has a higher reward, r_5 is allocated to w_5, while r_2 will go to the next batch waiting to be assigned. Assuming $\alpha=0.5$ and $\beta=0.3$, we get a final matching result $M=\{(w_1,r_1),(w_2,r_3),(w_4,r_4),(w_5,r_5)\}$ with total inner utility $12\times0.8+10\times0.9+6\times0.6+(9-3.6)\times0.8=26.52$.

Time Complexity: It costs $O(|W_{in}|\cdot|R|)$ in Phase 1, where $|R|$ and $|W_{in}|$ represent the task number and the inner worker number, respectively. The time complexity in Phase 2 is $O(|W_{in}|\cdot|R|)$. Besides, the time complexity in Phase 3 is $O((|W_{out}|\cdot|R|)$, where $|W_{out}|$ represents the outer worker number. Therefore, the time complexity of Algorithm 3 is $O((2\cdot|W_{in}|+|W_{out}|)\cdot|R|)$.

V. Improved Task Assignment Algorithm for Outer Workers

Our proposed KM-based and Greedy algorithms provide overall solutions for allocating inner and outer workers. It aims to maximise the utility of the inner platform but ignores the interests and willingness of outer workers. In order to realize a win-win goal, we further investigate a game-theoretic algorithm and a simulated annealing algorithm to raise the revenues of outer workers with no compromise on inner utility.

A. Game-Theoretic Algorithm

The task assignment process usually involves multiple outer workers simultaneously. Besides, the payment of each outer worker is associated with the task completion of other outer workers, which increases the complexity of task allocation. Hence the dynamic decision process among outer workers is modeled as a n-player strategic game in this section.

The methods proposed in [2] suppose that outer workers will perform allocated tasks as instructed by the inner platform. However, individuals in real life are selfish and strive to maximize their own interests. To solve this problem, we propose a game-theoretic model that considers the outer workers as the game players, the available task sets of the outer workers as their strategy sets, and the outer revenues of the outer workers as the utility function. We design an algorithm based on the best-response framework, where each game player iteratively adjusts its optimal strategy until it reaches a stable state where all outer workers are satisfied. This state is called the Nash equilibrium (NE) [15], i.e., there are no outer workers who can increase their revenues through unilaterally shifting from their allocated tasks to other tasks.

The formal definition of the outer worker task allocation game is given as follows.

$$G{=}\langle W_o, \{S_w\}_{w\in W_o}, \{U_w : \times_{w\in W_o} S_w\}_{w\in W_o} {\to} \mathbb{R}\rangle,$$

where W_o is the set of outer workers and we can obtain a strategy set $S_w{\subseteq}R$ for each outer worker $w{\in}W_o$ (a.k.a. game player), and $s_w{\in}S_w$ is a specific strategy for w, i.e., a task that w is eligible to perform. Given s_w and the strategies $\overline{s_w}$ of the other players, the utility function is to calculate the outer revenue of w. The goals of outer workers are to find the tasks that maximize their own utilities. According to Definition 5, the utility function of outer worker w can be calculated as

$$\begin{aligned} U_w(s_w, \overline{s_w}) &= p'(w, s_w) - c_w{\times}d(w, s_w) \\ &= \alpha \cdot p_{s_w} \cdot \rho_w + \beta \cdot \sum_{s_{\sim w}\in\mathbb{S}} p_{s_{\sim w}} \cdot \frac{SD_w}{\sum_{s_{\sim w}\in\mathbb{S}} SD_{\widetilde{w}}} \\ &\quad - c_w{\times}d(w, s_w) \end{aligned} \tag{4}$$

where the full set of currently selected strategies for all outer workers is $\mathbb{S}{=}\{s_w, \overline{s_w}\}$. It is clear that the utility of an outer worker is associated with his/her own strategy as well as with the strategies of other outer workers. As demonstrated in [16], all games with a finite number of players and strategies have a mixed Nash equilibrium. This only means a stable probability distribution in the strategy space, not a fixed play with a specific joint strategy space. In our problem, all workers need to either choose one of the available tasks or not perform any task, thus uncertainty cannot exist. We adopt pure strategies (i.e., deterministic strategies), which means that the outer worker w can choose a certain strategy from S_w with probability 1 while the probability of choosing other strategies from S_w is 0.

Next, we introduce the concept of Potential Game [17] and demonstrate that the CDTA game has pure NE where each player can choose a definite strategy.

Definition 10. (Potential Game [17]): A strategic game $G{=}\langle W_o, \{S_w\}_{w\in W_o}, \{U_w : \times_{w\in W_o} S_w\}_{w\in W_o} {\to} \mathbb{R}\rangle$ is a potential game, if there is a potential function $\Phi{:}\{S_w\}_{w\in W_o}{\to}\mathbb{R}$ such that for every player $w{\in}W_o$ and any two strategies $s_w, s'_w{\subseteq}S_w$ of player w satisfying

$$U_w(s_w, \overline{s_w}){-}U_w(s'_w, \overline{s_w}){=}\Phi_w(s_w, \overline{s_w}){-}\Phi_w(s'_w, \overline{s_w})$$

with any strategy profile $\overline{s_w}$ of other players.

Lemma 2: The CDTA Game is an Potential Game that has at least one pure NE.

Proof: Given s_w is a specific strategy of outer worker w and s'_w is any other response strategy of worker w. We define the potential function as

$$\begin{aligned} \Phi(S) = \alpha{\cdot} \sum_{w\in W_o} p_{s_w} \cdot \rho_w {+} \beta \cdot \sum_{w\in W_o} \frac{p_{s_w} \cdot SD_w}{\sum_{s_{\sim w}\in S} SD_{\widetilde{w}}} \\ - \sum_{w\in W_o} c_w{\times}d(w, s_w). \end{aligned} \tag{5}$$

Then, given the joint strategy $\overline{s_w}$ for workers other than w. We can calculate that

$$\begin{aligned} &\Phi_w(s_w, \overline{s_w}){-}\Phi_w(s'_w, \overline{s_w}) \\ &\quad = \alpha{\cdot}(p_{s_w}{-}p_{s'_w}){\cdot}\rho_w{+}\beta{\cdot}(p_{s_w}{-}p_{s'_w}){\cdot}\frac{SD_w}{\sum_{s_{\sim w}\in S} SD_{\widetilde{w}}} \\ &\qquad - c_w{\times}(d(w, s_w){-}d(w, s'_w)) \\ &\quad = U_w(s_w, \overline{s_w}){-}U_w(s'_w, \overline{s_w}). \end{aligned} \tag{6}$$

Therefore, the strategic game of the CDTA problem is a potential game and has at least one NE in pure strategy. □

Algorithm Details: In the best-response framework, potential games with finite strategies always converges to a pure NE [17]. We propose a game-theoretic algorithm based on this framework. First, Greedy is utilized to obtain the initial assignment result of inner workers and outer workers (line 1). Next, the algorithm iteratively chooses the best strategy of each outer worker which can maximize his/her utility defined in (4) according to the current joint strategy of the other outer workers. If the best response task of an outer worker w is occupied by another outer worker w', then their utilities are compared and the worker with the higher utility can occupy the task. This process is repeated until the NE is reached, i.e., no outer worker wants to switch from the existing strategy (lines 2-9). Finally, we get the allocation result M (line 10).

Algorithm 4: Game-Theoretic (Game) Algorithm.

Input: R, W_{in}, W_{out}
Output: Assignment result M
Use Greedy in Section IV to allocate inner workers and obtain an initial assignment of outer workers;
while *Not Nash Equilibrium* **do**
  foreach $w \in W_{out}$ **do**
    Find the best-response task $s_w^* \in S_w$ for w;
    $//s_w^*$ can be obtained from Equation (4)
    w follows the best strategy;
    if s_w^* *is occupied by another outer worker* w' **then**
      if w *has higher utility than* w' **then**
        w replaces w' and occupies s_w^*;
Obtain the matching result M according to NE;
return M

B. Simulated Annealing Optimization Algorithm

The Game algorithm aims to obtain stable allocation result that satisfy outer workers. The CDTA game is proven to have at least one pure NE (multiple pure NE may exist), and the result has a tendency to be trapped in a local optimum due to the strict constraints of the CDTA problem. The Simulated annealing (SA) algorithm searches for globally optimal solutions by simulating the annealing process of metal materials at a reduced temperature during metal heat processing. Unlike local search algorithms, it may accept non-optimal solutions in the solution space, which accelerates the search for the global optimal solution.

In the CDTA problem, the decision variables of the outer workers are discrete. The outer worker reaches a Nash equilibrium by continuously choosing the most satisfactory strategy. However, we propose an SA optimization algorithm that will accept other worse strategies with a certain probability. This approach enables escaping local optima to achieve better global results. In short, we first select a strategy, and if that strategy is better than the current strategy, we move to that strategy; if that strategy is worse than the current strategy, we calculate the success rate to decide whether to move.

Since each task is limited to be completed by one worker, if there is a conflict where multiple workers choose the same task, only the outer worker with higher utility will get the task. Assuming that the utility of the optimal strategy for worker w' is always lower than that of the other workers, w' tries to choose this strategy in each round, ignoring the other suboptimal strategies, and eventually has no task to complete. To avoid this, we define the preference value of outer worker w for each candidate task r as $Pre(w,r)$ and the initial value is set to 1. The selection probability of a worker w selecting a task r from the candidate task set $Cand_w$ depends on the preference value for r and is computed as follows:

$$Pro(w,r)=\frac{Pre(w,r)}{\sum_{r'\in Cand_w} Pre(w,r')}. \tag{7}$$

Next, a task is selected from the candidate tasks based on the selection probability distribution of w. The success rate $succ(w,r)$ of this task being occupied by w is calculated:

$$succ(w,r)=\begin{cases}1, & \text{if } U_w(r)\geq U_w(r^*) \text{ or } r^*=\emptyset\\ e^{\frac{U_w(r)-U_w(r^*)}{Tem(n_{round})}}, & \text{otherwise,}\end{cases} \tag{8}$$

where $U_w(r)$ denotes the utility of w if w chooses r (according to (4)), r^* denotes the currently occupied task of w (current strategy), and $Tem(n_{round})=\frac{1}{\log(n_{round}+1)}$ denotes the temperature value for the round. The higher the temperature, the higher the randomness of the results. As the number of rounds increases, the temperature gradually decreases and the results stabilize. From the (8), the strategy transfer must occur when the selected task r has a higher utility; otherwise, the transfer occurs with a certain probability.

In the process of competing for a task, if a worker who has already occupied that task is replaced by another worker, his/her preference value for that task will decrease. We get the new preference value for that outer worker by calculation:

$$Pre(w',r_{oc}) = Pre(w',r_{oc})\cdot e^{\frac{U_{w'}(r_{oc})-U_{w_{oc}}(r_{oc})}{Tem(n_{round})}}, \tag{9}$$

where w' denotes the worker whose task was replaced by another worker w_{oc}, and r_{oc} denotes the task that w' lost. In this way, outer workers no longer only consider the value of the task, but also whether they are competitive, thereby avoiding deadlocks where workers repeatedly compete for high-value tasks without reaching a stable allocation.

Algorithm Details: As shown in Algorithm 5, we first obtain an initial matching based on Greedy and an initial temperature (lines 1-2). In each round, for each outer worker w, the utility set $UCan_w$, preference value set Pre_w, and selection probability distribution Pro_w are calculated for candidate tasks (lines 5-7). Select a task r with probability distribution $Pro(w,r)$ and calculate the success rate for r. Then, generate a random number within [0,1] (lines 8-10). If the random number is not greater than the success rate and r is unoccupied, assign r to w. If r is already occupied by worker w' and the utility of w is higher than w', worker w replaces w' to get r, and $Pre(w',t)$ decreases (lines 11-16). Update n_{round} and Tem at the end of each round (line 17). Repeat the above steps until the maximum number of iterations $n_{\max}$ is reached.

Example 4 (SA): Assume that there are three outer workers w_1-w_3 and three tasks r_1-r_3, and the candidate set for each worker contains all three tasks. The utility sets of the workers for each task are $U_{w_1}=\{8,4,6\}$, $U_{w_2}=\{7,6,3\}$, and $U_{w_3}=\{5,3,7\}$ respectively. Workers have an initial preference value of 1 for each task, so the probability distributions are $\{\frac{1}{3},\frac{1}{3},\frac{1}{3}\}$. Calculate $Tem(1)$ for the first round as 1.44. In the first round, task r_3 is selected by sampling with the probability distribution of w_1. w_1 has no occupied task, so r_3 is occupied by w_1. Likewise, r_1 is occupied by w_2. Next, r_3 is selected by w_3. Although r_3 is occupied by w_1, w_3 has higher utility, so w_3 replaces w_1. Meanwhile, w_1 decreases the preference value $Pre(w_1,r_3)=1\cdot e^{\frac{6-7}{1.44}}\approx 0.5$ and $Pro(w_1,r_3)=\frac{0.5}{1+1+0.5}=0.2$. In the second round, Tem is updated to 0.91. First, w_1 chooses r_1 and w_2 is replaced by w_1. Therefore, updating $Pre(w_2,r_1)$ to 0.33. Next, based on the new probability distribution $Pro_{w_2}=\{0.14,0.43,0.43\}$, w_2

Algorithm 5: Simulated Annealing (SA) Algorithm.

Input: R, W_{in}, W_{out}
Output: Assignment result M
1 Use Greedy in Section IV.B to allocate inner workers and obtain an initial assignment of outer workers;
2 $n_{round} \leftarrow 1, Tem \leftarrow \frac{1}{\log(n_{round}+1)}$;
3 **while** $n_{round} \leq n_{max}$ **do**
4 **foreach** *outer worker* $w \in W_{out}$ **do**
5 $U_w \leftarrow$ calculate utility values of candidate tasks by Equation (4);
6 $Pre_w \leftarrow$ calculate preference values of candidate tasks;
7 $Pro_w \leftarrow$ calculate the selection probability of candidate tasks;
8 Select a task r with the selection probability Pro_w;
9 Calculate $succ(w, r)$ by Equation (8);
10 Randomly generate a real number $ran \in [0, 1]$;
11 **if** $ran \leq succ(w, r)$ **then**
12 **if** r *is not occupied* **then**
13 w occupies r;
14 **if** r *is occupied and* w *has higher utility than* w' **then**
15 w replaces w' and occupies r;
16 Update $Pre(w', r)$ based on Equation (9);
17 $n_{round} \leftarrow n_{round} + 1, Tem \leftarrow \frac{1}{\log(n_{round}+1)}$;
18 Obtain the assignment set M;
19 **return** M

selects and occupies r_2. w_3 selects r_2, but the success rate is 0.01, so occupancy fails. Through several rounds of iterations, a stable match is obtained as $\{(r_1, w_1), (r_2, w_2), (r_3, w_3)\}$. In the algorithm, each utility value changes as the workers' strategies change, and for simplicity it is treated as fixed here.

Time Complexity: The time complexity of the SA algorithm is $O(2|W_{in}| \cdot |R| + n_{\max} \cdot |W_{out}|)$.

VI. Adaptive Threshold Selection

In this section, we devise an adaptive threshold-based task assignment framework to further improve the results of task assignment.

A. Motivation and Overview

In order to ensure the utility of the inner platform and to avoid the loss of inner workers, we follow the principle of prioritizing the assignment of high-value tasks to inner workers. Therefore, a threshold must be given to determine which tasks are high-value tasks. For simplicity, this threshold is named Task Reward Threshold (TRT). The RamCOM algorithm [2] as well as our previously proposed algorithms use randomized thresholds to filter high-reward tasks to be allocated to inner workers (e.g., lines 2-3 in Algorithm 3). However, the assignment results under this strategy are unstable. When the selected threshold is small, most tasks are allocated to inner workers in priority, leading to a shortage of inner workers for high-value tasks that arrive later. When the selected threshold is large, most tasks are assigned to outer workers thus leaving many inner workers idle.

The random threshold selection strategy assumes that each threshold is chosen with equal probability. However, existing approaches ignore the potential to improve allocation results by adaptively selecting more appropriate thresholds using collected historical information. Since workers or tasks exhibit similar distributions over time, historical assignment results can provide guidance for setting thresholds. The multi-armed bandit (MAB) problem is an important part of reinforcement learning. It is formally expressed by assuming that there are K arms, each with an unknown expected reward, at each timestamp t, the agent chooses to pull an arm and receive a reward to maximize the cumulative reward. We transform the threshold selection process in the CDTA problem into the MAB problem. First, we need to discretize the threshold. Thus we can obtain a candidate TRT set and consider each TRT as an arm. The overall utility is maximised by selecting a TRT (i.e., pulling an arm) in each batch. The Upper Confidence Bound (UCB) algorithm is commonly used for MAB problems. In practice, we cannot predict which threshold should be chosen as the initial value, i.e., we face the problem of cold start. But the UCB algorithm can solve this problem by trying and exploring each threshold once.

B. Adaptive Threshold-Based Algorithm

We propose a UCB-based threshold selection algorithm, which finds a balance between exploration and exploitation. During the exploration phase, the inner platform tries different thresholds (TRTs) through several rounds of task allocation and learns the expected utility of TRTs. Over time, the platform adjusts the selection of TRT based on the collected data to improve overall utility.

Based on the maximum reward value for all tasks (predicted from historical data), we have the number of candidate thresholds $\phi = \lceil \ln(\max_{r \in R}(p_r)+1) \rceil$. Here, the threshold set with ϕ TRTs (i.e., arm set) is defined as $\mathcal{T} = \{\tau_1, \tau_2, \ldots, \tau_\phi\}$, where $\tau_n = e^{n-1}$, and $n \in \{1, 2, \ldots, \phi\}$. To prevent outcomes from disproportionately leaning toward any single party (such as the inner platform, inner workers, or outer workers), we define the overall utility that takes into account the interests of both the inner platform and the outer workers.

Definition 11. (Overall Utility): Given a task assignment set M, the overall utility U_M^{all} is the sum of the total utility of the inner platform and the total utility of the outer workers. In particular, according to Definition 6 and Definition 5, the overall utility of the task assignment set M is defined as

$$U_M^{all} = \omega \cdot U_M + (1-\omega) \cdot \sum_{(r, w_{out}) \in M} Rev(w_{out}, r), \quad (10)$$

where ω is used to regulate the importance of the two components.

Next, we use an incremental approach to calculate the utility mean for each threshold. The utility mean value $E_{\tau_n}^k$ of the threshold τ_n at the $k-th$ batch is expressed as:

$$E_{\tau_n}^k = \begin{cases} E_{\tau_n}^{k-1} + \frac{1}{k} \times [U_M^{all}(k) - E_{\tau_n}^{k-1}], & k>1, \\ U_M^{all}(k), & k=1, \end{cases} \quad (11)$$

Algorithm 6: Adaptive Threshold-Based (Adaptive) Algorithm.

Input: R, W_{in}, W_{out}, batch number k
Output: Assignment result M
1 Calculate the number of thresholds ϕ and the TRT set $\mathcal{T}$;
2 $E_{\mathcal{T}} \leftarrow 0$, $num_{\mathcal{T}} \leftarrow 0$;
3 **if** $k \leq \phi$ **then**
4 Select $\hat{\tau} = \tau_k$ as the threshold;
5 **else**
6 $E^k_{\tau_n}$ for each threshold is normalised to between $[0, 1]$;
7 Calculate $I^k_{\tau_n}$ by Equation (12) for each threshold and select the threshold $\hat{\tau}$ with maximum UCB index;
8 **if** *task reward* $p_r \geq$ *selected threshold* $\hat{\tau}$ **then**
9 Assign task r to an inner worker;
10 **else**
11 Assign task r to an outer worker;
12 Call the Algorithm 4 to perform task assignment;
13 Update the utility mean $E_{\hat{\tau}}$ and the number of pulls $num_{\hat{\tau}}$;
14 **return** M.

where $U^{all}_M(k)$ denotes the overall utility of the $k{-}th$ batch.

Therefore, we can obtain the UCB index for each threshold to evaluate the performance of the threshold as follows.

Definition 12. (UCB index): Given a threshold τ_n, its UCB index $I^k_{\tau_n}$ at the k-th batch is defined as

$$I^k_{\tau_n} = E^k_{\tau_n} + \sqrt{\frac{2 \cdot \ln N}{num_{\tau_n}}}, \tag{12}$$

where N is the total number of times all thresholds are pulled and num_{τ_n} is the number of times threshold τ_n is pulled.

The UCB index remains optimistic when facing uncertainty. The right part is a measure of the uncertainty in the value estimate for the k-th action, i.e., the exploration process. The fewer the number of attempts for a threshold, the higher the uncertainty, and hence the higher the probability of being selected. The left part is an assessment of the historical performance of the threshold, i.e., the exploitation process. Next, the platform determines the optimal bandit strategy and views it as a guideline for arm selection in each batch. The UCB indexs of TRTs change dynamically after each batch of task assignment is completed. If a TRT leads to a higher overall utility, it will have a rising probability of being chosen over time.

Algorithm Details: As shown in Algorithm 6, the maximum reward of the task is estimated based on the history and the candidate TRT set $\mathcal{T}$ (corresponding to the arm set) is obtained (line 1). Then, we initialize the mean value $E_{\mathcal{T}}$ and the number of pulled times $num_{\mathcal{T}}$ for each TRT to 0, respectively (line 2). We perform threshold selection in each batch and k represents the current batch count. To solve the cold-start problem, in the first ϕ rounds ($k \leq \phi$), we select each threshold in turn for one time to get their initial profile: the utility mean value and the UCB index (line 3-4). In subsequent rounds, to make the exploration and exploitation parts balanced, we first normalise the mean value $E^k_{\tau_n}$ of each candidate threshold. Then the UCB index is calculated for each TRT and the TRT with the largest UCB index is selected (lines 5-7). The selected threshold $\hat{\tau}$ is used to filter high-value tasks and low-value tasks and assign them to inner workers and outer workers, respectively (lines 8-12). After completing the processing of a batch, update $E_{\hat{\tau}}$ and $num_{\hat{\tau}}$ for the threshold $\hat{\tau}$ (line 13).

Example 5 (Adaptive): Assuming the maximum reward of tasks is 12, we have the number of candidate thresholds $\phi = \lceil \ln(12+1) \rceil = 3$ and the threshold set $\mathcal{T} = \{\tau_1, \tau_2, \tau_3\} = \{1, 2.7, 7.4\}$. During the initial cold start phase, τ_1, τ_2 and τ_3 are selected as TRT, respectively. In the first batch, assume that the overall utility generated by this batch after selecting τ_1 is 10, we update $E_{\tau_1} = 10$ and $num_{\tau_1} = 1$. Similarly, the profiles is updated for τ_2 and τ_3. Starting from batch 4, we calculate the UCB index of each TRT and select the one with the largest UCB index as the threshold. The UCB index of τ_1 is $I^4_{\tau_1} = 0.8 + \sqrt{\frac{2 \cdot \ln 3}{1}} \approx 2.3$ by (12) (0.8 is the normalized result). Assuming that τ_1 has the largest UCB index, τ_1 is selected again, and E_{τ_1} and num_{τ_1} are updated. Subsequently, the above steps are repeated in each batch to adaptively select the threshold.

Time Complexity: The overall computation complexity of Algorithm 6 is $O(\phi + 2 \cdot |W_{in}| \cdot |R| + n_{\max} \cdot |W_{out}|)$.

TABLE III
EXPERIMENTAL SETTINGS ON REAL DATA

Dataset	$\lvert R \rvert$	$\lvert W_{in} \rvert$ / $\lvert W_{out} \rvert$	c_w	due	(b_t, b_n)
Chicago	71,334	19,019 / 13,095	3	900	(300, 200)
NYC	347,569	35,935 / 35,935	2	40	(20, 500)

VII. EXPERIMENTS

A. Dataset and Setup

Datasets: We evaluate the proposed algorithm using synthetic dataset as well as two real datasets named the Chicago dataset[5] and the NYC dataset[6]. The Chicago dataset contains the Chicago taxi trips in January 2020, and the NYC dataset contains the New York City taxi trips in June 2016. The drop-off sites of drivers are considered the locations of workers; the pick-up sites of passengers are considered the locations of tasks. In the Chicago dataset, we extract the inner and outer workers from the City Service and Blue Ribbon Taxi Association, respectively. In the NYC dataset, the inner and outer workers are from Yellow Taxi and Green Taxi. As the task requester of Green Taxi can be dropped off within the same region as the service range of Yellow Taxi, we take the drop-off location as the worker's location and assume that the worker of Green Taxi is allowed to serve the request of Yellow Taxi. We also create synthetic datasets by randomly sampling the NYC dataset and generating workers following [2].

Tables III and IV show the parameter settings, where the default values are marked in bold. The task rewards of the synthetic datasets follow a normal distribution. In this process, because μ_p has more influence on performance [18], the parameter σ_p is set to 3.75 and the value of μ_p is varied. Moreover, the most

[5]https://data.cityofchicago.org/Transportation/Taxi-Trips-2020/r2u4-wwk3
[6]https://www.nyc.gov/site/tlc/about/tlc-trip-record-data.page

TABLE IV
EXPERIMENTAL SETTINGS ON SYNTHETIC DATA

Parameters	Setting
$\|R\|$	500, 1000, **2500**, 5k, 10k, 20k, 50k, 100k
$\|W\|$	100, 200, **500**, 1000, 2500, 5k, 10k, 20k
rad	1.0, 1.5, **2.0**, 2.5, 3.0
c_w	1, 2, **3**, 4, 5
ddl	40, 60, **80**, 100, 120
μ_p	**Normal:** μ = [2, 5, **10**, 20, 50], σ = 3.75
b_t	10, 20, **40**, 80, 160
b_n	25, 50, **100**, 200, 400
(α, β)	$(0.4, 0.4)$, $\mathbf{(0.5, 0.3)}$, $(0.6, 0.2)$, $(0.7, 0.2)$, $(0.8, 0)$

important part for the inner platform is the inner utility, so the parameter ω in UCB is set to 0.7.

Approaches and Measurements: We compare the proposed KM-based, Greedy, Game, SA and UCB with a multi-platform cooperative algorithm (RamCOM [2]) and an adaptive batch processing method (RQL [19]) using four metrics: 1) Inner utility: total utility of inner platform; 2) Outer revenue: total revenue received by outer workers; 3) NumOfTask: number of tasks completed; 4) Running time.

We partition the spatial region into 10×10 grids and compute the shortage degree of workers (SD) for each grid. Since RamCOM uses an online processing solution and a dynamic pricing approach [13], this algorithm is not involved in the evaluation and comparison of batch thresholds b_t and b_n, as well as pricing parameter (α, β) . In all experiments, we repeatedly test 10 times and record the average outcomes. All algorithms are run on an Intel(R) Xeon(R) Gold 5218R CPU @2.10 GHz with 255 GB GB RAM in C++.

B. Experiments on Real Datasets

We demonstrate the performance of the seven algorithms using two real datasets and the experimental results are shown in Tables V and VI. The bolded values in each column of these two tables indicate the best-performing value among all algorithms for the corresponding metric. It is obvious that KM-based yields the maximum inner utility, overall utility and the maximum NumOfTask over two real datasets, since it obtains the optimum allocation in every batch. Among all algorithms, SA has the largest outer revenue. This is because it avoids falling into a local optimum by the simulated annealing optimization process of the CDTA game. While KM-based outperforms the other algorithms we propose in terms of inner utility, the overall utility differences among them are greatly reduced by the cross-platform incentive mechanism and optimized outer worker allocation approaches. Besides, Greedy has the shortest running time due to the batch processing strategy and greedy strategy.

It is worth noting that Greedy, Game, SA and UCB have fast response speeds on both real datasets, with running times remaining under 1 second on the Chicago dataset and under 4 seconds on the NYC dataset, obviously outperforming the other algorithms. Except for KM-based, UCB yields the maximum inner utility, overall utility and NumOfTask among our proposed algorithms and is very close to KM-based, indicating that it improves efficiency while guaranteeing the quality of the results.

C. Experiments on Synthetic Datasets

In this section, we run experiments on synthetic datasets to evaluate the performance of the seven algorithms.

1) Effect of $|R|$: As illustrated in Fig. 5, the five metrics all increase as $|R|$ increases. Obviously, the reason for this is the increase in the number of tasks completed, which brings more revenues to both inner and outer workers, as well as an increase in processing time. For inner utility, outer revenue, overall utility and NumOfTask, all five of our proposed algorithms outperform RamCOM and RQL, but our proposed algorithms require more running time. Among all algorithms, KM-based obtains the largest inner utility and overall utility which demonstrates the superiority of the optimal task assignment algorithm. Game, SA and UCB all generate high outer revenue and assign more tasks.

2) Effect of $|W|$: As shown in Fig. 6, the inner utility, overall utility, NumOfTask, and the running time exhibits a similar increasing trend as $|W|$ increases in most cases. However, the outer revenue is not simply positively correlated with $|W|$ for our proposed algorithms. The default task number is 2,500. When $|W|$ is greater than 2,500 and gets larger, competition among workers becomes more intense and high-reward tasks are allocated in priority to inner workers, leading to a decline in outer revenue. After $|W|$ is larger than 2,500, the running time of Greedy, Game, SA, and UCB begins to decrease as well, due to the decrease in the time to perform task assignment for outer workers. Compared to RamCOM and RQL, our proposed algorithms obtain higher inner utility, more outer revenue, overall utility, and NumOfTask, but require more runtime in most cases. Among all the algorithms, KM-based has the largest inner utility, and Game generates the most outer revenue and NumOfTask for the most part. However, the overall utility of our five proposed algorithms is very close.

3) Effect of the worker service radius rad: In Fig. 7, the inner utility, overall utility and NumOfTask of each algorithm increase with the growth of rad, while the running time has little change among different algorithms. Outer revenues of Greedy, Game, SA and UCB first decrease and then increase with increasing rad, while the other algorithms show a decreasing trend. This is because a larger rad means that the task has more candidate outer workers competing for it. Similarly, the inner utility and the overall utility of KM-based are maximized. Also we can see that SA performs best in terms of outer revenue because SA avoids being trapped in local optimum. RQL, KM-based, Greedy, Game, SA and UCB can achieve much higher NumOfTask than RamCOM. RamCOM runs the fastest, while UCB runs the slowest.

4) Effect of the unit cost c_w: We proceed to consider the effect of c_w by varying it from 1 to 5 and the results are shown in Fig. 8. With an increase of c_w, the out revenues of all algorithms decrease. Because the higher the cost of mobility, the lower the revenue for the outer workers, the fewer outer workers are willing to participate in the task assignment. However, c_w has no significant effect on the inner utility, overall utility, NumOfTask, and running time of all algorithms. Among all the algorithms, KM-based generates the largest inner utility and overall utility and needs the least running time. Game, SA and

TABLE V
RESULTS ON THE CHICAGO DATASET

Methods	Inner Utility ($\times 10^5$)	Outer Revenue($\times 10^5$)	Overall Utility($\times 10^5$)	NumOfTask	Running Time
RamCOM	2.073	0.194	1.509	17,974	10s
RQL	4.043	0.633	3.020	29,750	9.6s
KM-based	**4.237***	0.536	**3.127***	**29,840***	5.3s
Greedy	3.524	1.077	2.790	26,239	**0.37s***
Game	3.641	1.215	2.913	28,016	0.54s
SA	3.607	**1.228***	2.893	27,552	0.42s
UCB	3.888	1.139	3.063	29,711	0.62s

TABLE VI
RESULTS ON THE NYC DATASET

Methods	Inner Utility ($\times 10^6$)	Outer Revenue($\times 10^5$)	Overall Utility($\times 10^5$)	NumOfTask	Running Time
RamCOM	0.462	0.337	3.335	33,726	124s
RQL	1.191	1.468	8.777	53,799	35s
KM-based	**1.218***	1.577	**8.999***	**55,047***	7s
Greedy	1.019	3.037	8.044	50,204	**2s***
Game	1.001	3.108	7.939	49,704	3s
SA	0.983	**3.113***	7.815	49,026	3s
UCB	1.113	3.009	8.694	54,665	4s

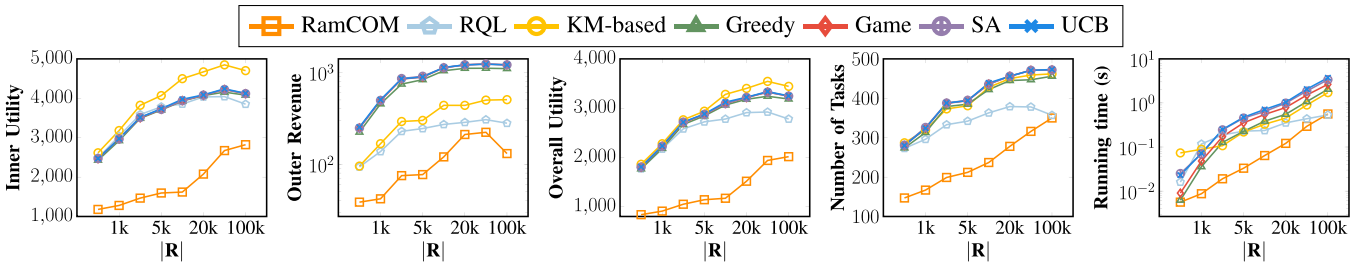


Fig. 5. Effect of the task number $|R|$.

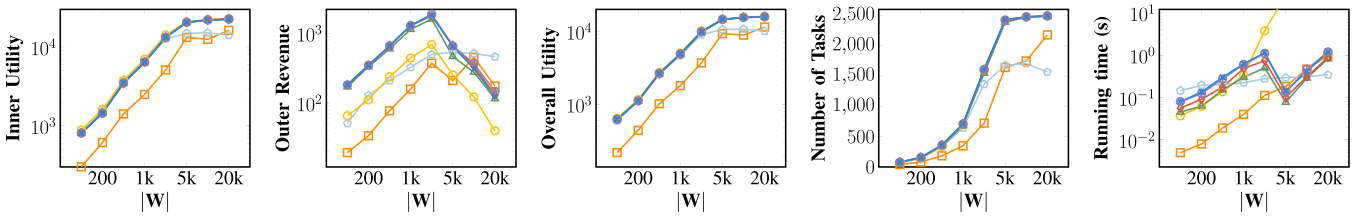


Fig. 6. Effect of the worker number $|W|$.

UCB produce higher outer revenues and can assign more tasks. For Game, outer workers are able to autonomously select the tasks that benefit the most, while the current platform can get higher utility. UCB requires the highest running time because it requires calculating the optimal threshold at each batch. Note that since RamCOM ignores the travel costs of outer workers, when $c_w \geq 4$, the outer revenue is less than 0.

5) Effect of batch time threshold b_t: It can be seen from Fig. 9 that changing b_t will have no impact on RQL since there is no batch time threshold set for RQL. All algorithms except RQL show a trend of increasing and then rapidly decreasing in terms of inner utility, outer revenue, overall utility, and NumOfTask. It is mainly due to the fact that the larger the value of b_t, the more tasks can be processed simultaneously, which leads to a better assignment result. But when b_t is too large, the number of workers in a batch can be much larger than b_n, and repeated processing of difficult-to-assign workers reduces the effectiveness and efficiency of assignment. Not surprisingly, the running time of our proposed algorithms increases as the batch time threshold b_t grows. Among our proposed algorithms, the inner utility and overall utility of KM-based is the largest, UCB obtains the highest outer revenue and the most NumOfTask, while it requires the most running time.

6) Effect of batch size threshold b_n: From Fig. 10, we observe that the five metrics of all algorithms increase with b_n in most cases. Among all the algorithms, KM-based still has the largest inner utility and overall utility, and Game, SA and UCB have higher outer revenue and NumOfTask. The reason for this is

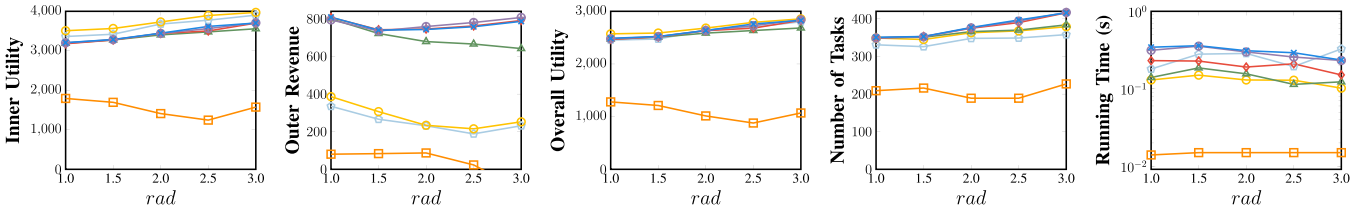


Fig. 7. Effect of the worker service radius rad.

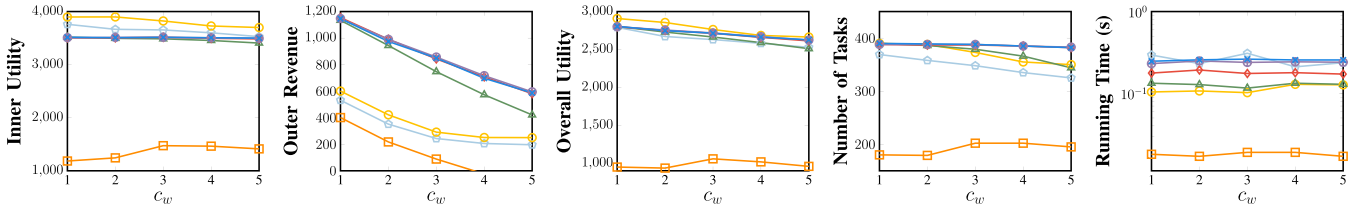


Fig. 8. Effect of the unit cost of worker c_w.

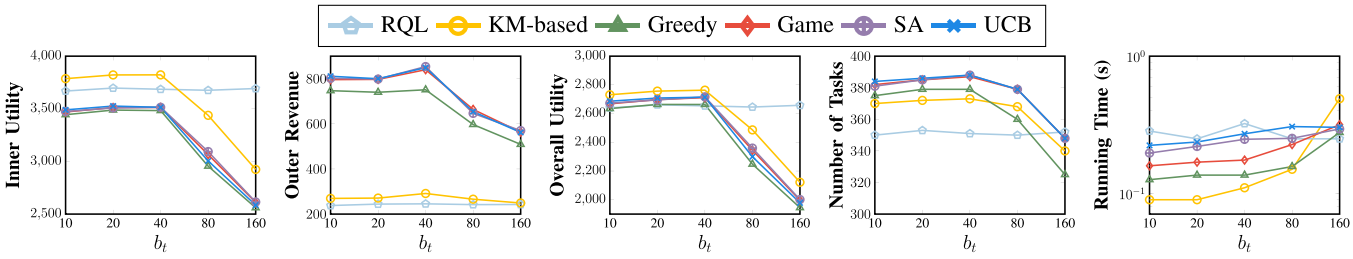


Fig. 9. Effect of the batch time threshold b_t.

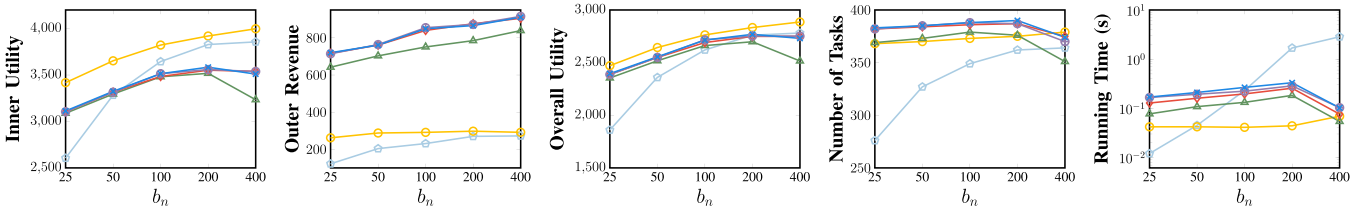


Fig. 10. Effect of the batch size threshold b_n.

that Greedy, Game, SA and UCB assign fewer tasks to internal workers when b_n is large. However, since Game, SA and UCB take into account the interests of outer workers, it alleviates the decrease in inner utility to some extent. It is clear that RQL has the fastest growing running time and KM-based has the shortest running time.

7) Effect of pricing parameters (α, β)*:* Both RQL and KM-based ignore the extra bonuses for outer workers, so only α has an impact on their results. In Fig. 11, as α increases, the inner utility of RQL and KM-based decreases because the inner platform needs to pay more to outer workers after they complete tasks. Accordingly, the outer revenue and NumOfTask tend to increase for RQL and KM-based. As a result, their overall utility declines at a relatively slower rate. When α=0.7 and β=0.2, Greedy, Game, SA and UCB have the lowest inner utility and overall utility, but significantly higher outer revenue. From Fig. 11, the pricing parameters have no significant influence on the running time. Among all the algorithms, the inner utility and overall utility of KM-based is the largest in most cases, Game, SA, and UCB have higher outer revenue, while UCB has the largest NumOfTask.

8) Effect of the task reward μ_p *and the task deadline* ddl*:* Due to space limitations, we omit the figures of the experimental results that vary the parameters μ_p and ddl. By varying the task reward μ_p, the inner utility, outer revenue, and overall utility of all algorithms increase as μ_p increases, as the higher the value of the task, the higher the overall benefit to the platform and workers. In particular, KM-based has the largest inner utility and overall utility, while SA has the highest outer revenue. RamCOM still requires the least runtime, but UCB costs the most runtime.

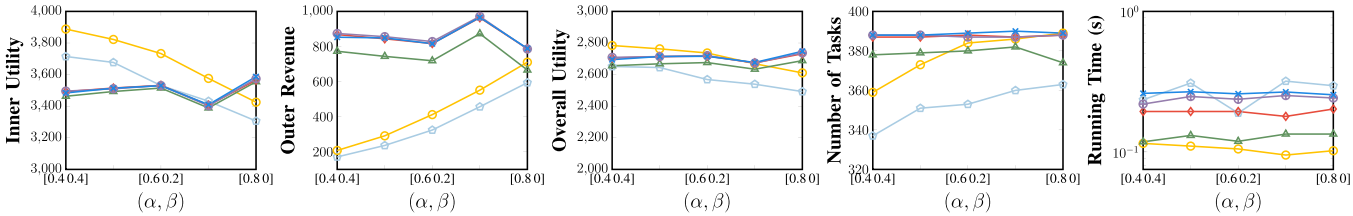


Fig. 11. Effect of the outer payment parameter (α, β).

As the task deadline *ddl* becomes longer, the inner utility and overall utility of the six algorithms grows, which is reasonable because the tasks have a higher probability of being successful in assignment. Among all the algorithms, KM-based produces the highest inner utility and overall utility. With respect to outer revenue and NumOfTask, Game, SA and UCB are close and outperform the other methods. In addition, when the deadline increases, the running time of the six algorithms does not change significantly.

D. Summary

Our main experimental findings are as follows.

1) KM-based has the highest inner utility and overall utility and SA produces the highest outer revenue compared to other algorithms in most cases. Among our proposed algorithms, KM-based is the most efficient on small-scale datasets, but requires much higher runtime on large-scale datasets.
2) Our proposed algorithms outperform RamCOM in terms of inner utility, outer revenue, overall utility, and NumOfTask. For most cases, KM-based has higher inner utility, overall utility, and NumOfTask than RQL, and the outer revenue of Greedy, Game, SA and UCB are both larger than RQL.
3) The running time of the proposed algorithms is much less than RamCOM and RQL over real datasets. Greedy consumes the least amount of runtime on real datasets and RamCOM costs the least runtime on synthetic datasets. Importantly, the runtime of our proposed algorithms are all less than 1 second, which can satisfy the practical needs in most applications.
4) On synthetic datasets, Game, SA, and UCB perform similarly. However, on real datasets, UCB has the highest inner utility, overall utility and NumOfTask, and SA has the highest outer revenue. When the number of workers/tasks within a batch is small, the solution space is small, thus the probability of falling into a local optimum in SA is small. When the batch number is small, UCB may still be in the exploration phase, with higher uncertainty in the results. Therefore, SA and UCB show less significant performance on synthetic datasets. However, on real datasets, both SA and UCB can obtain better results through iteration or adaptive adjustment.

Our proposed algorithms are applicable to different scenarios in spatial crowdsourcing.

Small-scale Task Scenarios: For cross-platform SC in small-scale task scenarios (e.g., photo-taking tasks, renovation tasks), KM-based can be chosen if faster response time and higher inner utility are required, and Greedy, Game, SA, or UCB can be chosen if the inner platform is expected to be sustainable and to attract more outer workers to participate and thus achieve a higher number of task matches.

Large-scale Task Scenarios: For cross-platform SC in large-scale task scenarios (e.g., ride-hailing services), if real-time feedback of matching results is not required, KM-based can be chosen for platforms with high inner utility requirements, but it only considers the interests of inner platform and may ignore the benefits of outer workers. Greedy, Game, SA, and UCB are suitable for real-time demanding scenarios and guarantee the interests of both the inner platform and outer workers. In these algorithms, SA can achieve higher outer revenue to motivate outer workers, and UCB has higher overall utility and NumOfTask to achieve the global optimum.

VIII. Related Work

In this section, we discuss some studies in spatial crowdsourcing (SC), mainly on task assignment, incentive mechanism, and Multi-Armed Bandit (MAB).

Task Assignment: As a core issue in SC [20], most existing studies focus on task assignment [21], [22], [23], [24], [25], [26], [27], [28], [29], [30]. The task processing modes in spatial crowdsourcing are divided into two types: real-time mode [31], [32], [33] and batch-based mode [34], [35], [36], [37], [38], [39]. In this paper, we adopt a hybrid batch processing framework for batch processing of tasks and workers. Wang et al. [19] designed an adaptive batch processing framework and a Restricted Q-learning (RQL) algorithm. However, when processing tasks of large scale, RQL requires a large amount of time to search and update the Q-table.

Game theory is applicable to deal with many types of task allocation problems in SC [34], [37], [40], [41], [42], [43], [44]. The problems in [40], [41], [42], [43] require multiple workers to collaborate on a task. Studies in [34], [37] focus on task assignment with dependency constraints, aiming to maximize total payoff [34] and the completed task number [37]. The issue of fairness is a key concern in SC. In order to ensure payoff fairness among workers, Zhao et al. [44] formulated the task allocation process as a multi-party game and proposed an improved evolutionary game-theoretic algorithm. Xie et al. [45] et al. adopted a game-theoretic based approach to maximize

overall user satisfaction in complex task scenarios. Li et al. [46] solved the fairness issue by the multi-round n-player strategy game model and proposed a multi-round task assignment algorithm as well as a series of optimization strategies, which aims to maximize total revenue for workers while ensuring fairness constraints.

To our best knowledge, several studies on cross-platform collaboration in SC has been conducted recently. Cheng et al. [2] first investigated the Cross Online Matching (COM) problem and proposed two algorithms, DemCOM and RamCOM. Peng et al. [3] investigated the task assignment methods to achieve the goal of maximising overall sensing quality in multi-platform scenarios where the sensing quality of each user is unknown. Li et al. [47] designed an autonomy and coordination task assignment framework with the goal of improving revenue and ensuring fairness. Yang et al. [4] employed a third-party platform to globally match public tasks across all platforms. Privacy protection is a core issue to ensure the sustainability and practicality of cross-platform SC scenario. Wang et al. [9] studied the cross-platform order scheduling problem and designed a Federated Learning-to-Dispatch framework. Zhong et al. [10] designed a federated learning framework for cross-platform task assignment, which combines personalized location preference learning and efficient algorithms to address privacy concerns and data heterogeneity. The work in [11], [48] investigates how to optimize cross-platform task allocation by considering task preferences of workers while preserving privacy. Each platform trains the personalized worker preference model using local historical data, and then the central server performs preference-driven task assignment. These existing approaches effectively preserve privacy in cross-platform SC, thus our study aims to improve the effectiveness and efficiency of the task assignment process. Cui et al. [49] adopted Cooperative Global Path Planning framework to solve the traffic congestion problem caused by the isolation of platforms. Cheng et al. [50] proposed a cross online ride-sharing framework with a worker selection algorithm based on direction prediction.

Incentive Mechanism: Incentive mechanisms are designed to encourage workers to engage in SC. Hu et al. [51] proposed an incentive mechanism based on reverse and multi-attribute auctions in mobile crowdsourcing. Liu et al. [52] formulated a price adjustment function and two budget allocation algorithms to solve the imbalanced sensory data distribution problem. Zhao et al. [53] investigated the real-time auction problem. Xu et al. [54] designed an incentive mechanism based on the impact of social networks and the utility function. Zhang et al. [55] addressed the problem of reliable task allocation. Tong et al. [13] considered the spatio-temporal distribution of tasks and workers as well as the worker mobility during the dynamic pricing. Wang et al. [56] studied a novel dynamic incentive mechanism to recruit workers by utilizing social networks.

Multi-Armed Bandit (MAB) Model: Reinforcement learning, which learns optimal decision-making strategies by interacting with the environment, shows wide application potential in SC. Li et al. [57] proposed an adaptive sliding window decision algorithm based on deep reinforcement learning to solve the city express delivery problem, which yields better results compared to manually adjusting batch size. The MAB model illustrates the sequential decision-making process under incomplete information [58]. Some previous studies have applied this concept to related fields [59], [60]. Yang et al. [61] evaluated worker performance information through a UCB-based online learning algorithm. Gao et al. [62] designed a MAB-based reinforced worker selection framework and model the quality of workers through bias and variance to obtain reliable workers. Peng et al. [63] investigated the multi-platform task allocation problem with unknown qualities of users and proposed two online MAB-based user selection algorithms. However, their solution ignores the priority of inner workers (resulting in the loss of inner workers) and the interests of outer workers. Most existing MAB-based algorithms in SC are used for worker selection and cannot be applied to our threshold selection problem.

IX. Conclusion

We formalize and provide solutions to a novel SC problem, named Cross Dynamic Task Assignment (CDTA). We propose a hybrid batch processing framework and a pricing strategy applicable to outer workers. In order to realize the global task allocation for inner workers and outer workers, we design the KM-based algorithm and the Greedy algorithm. To guarantee the benefits of outer workers, we propose the Game algorithm and the SA algorithm. An adaptive threshold selection method is developed to improve the overall utility. We validate the effectiveness and efficiency of our proposed algorithm by conducting extensive experiments on real and synthetic datasets. Our algorithms are adaptable to cross-platform task assignment scenarios with different requirements.

References

[1] T. Ren, X. Zhou, K. Li, Y. Gao, J. Zhang, and K. Li, "Efficient cross dynamic task assignment in spatial crowdsourcing," in *Proc. IEEE 39th Int. Conf. Data Eng.*, 2023, pp. 1420–1432.
[2] Y. Cheng, B. Li, X. Zhou, Y. Yuan, G. Wang, and L. Chen, "Real-time cross online matching in spatial crowdsourcing," in *Proc. IEEE 36th Int. Conf. Data Eng.*, 2020, pp. 1–12.
[3] S. Peng, B. Zhang, Y. Yan, and C. Li, "A multi-platform cooperation based task assignment mechanism for mobile crowdsensing," *IEEE Internet Things J.*, vol. 10, no. 19, pp. 16881–16894, Oct. 2023.
[4] Y. Yang, Y. Cheng, Y. Yang, Y. Yuan, and G. Wang, "Batch-based cooperative task assignment in spatial crowdsourcing," in *Proc. IEEE 39th Int. Conf. Data Eng.*, 2023, pp. 1180–1192.
[5] D. Zhao, X.-Y. Li, and H. Ma, "How to crowdsource tasks truthfully without sacrificing utility: Online incentive mechanisms with budget constraint," in *Proc. IEEE Conf. Comput. Commun.*, 2014, pp. 1213–1221.
[6] Z. Liu, K. Li, X. Zhou, N. Zhu, and K. Li, "Incentive mechanisms for crowdsensing: Motivating users to preprocess data for the crowdsourcer," *ACM Trans. Sensor Netw.*, vol. 16, no. 4, pp. 1–24, 2020.
[7] Y. Tong, J. She, B. Ding, L. Wang, and L. Chen, "Online mobile micro-task allocation in spatial crowdsourcing," in *Proc. IEEE 32th Int. Conf. Data Eng.*, 2016, pp. 49–60.
[8] V. V. Vazirani, *Approximation Algorithms*. Berlin, Germany: Springer, 2013.
[9] Y. Wang et al., "Fed-LTD: Towards cross-platform ride hailing via federated learning to dispatch," in *Proc. 28th ACM SIGKDD Conf. Knowl. Discov. Data Mining*, 2022, pp. 4079–4089.
[10] X. Zhong, H. Miao, D. Qiu, Y. Zhao, and K. Zheng, "Personalized location-preference learning for federated task assignment in spatial crowdsourcing," in *Proc. Conf. Inf. Knowl. Manage.*, 2023, pp. 3534–3543.
[11] H. Miao et al., "Task assignment with efficient federated preference learning in spatial crowdsourcing," *IEEE Trans. Knowl. Data Eng.*, vol. 36, no. 4, pp. 1800–1814, Apr. 2024.

[12] P. Cheng, L. Chen, and J. Ye, "Cooperation-aware task assignment in spatial crowdsourcing," in *Proc. IEEE 35th Int. Conf. Data Eng.*, 2019, pp. 1442–1453.

[13] Y. Tong, L. Wang, Z. Zhou, L. Chen, B. Du, and J. Ye, "Dynamic pricing in spatial crowdsourcing: A matching-based approach," in *Proc. Int. Conf. Manage. Data*, 2018, pp. 773–788.

[14] H. W. Kuhn, "The Hungarian method for the assignment problem," *Nav. Res. Logistics Quart.*, vol. 2, no. 1/2, pp. 83–97, 1955.

[15] E. Maskin, "Nash equilibrium and welfare optimality," *Rev. Econ. Stud.*, vol. 66, no. 1, pp. 23–38, 1999.

[16] J. F. Nash Jr., "Equilibrium points in n-person games," *Proc. Nat. Acad. Sci. USA*, vol. 36, no. 1, pp. 48–49, 1950.

[17] D. Monderer and L. S. Shapley, "Potential games," *Games Econ. Behav.*, vol. 14, pp. 124–143, 1996.

[18] Y. Tong, Y. Zeng, B. Ding, L. Wang, and L. Chen, "Two-sided online micro-task assignment in spatial crowdsourcing," *IEEE Trans. Knowl. Data Eng.*, vol. 33, no. 5, pp. 2295–2309, May 2021.

[19] Y. Wang, Y. Tong, C. Long, P. Xu, K. Xu, and W. Lv, "Adaptive dynamic bipartite graph matching: A reinforcement learning approach," in *Proc. IEEE 35th Int. Conf. Data Eng.*, 2019, pp. 1478–1489.

[20] Y. Tong, Z. Zhou, Y. Zeng, L. Chen, and C. Shahabi, "Spatial crowdsourcing: A survey," *VLDB J.*, vol. 29, no. 1, pp. 217–250, 2020.

[21] J. Xia, Y. Zhao, G. Liu, J. Xu, M. Zhang, and K. Zheng, "Profit-driven task assignment in spatial crowdsourcing," in *Proc. Int. Joint Conf. Artif. Intell.*, 2019, pp. 1914–1920.

[22] H. To, C. Shahabi, and L. Kazemi, "A server-assigned spatial crowdsourcing framework," *ACM Trans. Spatial Algorithms Syst.*, vol. 1, no. 1, pp. 1–28, 2015.

[23] Y. Tong, J. She, B. Ding, L. Chen, T. Wo, and K. Xu, "Online minimum matching in real-time spatial data: Experiments and analysis," *Proc. VLDB Endowment*, vol. 9, no. 12, pp. 1053–1064, 2016.

[24] P. Cheng, X. Lian, L. Chen, and C. Shahabi, "Prediction-based task assignment in spatial crowdsourcing," in *Proc. IEEE 33th Int. Conf. Data Eng.*, 2017, pp. 997–1008.

[25] Z. Wang, Y. Zhao, X. Chen, and K. Zheng, "Task assignment with worker churn prediction in spatial crowdsourcing," in *Proc. Conf. Inf. Knowl. Manage.*, 2021, pp. 2070–2079.

[26] Y. Li, Y. Zhao, and K. Zheng, "Preference-aware group task assignment in spatial crowdsourcing: A mutual information-based approach," in *Proc. IEEE Conf. Data Mining*, 2021, pp. 350–359.

[27] X. Chen, Y. Zhao, K. Zheng, B. Yang, and C. S. Jensen, "Influence-aware task assignment in spatial crowdsourcing," in *Proc. IEEE 38th Int. Conf. Data Eng.*, 2022, pp. 2141–2153.

[28] B. Zhao, P. Xu, Y. Shi, Y. Tong, Z. Zhou, and Y. Zeng, "Preference-aware task assignment in on-demand taxi dispatching: An online stable matching approach," in *Proc. AAAI Conf. Artif. Intell.*, 2019, pp. 2245–2252.

[29] Y. Zhao, J. Liu, Y. Li, D. Zhang, C. S. Jensen, and K. Zheng, "Preference-aware group task assignment in spatial crowdsourcing: Effectiveness and efficiency," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 10, pp. 10722–10734, Oct. 2023.

[30] Y. Zhao, T. Lai, Z. Wang, K. Chen, H. Li, and K. Zheng, "Worker-churn-based task assignment with context-LSTM in spatial crowdsourcing," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 9, pp. 9783–9796, Sep. 2023.

[31] Y. Tong et al., "Flexible online task assignment in real-time spatial data," *Proc. VLDB Endowment*, vol. 10, no. 11, pp. 1334–1345, 2017.

[32] D. Sun et al., "Online delivery route recommendation in spatial crowdsourcing," in *Proc. Conf. World Wide Web*, 2019, pp. 2083–2104.

[33] Y. Zeng, Y. Tong, L. Chen, and Z. Zhou, "Latency-oriented task completion via spatial crowdsourcing," in *Proc. IEEE 34th Int. Conf. Data Eng.*, 2018, pp. 317–328.

[34] Z. Liu, K. Li, X. Zhou, N. Zhu, Y. Gao, and K. Li, "Multi-stage complex task assignment in spatial crowdsourcing," *Inf. Sci.*, vol. 586, pp. 119–139, 2022.

[35] Z. Chen, P. Cheng, L. Chen, X. Lin, and C. Shahabi, "Fair task assignment in spatial crowdsourcing," *Proc. VLDB Endowment*, vol. 13, no. 12, pp. 2479–2492, 2020.

[36] P. Cheng, X. Lian, L. Chen, J. Han, and J. Zhao, "Task assignment on multi-skill oriented spatial crowdsourcing," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 8, pp. 2201–2215, Aug. 2016.

[37] W. Ni, P. Cheng, L. Chen, and X. Lin, "Task allocation in dependency-aware spatial crowdsourcing," in *Proc. IEEE 36th Int. Conf. Data Eng.*, 2020, pp. 985–996.

[38] Y. Li, H. Li, X. Huang, J. Xu, Y. Han, and M. Xu, "Utility-aware dynamic ridesharing in spatial crowdsourcing," *IEEE Trans. Mobile Comput.*, vol. 23, no. 2, pp. 1066–1079, Feb. 2024.

[39] Y. Xie et al., "Trajectory-aware task coalition assignment in spatial crowdsourcing," *IEEE Trans. Knowl. Data Eng.*, vol. 36, no. 11, pp. 7201–7216, Nov. 2024.

[40] Y. Zhao, J. Guo, X. Chen, J. Hao, X. Zhou, and K. Zheng, "Coalition-based task assignment in spatial crowdsourcing," in *Proc. IEEE 37th Int. Conf. Data Eng.*, 2021, pp. 241–252.

[41] Y. Zhao et al., "Coalition-based task assignment with priority-aware fairness in spatial crowdsourcing," *VLDB J.*, vol. 33, no. 1, pp. 163–184, 2024.

[42] P. Cheng, L. Chen, and J. Ye, "Cooperation-aware task assignment in spatial crowdsourcing," in *Proc. IEEE 35th Int. Conf. Data Eng.*, 2019, pp. 1442–1453.

[43] Z. Wang, Y. Li, K. Zhao, W. Shi, L. Lin, and J. Zhao, "Worker collaborative group estimation in spatial crowdsourcing," *Neurocomputing*, vol. 428, pp. 385–391, 2021.

[44] Y. Zhao, K. Zheng, J. Guo, B. Yang, T. B. Pedersen, and C. S. Jensen, "Fairness-aware task assignment in spatial crowdsourcing: Game-theoretic approaches," in *Proc. IEEE 37th Int. Conf. Data Eng.*, 2021, pp. 265–276.

[45] Y. Xie, Y. Wang, K. Li, X. Zhou, Z. Liu, and K. Li, "Satisfaction-aware task assignment in spatial crowdsourcing," *Inf. Sci.*, vol. 622, pp. 512–535, 2023.

[46] Y. Li, H. Li, B. Mei, X. Huang, J. Xu, and M. Xu, "Fairness-guaranteed task assignment for crowdsourced mobility services," *IEEE Trans. Mobile Comput.*, vol. 23, no. 5, pp. 5385–5400, May 2024.

[47] B. Li, Y. Cheng, Y. Yuan, Y. Yang, Q. Jin, and G. Wang, "ACTA: Autonomy and coordination task assignment in spatial crowdsourcing platforms," *Proc. VLDB Endowment*, vol. 16, no. 5, pp. 1073–1085, 2023.

[48] J. Liu, L. Deng, H. Miao, Y. Zhao, and K. Zheng, "Task assignment with federated preference learning in spatial crowdsourcing," in *Proc. Conf. Inf. Knowl. Manage.*, 2022, pp. 1279–1288.

[49] X. Cui, Y. Cheng, S. Zhang, Y. Yuan, and G. Wang, "Cooperative global path planning for multiple platforms," in *Proc. IEEE 40th Int. Conf. Data Eng.*, 2024, pp. 303–316.

[50] Y. Cheng et al., "Cross online ride-sharing for multiple-platform cooperations in spatial crowdsourcing," in *Proc. IEEE 40th Int. Conf. Data Eng.*, 2024, pp. 4140–4152.

[51] Y. Hu, Y. Wang, Y. Li, and X. Tong, "An incentive mechanism in mobile crowdsourcing based on multi-attribute reverse auctions," *Sensors*, vol. 18, no. 10, 2018, Art. no. 3453.

[52] J.-X. Liu, Y.-D. Ji, W.-F. Lv, and K. Xu, "Budget-aware dynamic incentive mechanism in spatial crowdsourcing," *J. Comput. Sci. Technol.*, vol. 32, no. 5, pp. 890–904, 2017.

[53] D. Zhao, H. Ma, and L. Liu, "Frugal online incentive mechanisms for mobile crowd sensing," *IEEE Trans. Veh. Technol.*, vol. 66, no. 4, pp. 3319–3330, Apr. 2017.

[54] Y. Xu, M. Xiao, J. Wu, S. Zhang, and G. Gao, "Incentive mechanism for spatial crowdsourcing with unknown social-aware workers: A three-stage Stackelberg game approach," *IEEE Trans. Mobile Comput.*, vol. 22, no. 8, pp. 4698–4713, Aug. 2023.

[55] X. Zhang, Z. Yang, Y. Liu, and S. Tang, "On reliable task assignment for spatial crowdsourcing," *IEEE Trans. Emerg. Topics Comput.*, vol. 7, no. 1, pp. 174–186, First Quarter 2019.

[56] Z. Wang, Y. Huang, X. Wang, J. Ren, Q. Wang, and L. Wu, "SocialRecruiter: Dynamic incentive mechanism for mobile crowdsourcing worker recruitment with social networks," *IEEE Trans. Mobile Comput.*, vol. 20, no. 5, pp. 2055–2066, May 2021.

[57] Y. Li, Q. Wu, X. Huang, J. Xu, W. Gao, and M. Xu, "Efficient adaptive matching for real-time city express delivery," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 6, pp. 5767–5779, Jun. 2023.

[58] H. Robbins, "Some aspects of the sequential design of experiments," *Bull. Amer. Math. Soc.*, vol. 58, no. 5, pp. 527–535, 1952.

[59] B. An, M. Xiao, A. Liu, X. Xie, and X. Zhou, "Crowdsensing data trading based on combinatorial multi-armed bandit and Stackelberg game," in *Proc. IEEE 37th Int. Conf. Data Eng.*, 2021, pp. 253–264.

[60] H. Zhao, M. Xiao, J. Wu, Y. Xu, H. Huang, and S. Zhang, "Differentially private unknown worker recruitment for mobile crowdsensing using multi-armed bandits," *IEEE Trans. Mobile Comput.*, vol. 20, no. 9, pp. 2779–2794, Sep. 2021.

[61] P. Yang, N. Zhang, S. Zhang, K. Yang, L. Yu, and X. Shen, "Identifying the most valuable workers in fog-assisted spatial crowdsourcing," *IEEE Internet Things J.*, vol. 4, no. 5, pp. 1193–1203, Oct. 2017.

[62] X. Gao, S. Chen, and G. Chen, "MAB-based reinforced worker selection framework for budgeted spatial crowdsensing," *IEEE Trans. Knowl. Data Eng.*, vol. 34, no. 3, pp. 1303–1316, Mar. 2022.

[63] S. Peng, B. Zhang, Y. Yan, and C. Li, "A multiplatform-cooperation-based task assignment mechanism for mobile crowdsensing," *IEEE Internet Things J.*, vol. 10, no. 19, pp. 16881–16894, Oct. 2023.

Tianyue Ren is currently working toward the doctoral degree with the College of Computer Science and Electronic Engineering, Hunan University, Changsha, China. Her research interests include spatial crowdsourcing and location privacy protection.

Zhibang Yang received the PhD degree in computer science from Hunan University, China. He is currently a professor with Changsha University. His research interests include data management, parallel computing, and network security.

Yan Ding (Member, IEEE) received the PhD degree in computer science from Hunan University, China, in 2021. Currently, he is an assistant professor with Hunan University. His research interests include parallel computing, mobile edge computing, Big Data, artificial intelligence, and architecture.

Xu Zhou received the master's degree from the College of Computer Science and Electronic Engineering, Hunan University, in 2009. She is currently an associate professor with the Department of Information Science and Engineering, Hunan University, Changsha, China. Her research interests include parallel computing and data management.

Kenli Li (Senior Member, IEEE) received the PhD degree in computer science from the Huazhong University of Science and Technology, China, in 2003. He is currently a Cheung Kong professor of computer science and technology with Hunan University, the dean with the College of Computer Science and Electronic Engineering, Hunan University. His major research interests include parallel and distributed processing. He serves on the editorial board of the *IEEE Transactions on Computers*.

Yunjun Gao (Senior Member, IEEE) received the PhD degree in computer science from Zhejiang University, China, in 2008, where he is currently a professor with the College of Computer Science. His research interests include database, Big Data management and analytics, and AI interaction with DB technology.

Keqin Li (Fellow, IEEE) is a SUNY distinguished professor of computer science with the State University of New York. He is also a National distinguished professor with Hunan University, China. His current research interests include cloud computing, fog computing and mobile edge computing, energy-efficient computing and communication, embedded systems, and computer architectures and systems. He has authored or coauthored more than 860 journal articles, book chapters, and refereed conference papers, and has received several best paper awards. He is an AAIA fellow. He is also a member of Academia Europaea.