Joint Dependency and Conflicting Task Allocation in Collaboration-aware Spatial Crowdsourcing

Jiajun Yao[1], Lei Yang[2] *, Hao Liu[1], Hui Xiong[1,3]
[1]Artificial Intelligence Thrust, Hong Kong University of Science and Technology (Guangzhou), Guangzhou, China
[2]School of Software Engineering, South China University of Technology, Guangzhou, China
[3]Department of Computer Science and Engineering, The Hong Kong University of Science and Technology, Hong Kong SAR, China
Email: 201810102795@mail.scut.edu.cn, sely@scut.edu.cn, liuh@ust.hk, xionghui@ust.hk

2025 IEEE 41st International Conference on Data Engineering (ICDE) | 979-8-3315-3603-9/25/$31.00 ©2025 IEEE | DOI: 10.1109/ICDE65448.2025.00266

***Abstract*—Spatial crowdsourcing (SC) is a new form of crowdsourcing that utilizes users (i.e., workers) equipped with smart devices to complete tasks at specific locations. Previous studies usually focus on single task relationships (e.g., dependencies or conflicts) without considering task allocation under multiple relationships. To address this limitation, we jointly consider task dependency and conflict while also considering collaboration among workers for task allocation. In this paper, we define and formulate a new problem, called Joint Dependency and Conflicting Task Allocation in Collaboration-aware Spatial Crowdsourcing (JDCTA), which is proved to be NP-hard. To tackle the JDCTA problem, we first design an approximation algorithm, JDCTA-Greedy, which constructs a set of associated task groups based on task relationships and then greedily allocates these groups, in which we can obtain results with a theoretical bound on the approximate ratio. We then propose JDCTA-Game, a both dependency and conflict aware game approach. JDCTA-Game reduces the strategy space by defining dependency and conflict trees, combined with a dynamic payoff function based on the multiple relationships between tasks, to achieve high-quality solutions. Theoretical analysis demonstrates that this method guarantees the existence of at least one Nash equilibrium, and the solution quality is bounded. Experimental results on both synthetic and real datasets show that our proposed approach outperforms the representative approaches in terms of overall utility.**

***Index Terms*—Spatial crowdsourcing, task dependencies, task conflicts, task allocation, collaboration.**

I. Introduction

Recently, Spatial Crowdsourcing (SC) has played a crucial role in academic research and industrial applications implementing spatial data collection, location identification, and task allocation as they can recruit large numbers of volunteers or participants (i.e., workers) with high flexibility and low cost, utilizing their geographical location information, spatial sensing and mobile devices to complete specific tasks [1]–[3]. In SC, requesters post location-based tasks to a platform, which assigns them to workers based on set rules, and workers travel to specified locations to perform them.

A fundamental issue in SC is task assignment [4]–[9], which assigns spatial tasks to appropriate workers such that the optimization goals are achieved. Despite extensive research on task assignment, these studies exhibit the following limitations.

- *Few efforts have been made to address real-world applications that require task assignments based on multiple relationships (e.g., dependencies or conflicts) between tasks.* In practice, it is essential for the SC platform to jointly consider these relationships to enhance the system's adaptability. For example, as shown in Fig. 1(a) for house decoration, the platform should first assign workers to task t_1 (main house maintenance), followed by task t_3 (installing water and electrical systems) only after t_1 is completed, since t_3 depends on t_1. Additionally, a user may post two floor installation tasks, t_5 (wooden floor installation) and t_6 (ceramic floor installation), for workers to choose from. Once a worker selects task t_5, t_6 will no longer be assigned, as t_5 and t_6 conflict with each other. Meanwhile, some tasks exist independently, such as t_2 (purchasing materials).

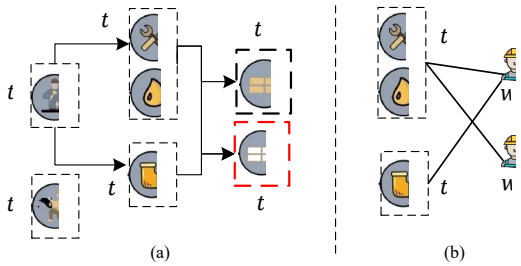


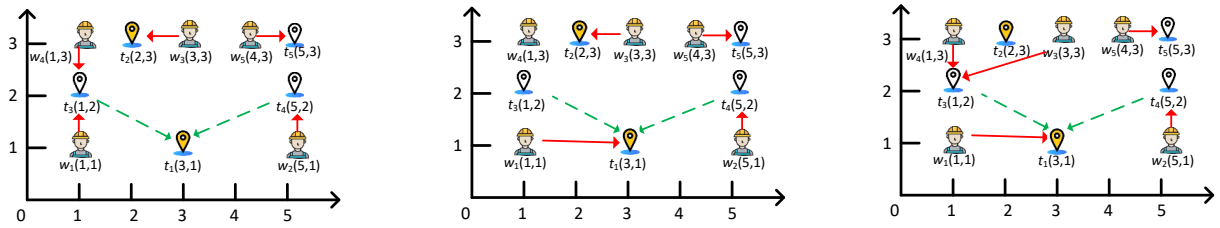
Fig. 1. An example of house decoration.

- *Simple task allocation models, such as the one-to-one model, are inadequate to meet the rapid evolution of crowdsourcing applications.* Existing studies [10], [11] mainly treat tasks as atomic tasks, meaning each task is assigned to only one worker per round. Other studies [12], [13] consider many-to-one allocation models, where tasks can be complex and require the collaboration of multiple workers. Similarly, workers with multiple skill sets can be assigned multiple tasks [14], representing a one-to-many model. In practice, a many-to-many model is required, as shown in Fig. 1(b). The task t_3 is a complex task that requires two workers to complete. At the same time, worker w_1 can handle multiple tasks.

Next, we further illustrate the JDCTA problem by a motivation example of task allocation in Fig. 2.

As shown in Fig. 2, there are five workers $(w_1, \ldots, w_5)$ *and five tasks* $(t_1, ..., t_5)$ *in a two-dimensional (2D) space. Workers and tasks are labeled with a location (i.e., 2D coordinates) when they appear on the platform. Each*

2375-026X/25/$31.00 ©2025 IEEE
DOI 10.1109/ICDE65448.2025.00266



（a）Conflicting assignments　（b）Dependency assignments　（c）Combined dependencies and conflicts

Fig. 2. This is a motivational example, where green dashed arrows represent task dependencies (e.g., t_3 depends on t_1), the two yellow coordinates indicate conflicting tasks (i.e., t_1 and t_2 are in conflict), and the red arrows represent the allocation results.

TABLE I
TASKS' DETAIL.

Task	t_1	t_2	t_3	t_4	t_5
Arrived time	1	1	1	2	2
Required skill	$\{\psi_1\}$	$\{\psi_2\}$	$\{\psi_1, \psi_3\}$	$\{\psi_4\}$	$\{\psi_4\}$
Dependency	$\emptyset$	$\emptyset$	$\{t_1\}$	$\{t_1\}$	$\emptyset$
Conflict	$\{t_2\}$	$\{t_1\}$	$\emptyset$	$\emptyset$	$\emptyset$
Reward	5	7	7	5	2

TABLE II
WORKERS' DETAIL.

Worker	w_1	w_2	w_3	w_4	w_5
Arrived time	1	1	1	2	2
Skill set	$\{\psi_1\}$	$\{\psi_4\}$	$\{\psi_2, \psi_3\}$	$\{\psi_1\}$	$\{\psi_4\}$

worker and task is associated with a skill set and a required skill set, respectively. Tables I and II illustrate the detailed information of workers and tasks, respectively. First, as illustrated in Fig. 2(a), this gives a conflicting task allocation without considering the dependencies among tasks, $\{\{t_2,(w_3)\}, \{t_3,(w_1,w_4)\}, \{t_4,(w_2)\}, \{t_5,(w_5)\}\}$, *where the match pairs* $\{t_3,(w_1)\}$ *and* $\{t_3,(w_4)\}$ *are invalid assignment because the dependent tasks of* t_3 *and* t_5 *have not been assigned. Thus, the total utility only is 9. Second, we also present a dependency task assignment* $\{\{t_1,(w_1)\}, \{t_2,(w_3)\}, \{t_4,(w_2)\}, \{t_5,(w_5)\}\}$, *as shown in Fig. 2(b). Since this allocation does not account for task conflicts, the match pair for either* $\{t_1,(w_1)\}$ *or* $\{t_2,(w_3)\}$ *is invalid. As a result, the total utility of this allocation is at most 14. Finally, we propose an allocation that jointly considers task dependencies and conflicts,* $\{\{t_1,(w_1)\}, \{t_3,(w_3,w_4)\}, \{t_4,(w_2)\}, \{t_5,(w_5)\}\}$, *as shown in Fig. 2(c). We can observe that this allocation achieves the highest allocation result with a total reward of 19.*

With the rapid development of spatial crowdsourcing applications, various tasks are continuously emerging, and the relationships between these tasks are becoming increasingly complex [15]. Previous allocation methods that focused on single-task relationships have struggled to meet these evolving demands. Both [10] and [16] highlighted the existence of dependencies among tasks and proposed solutions for handling dependent tasks in offline and online settings, respectively. Task dependency refers to the condition where a task can only be assigned once all its dependent tasks have been allocated. However, these studies only addressed task dependencies and did not consider potential conflicts between tasks. On the other hand, some research has examined task conflicts while overlooking task dependencies [17]–[19]. Therefore, we aim to investigate a task allocation problem in SC that jointly considers both task dependencies and conflicts, with the added complexity that a task may require the cooperation of multiple workers for completion.

Inspired by the above issues, in this paper, we formally define a novel problem, called Joint Dependency and Conflicting Task Allocation (JDCTA) in collaboration-aware SC. Our proposed method is mainly based on a batch processing framework to better adapt to dynamic SC scenarios. We first prove that the problem is NP-hard by reducing it from the Subset Sum Optimization. We then propose an approximation algorithm and a game-based algorithm to approximate the optimal solution while providing theoretical guarantees. To the best of our knowledge, this is the first study to address the task allocation that combines both task dependencies and conflicts in SC. Our main contributions can be summarized as follows:

- We formally define a novel task assignment in SC, namely Joint Dependency and Conflicting Task Allocation in collaboration-aware spatial crowdsourcing, where tasks have dependencies and conflicts, and complex tasks are completed through the cooperation of multiple workers. And we prove it is NP-hard.
- We develop a grouping-based approximation algorithm to efficiently allocate tasks, in which an associative task set is proposed to obtain a theoretical performance guarantee with an approximation ratio of $1/(1+d)$, where d denotes the maximum number of conflicting tasks for any task.
- We further propose a both dependency and conflict aware game approach, JDCTA-Game, which reduces the game space by defining dependency and conflict trees and a dynamic payoff function based on multiple relationships between tasks, leading to high-quality solutions.
- We evaluate the effectiveness and efficiency of our proposed methods on both real-world and synthetic datasets. Experimental results demonstrate that our methods outperform representative approaches.

II. PROBLEM FORMULATION

In this section, we first introduce a set of necessary preliminaries and formally define the JDCTA problem.

A. Problem Definitions

Definition 1. (Spatial Task) A task $t = \langle t.loc, t.b, t.d, t.\psi, t.\varphi, t.\mathcal{C}, t.\mathcal{D} \rangle$ is submitted to the SC

platform by a user. Here, $t.loc$ is the specific location where the task is performed, $t.b$ is the release time, $t.d$ is the due time, $t.\varphi$ is the reward for completing the task, and $t.\psi$ represents the set of skill requirements. Additionally, $t.\mathcal{C}$ and $t.\mathcal{D}$ denote the sets of tasks that conflict with and depend on task t, respectively. A detailed explanation of task dependencies and conflicts is given below.

- Dependency: The allocation of a task depends on the allocation status of other tasks, meaning a task can only be assigned after its dependent tasks have been allocated. For example, in Example 1, tasks t_3 and t_4 can only be assigned after task t_1 has been allocated.
- Conflict: Task conflict means that once a task is assigned to a worker, other tasks that conflict with it cannot be assigned to other workers. In Example 1, if tasks t_1 and t_2 are in conflict, then once task t_1 is assigned to a worker, task t_2 cannot be assigned to either the same worker or any other worker.

Definition 2. (Worker) A worker $w = \langle w.loc, w.b, w.d, w.v, w.c, w.r, w.\psi, w.\theta \rangle$ has a current location $w.loc$, appears on the platform at time $w.b$, and will stop providing services at time $w.b + w.d$. In addition, $w.v$ denotes the worker's movement velocity, $w.r$ represents the farthest distance the worker can arrived, $w.c$ indicates the maximum number of tasks the worker can complete, $w.\psi$ represents a set of skills the worker possesses and $w.\theta$ is the overall rating given by users for the quality of tasks completed by the worker. Note that, each worker has a service time window $[w.b, w.b + w.d]$, and they can perform multiple tasks within their time window, but can only service one task at a time point.

Definition 3. (Valid Workers-and-Task Matching Set) Given a set of workers and a task t, we say that this set is the *Valid Workers-and-Task Matching Set* ($VWTS_t$) for task t if it satisfies the following constraints,

- The skills of the workers in $VWTS_t$ can completely cover the skills required for the task and the number of the workers in $VWTS_t$ is equal to the number of skills required for the task, i.e., $\cup_{w \in VWTS_t} w.\psi = t.\psi$ and $|VWTS_t| = |t.\psi|$.
- The worker in $VWTS_t$ should arrive at the location of t before the deadline of task t and the worker, i.e., $w.b + d(w.loc, t.loc)/w.v \leq w.b + w.d$ and $t.b + t.d$, where $d(w.loc, t.loc)$ denotes the distance that the worker w moves from her/his current location to the location of t, which can be calculated by Euclidean distance.
- The location of task t is located within reachable activity range of all workers in $VWTS_t$, i.e., $(w.loc - t.loc)^2 \leq w.r^2$.

Definition 4. (The Preference of Task). Given a worker w and task t, the preference of task t for w is defined as $P_t(w) = \frac{w_i.\theta}{max(w.\theta)}$, where $max(w.\theta)$ means the highest rating in the worker list.

Definition 5. (The Preference of Worker). Given a worker w and a task t, the preference of w for t is defined as $P_w(t) = t.\varphi - d(w_i.loc, t.loc)\,\rho$, where ρ is an adjustable parameter that can be determined by correlation factors such as gasoline prices, weather, etc [16].

Definition 6. (Utility Score). Given a valid task-and-workers matching set $VWTS_t$ and a task t. The utility score $P(t, VWTS_t)$ is defined as the sum of the two-sided preferences from task and workers.

$$P(t, VWTS_t) = a \cdot P_t(VWTS_t) + (1 - a) \cdot P_{VWTS_t}(t) \quad (1)$$

where a is the normalization parameter, $P_t(VWTS_t)$ is preference score of t for worker set $VWTS_t$, $P_t(VWTS_t)$ is preference score of $VWTS_t$ for task t. The tasks' preference score $P_t(VWTS_t)$ can be computed by $P_t(VWTS_t) = \sum_{w \in VWTS_t} \frac{P_t(w)}{|VWTS_t|}$, where $|VWTS_t|$ is the number of valid workers-and-task matching set. Similarly, $P_{VWTS_t}(t) = \sum_{w \in VWTS_t} \frac{P_w(t)}{|VWTS_t|}$.

Note that, we use utility function $P(,)$ to better reflect the generalized setting for many practical applications of SC, including the total reward of assigned tasks [20], [21] ($a = 0$), the total quality of the tasks [22], [23] ($a = 1$), the total bilateral satisfaction [24], [25] ($a = 0.5$).

Definition 7. (JDCTA problem). Given a worker set $W = \{w_1, w_2, \ldots, w_m\}$, a task set $T = \{t_1, t_2, \ldots, t_n\}$, and an overall preference function $P(,)$. The JDCTA problem is to find an assignment A to maximize the overall utility score $P_A = \sum_{t_i \in T} P(t_i, (VWTS_{t_i}))$ without violating the following constraints: ($w_j \in VWTS_{t_i}, t_i \in T$)

$$max\ P_A \ \ s.t.$$

$$w_j.b + \frac{d(w_j.loc, t_i.loc)}{w_j.v} \leq w_j.b + w_j.d, \quad (2)$$

$$w_j.b + \frac{d(w_j.loc, t_i.loc)}{w_j.v} \leq t_i.b + t_i.d, \quad (3)$$

$$w_j.b \geq t_i.b, \quad (4)$$

$$d(w_j.loc, t_i.loc) \leq w_j.r, \quad (5)$$

$$P_{w_j}(t_i) \geq 0 \text{ and } P_{t_i}(w_j) \geq 0, \quad (6)$$

$$t_i.\varphi - d(w_j.loc, t_i.loc)\,\rho > 0, \quad (7)$$

$$\sum_{j=1}^{|W|} x_{ij} \leq 1, \quad x_{ij} \in \{0, 1\}, \quad (8)$$

$$\sum_{i=1}^{|t.C|} \sum_{j=1}^{|W|} x_{ij} \leq 1, \quad x_{ij} \in \{0, 1\}, t_i \in t.C, \quad (9)$$

$$T(t_j) \geq T(t_i), \quad t_i \to D_i, t_j \in D_i \quad (10)$$

where constraints (2) , (3) and (4) ensure that the worker can complete the task within both the task's and the worker's valid time window. Constraint (5) denotes that the assigned tasks must be within the radius of workers. (6) means that any valid utility score is non-negative. (7) ensures that the reward for completing tasks is positive. (8) means that each task can be completed at most once. (9) only one of t_i and t_j is assigned if t_i and t_j conflict. (10) means that if task t_i depends on t_j, t_j must be assigned before t_i.

Theorem 1. *The JDCTA problem is NP-hard.*

Proof. The NP-hardness proof can be obtained through a reduction from the Subset Sum Optimization (SSO) problem, which can be stated as follows: given a set $S = \{s_1, \ldots, s_{|S|}\}$ and a positive number K, where for each subset $s_i \in S$ is associated with a weight $u_i > 0$. The SSO is to find a subset S' that maximizes $P(S') = \sum_{s_i \in S'} u_i$ satisfying $P(S') \leq K$ for $S' \subseteq S$.

Consider the following instance of the JDCTA problem. Assuming that the reward of each task is set to 1, we construct n task groups $\check{T} = \{T_1, T_2, \ldots, T_n\}$ and $\check{D} = \{D_1, D_2, \ldots, D_n\}$. Each group T_i contains a_i, and each D_i contains b tasks, with tasks in D_i depending on those in T_i. Additionally, T_1 conflicts with T_2, and $|T_1| > |T_2|$. The skill requirement of tasks in T_i is ψ_1 and the skill requirement of tasks in D_i is ψ_2. There are K workers with the skill ψ_1 and enough workers with skill ψ_2. Suppose that there are some task groups $\bar{T}(\bar{T} \subset \check{T})$ that can be fully allocated, and then the goal of instance of the JDCTA problem is to maximize $b\sum_{T_i \in \bar{T}} a_i$ among task groups $\check{T}$, which is equivalent to maximize $b\sum_{T_i \in \bar{T}} a_i$ among $\bar{T}$ ($\bar{T} = \{T_1, T_3 \ldots, T_n\}$ and $|\bar{T}| = n-1$). Since b is constant, the objective of the instance is equivalent to the SSO problem, which finds a subset $\bar{T}$ that maximizes $\sum_{T_i \in \bar{T}} s_i$ but does not exceed K.

Given this mapping, the SSO problem can be solved by transforming it into the corresponding JDCTA problem instance. As SSO problem is known to be NP-hard [26], thus, the theorem is proved.

III. Approximate Algorithm

In this section, we propose an approximation algorithm, JDCTA-Greedy, to obtain results with theoretical bounds. Specifically, we combine each task and its related tasks into a set, and iteratively assign the related task set to multiple worker coalitions to obtain the highest overall utility.

First, we introduce the concept of an associative task group. We put a task and its dependent tasks into a set, and call the set an Associative Task Group (ATG). For example, there is task t_3 and its dependent task t_1 in Example 1, we then can obtain one associative task group $ATG = (t_1, t_3)$ of t_3. Based on this, a total $ATG = \{t_1, t_2, (t_1, t_3), (t_1, t_4), t_5\}$ can be obtained. Specifically, if an associated task group ATG_{t_1} is assigned, the total ATG will be updated as $ATG = \{t_2, t_3, t_4, t_5\}$.

A. JDCTA-Greedy Algorithm

The details of the JDCTA-Greedy algorithm are shown in Algorithm 1. Firstly, JDCTA-Greedy initializes three empty sets: S^* to store temporary matching results, S to hold the best matching result returned by the Hungarian algorithm, and A to store the final allocation results (lines 1-2). It then generates a total ATG for all tasks arriving before time b (line 2). For each round (lines 3-17), we obtain an atg with the highest benefit score by running the optimal bipartite graph algorithm (i.e., the Hungarian algorithm) and iterates repeatedly until no additional atg can be assigned. Notably, since each task t_i may require multiple skills, we decompose it into multiple independent tasks (Lines 7-9). Finally, all invalid objects that have passed their deadlines are removed (line 18).

Algorithm 1 JDCTA-Greedy algorithm.

Input: Worker set W_b, task set T_b, utility function $U(,)$
Output: Task assignment: A

1 Multiset $S^* \leftarrow \emptyset, S \leftarrow \emptyset, A \leftarrow \emptyset$
2 Generate a total associated task group ATG;
3 **while** there is $agt_i \in ATG$ can be assigned **do**
4 $Best_{atg} = 0$;
5 **foreach** $atg_i \in ATG$ **do**
6 **if** atg_i is a valid associated task group **then**;
7 **for** $t_i \in atg_i$ **do**;
8 **for** $s \in t_i.\psi$ **do**;
9 $T^{\Delta} \leftarrow s$;
10 $W^{\Delta} \leftarrow W_{agt_i}$;
11 $S \leftarrow$ optimal matching on $\left(W^{\Delta} \cup T^{\Delta}\right)$;
12 **if** $U(S) > Best_{atg}$ **then**;
13 $S^* \leftarrow S$;
14 $Best_{atg} = U(S)$;
15 **if** $S^* \neq \emptyset$ **do**
16 $A \leftarrow S^*$;
17 Update ATG and delete tasks that conflict with tasks in S^*;
18 remove workers/tasks whose deadlines have passed;
19 **return** A;

Example. Go back to Example 1. The JDCTA-Greedy algorithm first generates a total ATG for all tasks in the current batch. From this, we can obtain five sub-associative task groups $ATG = \{t_1, t_2, (t_1, t_3), (t_1, t_4), t_5\}$. As shown in Fig. 2, in the first iteration, we select the largest utility score ATG for allocation, $atg_{t_4} = \{t_1, t_4\}$, with the allocation result being $\{(t_1, w_1), (t_4, w_2)\}$. Since tasks t_1 and t_4 have already been assigned, we remove the invalid task t_2 and update the total ATG. In the second round, the current largest associative task group is $atg_{t_3} = \{t_3\}$, but since t_3 requires two workers to complete it, we treat t_3 as two separate independent tasks for allocation. Through multiple rounds of iteration, we finally achieve an allocation $\{\{t_1, (w_1)\}, \{t_3, (w_3, w_4)\}, \{t_4, (w_2)\}, \{t_5, (w_5)\}\}$ with a total utility of 19.

B. Theoretical analysis

Complexity Analysis. We use $\bar{W}$ and $\bar{T}$ as the maximum cardinality of available workers and tasks in all iterations, respectively. The loop will be ended until there is no associative task set that can be assigned. Thus, the number of iterations (line 3) is at most O $(\min(\bar{W}, \bar{T}))$. In each iteration, there are O$(\bar{T})$ associative task sets that should be scanned (line 5). In line 11, it takes O $\left(\min\left(\left(T^{\Delta}\right)^3, \left(W^{\Delta}\right)^3\right)\right)$ to obtain an optimal matching result by running the Hungarian algorithm. However, the length of the associated task group and the skills required for the tasks are limited and are small

relative to the number of tasks and workers at time b (i.e., $W^{\Delta} \ll \bar{W}$ and $T^{\Delta} \ll \bar{T}$). Thus, JDCTA-Greedy has a time complexity of $O(\min(|\bar{W}|, |\bar{T}|) * |\bar{T}|)$.

Approximate Ratio Analysis. Inspired theoretical analysis in [27] and [10], [28], we first demonstrate that $U(A)$, as derived by JDCTA-Greedy, is both monotone and submodular. We then establish that it satisfies the k-extendible system property, and finally obtain its approximation ratio based on results in [29].

Theorem 2. The overall utility $U(A)$ returned by JDCTA-Greedy is monotonic and submodular.

Proof. We can be rewritten to the Equation 1, $U(A) = \sum_{t \in T, w \in W} U(t, w) = \sum_{atg \in ATG, ws \in W} U(atg, ws)$, where ws represents a set of workers. Since $\forall U(atg, ws) > 0$, so $U(A)$ is monotone.

According to the submodularity theory [27], we have,

$$\begin{gathered} \forall\left(atg_i, ws_i\right) \notin A, \forall\left(\mathrm{atg}_j, ws_j\right) \notin A, \\ U\left\{A \cup x\right\}-U\{A\} \geq U\left\{\bar{A} \cup x\right\}-U\{\bar{A}\} \end{gathered} \tag{11}$$

where $x = (atg_i, ws_i)$, $A \subseteq \bar{A}$ and $\bar{A} = \{A \cup (atg_j, ws_j)\}$.

To prove the above inequality, we give two cases,

- Case1: x is added to both A and $\bar{A}$, and there is no conflict between tasks in atg_i and tasks in $\bar{A}$, we can obtain $U\{A \cup x\} - U\{A\} = U(x)$. Since atg_j does not become larger with the addition of atg_i, $U\{\bar{A} \cup x\} - U\{\bar{A}\} = U(x)$. Thus, the above inequality satisfies. Assuming there is a conflict between tasks in atg_j and atg_i, we can still obtain $U\{A \cup x\} - U\{A\} = U(x)$, but $U\{\bar{A} \cup x\} - U\{\bar{A}\} = 0$. Thus, we conclude that: $U\{A \cup x\} - U\{A\} \geq U\{\bar{A} \cup x\} - U\{\bar{A}\}$.
- Case2: There are two worker sets WS and $\overline{WS}$. WS and $\overline{WS}$ represent the sets of active workers who have not performed, and those who have performed, the associated task group atg_i, respectively. Therefore, we have $\overline{WS} \subseteq WS$. If atg_j can be completed by workers in WS but not by those in $\overline{WS}$. As a result, $U\{\bar{A} \cup x\} - U\{\bar{A}\} = 0$, but $U\{A \cup x\} - U\{A\} = U\left(atg_j, ws_j\right)$.

Thus, the theorem 2 is proved. □

Theorem 3. The overall utility $U(A)$ returned by JDCTA-Greedy algorithm is at least $\left(\frac{1}{1+d}\right) OPT(A)$, where $d = max_{i \in T} |t_i.\mathcal{C}|$.

Proof. We leverage the concept of a k-extendible system to derive performance guarantees for the JDCTA-Greedy algorithm. Let $\mathcal{N} = ATG$ and I represent the set of all subsets of ATG that satisfy the dependence and conflict constraints. First, it is evident that $(\mathcal{N}, I)$ is downward closed. Regarding the exchange property, two observations can be made: (1) adding a new atg_i does not violate the dependence constraint, and (2) adding a new atg_i may violate the conflict constraint, which can be corrected by removing at most d associated task groups. Therefore, we conclude that the JDCTA-Greedy system is d-extendible, where $d = max_{i \in T} |t_i.\mathcal{C}|$.

Based on the theoretical results in [29], for a k-extendible system with a positive, monotone submodular objective function, a greedy strategy can produce a $(k+1)$-approximate solution. □

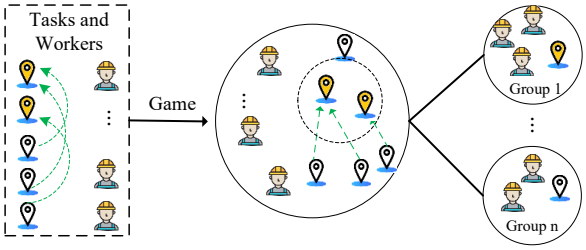


Fig. 3. This is an example of both dependency and conflict aware allocation. Task allocation depends not only on the task itself but also on its dependent subtasks and conflicting tasks. Moreover, workers need to form coalitions based on constraints to effectively conduct task.

IV. Game Theory Approach

Although the JDCTA-Greedy algorithm offers a theoretical guarantee on the approximation ratio, its effectiveness is limited and depends on a centralized server. In practice, workers may prefer to communicate and collaborate with each other to achieve higher task reward. To accommodate these individual incentives, game theory has been widely applied across various fields [6], [11], [30]. In the JDCTA problem, can be viewed as players who both cooperate and compete, each striving to maximize their own benefits, which aligns well with game theory principles. Thus, in this section, we introduce the JDCTA-Game, a dependency-and-conflict aware game algorithm, to further improve the effectiveness of task allocation. We also provide a theoretical analysis of its convergence speed and solution quality.

A. Game Modeling

In the JDCTA problem, task dependencies and conflicts expand the strategy space for worker games, increasing complexity, solving time, and the difficulty of theoretical analysis. As shown in Fig. 3, task allocation must consider both task dependencies and conflicts, while the assignment of preceding tasks depends on subsequent dependent tasks. For workers, the game evolves from a competition between individual workers to one between worker groups. Initially, workers autonomously form multiple worker group sets to ensure all constraints are met, and these groups ultimately converge into a final coalition to guide task allocation.

We formulate the JDCTA problem as a Multi-player game, which is defined by a tuple $< W, \{S_w\}_{w \in W}, \{U_w : \times_{w \in W} S_w\}_{w \in W} \rightarrow R >$ as:

- $W = \{w_1, w_2, \ldots, w_n\}$ denotes the set of players i.e., workers. Note that, we will use the concepts of player and worker exchangeably in the rest of the paper.
- $S = \{ST_1, ST_2, \ldots, ST_n\}$ denotes the total strategy set of all the players. ST_i is a limited set of tasks or an empty task set (that means w_i currently has no valid tasks that can satisfy the matching constraints with him/her). The task t in ST_i means that the dependencies and conflicts are completely satisfied, and the worker can reach location of t before deadline of t, and t is within the reachable distance radius of the worker, and the worker has not exceeded his/her own time window.

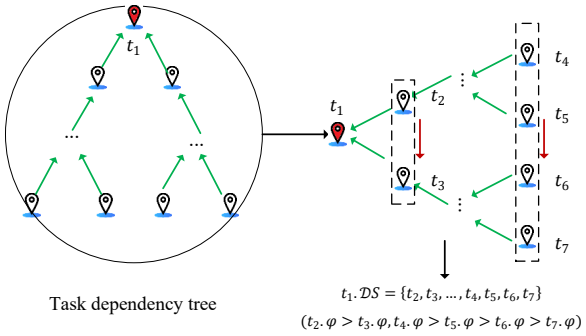


Fig. 4. An illustration of task dependency tree.

- $U = \{U_1, U_2, \ldots, U_n\}$ represents the total utility scores of all the players, where U_i denotes the utility score returned by worker w_i after executing its assigned strategy. It is important to note that this utility function differs from the Definition 6, as it explicitly accounts for both task dependencies and conflicts. Thus, we will redefine the task utility function to better reflect these multiple relationships.

To obtain a relatively global optimal allocation, we establish a dependency subset $t_i.\mathcal{DS}$ for each task. This dependency set $t_i.\mathcal{DS}$ can be represented as a tree structure, where directly dependent tasks form the first level, with further dependencies branching layer by layer. As shown in Fig. 4, task t_1 serves as the root node, t_2 and t_3 form the first level, while t_4, t_5, t_6, and t_7 form the second level. We establish the following rules for defining the dependency subset $t.\mathcal{DS}$:

- In the dependency tree, tasks from lower levels are always arranged before tasks from higher levels.
- Within the same level of the dependency tree, tasks with higher rewards are arranged before those with lower rewards.

Moreover, we define a conflict tree based on the previously established dependency tree. For two conflicting tasks, t_i and t_j, if they are in conflict with each other, we refer to their respective dependency trees as conflict trees.

Building on the above analysis, we give the utility function for workers performing tasks. Given the task strategy st_i and the strategies of other workers $\vec{st}_{-i}$, the utility $U\left(st_i, \vec{st}_{-i}\right)$ for worker w_i depends not only on the utility derived from executing its own task st_i but also on the aggregated impact of tasks that depend on st_i. Specifically, when $st_i.\mathcal{C} = \emptyset$, the overall utility can be expressed as follows:

$$U\left(st_i, \vec{st}_{-i}\right) = \prod_{t \in st_i.\mathcal{D}} a_t \cdot max\{P\left(st_i, w_i\right), \frac{P\left(st_i, w_i\right) + \sum_{j=1}^{st_i.\mathcal{DS}} P\left(st_i, VWTS(st_i)\right)}{|st_i.\mathcal{DS}| + 1}\} \tag{12}$$

Where $st_i.\mathcal{DS}$ represents a dependent tree for task st_i. a_t denotes whether its dependent tasks have been assigned, if $a_t = 1$ indicates that t has been assigned, otherwise vice versa.

From Equation 12, it is clear that the utility score of worker w_i for executing strategy st_i is determined not only by the utility score of st_i itself but also by the combined utility of the subsequent dependent tasks. If the average utility score of the dependent tasks is lower than that of st_i, the utility for worker w_i is solely based on st_i's own score. Conversely, if the average utility score of the dependent tasks is higher, strategy st_i is given a higher priority.

When $st_i.\mathcal{C} \neq \emptyset$, we assume that the conflicting task for st_i is st_j. In this case, we use the following equation to calculate the potential contributions of these two conflicting tasks to the overall allocation:

$$P(st_i) = \prod_{t \in t_i.D} a_t \left(P(t_i, w_i) + \sum_{i=1}^{|t_i.\mathcal{DS}|} P(s_i, VWTS(s_i)) \right) \tag{13}$$

By the above equation, if $P(st_i) \geq P(st_j)$, $U(st_i, \vec{st}_{-i})$ can be obtained using Equation 13; otherwise, it is 0.

B. JDCTA-Game Algorithm

Algorithm 2 JDCTA-Game Algorithm.

Input: Worker set W_b, task set T_b, utility function $U(,)$
Output: Task assignment: A

Multiset $A^* \leftarrow \emptyset, A \leftarrow \emptyset$
for each task $t_i \in T_b$ **do**;
  Generate the dependent tree $t_i.\mathcal{DS}$ of t_i;
for each worker $w_j \in W_b$ **do**;
  Obtain w_j strategic space $ST(w_j)$;
for each task $t_i \in T_b$ **do**;
  $VWTS(t_i) \leftarrow$ Randomly assign a worker group to t_i and it satisfies all constraints;
  if $VWTS(t_i) \neq \emptyset$ **then**;
    for each worker $w_j \in VWTS(t_i)$ **do**;
      $w_j.st = t_i$;
      $A^* \leftarrow (t_i, VWTS(t_i))$;
while Not Nash Equilibrium **do**
  for each $w_j \in W_b$ **do**
  $*$ Find the best response task t_i to w_j, and let t^* is a task that conflicts with t_i
    for $t_i \in ST(w_j)$ **do**
      for $w_j.\psi_k \in w_j.\psi$ **do**
        if $w_j.\psi_k$ in $A^*(t_i)$ **then**
          continue
        for $t_i.\psi \in w_j.\psi$ **do**
          if $S(t_i) =$ null **then**
            $w_j.st = t_i$;
            $A^*(t_i) \leftarrow w_j$;
          else
            if conflicting task $t^* \in A$
              if $U(t^*, w_j) > U(t_i, w_j)$
                continue;
              else $A^*(t_i) \leftarrow w_j$;
            else
              if $U(t_i, w_j) > U(t_k, w_j)$;
                $A^*(t_i) \leftarrow w_j$;
return A;

Algorithm 2 illustrates the details of the JDCTA-Game algorithm. Initially, the algorithm is given a set of workers W_b and tasks T_b before time b. In lines 1-11, both the preallocated set A^* and the assignment set A are initialized to $\emptyset$. Before the workers engage in the game, we first generate a dependency tree for each task, compute each worker's strategy space, and then use a random strategy to assign a worker group that satisfies all constraints for each task, adding the results to A^*. Next, in lines 12-29, the workers begin the game phase, adjusting their assigned tasks. Each worker iteratively modifies their strategy according to the best-response mechanism, which aims to maximize their total utility scores based on the current joint strategies of others. Since each worker has multiple skills and tasks may require multiple skills, we loop through to find a worker with the best skills that match the skills required for the task (lines 15-19). The best response strategy of a worker w can be found by calculating the overall utility brought by this allocation through Equations 12 and 13. Finally, we add valid matching pairs that satisfy dependency and conflict constraints to allocation A.

Example. Go back to Example 1 in Fig. 2, we can obtain three task dependency trees $ATG = \{(t_1, t_3, t_4), t_2, t_5\}$. In the first round of the game, worker w_1 has candidate strategies t_1 and t_3. Since the dependency task t_1 for t_3 is not yet assigned, w_1 opts for t_1. However, t_1 conflicts with t_2, with potential contributions of 19 and 7, respectively, to the overall allocation. Thus, the best response for w_1 is still to choose t_1. When worker w_2 arrives, since the dependency of task t_4 has been satisfied, the best response strategy for w_2 is t_4. Similarly, eventually each worker is able to determine their best response strategy. As a result, we obtain an optimal allocation, $\{\{t_1, (w_1)\}, \{t_3, (w_3, w_4)\}, \{t_4, (w_2)\}, \{t_5, (w_5)\}\}$, with a total utility of 19.

C. Theoretical analysis

Complexity Analysis. We ignore the time complexity involved in generating dependency subsets for each task and strategy spaces for each worker (lines 2-5), as well as the time required to initialize each worker coalition (lines 6-11). Our primary analysis focuses on the time complexity of lines 12-29, as the algorithm's main time consumption occurs during the worker game phase. Let L represent the number of iterations, i.e., the algorithm reaches Nash equilibrium after L rounds. Consequently, each worker may spend up to T_b time per round to obtain the optimal response task. Hence, the time complexity of the JDCTA-Game can be expressed as $O(L * W_b * T_b)$, where L is provided in Lemma 1.

Before analyzing the maximum number of rounds for the algorithm, we first show that at least one Nash equilibrium exists in the JDCTA-Game. Within the JDCTA framework, each worker's strategy is deterministic: they either choose a task to execute or do nothing, aligning with the concept of a pure strategy in Nash equilibrium [31]. A pure Nash equilibrium comprises a set of strategies where no player has an incentive to deviate from their strategy, provided that the strategies of other players remain constant.

Theorem 4. *The JDCTA game is an exact potential game that has at least one pure Nash Equilibrium.*

Proof. A strategic game is an exact potential game if a function Φ exists, such that for all $\vec{st}_i \in \times_{j \in T \setminus \{i\}} S_j$, the following condition holds:

$$U\left(st_i, \vec{st}_{-i}\right) - U\left(st'_i, \vec{st}_{-i}\right) = \Phi\left(st_i, \vec{st}_{-i}\right) - \Phi\left(st'_i, \vec{st}_{-i}\right) \quad (14)$$

In the JDCTA problem, it suffices to have that for every worker w, who changes his strategy from the current one st_w to his/her best-response st'_w. When $st_i.\mathcal{D} \neq \emptyset$ and $st_i.\mathcal{C} \neq \emptyset$, we define the potential function as

$$\Phi(S) = \sum_{t_i \in T - \{t_{\text{conflit}}\}} \prod_{t_i \in t_i.\mathcal{D}} a_t \cdot U\left(t_i, \text{VWTS}\left(t_i\right)\right) \quad (15)$$

Thus, we have

$$\begin{aligned}
&\Phi\left(st_i, \vec{st}_{-i}\right) - \Phi\left(st'_i, \vec{st}_{-i}\right) \\
&= \left(\prod_{t_j \in t_j.\mathcal{D}} a_{t_j} \cdot U\left(VTWS\left(t_j\right) \cup \{w_i\}\right) + \prod_{t_k \in t_k.\mathcal{D}} a_{t_k} \cdot U\left(VTWS\left(t_k\right)\right) \right. \\
&\qquad \left. + \sum_{t_i \in T - \{t_j, t_k, t_{\text{conflit}}\}} \prod_{t_i \in t_i.\mathcal{D}} a_t \cdot U\left(t_i, \text{VWTS}\left(t_i\right)\right) \right) \\
&- \left(\prod_{t_j \in t_j.\mathcal{D}} a_{t_j} \cdot U\left(VTWS\left(t_j\right)\right) + \prod_{t_k \in t_k.\mathcal{D}} a_{t_k} \cdot U\left(VTWS\left(t_k\right) \cup \{w_i\}\right) \right. \\
&\qquad \left. + \sum_{t_i \in T - \{t_j, t_k, t_{\text{conflit}}\}} \prod_{t_i \in t_i.\mathcal{D}} a_t \cdot U\left(t_i, VWTS\left(t_i\right)\right) \right) \\
&= \textstyle\prod_{t_j \in t_j.\mathcal{D}} a_{t_j} \cdot \left(U\left(VTWS\left(t_j\right) \cup \{w_i\}\right) - U\left(VTWS\left(t_j\right)\right)\right) \\
&\qquad + \prod_{t_k \in t_k.\mathcal{D}} a_{t_k} \cdot \left(U\left(VTWS\left(t_k\right)\right) - U\left(VTWS\left(t_k\right) \cup \{w_i\}\right)\right) \\
&= U\left(st_i, \vec{st}_{-i}\right) - U\left(st'_i, \vec{st}_{-i}\right)
\end{aligned}$$

Similarly, when $st_i.\mathcal{D} = \emptyset$ and $st_i.\mathcal{C} \neq \emptyset$, $st_i.\mathcal{D} \neq \emptyset$ and $st_i.\mathcal{C} = \emptyset$, $st_i.\mathcal{D} = \emptyset$ and $st_i.\mathcal{C} = \emptyset$, we can have the same result: $U\left(st_i, \vec{st}_{-i}\right) - U\left(st'_i, \vec{st}_{-i}\right) = \Phi\left(st_i, \vec{st}_{-i}\right) - \Phi\left(st'_i, \vec{st}_{-i}\right)$. □

Based on the theory of potential games and Theorem 4, we can conclude that there is at least one Nash equilibrium point in the JDCTA-Game. In other words, the JDCTA-Game will converge to a pure Nash equilibrium within a finite number of rounds and in a limited amount of time [32].

To determine the maximum number of rounds required to reach a pure Nash equilibrium using JDCTA-Game, we consider a scaled version of JDCTA with an integer-valued objective function. We define a potential function $\Phi_z(S) = \lambda * \Phi(S)$ as an equivalent game of the JDCTA potential game, where λ is a positive scaling factor that ensures $\Phi_z(S) \in Z$ for all S. Through the scaled potential function, we ascertain that JDCTA-Game requires at most $\Phi_z\left(S^*\right)$ rounds to achieve a pure Nash equilibrium, where S^* represents the optimal equilibrium in this potential JDCTA game.

Lemma 1. *The JDCTA-Game requires at most $\Phi_z(S^*)$ rounds in each time batch to converge to a pure Nash equilibrium, where $\Phi_z(S^*)$ $(= \lambda * \Phi(S^*))$ is a scaled potential function with only integer values. Here, $\Phi(S^*)$ represents the optimal total utility score obtained by the JDCTA-Game, and S^* is the optimal joint strategy in the potential JDCTA game.*

Proof. In Algorithm 2, JDCTA-Game achieves a Nash equilibrium when no worker deviates his/her current strategies. This means that in each round there is at least one worker who changes his/her current strategies, that is, our scaled problem can obtain at least 1 (i.e., $\Phi_z(st^*, st_{-i}) - \Phi_z(st, st_{-i}) \geq 1$) increase in utility in each round. Because the utility score for any valid matching is a positive integer number and $\Phi_z(ST^*)$ is the total utility of the optimal allocation. Therefore, the JDCTA method reaching the pure Nash equilibrium requires at most $\Phi_z(ST^*)$ rounds. □

We analyze the upper and lower bounds of the result quality returned by JDCTA-Game. The quality of the equilibrium is typically evaluated using three metrics: 1) Social Optimum (OPT): This metric refers to achieving the globally optimal quality of the joint policy. 2) Price of Stability (PoS): This represents the ratio of the utility of the best Nash equilibrium to the utility of the global optimum. 3) Price of Anarchy (PoA): This denotes the ratio of the utility of the worst Nash equilibrium to that of the globally optimal outcome.

Lemma 2. *In the pure strategy game of JDCTA, the quality $A(S)$ of the results returned by the JDCTA-Game algorithm is bounded as follows:*

$$\frac{\left(a\frac{\min(w.\theta)}{\max(w.\theta)} + (1-a)(\min(t.\varphi) - \max(w.r)\rho)\right)}{(a + (1-a)(\max(t.\varphi)))} \leq A(\mathrm{S}) \leq 1 \quad (16)$$

Proof. Let $Q(S)$ represent the overall task reward of the joint strategy S for task A. Additionally, we denote the globally optimal joint strategy as OPT, the equilibrium strategy with the highest overall task reward as S^*, and the equilibrium strategy with the lowest overall task reward as $S^{\#}$. Since OPT is the joint strategy of solutions in the optimal algorithm and S^* is the equilibrium that obtains the highest overall satisfaction value, it is obvious that $Q(OPT) \geq Q(S^*)$. Thus, we have

$$A(S) \leq PoS = \frac{\mathrm{Q}(S^*)}{\mathrm{Q}(OPT)} \leq 1$$

To find the lower bound of $A(S)$, we need to analyze the minimum global utility returned by JDCTA-Game. In each round, if workers change their strategy, the total utility score will increase. Therefore, the strategy game converges to a Nash equilibrium, the resulting assignment must yield a higher utility than the initial assignment. Let $|A|$ denote the number of matching pairs in A, we have

$$\begin{aligned}
&\mathrm{Q}\left(S^{\#}\right) \geq |A| \min(U(t, VWTS(t))) \\
&= |A| \left(a \min\left(P_t(VWTS(t))\right) + (1-a) \min\left(P_{VWTS}(t)(t)\right)\right. \\
&\geq |A| \left(a\frac{\min(w.\theta)}{\max(w.\theta)} + (1-a)(\min(t.\varphi) - \max d(\mathrm{w}.loc, t.loc)\rho)\right) \\
&\geq |A| \left(a\frac{\min(w.\theta)}{\max(w.\theta)} + (1-a)(\min(t.\varphi) - \max(w.r)\rho)\right)
\end{aligned}$$

TABLE III
SYNTHETIC DATASET.

Factor	Setting
$\|T\|$	500,1000,**2500**,5000,10000
$\|W\|$	100,200,**500**,1000,2000
$w.v$	0.1,0.2,**0.3**,0.4,0.5
$w.c$	1,2,**3**,4,5
$w.r$	0.2,0.4,**0.6**,0.8,1
$w.cost$	10,20,**30**,40,50
$w.due, t.due$	2,4,**6**,8,10
$t.\psi$	1,2,**3**,4,5
$w.\psi$	**1**,2,3,4,5
$d.pro$	[0.1,0.1],[0.1,0.3],**[0.1,0.5]**,[0.1,0.7],[0.1,0.9]
$d.bra$	[1,1],[1,2],**[1,3]**,[1,4],[1,5]
$c.pro$	[0.1,0.1],[0.1,0.3],**[0.1,0.5]**,[0.1,0.7],[0.1,0.9]

TABLE IV
REAL DATASET.

Platform	$\|T\|$	$\|W\|$	$w.d, t.d$	$w.\theta$	$w.r$
EverySender	4000	100,200,400,600,800	600	(0,1]	10
DiDi	5000	1K,2K,3K,4K,5K	300	(0,1]	300

The upper bound of the total utility can be calculated as

$$\begin{aligned}
&\mathrm{Q}(OPT) \leq |A| \max(U(t, VWTS(t))) \\
&= |A| \left(a \max\left(P_t(VWTS(t))\right) + (1-a) \max\left(P_{VWTS}(t)(t)\right)\right. \\
&\leq |A|(a + (1-a)(\max(t.\varphi)))
\end{aligned}$$

Thus, we have

$$\begin{aligned}
A(S) \quad &\geq AoS \geq \frac{\mathrm{Q}\left(S^{\#}\right)}{\mathrm{Q}(OPT)} \\
&= \frac{\left(a\frac{\min(w.\theta)}{\max(w.\theta)} + (1-a)(\min(t.\varphi) - \max(w.r)\rho)\right)}{(a + (1-a)(\max(t.\varphi)))}
\end{aligned}$$

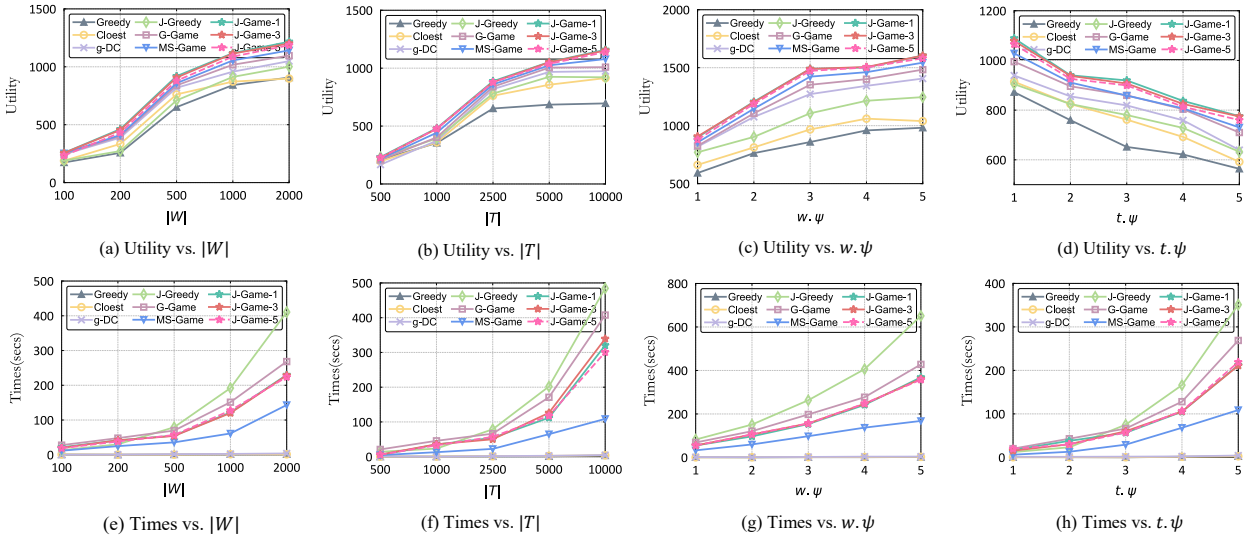
□

V. EXPERIMENTAL EVALUATION

In this section, we evaluate the performance of the proposed approaches based on two real-world datasets and a synthetic dataset. All methods are performed on a machine equipped with an Intel i7-8750H 2.20GHz and 16GB RAM.

A. Experimental Setup

Datasets. We evaluated the performance of our proposed method using two widely-used real-world datasets: EverySender (from a generic spatial crowdsourcing platform) [14], [33] and DiDi (from the Didi Chuxing ride-hailing app's open dataset) [12]. EverySender is a spatial crowdsourcing platform where each worker/task in the dataset is associated with attributes such as location, release time, deadline, reachable distance, and task reward. DiDi is a ride-hailing platform where users post ride requests with pickup and drop-off locations. The DiDi dataset includes information such as drop-off time, pickup time, and order price. For our purposes, we treat the drop-off and pickup times as the task and worker arrival times, respectively, and use their corresponding locations as the task and worker initial positions. Since explicit task dependencies or conflicts are not available in these real-world datasets, we generated task dependencies and conflicts following the methods described in [10], [16], [17]. Table III the statistics of the real datasets.



(a) Utility vs. $|W|$ (b) Utility vs. $|T|$ (c) Utility vs. $w.\psi$ (d) Utility vs. $t.\psi$

(e) Times vs. $|W|$ (f) Times vs. $|T|$ (g) Times vs. $w.\psi$ (h) Times vs. $t.\psi$

Fig. 5. Results on varying the number of workers ($|W|$), tasks ($|T|$), worker skills ($w.\psi$), and task skill requirements ($t.\psi$).

For the synthetic dataset, we use the same generation method as the synthetic dataset setup in the previous work [14], [16]. We vary parameters such as the number of workers and tasks, the number of skills possessed by workers, the number of skills required by tasks, workers' travel costs ($w.cost$), and the due times ($t.due$ and $w.due$) for tasks and workers, to better reflect real-world application scenarios. Specifically, we generate the locations of workers and tasks using a uniform distribution within the unit square [0,1]×[0,1]. Additionally, we set the probability $d.pro$ and $c.pro$ that each task has dependencies and conflicts within the range $[0.1, 0.1]$ to $[0.1, 0.9]$, and define the size $d.bra$ of dependent task sets for each task within the range $[1, 1]$ to $[1, 5]$. Other attributes (e.g., $|W|$, $|T|$ and $w.r$) are set in the same way as in [12]. For simplicity, the size of the conflict set is set to 1 by default. Table IV presents the statistics of the synthetic dataset, with the default settings shown in bold.

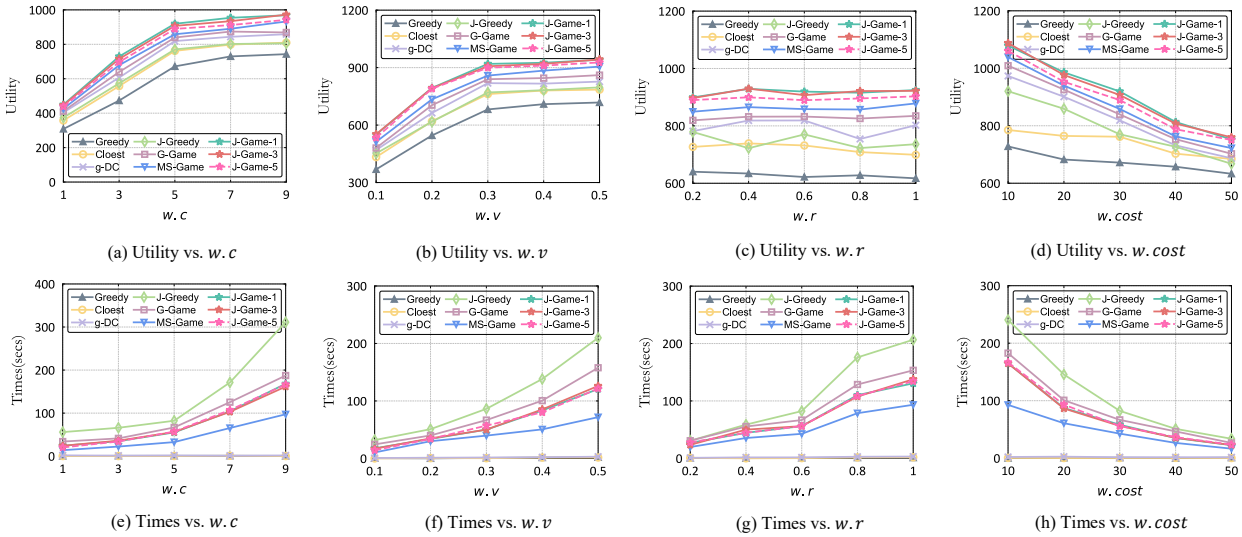
Comparison Approaches. Our proposed algorithm is compared with the following algorithms.

- **Greedy**: This is a greedy approach for task assignment [11], [34]. It greedily selects a matching pair with the largest utility score without violating all constraints.
- **Cloest**: This is a simple heuristic approach that assigns the closest worker to task form a matching pair [10], [35].
- **g-D&C** (g-DC for short): The algorithm iteratively divides the problem into g subproblems at each level until each subproblem contains only a single task. Each of these single-task subproblems is then solved using a greedy algorithm [10].
- **Mutil_Stage-Game** (MS-Game for short): This is multi-stage game theory method iteratively filters out tasks that do not meet dependency constraints. It begins with a filtering module to identify available tasks and workers, followed by task allocation through a best-response mechanism. The process is repeated until the final assignment is obtained [35], [36].
- **Globle-Game** (G-Game for short): This is a global game method that allocates all currently arrived tasks and workers, and then filters out invalid allocations, i.e. allocations where dependencies and conflicts have not been satisfied [11]–[13].
- **JDCTA-Greedy** (J-Greedy for short): Our JDCTA-Greedy method is a grouping-based greedy algorithm for obtaining approximate solutions, as described in Section III.
- **JDCTA-Game** (J-Game for short): The JDCTA-Game approach, detailed in Algorithm 2, is our proposed method. Here, J-Game-k denotes that the utility of a task is influenced by the utilities of the first $|t.\mathcal{DS}| = k$ dependent tasks. All game methods use a threshold stopping optimization strategy to shorten the running time, where the threshold defaults to 0.05.

Metrics and Implementation. In the experiments, we utilize the following two metrics to investigate the performance of our proposed approaches. 1) Overall utility (utility for short): The overall utility represents the total benefit from assigning worker-task pairs, which is the objective of the JDCTA problem [4], [21]; 2) Running time (time for short): The running time represents the time it takes to execute an algorithm to solve the JDCTA problem [10], [24].

B. Experimental Results

Effect of $|W|$. The first column in Fig. 5 shows the effect of varying $|W|$. In Fig. 5(a), the total utility of the nine approaches naturally increases when the value range of $|W|$ increases from 100 to 2000. This is because more tasks can be conducted when $|W|$ increases. Our proposed J-Game-1, J-Game-3, and J-Game-5 outperform the baseline methods MS-Game, G-Game, Greedy, Closest, and g-DC. Specifically, all game-based methods consistently perform better than heuristic approaches such as g-DC. This advantage arises from



(a) Utility vs. $w.c$ (b) Utility vs. $w.v$ (c) Utility vs. $w.r$ (d) Utility vs. $w.cost$

(e) Times vs. $w.c$ (f) Times vs. $w.v$ (g) Times vs. $w.r$ (h) Times vs. $w.cost$

Fig. 6. Results on varying worker capacity ($w.c$), movement speed ($w.v$), reachable distance ($w.r$), and movement cost ($w.cost$).

the fact that the Game algorithms conduct a global search, whereas the Greedy, Closest, and g-DC methods only search for a single object (worker or task) or a part of the problem space. Additionally, in most cases, J-Game-1 and J-Game-3 are more effective than J-Game-5, possibly because excessive dependence on subsequent subtasks can lead to inefficiencies. Fig. 5(e) presents the runtime of these nine methods, showing a natural increase as the number of workers grows.

Effect of $|T|$. The second column in Fig. 5 presents the effect of varying $|T|$. We can see that when $|W|$ is fixed and $|T|$ increases from 500 to 10000, the overall utility follows a similar pattern as when $|W|$ increases. Our J-Game-k achieves the highest utility score, followed by MS-Game, G-Game, g-DC, J-Greedy, Closest, and Greedy. This trend is consistent with the results discussed for varying $|W|$. Fig. 5(f) shows that J-Greedy and G-Game have the longest runtime, which is expected due to the approximately cubic time complexity of these two algorithms.

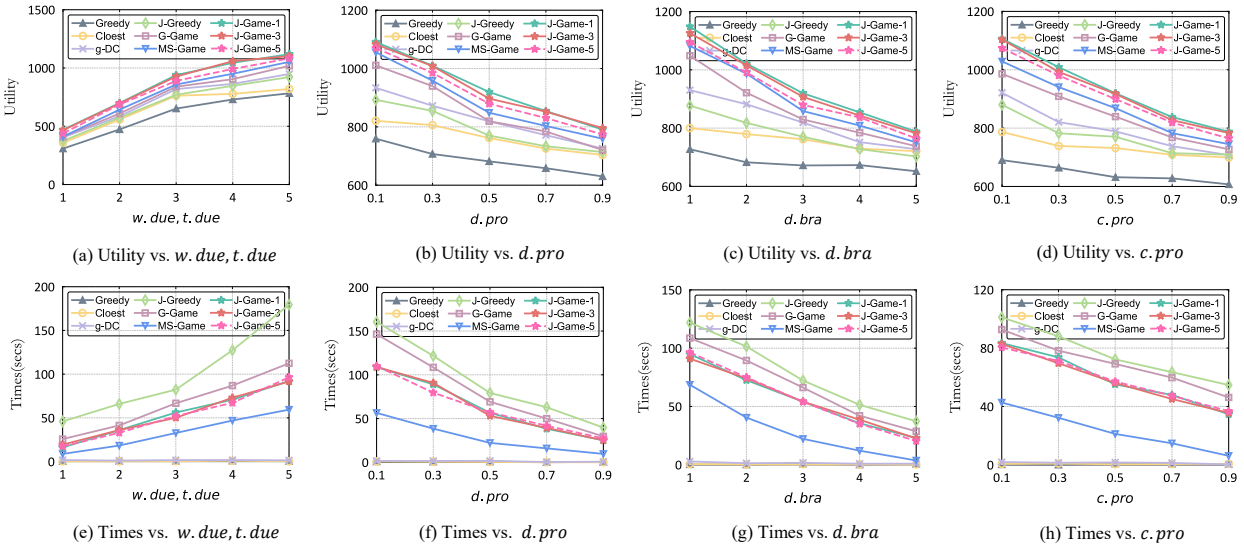
Effect of $w.\psi$. Fig. 5(c) shows the variation in overall utility results with $w.\psi$. As the workers' skills $w.\psi$ increase from 1 to 5, the total utility of all methods initially rises rapidly and then stabilizes. This is because the likelihood of assigning tasks increases as $w.\psi$ increases. However, when $w.\psi$ increases from 3 to 5, the overall utility remains largely unchanged since the default skill requirement for each task is 3. Our J-Game-k approach outperforms the baseline methods, MS-Game and G-Game, in terms of utility. Regarding time consumption, Fig. 5(g) shows that the runtime of all algorithms gradually increases. Among them, Greedy and Closest are the most efficient, while J-Greedy has the highest runtime.

Effect of $t.\psi$. Fig. 5(d) shows the variation of the overall utility results with $t.\psi$. As $t.\psi$ increases from 1 to 5, the total utility of all methods gradually decreases. The reason is that a larger $t.\psi$ means more workers are required for each task, resulting in fewer completed tasks. However, the overall utility increase of J-Game-k is still the largest, while that of the baseline Greedy is the smallest. In Fig. 5(h), the running time of all nine algorithms also gradually increases when $t.\psi$ increases. This is due to the fact that a larger $t.\psi$ results in a reduced chance of each task being completed, causing each task to continue to stay in each batch for the workers to search.

Effect of $w.c$. The first column in Fig. 6 illustrates the impact of varying $w.c$. In Fig. 6(a), the total utility of the nine algorithms first increase and then stabilize. This is because when $w.c$ increases from 1 to 5, each worker gets more capacity, i.e., the worker can conduct more tasks during they service time window. Since $|T| = 2500$ and $|W| = 500$, we can know that each worker can perform five tasks on average, so the total utility only slightly increase when $w.c$ increases from 5 to 9. Our proposed J-Game-k still achieves the highest overall utility. In terms of runtime, J-Game-k is faster than G-Game, as shown in Fig. 6(e). This is because G-Game spends additional time dealing with invalid matches where dependency and conflict constraints are not satisfied.

Effect of $w.v$. In Fig. 6(b), we can see that the overall utility of all methods gradually increase and then stabilize when the worker speed $w.v$ increases from 0.1 to 0.5. This is due to the fact that with $w.v$ increases, each worker is able to complete tasks faster, resulting in more tasks being assigned within their time window. When $w.v$ falls within the range of $[0.3, 0.5]$, although the speed of each worker becomes larger, the tasks that can be completed by them are limited, so the total utility becomes stable. However, J-Game-k still achieves higher overall utility scores than all other baselines. Fig. 6(f) also shows that the runtime of all algorithms gradually increases. This is due to the increase in $w.v$, which allows workers to become active more quickly, leading to more workers participating in subsequent rounds, thus increasing the overall runtime of each algorithm.

Effect of $w.r$. The third column in Fig. 6 shows the effect



(a) Utility vs. $w.due, t.due$ (b) Utility vs. $d.pro$ (c) Utility vs. $d.bra$ (d) Utility vs. $c.pro$

(e) Times vs. $w.due, t.due$ (f) Times vs. $d.pro$ (g) Times vs. $d.bra$ (h) Times vs. $c.pro$

Fig. 7. Results on worker and task durations ($w.due, t.due$), the probability ($d.pro$) of a task having dependencies, the size of the dependency set ($d.bra$), and the probability ($c.pro$) of task conflicts.

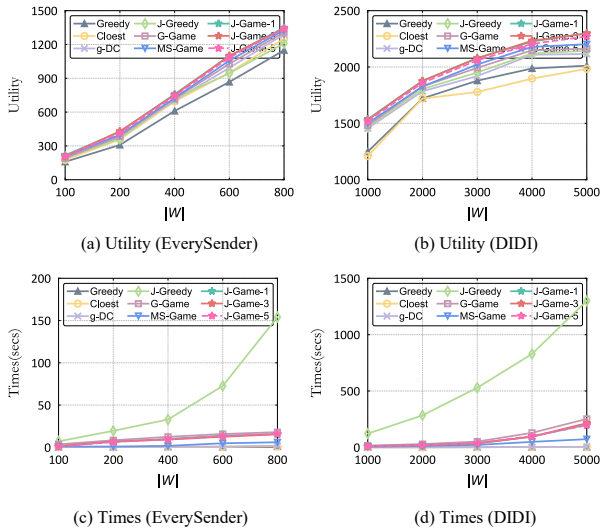
of varying $w.r$. Fig. 6(c) shows that the total utility of the nine methods changes only slightly as $w.r$ grows. This is because, despite the increase in $w.r$, each worker can perform requests at greater distances, which means s/he will incur more movement costs and hence only a slight change. In terms of CPU time, as shown in Fig. 6(g), each worker can direct more tasks with $w.r$ increases, resulting in an increase in the running time of the nine methods.

Effect of $w.cost$**.** Fig. 6(d) shows the variation of overall utility with $w.cost$. When movement cost $w.cost$ of workers increases from 10 to 50 and other attributes are fixed. We can note that the trend in total utility is similar to the trend in changing $w.r$. As shown in Fig. 6(h), the runtimes of all algorithms decrease gradually. This is because a larger $w.cost$ reduces the number of tasks that workers can complete.

Effect of $w.due, t.due$**.** Fig. 7(a) shows the variation of overall utility with $w.due, t.due$. When the duration of workers and tasks increase from 2 to 10 and other attributes are fixed. We can notice that the trend of total utility is similar to the trend of varying $|W|$ or $|T|$. As for CPU time, the execution time of the algorithm increases continuously when $w.due$ and $t.due$ get larger, as illustrated in Fig. 7(e). This is due to the fact that a larger deadline window means more time is needed to traverse the tasks or workers in each iteration. Our J-Game-k is still more efficient than the baseline J-Greedy and G-Game in terms of time consumption.

Effect of $d.pro$**.** In Fig. 7(b), when both $|W|$ and $|T|$ are fixed and the probability $d.pro$ increases from $[0.1, 0.1]$ to $[0.1, 0.9]$, we can see that the overall utility score of all the approaches decreases gradually. This is due to the fact that with $d.pro$ increases, there are more tasks that have dependencies, which may lead to fewer tasks available. When $d.pro$ falls within $[0.1, 0.9]$, the total utility of all the algorithms is the worst.

Effect of $d.bra$**.** Fig. 7(c) reports the results of the total utility for varying $d.bra$. When $d.bra$ increases from $[1, 1]$ to $[1, 5]$, the overall utility calculated by all methods gradually decreases. This is because a larger $d.bra$ means that the set of dependent tasks for each task becomes larger, resulting in fewer tasks being allocated. Furthermore, our proposed J-Game-k and J-Greedy achieve more overall utility than the baseline Greedy and Closest but sacrifice some efficiency. Fig. 7(g) shows that the running time of these nine algorithms gradually decreases. This is because fewer tasks can be assigned, resulting in less search time.



(a) Utility (EverySender) (b) Utility (DIDI)

(c) Times (EverySender) (d) Times (DIDI)

Fig. 8. Results on real datasets.

Effect of $c.pro$**.** Fig. 7(d) reports the results of the total utility with $c.pro$. The range of the probability that a task has conflicting tasks increases from $[0.1, 0.1]$ to $[0.1, 0.9]$, and the number of workers and tasks remains unchanged. We can observe that the trends in utility scores and CPU time are similar patterns to those when the increment of $d.pro$.

Performance on real datasets. To further evaluate the performance of our proposed method, we utilize two real-world datasets: EverySender and DIDI. Figs. 8(a) and 8(b) show the effect of varying $|W|$ on the total utility. We observe that as $|W|$ increases, the overall utility of all nine methods also increases. Notably, the total utility of J-Game-k grows faster compared to the baseline methods MS-Game and G-Game as $|W|$ increases. Regarding runtime, as shown in Figs. 8(c) and 8(d), all methods exhibit a similar increasing trend as $|T|$ increases. The runtime of J-Game-k is higher than that of g-DC, MS-Game, Greedy, and Closest, but lower than G-Game and J-Greedy. Importantly, both J-Game-k and J-Greedy outperform the baseline methods Greedy and Closest in terms of overall utility.

Summary of Results. The experimental study can be summarized as follows.

- JDCTA-Game significantly outperforms other algorithms in terms of total utility, albeit at the cost of some efficiency. Specifically, J-Game-1 and J-Game-3 achieve higher utility than J-Game-5 in most scenarios, while maintaining a similar level of efficiency.
- MS-Game provides a good balance between efficiency and effectiveness.
- The JDCTA-Greedy algorithm yields higher utility compared to the baselines Closest and Greedy, while also offering a theoretical performance guarantee.

Though the proposed methods manage large-scale problems via batch size adjustments, performance may degrade. Lowering their time complexity is critical for enhanced scalability, yet such optimization may weaken theoretical guarantees, requiring trade-off analysis.

VI. Related Work

In recent years, spatial crowdsourcing (SC) has emerged as a growing topic, garnering widespread attention from both industry and academia [2], [37], [38]. Tasks can be complex, with various relationships among them.

1) Task Allocation with Dependencies. Traditional SC often assumes task independence, focusing on maximizing assignments or optimizing quality, while overlooking task dependencies [14], [37]. However, real-world tasks, like home renovations and sports events, often involve interdependencies that need consideration [39]–[41]. Recently, Liu et al. [10], have proposed models that incorporate these dependencies to maximize task assignments while satisfying constraints. Game theory based methods have also been explored to achieve near-optimal allocations while respecting task dependencies [35]. Considering the online task allocation scenario. Yao et al. proposed an online dependent task allocation problem in preference-aware spatial crowdsourcing [16]. They introduced different methods under various order models and achieved the same theoretical bounds as in micro-task matching. Additionally, task dependencies have been explored in various allocation contexts. In influence-aware allocation, task dependencies are optimized through a data-driven framework [7]. A local ratio-based algorithm improves location-dependent task allocation by addressing dependencies [42]. In mobile crowdsourcing, an optimal stopping-based algorithm enhances satisfaction and stability by considering task dependencies and worker trajectory uncertainties [20].

2) Task Allocation with Conflicts. In task allocation, tasks can conflict with each other [27], [43]. Chen et al. [44] introduced the Global Event Participant Assignment Problem, which matches tasks with workers globally while accounting for conflicts. They extended bipartite matching by adding conflict constraints to prevent conflicting tasks from being assigned together. Further research on online conflict task allocation tackled dynamic changes in task allocation. [17] proposed a task matching approach with degree and budget constraints, solved through linear programming and greedy algorithms, effectively resolving conflicts. Additionally, worker-side conflicts, especially in mobile crowdsensing, have been addressed, with Zhang et al. [19] proposing a conflict-aware participant recruitment mechanism that also incorporates participant interest conflicts. However, none of the aforementioned work jointly considers both task dependencies and conflicts, while also addressing the need for multiple workers to collaborate on a single task.

VII. Conclusions

In this paper, we introduce a novel problem called Joint Dependency and Conflicting Task Allocation (JDCTA) in collaboration-aware spatial crowdsourcing, where tasks may not be independent due to dependencies and conflicts. The presence of these relationships adds complexity to the cooperative task assignment problem, which we prove to be NP-hard. We first propose a grouping-based greedy algorithm JDCTA-Greedy that provides a theoretical bound on the approximation ratio. Recognizing that workers prefer communication and collaboration to ensure high-quality task completion, we further develop a dependency and conflict aware game approach, JDCTA-Game, in which the Nash equilibrium is obtained using a best-response strategy. Experimental results demonstrate that JDCTA-Game significantly enhances the effectiveness of task assignments. In the future, we plan to focus on modeling task allocation in dynamic environments, taking into account real-time changes and constraints, while also investigating problems such as predictive modeling of worker behavior and preemptive task priority allocation.

VIII. Acknowledgment

This work is supported by Guangdong Basic and Applied Basic Research Foundation under Grant 2025A1515011996, in part by the National Key R&D Program of China (Grant No.2023YFF0725001),in part by the National Natural Science Foundation of China (Grant No.92370204), in part by the guangdong Basic and Applied Basic Research Foundation (Grant No.2023B1515120057), in part by Guangzhou-HKUST(GZ) Joint Funding Program (Grant No.2023A03J0008), Education Bureau of Guangzhou Municipality.

REFERENCES

[1] L. Kazemi and C. Shahabi, "Geocrowd: enabling query answering with spatial crowdsourcing," in *Proceedings of the 20th international conference on advances in geographic information systems*, 2012, pp. 189–198.

[2] S. R. B. Gummidi, X. Xie, and T. B. Pedersen, "A survey of spatial crowdsourcing," *ACM Transactions on Database Systems (TODS)*, vol. 44, no. 2, pp. 1–46, 2019.

[3] Z. Chen, R. Fu, Z. Zhao, Z. Liu, L. Xia, L. Chen, P. Cheng, C. C. Cao, Y. Tong, and C. J. Zhang, "gmission: A general spatial crowdsourcing platform," *Proceedings of the VLDB Endowment*, vol. 7, no. 13, pp. 1629–1632, 2014.

[4] Y. Li, Q. Wu, X. Huang, J. Xu, W. Gao, and M. Xu, "Efficient adaptive matching for real-time city express delivery," *IEEE Transactions on Knowledge and Data Engineering*, 2022.

[5] L. Yang, X. Yu, J. Cao, X. Liu, and P. Zhou, "Exploring deep reinforcement learning for task dispatching in autonomous on-demand services," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 15, no. 3, pp. 1–23, 2021.

[6] H. Wang, W. Liu, A. Liu, T. Wang, H. Song, and S. Zhang, "Sqcs: A sustainable quality control system for spatial crowdsourcing via three-party evolutionary game: Theory and practice," *Expert Systems with Applications*, vol. 238, p. 122132, 2024.

[7] X. Chen, Y. Zhao, K. Zheng, B. Yang, and C. S. Jensen, "Influence-aware task assignment in spatial crowdsourcing," in *2022 IEEE 38th International Conference on Data Engineering (ICDE)*. IEEE, 2022, pp. 2141–2153.

[8] Y. Zhao, X. Chen, G. Ye, F. Guo, K. Zheng, and X. Zhou, "Task allocation in spatial crowdsourcing: An efficient geographic partition framework," *IEEE Transactions on Knowledge and Data Engineering*, 2024.

[9] J. Akram, A. Anaissi, R. S. Rathore, R. H. Jhaveri, and A. Akram, "Galtrust: Generative adverserial learning-based framework for trust management in spatial crowdsourcing drone services," *IEEE Transactions on Consumer Electronics*, 2024.

[10] W. Ni, P. Cheng, L. Chen, and X. Lin, "Task allocation in dependency-aware spatial crowdsourcing," in *2020 IEEE 36th International Conference on Data Engineering (ICDE)*. IEEE, 2020, pp. 985–996.

[11] 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 *2021 IEEE 37th International Conference on Data Engineering (ICDE)*. IEEE, 2021, pp. 265–276.

[12] Y. Zhao, K. Zheng, Z. Wang, L. Deng, B. Yang, T. B. Pedersen, C. S. Jensen, and X. Zhou, "Coalition-based task assignment with priority-aware fairness in spatial crowdsourcing," *The VLDB Journal*, vol. 33, no. 1, pp. 163–184, 2024.

[13] P. Cheng, L. Chen, and J. Ye, "Cooperation-aware task assignment in spatial crowdsourcing," in *2019 IEEE 35th International Conference on Data Engineering (ICDE)*. IEEE, 2019, pp. 1442–1453.

[14] Y. Tong, Y. Zeng, B. Ding, L. Wang, and L. Chen, "Two-sided online micro-task assignment in spatial crowdsourcing," *IEEE Transactions on Knowledge and Data Engineering*, vol. 33, no. 5, pp. 2295–2309, 2019.

[15] X. Lin, K. Wei, Z. Li, J. Chen, and T. Pei, "Aggregation-based dual heterogeneous task allocation in spatial crowdsourcing," *Frontiers of Computer Science*, vol. 18, no. 6, p. 186605, 2024.

[16] J. Yao, L. Yang, and X. Xu, "Online dependent task assignment in preference aware spatial crowdsourcing," *IEEE Transactions on Services Computing*, 2022.

[17] C. Chen, L. Zheng, V. Srinivasan, A. Thomo, K. Wu, and A. Sukow, "Conflict-aware weighted bipartite b-matching and its application to e-commerce," *IEEE Transactions on Knowledge and Data Engineering*, vol. 28, no. 6, pp. 1475–1488, 2016.

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

[19] L. Zhang, Y. Ding, X. Wang, and L. Guo, "Conflict-aware participant recruitment for mobile crowdsensing," *IEEE Transactions on Computational Social Systems*, vol. 7, no. 1, pp. 192–204, 2019.

[20] F. Yucel and E. Bulut, "Online stable task assignment in opportunistic mobile crowdsensing with uncertain trajectories," *IEEE Internet of Things Journal*, 2021.

[21] Y. Cheng, B. Li, X. Zhou, Y. Yuan, G. Wang, and L. Chen, "Real-time cross online matching in spatial crowdsourcing," in *2020 IEEE 36th International Conference on Data Engineering (ICDE)*. IEEE, 2020, pp. 1–12.

[22] X. Miao, Y. Kang, Q. Ma, K. Liu, and L. Chen, "Quality-aware online task assignment in mobile crowdsourcing," *ACM Transactions on Sensor Networks (TOSN)*, vol. 16, no. 3, pp. 1–21, 2020.

[23] L. Zhao, W. Tan, B. Li, L. D. Xu, and Y. Yang, "Multiple cooperative task assignment on reliability-oriented social crowdsourcing," *IEEE Transactions on Services Computing*, 2021.

[24] Q. Zhang, Y. Wang, G. Yin, X. Tong, A. M. V. V. Sai, and Z. Cai, "Two-stage bilateral online priority assignment in spatio-temporal crowdsourcing," *IEEE Transactions on Services Computing*, 2022.

[25] X. Zhou, S. Liang, K. Li, Y. Gao, and K. Li, "Bilateral preference-aware task assignment in spatial crowdsourcing," in *2022 IEEE 38th International Conference on Data Engineering (ICDE)*. IEEE, 2022, pp. 1687–1699.

[26] V. V. Vazirani, *Approximation algorithms*. Springer Science and Business Media, 2013, vol. 1.

[27] C. Chen, S. Chester, V. Srinivasan, K. Wu, and A. Thomo, "Group-aware weighted bipartite b-matching," in *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, 2016, pp. 459–468.

[28] S. Khuller, A. Moss, and J. S. Naor, "The budgeted maximum coverage problem," *Information processing letters*, vol. 70, no. 1, pp. 39–45, 1999.

[29] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher, "An analysis of approximations for maximizing submodular set functions—i," *Mathematical programming*, vol. 14, pp. 265–294, 1978.

[30] Z. Han, *Game theory in wireless and communication networks: theory, models, and applications*. Cambridge university press, 2012.

[31] E. Maskin, "Nash equilibrium and welfare optimality," *The Review of Economic Studies*, vol. 66, no. 1, pp. 23–38, 1999.

[32] D. Fudenberg and J. Tirole, *Game theory*. MIT press, 1991.

[33] T. Song, Y. Tong, L. Wang, J. She, B. Yao, L. Chen, and K. Xu, "Trichromatic online matching in real-time spatial crowdsourcing," in *IEEE International Conference on Data Engineering*, 2017, pp. 1009–1020.

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

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

[36] 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 Transactions on Mobile Computing*, vol. 22, no. 8, pp. 4698–4713, 2022.

[37] J. Yao, L. Yang, Z. Wang, and X. Xu, "Non-rejection aware online task assignment in spatial crowdsourcing," *IEEE Transactions on Services Computing*, 2023.

[38] H. Bao, Z. Wang, R. Lu, C. Huang, and B. Li, "Tamt: Privacy-preserving task assignment with multi-threshold range search for spatial crowdsourcing applications," *IEEE Transactions on Big Data*, 2024.

[39] N. Castelo, M. W. Bos, and D. R. Lehmann, "Task-dependent algorithm aversion," *Journal of marketing research*, vol. 56, no. 5, pp. 809–825, 2019.

[40] J. Chen, Y. He, Y. Zhang, P. Han, and C. Du, "Energy-aware scheduling for dependent tasks in heterogeneous multiprocessor systems," *Journal of Systems Architecture*, vol. 129, p. 102598, 2022.

[41] S. Sundar and B. Liang, "Offloading dependent tasks with communication delay and deadline constraint," in *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*. IEEE, 2018, pp. 37–45.

[42] S. He, D.-H. Shin, J. Zhang, and J. Chen, "Toward optimal allocation of location dependent tasks in crowdsensing," in *IEEE INFOCOM 2014-IEEE Conference on Computer Communications*. IEEE, 2014, pp. 745–753.

[43] S. Braem, J. M. Bugg, J. R. Schmidt, M. J. Crump, D. H. Weissman, W. Notebaert, and T. Egner, "Measuring adaptive control in conflict tasks," *Trends in cognitive sciences*, vol. 23, no. 9, pp. 769–783, 2019.

[44] J. She, Y. Tong, L. Chen, and C. C. Cao, "Conflict-aware event-participant arrangement," in *2015 IEEE 31st International Conference on Data Engineering*. IEEE, 2015, pp. 735–746.