Fairness-aware Task Assignment in Spatial Crowdsourcing: Game-Theoretic Approaches

Yan Zhao
Department of Computer Science
Aalborg University
Aalborg, Denmark
yanz@cs.aau.dk

Kai Zheng*
University of Electronic Science and Technology of China
Chengdu, China
zhengkai@uestc.edu.cn

Jiannan Guo
China Mobile Cloud Centre
Suzhou, China
guojiannan@cmss.chinamobile.com

Bin Yang
Department of Computer Science
Aalborg University
Aalborg, Denmark
byang@cs.aau.dk

Torben Bach Pedersen
Department of Computer Science
Aalborg University
Aalborg, Denmark
tbp@cs.aau.dk

Christian S. Jensen
Department of Computer Science
Aalborg University
Aalborg, Denmark
csj@cs.aau.dk

***Abstract*—The widespread diffusion of smartphones offers a capable foundation for the deployment of Spatial Crowdsourcing (SC), where mobile users, called workers, perform location-dependent tasks assigned to them. A key issue in SC is how best to assign tasks, e.g., the delivery of food and packages, to appropriate workers. Specifically, we study the problem of Fairness-aware Task Assignment (FTA) in SC, where tasks are to be assigned in a manner that achieves some notion of fairness across workers. In particular, we aim to minimize the payoff difference among workers while maximizing the average worker payoff. To solve the problem, we first generate so-called Valid Delivery Point Sets (VDPSs) for each worker according to an approach that exploits dynamic programming and distance-constrained pruning. Next, we show that FTA is NP-hard and proceed to propose two heuristic algorithms, a Fairness-aware Game-Theoretic (FGT) algorithm and an Improved Evolutionary Game-Theoretic (IEGT) algorithm. More specifically, we formulate FTA as a multi-player game. In this setting, the FGT approach represents a best-response method with sequential and asynchronous updates of workers' strategies, given by the VDPSs, that achieves a satisfying task assignment when a pure Nash equilibrium is reached. Next, the IEGT approach considers a setting with a large population of workers that repeatedly engage in strategic interactions. The IEGT approach exploits replicator dynamics that cause the whole population to evolve and choose better resources, i.e., VDPSs. Using the property of evolutionary equilibrium, a satisfying task assignment is obtained that corresponds to a stable state with similar payoffs among workers and good average worker payoff. Extensive experiments offer insight into the effectiveness and efficiency of the proposed solutions.**

I. Introduction

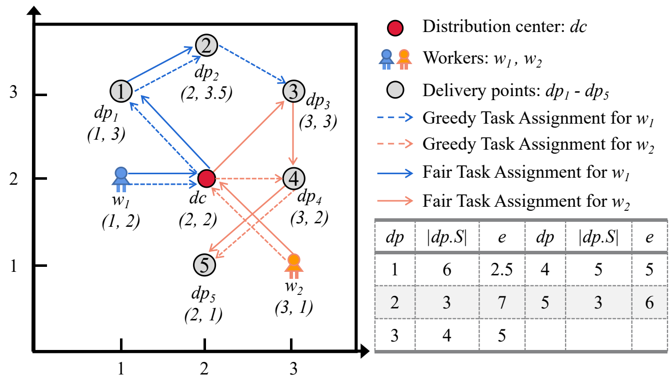
The widespread diffusion of smartphones enables a new class of crowdsourcing, called Spatial Crowdsourcing (SC), where a server assigns location-dependent tasks to smartphone users, called workers.

It is natural to view an SC system as a multi-player system that is designed based on the principles of a standard game-theoretic model. For example, a recent study [1] provides a game-theoretic approach, in which workers perform tasks cooperatively to achieve high total cooperation quality scores. However, the study does not address fairness during task assignment, i.e., how to ensure that task assignment does not discriminate against certain workers or worker groups. The standard game-theoretic approaches typically assume that players are rational, meaning that individual players maximize their own utility selfishly, while disregarding the effects on the population-wide utility. However, studies in behavioral economics reveal that humans often do not behave selfishly [2]. Instead, they consider the effects of their actions or strategies on others and strive for fair solutions while expecting others to do the same. Therefore, multi-player systems using only standard game-theoretic principles risk being misaligned with human expectations, which motivates the study of fairness in SC task assignment.

Fairness is a critical aspect of an SC platform: the social psychology literature shows that fairness is key to ensuring continuous and high worker participation and satisfaction [3]. Put differently, to ensure long-term commitment and participation by workers, fairness needs to be considered as a first class citizen when designing SC applications. Borromeo et al. [4] define the concept of unfairness as discrimination against individuals, while Durward et al. [5] differentiate among a variety of perspectives on fairness and ethics in crowdsourcing. However, these studies consider traditional crowdsourcing settings and do not take into account spatio-temporal aspects. Recently, Chen et al. [6] study a Fair and Effective Task Assignment (FETA) problem in SC and define worker fairness based on the notion of Fagin-Williams share. We will go further in this direction by providing an assignment notion of fairness and by designing game-theoretic approaches to achieving fair task assignment in SC.

Because delivery logistics, e.g., on-demand local delivery and transportation, is an important use case for SC, we investigate task assignment, called Fairness-aware Task Assignment

* Corresponding author: Kai Zheng.



● Distribution center: *dc*
Workers: w_1, w_2
Delivery points: dp_1 - dp_5
Greedy Task Assignment for w_1
Greedy Task Assignment for w_2
Fair Task Assignment for w_1
Fair Task Assignment for w_2
dp_1 (1, 3)
dp_2 (2, 3.5)
dp_3 (3, 3)
dp_4 (3, 2)
dp_5 (2, 1)
w_1 (1, 2)
w_2 (3, 1)
dc (2, 2)

dp	|*dp.S*|	*e*	*dp*	|*dp.S*|	*e*
1	6	2.5	4	5	5
2	3	7	5	3	6
3	4	5			

Fig. 1. Running Example

(FTA), in such a setting. Specifically, given a distribution center, a set of workers, delivery points, and tasks (deliveries from the distribution center to delivery points), the goal is to find a fair assignment of tasks to workers. We assume so-called assignment fairness, which is usually related to the distribution of goods and resources. We first illustrate the FTA problem through the example in Figure 1, that shows a distribution center (dc, located at $(2, 2)$), two workers (e.g., w_1 located at $(1, 2)$ and w_2 located at $(3, 1)$), and five delivery points (i.e., dp_1–dp_5). Each delivery point dp is associated with a set of tasks $dp.S$ (the deliveries to the location of dp), and the number of tasks is denoted by $|dp.S|$ (e.g., dp_1 has 6 tasks). Further, tasks expire after a certain time, and the earliest expiration time e among the tasks in a task set is recorded (e.g., the earliest expiration time of $dp.S$ is 2.5). The FTA problem is to assign tasks to workers so as to minimize the unfairness (i.e., payoff difference) among workers while maximizing the average worker payoff. Here the payoff of a worker is the ratio between the sum of the rewards of the assigned tasks and the sum of the worker's travel times needed to complete the tasks. For simplicity, we assume that each task has the same reward (1) and that each worker has the same speed (1). Next, each worker has to move to the distribution center to receive a task (e.g., a package to be delivered) and must then travel to an assigned delivery point to complete the task (perform the delivery). If worker w_i successively visits dp_1, dp_2, and dp_3 to perform tasks, the payoff is $\frac{6+3+4}{1+1.41+1.12+1.12} = \frac{13}{4.65} = 2.80$, where 13 is the sum of rewards and 4.65 is the sum of the travel times. A simple greedy approach is to ask each worker to move to the delivery points with the highest payoff without violating the spatio-temporal constraints, referred to as *Greedy Task Assignment*. This gives the task assignment $\{(w_1, \{dp_1, dp_2, dp_3\}), (w_2, \{dp_4, dp_5\})\}$, where the payoff difference between the two workers is 0.71 and the average worker payoff is 2.44. However, adopting a fair task assignment approach, we can achieve a difference of only 0.26 and a comparable average payoff of 2.42 with the assignment $\{(w_1, \{dp_1, dp_2\}), (w_2, \{dp_3, dp_4, dp_5\})\}$.

We show that the FTA problem is NP-hard and thus design two heuristic approaches, specifically a Fairness-aware Game-Theoretic (FGT) approach and an Improved Evolutionary Game-Theoretic (IEGT) approach. We first present an algorithm based on dynamic programming that generates so-called Valid Delivery Point Sets (VDPSs) for each worker. These represent the strategies of workers in games. We then improve the efficiency of the algorithm by means of a distance-constrained pruning strategy. The FGT and IEGT approaches are designed by converting the FTA problem into a multi-player game. In the FGT approach, we adopt Inequity Aversion based Utility (IAU), a common descriptive model for fairness, as the utility function. A best-response mechanism is employed to sequentially and asynchronously update workers' strategies (of selecting VDPSs) to reach a pure Nash equilibrium, corresponding to a satisfying fair task assignment. A Nash equilibrium holds that no worker can improve their utility by means of unilateral changes to their strategies when other workers persist in their existing strategies.

However, the assumption of rationality of workers implies complete information of each worker to calculate the best-response strategies given other workers' strategies, which is often not realistic. Therefore, we design an Improved Evolutionary Game-Theoretic (IEGT) approach based on a replicator dynamics mechanism. Using the property of improved evolutionary equilibrium, the algorithm achieves an improved stable state, where workers have similar payoffs and high average payoff. Due to the rationality limitations of workers, the improved evolutionary equilibrium can only be reached through repeated games. In our case, such an approach has the advantage that the rationality of workers can be improved by a dynamic studying process. Through repeated games, workers can adjust their strategies to arrive at a final evolutionary equilibrium that leads to a fair allocation of tasks.

Our contributions can be summarized as follows:

i) We identify and study in depth a Fairness-aware Task Assignment (FTA) problem in the context of SC.

ii) We formulate the FTA problem as a multi-player game and propose a classical game-theoretic approach, where fairness is considered and a Nash equilibrium is found based on the best-response framework.

iii) We also design an Improved Evolutionary Game-Theoretic task assignment approach that works by achieving an improved evolutionary equilibrium and that solves the FTA problem.

iv) We report experiments that offer insight into the impact of key parameters and into the effectiveness of the proposed techniques. The convergence of the game-theoretic approaches is also studied and analyzed.

II. Related Work

Research on Spatial Crowdsourcing (SC) [7]–[15] has attracted substantial attention in recent years; consequently, many task assignment techniques have been proposed for different application scenarios. As the task assignment problem is NP-hard in its general form [16], [17], many heuristic algorithms have been proposed that aim for near-optimal solutions [18]. Since SC systems can be regarded naturally as multi-player systems, some studies [1], [16] have proposed game-theoretic methods to solve the optimization problem of task allocation from the viewpoint of workers. However, the above studies focus mainly on the spatio-temporal availability of workers and tasks, thus leaving many problems regarding effective and efficient task assignment largely unaddressed, including how to consider fairness when assigning tasks.

An attractive SC platform offers means of promoting continuous participation of workers and avoiding worker turnover. Naturally, a negative correlation is expected between job satisfaction and worker turnover. Fairness is the most important factor for job satisfaction [19]. Ye et al. [20] study the fair assignment of tasks to heterogeneous workers with different capacities and costs in relation to task execution. The main idea to achieve fairness is to maximize the minimum utility (i.e., number of allocations) for all workers. However, most of these studies disregard the spatio-temporal information of workers and tasks, and thus do not apply readily to an SC application. Zhao et al. [21] propose a preference-aware task assignment problem for on-demand taxi dispatching, which considers the degree of dissatisfaction for both workers and tasks (i.e., passengers). Their problem setting differs substantially from ours, and thus their algorithm does not solve our problem. Basik et al. [22] propose a fair task allocation framework for crowdsourced delivery, focusing on distributive fairness. Specifically, distributive fairness is defined as the proximity between a worker's own input/output ratio and the input/output ratio of a referent, where the input to a worker is the total reward of the offers accepted and the output of a worker is the amount of reward earned. This notion of fairness differs from our definition of fairness. Further, Basik et al. assign tasks to workers, they do not schedule task location sequences. In addition, they assign tasks in ascending order of the number of available workers and assign workers with the lowest local assignment ratio. This arrangement does not apply in our setting, where each task has the same number of available workers and each worker has the same local assignment ratio. Recently, Chen et al. [6] address the worker fairness problem in SC by transforming it into an online bi-objective matching problem. It differs from our work in terms of the fairness definition, the problem settings, and the objectives. First, they define the worker fairness by Fagin-Williams share, which denotes the ideal credit each person in a carpool should have to find the deserved amount of task requests. In contrast, we employ the concept of Inequity Aversion based Utility (IAU), which offers a common and intuitive descriptive model for fairness. Further, we improve the notion of an evolutionary stable strategy in an evolutionary game to achieve a satisfied fair task assignment that affords workers' similar payoffs. Second, while Chen et al. [6] assign each worker a suitable task in each assignment, we assign tasks that are scheduled delivery point sequences to worker. Third, Chen et al. [6] aim to maximize a linear combination of the minimum individual worker fairness cost and the total utility, while we aim to minimize the payoff difference among workers to achieve more fair and equal task assignments. In particular, we aim to find the best-response strategy (i.e., the delivery point set with the highest IAU) for each worker in a classical game and pursue evolutionary stability in an evolutionary game, which can also bring a satisfying average payoff for each worker.

III. Problem Statement

We proceed to present necessary preliminaries and then define the problem addressed.

Definition 1 (Distribution Center): A distribution center, denoted by $dc = (l, S, DP)$, has a location $dc.l$, a set of tasks $dc.S$ to be distributed, and a set of delivery points $dc.DP$ (see Definition 2), at which tasks in $dc.S$ are to be delivered.

Definition 2 (Delivery Point): A delivery point, denoted by $dp = (l, S)$, consists of a location $dp.l$, and a set of tasks $dp.S$ that are deliveries to the delivery point, meaning that $dc.S = \cup_{dp \in dc.DP} dp.S$.

Definition 3 (Spatial Task): A spatial task, denoted by $s = (dp, e, r)$, encompasses a delivery point $s.dp$, a task expiration deadline $s.e$, and a reward $s.r$ that the worker who completes s will obtain.

A spatial task s can be completed only if a worker is physically located at its location (i.e., the location of $s.dp$). Next, a task s can be assigned to a worker only if the worker arrives at its location before the deadline $s.e$. With single-task assignment mode, the server assigns each task to a worker at a time. For simplicity and without loss of generality, we assume that the processing time of a task is zero, which means that a worker will proceed to the location of the next task immediately upon finishing the current task.

Definition 4 (Worker): A worker, denoted as $w = (l, maxDP)$, is able to perform spatial tasks. A worker can be in either online or offline mode. A worker is online when the worker is ready to accept tasks and offline when unavailable to perform tasks. An online worker w is associated with a current location $w.l$ and a maximum acceptable number of delivery points $w.maxDP$ that the worker is willing to deliver to.

We assume that a worker can only work for a single distribution center, which is reasonable in practice. The server will consider all the available tasks and workers at a particular time instance, and it returns a sequence of delivery points (each with a set of tasks) for each worker to visit in order to complete the delivery tasks while observing the spatio-temporal constraints. Once a delivery point sequence is assigned to a worker, the worker is offline until the assigned tasks are completed.

Definition 5 (Delivery Point Sequence): Given an online worker w and a set of assigned delivery points DP_w, a delivery point sequence on DP_w, denoted as $R(DP_w)$, represents the order in which w visits the delivery points in DP_w. The arrival time of w at delivery point $dp_i \in DP_w$ (the time of completing tasks related to dp_i) is computed as follows:

$$t_{w,R}(dp_i) = \begin{cases} c(w.l, dc.l) + c(dc.l, dp_i.l) & \text{if } i = 1 \\ t_{w,R}(dp_{i-1}) + c(dp_{i-1}.l, dp_i.l) & \text{if } i > 1, \end{cases}$$

where $c(a, b)$ is the travel time from location a to location b. When w and R are clear from the context, we use $t(dp_i)$ to denote $t_{w,R}(dp_i)$.

Definition 6 (Valid Delivery Point Set): A delivery point set DP_w is called a Valid Delivery Point Set (VDPS) for a worker w, denoted as $VDPS(w)$, if a delivery point sequence

$R(DP_w)$ exists, such that all the tasks located in dp ($\in DP_w$) can be completed before their expiration times, i.e., $\forall s_i \in dp.S$, $\forall dp \in DP_w$ ($t(dp) \leq s_i.e$).

Definition 7 (Worker Payoff): The payoff of a worker w is denoted by $P(w, VDPS(w))$, which is the ratio between the rewards for the tasks in $VDPS(w).S$ and worker w's travel time. It is calculated as follows:

$$P(w, VDPS(w)) = \frac{\sum_{s \in VDPS(w).S} s.r}{t(dp_{|VDPS(w)|})}, \tag{1}$$

where $VDPS(w).S$ denotes the tasks located in all the delivery points in $VDPS(w)$, $\sum_{s \in VDPS(w).S} s.r$ is the sum of the rewards of tasks in $VDPS(w).S$, and $t(dp_{|VDPS(w)|})$ denotes the arrival time of w at the final delivery point $dp_{|VDPS(w)|}$, which represents w's travel time from the current location to $dp_{|VDPS(w)|}$. The payoff difference is used to balance fairness and travel time. When $VDPS(w)$ is clear from the context, we use simply $P(w)$ instead of $P(w, VDPS(w))$.

Note that more than one delivery point sequences may exist for a given VDPS. Among these, we consider only the one with the minimal travel time (which brings workers the highest payoffs) for each VDPS. In Figure 1, $\{dp_1, dp_2, dp_3\}$ is a VDPS for worker w_1, and (dp_1, dp_2, dp_3) and (dp_1, dp_3, dp_2) are delivery point sequences. However, we only consider (dp_1, dp_2, dp_3) and compute the payoff of w_i based on it because it has the lowest travel time among the two.

Definition 8 (Spatial Task Assignment): Assume a set of distribution centers, a set of delivery points, each with a set of tasks, and a set of workers. Then a spatial task assignment, denoted by A, consists of a set of pairs of a worker and a VDPS for the worker: $(w_1, VDPS(w_1))$, $(w_2, VDPS(w_2))$,...,$(w_{|W|}, VDPS(w_{|W|}))$, where $VDPS(w_i) \cap VDPS(w_j) = \emptyset$, $1 \leq i \neq j \leq |W|$, and W denotes the worker set.

Each $(worker, VDPS)$ pair has a payoff, $P(w, VDPS(w))$. We use the payoff difference (denoted as P_{dif}) between workers to denote the assignment unfairness among them, which is calculated in Equation 2.

$$P_{dif} = \frac{\sum_{w_i \in W, w_j \in W, w_i \neq w_j} |P(w_i) - P(w_j)|}{|W|(|W| - 1)}, \tag{2}$$

where $P(w)$ denotes the payoff of worker w. We use $A.P_{dif}$ to denote the payoff difference among all workers in task assignment A.

FTA Problem Statement. Given a set of distribution centers, a set of delivery points, each with a set of tasks, and a set of workers at the current time instance on an SC platform, our problem is to find a task assignment A_{opt} that achieves the following goals:

1) primary optimization goal: minimize the payoff difference among workers (i.e., $\forall\ A_i \in \mathbb{A}$ $(A_i.P_{dif} \geq A_{opt}.P_{dif})$), where $\mathbb{A}$ denotes all possible assignments; and

2) secondary optimization goal: maximize the average worker payoff.

Lemma 1: The FTA problem is NP-hard.

Proof 1: The lemma can be proved through a reduction from the 0-1 knapsack problem that can be described as follows: given a set C with n items, in which each item $c_i \in C$ is labeled with a weight m_i and a value v_i, the 0-1 knapsack problem is to identify a subset C' of C that maximizes $\sum_{c_i \in C'} v_i$ subjected to $\sum_{c_i \in C'} m_i \leq M$, where M is a maximum weight capacity.

Consider the following instance of the FTA problem. We are given a worker set W with n workers, in which each worker $w_i \in W$ is associated with the maximum acceptable number of delivery points $w_i.maxDP = m_i$ (corresponding to the weight m_i of the 0-1 knapsack problem). Here, the number of delivery points is sufficiently large, and each worker can move to $w_i.maxDP$ delivery points to perform tasks. The value v_i of each worker w_i, which is a function, is at least as hard as the v_i (that is a constant) in the 0-1 knapsack problem, so that this difference does not make our problem easier. Additionally, we give M delivery points. Therefore, the FTA problem is to identify a worker subset W' of W that maximizes $\sum_{w_i \in W'} v_i$ subjected to $\sum_{w_i \in W'} m_i \leq M$.

If we can solve the FTA problem instance efficiently (i.e., in polynominal time), we can solve a 0-1 knapsack problem by transforming it to the corresponding FTA problem instance and then solve it efficiently. This contradicts the fact that the 0-1 knapsack problem is NP-hard [23], and so there cannot be an efficient solution to the FTA problem instance that is then NP-hard. Since the FTA problem instance is NP-hard, the FTA problem is also NP-hard.

IV. Valid Delivery Point Set (VDPS) Generation

We proceed to detail how to generate VDPS sets for workers, which will be used throughout the task assignment process. We devise a dynamic programming algorithm that finds the set of valid delivery point sequences for each worker. It can be shown that the global optimal result is the union of a possible valid delivery point sequence for each worker.

Each worker has to move to the distribution center first and then go to assigned delivery points to complete tasks. Therefore, we only need to calculate the VDPS set starting from the distribution center, called *Center-origin VDPS (C-VDPS)*, after which we check if the C-VDPSs are valid for each worker according to the travel time between the worker and the distribution center, and the task expiration times.

We proceed to detail the C-VDPS generation based on a dynamic programming algorithm that iteratively expands the sets of delivery points in ascending order of set size and finds all VDPSs in each iteration. Specifically, given a distribution center dc and a set of delivery points Q that is a subset of all delivery points associated with dc, we define $opt(Q, dp_j)$ as the maximum number of delivery points by scheduling all the delivery points in Q with constraints starting from $dc.l$ and ending at $dp_j.l$, and R as the corresponding delivery point sequence on Q to achieve this optimum value. We also use dp_i to denote the second-last delivery point in R before arriving at dp_j, and we use R' to denote the corresponding task sequence for $opt(Q - \{dp_j\}, dp_i)$. The calculation of $opt(Q, dp_j)$ depends on $|Q|$, the number of delivery points in Q:

i) when $|Q| = 1$,

$$opt(Q, dp_j) = \begin{cases} 1 & \text{if } t'_{dc,R}(dp_j) \leq dp_j.e \\ 0 & \text{otherwise} \end{cases} \quad (3)$$

$$t'_{dc,R}(dp_j) = \begin{cases} c(dc.l, dp_j.l), & \text{if } j = 1 \\ t'_{dc,R}(dp_{j-1}) + c(dp_{j-1}.l, dp_j.l) & \text{if } j > 1, \end{cases}$$

where $t'_{dc,R}(dp_j)$ denotes the arrival time of a worker (who starts from distribution center dc and follows delivery point sequence R) at delivery point $dp_j \in dc.DP$. Next, $dp_j.e$ denotes the earliest expiration time among the tasks located in dp_j, and $c(a, b)$ is the travel time from location a to b.

ii) when $|Q| > 1$,

$$opt(Q, dp_j) = \max_{dp_i \in Q, dp_i \neq dp_j} \{opt(Q - \{dp_j\}, dp_i) + \delta_{ij}\} \quad (4)$$

$$\delta_{ij} = \begin{cases} 1 & \text{if } t'_{dc,R}(dp_j) \leq dp_j.e \\ 0 & \text{otherwise} \end{cases}$$

Here, $\delta_{ij} = 1$ means that dp_j can be finished after appending dp_j to R'.

When Q contains only one delivery point dp_j, the problem is trivial. When $|Q| > 1$, we need to search through Q to examine all possibilities of valid delivery point sets and find the particular dp_i that achieves the optimum value of $opt(Q, dp_j)$. Algorithm 1 outlines the structure of this procedure. Note that we use $pre(Q, dp_j)$ to record the second-last delivery point dp_i before achieving $opt(Q, dp_j)$ to facilitate the reconstruction of the valid task sequence R^*. After initialization, the algorithm generates and processes sets in the increasing order of their size from 2 to n (lines 6–7). For each delivery point $dp_j \in Q$, it computes $opt(Q, dp_j)$ and $pre(Q, dp_j)$ according to Equation 4 (lines 9–10). If $\delta_{ij} = 1$, Q is added to Q_{dc} (lines 11–12). Then we construct R^* from tables opt and pre (line 13). Algorithm 1 has time complexity $O(2^{|dc.DP|} \cdot |dc.DP|^3)$, where $|dc.DP|$ is the number of delivery points for delivery center dc.

Algorithm 1: C-VDPS

```
Input: $dc$, $dc.DP$
Output: $Q_{dc}$
1  $Q_{dc} \leftarrow null$;
2  for each task $dp_i$ in $dc.DP = \{dp_1, dp_2, ..., dp_n\}$ do
3      $opt(\{dp_i\}, dp_i) \leftarrow 1$;
4      $Q_{dc} \leftarrow Q_{dc} \cup \{\{dp_i\}\}$;
5      $pre(\{dp_i\}, dp_i) \leftarrow null$;
6  for $len = 2$ to $n$ do
7      for each subset $Q \subseteq dc.DP$ of size $len$ do
8          for each $dp_j \in Q$ do
9              $opt(Q, dp_j) \leftarrow \max_{dp_i \in Q, dp_i \neq dp_j} opt(Q - \{dp_j\}, dp_i) + \delta_{ij}$;
10             $pre(Q, dp_j) \leftarrow arg \max_{dp_i \in Q, dp_i \neq dp_j} opt(Q - \{dp_j\}, dp_i) + \delta_{ij}$;
11             if $\delta_{ij} = 1$ then
12                 $Q_{dc} \leftarrow Q_{dc} \cup \{Q\}$;
13 compute $R^*$ based on $opt$ and $pre$;
14 return $Q_{dc}$
```

During the process of C-VDPS generation, for each C-VDPS, we only record the delivery point sequence with the minimal travel time (which means the tasks on this delivery point sequence can bring workers the highest payoffs based on Definition 7). After obtaining the C-VDPSs, we check whether a C-VDPS is valid for each worker by considering the travel time between the worker and the distribution center, as well as the tasks' expiration times.

Distance-constrained Pruning Strategy. We observe that a C-VDPS in which adjacent delivery points are far away from each other tends yield a low worker payoff, which means that workers are unwilling to move to the delivery points in this C-VDPS. Based on this observation, we design a distance-constrained pruning strategy that prunes the long-distance delivery points when generating a delivery point sequence, thus improving the efficiency of the algorithm. Specifically, when performing Equation 4, we only search for a delivery point set (denoted as $D(dp_j)$) with less than ϵ travel distance to the current delivery point (dp_j), i.e., $D(dp_j) = \{dp_g | d(dp_j.l, dp_g.l) \leq \epsilon\}$, where $d(a, b)$ is the travel distance between location a and b. Here, ϵ is a threshold that can be set by observation from a real application or specified by the SC platform. We study the effect of ϵ in our experimental part in Section VII-B. The studies show that the pruning strategy with a suitable ϵ can result in the same task assignment result as does VDPS generation without pruning, and it can improve the efficiency of the VDPS generation. We take travel time into account when computing the VDPS sets for workers, and we provide a distance-constrained pruning strategy to improve the calculation efficiency of VDPS. A worker who is far from the distribution center often needs longer travel time and is less likely to be assigned suitable tasks. Moreover, a worker with longer travel distance tends to obtain a lower payoff, meaning that the worker is more likely to be disregarded in the task assignment due to the goal of the assignment.

V. Fairness-aware Game-Theoretic Approach

The fundamental nature of the FTA problem is that each worker needs to choose a set of delivery points to perform tasks to obtain the payoffs while interacting with other workers during the task assignment process, which implies that the choice of delivery points for a worker depends on the choices made by other workers. Game theory is widely used for modeling such interdependent decisions, where workers can be treated as independent players. Thus, the FTA system can be suitably formalized as a multi-player game. Game theory [24] studies problems in which players maximize their utilities and where the payoffs depend in part on the strategies of other players. Consequently, game theory seems to be an adequate tool to study task assignment in SC. Here we design a Fairness-aware Game-Theoretic (FGT) approach to solving the FTA problem while taking fairness and worker average payoff into account.

A. Inequity Aversion based Utility

When quantifying fairness, we use assignment fairness, which is related to the task assignment. *Inequity Aversion* is a common descriptive model for fairness, according to which people resist inequitable outcomes, i.e., they are willing to give up payoffs to move in the direction of more equitable outcomes [25]. Inspired by this, we define an Inequity Aversion based Utility (IAU) [26] that we will use in the game. Specifically, the IAU function is defined as follows.

$$IAU(w_i, VDPS(w_i)) = P(w_i) - \Big(\frac{\alpha}{|W|-1} MP(w_i) + \frac{\beta}{|W|-1} LP(w_i)\Big) \tag{5}$$

$$MP(w_i) = \sum_{P(w_j)>P(w_i), w_j \in W} (P(w_j) - P(w_i)) \tag{6}$$

$$LP(w_i) = \sum_{P(w_i)>P(w_j), w_j \in W} (P(w_i) - P(w_j)), \tag{7}$$

where $IAU(w_i, VDPS(w_i))$ denotes the inequity aversion-based utility of worker w_i achieved by finishing the tasks in $VDPS(w_i)$. Next, $P(w_i) = P(w_i, VDPS(w_i))$ denotes w_i' payoff (see Definition 7), MP is the total extra payoffs that other workers (whose payoffs are higher than that of w_i) can obtain, and LP is the total extra payoffs w_i can gain compared with those of others (whose payoffs are lower than that of w_i). Further, α and β are parameters that control the contributions of MP and LP. Overall, $IAU(w_i, VDPS(w_i))$ consists of two parts: the first part quantifies w_i's payoff by selecting $VDPS(w_i)$, while the second part quantifies the penalty on w_i due to the unfair task assignment. When $VDPS(w_i)$ is clear from the context, we use $IAU(w_i)$ to denote $IAU(w_i, VDPS(w_i))$.

B. Game Formulation and Pure Nash Equilibrium

Our FTA problem can be formulated as an n-player strategic game, $\mathcal{G} = (W, \mathbb{ST}, \mathbb{U})$, which consists of players, strategy spaces, and utility functions as follows:

i) $W = \{w_1, ..., w_n\}$ $(n \geq 2)$ is a finite set of workers that serve as game players. In the rest of the paper, we will use the terms player and worker interchangeably.

ii) $\mathbb{ST} = \cup_{i=1}^{n} ST_i$ is the overall strategy set for the players, i.e., the strategy space of the game. ST_i is a finite set of strategies available to worker w_i, which contains w_i's VDPS set and that null delivery point set $null$ (that captures the case where w_i does not choose any delivery tasks), denoted as $ST_i = \{\mathbb{VDPS}(w_i), null\}$ (where $\mathbb{VDPS}(w_i)$ indicates the VDPS set of worker w_i). A joint strategy is denoted as $\vec{st} = (st_1, st_2, ..., st_n) \in \mathbb{ST}$, where $st_i \in ST_i$ is the strategy chosen by player w_i $(0 < i \leq n)$.

iii) $\mathbb{U} = \cup_{i=1}^{n} U_i$ denotes the utility functions of the players, and U_i: $\mathbb{ST} \rightarrow \mathbb{R}$ is the utility function of player w_i. For every joint strategy $\vec{st} \in \mathbb{ST}$, $U_i(\vec{st}) = IAU(w_i, VDPS(w_i)) \in \mathbb{R}$ represents the utility of player w_i, where tasks in $VDPS(w_i)$ are assigned to worker w_i under the joint strategy $\vec{st}$. When $\vec{st}$ is clear from the context, we use U_i to denote $U_i(\vec{st})$.

Taking the example in Figure 1, workers w_1 and w_2 are the players. For worker w_1, $\{dp_1\}$, $\{dp_2\}$, $\{dp_1, dp_2\}$, and $\{dp_1, dp_2, dp_3\}$ are four of the worker's VDPSs, each of which represents a strategy. For workers w_1 and w_2, $\vec{st} = (\{dp_1, dp_2\}, \{dp_3, dp_4, dp_5\})$ represents one of their joint s-trategies, based on which we can calculate w_1's utility, i.e., $U_1(\vec{st}) = IAU(w_1, \{dp_1, dp_2\}) = 2.42$, based on Equation 5 (where α and β are set to 0.5).

In our problem, since each worker needs to have a deterministic strategy, i.e., selecting a VDPS with delivery tasks or doing nothing, we only consider pure strategies (i.e., deterministic strategies), which means that the probability of a strategy that worker w_i can choose from ST_i is 1, while the probabilities of the remaining strategies from ST_i are 0. Next we prove that the FTA game has a pure Nash Equilibrium (NE), where each player performs a deterministic strategy. A pure NE is a state of the game (that is a satisfying task assignment), where no single worker is able to improve their utility by changing their strategy when other workers do not change their strategies. We first introduce the concept of Exact Potential Game (EPG) that has at least one pure NE [27].

Definition 9 (Exact Potential Game): A strategic game, $\mathcal{G} = (W, \mathbb{ST}, \mathbb{U})$, is an EPG if a function $\Phi : \mathbb{ST} \rightarrow \mathbb{R}$ exists, such that for all $\vec{st}_i \in \mathbb{ST}$, it holds that for all $w_i \in W$ that

$$U_i(st_i', \vec{st}_{-i}) - U_i(st_i, \vec{st}_{-i}) = \Phi(st_i', \vec{st}_{-i}) - \Phi(st_i, \vec{st}_{-i}), \tag{8}$$

where st_i' and st_i are the strategies that can be selected by worker w_i, $\vec{st}_{-i}$ is the joint strategy of the other workers except for worker w_i, and the function Φ is an exact potential function for game $\mathcal{G}$.

Lemma 2: The FTA game is an EPG that has at least one pure NE.

Proof 2: In the FTA problem, we define the potential function as $\Phi(\vec{st}) = \sum_{w_i \in W} U_i(\vec{st}) = \sum_{w_i \in W} IAU(w_i, VDPS(w_i))$, which represents the sum of utilities of all workers in W. Then it can be obtained that,

$$\begin{aligned}
&\Phi(st_i', \vec{st}_{-i}) - \Phi(st_i, \vec{st}_{-i}) \\
=&\Big(IAU(w_i, VDPS'(w_i)) + \sum_{w \in W - w_i} IAU(w, VDPS(w))\Big) \\
&- \Big(IAU(w_i, VDPS(w_i)) + \sum_{w \in W - w_i} IAU(w, VDPS(w))\Big) \\
=&IAU(w_i, VDPS'(w_i)) - IAU(w_i, VDPS(w_i)) \\
=&U_i(st_i', \vec{st}_{-i}) - U_i(st_i, \vec{st}_{-i}),
\end{aligned} \tag{9}$$

where the VDPSs selected in strategies st_i' and st_i are $VDPS'(w_i)$ and $VDPS(w_i)$, respectively. From Equation 9, we can see that the strategic game of the FTA problem is an EPG; thus, the FTA game has at least one pure NE.

C. Best-response Mechanism

Next, we employ a best-response mechanism, a basic tool in EPGs to address conflicts that occur among players, which finally achieves a pure NE [24]. Specifically, the best-response mechanism consists of players taking turns to find their best-response strategies (with maximal utilities) based on the most recent strategies chosen by the others, which ends up reaching the pure NE, corresponding to a satisfying task assignment. We detail the best-response mechanism in Algorithm 2.

Given a distribution center set DC, a worker set W, a delivery point set DP, and a task set S to be assigned, the task assignment A is initialized as $\emptyset$ (line 1). For each worker $w_i \in W$, the worker's VDPS set $\mathbb{VDPS}(w_i)$ is first obtained from DP as described in Section IV (line 4). Then the algorith-m randomly assigns each worker w_i a $VDPS(w_i)$ (with only

Algorithm 2: Fairness-aware Game-Theoretic (FGT) Approach

```
Input: Distribution center set $DC$, worker set $W$, delivery point set $DP$, task set $S$
Output: Task assignment: $A$
1  $A \leftarrow \emptyset$;
2  $W' \leftarrow W$;
3  for each worker $w_i \in W$ do
4      Obtain the VDPS set, $\mathbb{VDPS}(w_i)$, from $DP$ (according to Section IV);
5      $\mathbb{VDPS}'(w_i) \leftarrow \mathbb{VDPS}(w_i)$;
6  for each worker $w_i \in W'$ do
7      if $w_i$ has VDPSs then
8          Randomly assign a $VDPS(w_i) \in \mathbb{VDPS}'(w_i)$ (where $|VDPS(w_i)| = 1$) to $w_i$;
9          $w_i.st \leftarrow VDPS(w_i)$;
10         Compute $w_i$'s utility based on IAU (Equation 5);
11         $W' \leftarrow W' - w_i$;
12         for each worker $w_j \in W'$ do
13             $\mathbb{VDPS}'(w_j) \leftarrow \mathbb{VDPS}'(w_j) - VDPS(w_i)$;
14     else
15         $w_i.st \leftarrow null$;
16         $W' \leftarrow W' - w_i$;
17 $t \leftarrow 1$;
18 repeat
19     for each worker $w_i \in W$ do
20         Find the best-response VDPS, $VDPS^*(w_i)$, for $w_i$;
21         /*$VDPS^*(w_i)$ can be calculated by Equation 10; */
22         $w_i.st \leftarrow VDPS^*(w_i)$;
23     $t \leftarrow t + 1$;
24 until $W.\vec{st}^t = W.\vec{st}^{t-1}$;
25 /*$W.\vec{st}^t$ denotes the strategies of all the workers in the $t$th iteration*/
26 update $A$;
27 return $A$;
```

one valid delivery point) as the worker's strategy and computes the utility accordingly (lines 6–16). After that, Algorithm 2 iteratively adjusts each worker's strategy according to the worker's best-response strategy that maximizes the worker's utility based on the current joint strategies of the others until a NE (i.e., no one changes their strategy) is found (lines 18–24). The best-response VDPS with the maximal utility, denoted by $VDPS^*(w_i)$, for a worker $w_i \in W$ can be calculated by Equation 10.

$$\begin{aligned} VDPS^*(w_i) &= argmax_{VDPS(w_i) \in \mathbb{VDPS}(w_i)} U_i(\vec{st}) \\ &= argmax_{VDPS(w_i) \in \mathbb{VDPS}(w_i)} \Big(IAU\big(w_i, VDPS(w_i)\big)\Big) \end{aligned} \tag{10}$$

In each iteration, only one worker is allowed to adapt their strategy (i.e., selecting their best-response strategy) to maximize their utility, and the game is supposed to be played in sequence. Finally, an NE is achieved (line 24), and we update the task assignment A according to the NE (line 26). The time complexity of Algorithm 2 is $O(2^{|dc.DP|} \cdot |dc.DP|^3 + |W|^2 + \mathcal{T} \cdot |W| \cdot |maxVDPS|)$, where the first term is the complexity of Algorithm 1, $|dc.DP|$ is the number of delivery points for delivery center dc, $|W|$ is the number of workers, $\mathcal{T}$ is the number of iterations needed to adjust each worker's best-response strategy until a Nash equilibrium is reached, and $|maxVDPS|$ is the maximum number of valid delivery points among all workers (i.e., $|maxVDPS| = \max_{w \in W, VDPS(w) \in \mathbb{VDPS}(w)} |VDPS(w)|$).

VI. Improved Evolutionary Game-Theoretic Approach

A. Motivation and Overview

The classical game-theoretic approach assumes that players are rational, meaning that the players always aim to maximize their own utilities (i.e., IAUs) and have the ability of making correct decisions. However, in real SC scenarios, the workers (i.e., players) are better described as being bounded rational [28], meaning that they cannot always optimize their own utilities. Motivated by these concerns, we design a task assignment algorithm based on evolutionary game theory that offers means of contending with bounded rationality by learning during strategic interactions. The existing evolutionary game-theoretic algorithms generally concentrate on two or several strategies for workers, in which multiple players are allowed to choose the same strategy and obtain identical payoffs [28]. However, in our FTA problem, the strategy of each worker is different, which often leads to a different payoff. Therefore, we design an Improved Evolutionary Game-Theoretic (IEGT) approach that fits our problem. More specifically, we first model the task assignment game and then define an Improved Evolutionary Stable Strategy (IESS), which leads to an evolutionary equilibrium. Next, a replicated dynamic mechanism is used to produce the evolutionary equilibrium that leads to a satisfied fair task assignment, where workers can obtain similar payoffs. The experimental results (in Section VII) show that the IEGT algorithm converges and generates better task assignments (in terms of workers' payoff differences) than the classical game-theoretic algorithm.

B. Evolutionary Game Formulation and Improved Evolutionary Stable Strategy

The FTA problem can be formulated as an evolutionary game, $\mathcal{G} = (W, \mathbb{ST}, \mathbb{U}, \mathbb{G})$, which is comprised of players, strategy spaces, utility functions, and populations. The settings of players and strategy spaces are same as those in the classical game described in Section V-B. In this evolutionary game, the utility functions are represented by workers' payoffs, i.e., given a player w_i and a joint strategy $\vec{st} \in \mathbb{ST}$, the worker's utility is defined as follows: $U_i(\vec{st}) = P(w_i, VDPS(w_i))$. In the FTA problem, workers who work for a certain distribution center compete for tasks. Therefore, these workers can be regarded as a population, and they co-evolve (towards an evolutionary equilibrium) in a repeated game framework. There exist K populations ($\mathbb{G} = \{G_1, G_2, ..., G_K\}$, that work for the distribution centers $\{dc_1, dc_2, ..., dc_K\}$.

The notion of evolutionary stable strategy is adopted widely in evolutionary game theory. To design an Improved Evolutionary Stable Strategy (IESS), we use $U_i(st_i, \vec{st}_{-i})$ to denote the utility of player w_i using strategy st_i to play against the other players that use strategy $\vec{st}_{-i}$. The following is a formal definition of an IESS.

Definition 10 (Improved Evolutionary Stable Strategy): A strategy $\vec{st}^* = \{st_i^*, \vec{st}^*_{-i}\}$ is an IESS if and only if for all $st_i \neq st_i^*$, it satisfies the following two conditions:

i) equilibrium condition: $U_i(st_i, \vec{st}^*_{-i}) \leq U_i(st^*_i, \vec{st}^*_{-i})$

ii) weak stability condition: if $U_i(st_i, \vec{st}^*_{-i}) = U_i(st^*_i, \vec{st}^*_{-i})$ then $U_i(st_i, \vec{st}_{-i}) \leq U_i(st^*_i, \vec{st}_{-i})$

Condition i) guarantees that st^*_i is the best-response strategy to player w_i and can achieve an NE. Condition ii) is a weak stability condition. If all members of a population adopt the IESS, no mutant strategy (leading to a different payoff for a worker) can invade the population under the influence of natural selection, called *evolutionary equilibrium*. In other words, if a strategy is evolutionarily stable, it must have the property that the utility of a player following an IESS must exceed the utility of a player following a mutant strategy; otherwise, players following a mutant strategy would be able to invade the population.

C. Replicator Dynamics Mechanism

The key concept of evolutionary games is *replicator dynamics*, which describes the evolution of strategies over time. During the evolution process, the percentage of players (called *population share*) using a certain pure strategy may change, and each player will adapt their strategy according to the utility achieved. This adaptation process can be modeled by a set of *replicator dynamics equations*:

$$\dot{\sigma}_{km}(t) = \sigma_{km}(t)\left(U_i^{km}(t) - \bar{U}^k(t)\right) \tag{11}$$

$$\sigma_{km}(t) = \frac{N(VDPS_m)}{|G_k|} \tag{12}$$

$$N(VDPS_m) = \begin{cases} 1 & \text{if a worker in group } G_k \text{ choose } VDPS_m \\ 0 & \text{otherwise} \end{cases} \tag{13}$$

$$\bar{U}^k(t) = \sum_{m=1}^{|\mathbb{VDPS}(G_k)|} \sigma_{km}(t) \cdot U_i^{km}(t), \tag{14}$$

where $\dot{\sigma}_{km}(t)$ denotes the replicator dynamics for players in population G_k under strategy $VDPS_m$ at time t. Further, $\sigma_{km}(t)$ is the population share of strategy $VDPS_m$ in population G_k at time t (i.e., the fraction of population G_k that uses strategy $VDPS_m$ at time t), which is calculated by Equations 12 and 13. We have $\sigma_{km}(t) \geq 0$. Next, $U_i^{km}(t) = P_{w_i \in G_k, VDPS_m \in \mathbb{VDPS}(w_i)}(w_i, VDPS_m)$ denotes the utility for worker w_i (who selects strategy $VDPS_m$ in population G_k at time t), and $\mathbb{VDPS}(w_i)$ and $\mathbb{VDPS}(G_k)$ denote the VDPS sets of worker w_i and population G_k, respectively. Finally, $\bar{U}^k(t)$ is the average utility for players in population G_k at time t, which is calculated by Equation 14.

An evolutionary equilibrium is defined as a stable fixed point of the replicator dynamics. When a population of players evolves over time following the replicator dynamics, it will converge to the evolutionary equilibrium, which can be obtained only if the payoffs for all players in a population are equal. In our solution, we improve the original evolutionary equilibrium considering the fact workers are assigned different delivery point sets with different payoffs. In the improved evolutionary equilibrium, the payoffs for all players in a population are similar, and the state of each workers is stable.

Based on the improved evolutionary equilibrium and replicator dynamics mechanism, we propose a dynamic algorithm (see Algorithm 3) for the evolutionary game. The evolutionary equilibrium is achieved through strategy adaptation. In each iteration, all players in a population compare their utilities with the average utility of the whole population. Once their utilities are lower than the average utility, they must select another strategy with a higher payoff to avoid being eliminated. Specifically, Algorithm 3 takes a population G_k, delivery point set DP, and the corresponding task set S as input. Like Algorithm 2, this algorithm starts by calculating valid delivery point sets for each worker (lines 3–4) and randomly assigning a strategy to each worker (lines 6–16). Then we take a random assignment scheme to attain the evolutionary equilibrium (lines 17–27). Specifically, in each iteration, we adjust the strategy for each worker $w_i \in G_k$ based on the current replicator dynamics $\dot{\sigma}_{km}(t)$, i.e., we randomly assign w_i another valid delivery point set ($VDPS'(w_i)$) that can bring the worker a higher payoff when $\dot{\sigma}_{km}(t) < 0$ (lines 22–25). This process follows Darwin's On the Origin of Species, which posits that most individuals in a population tend to evolve to higher levels due to natural selection. This means that workers whose payoffs are lower than the average have only two choices: to evolve or to be eliminated. The VDPSs that bring lower payoffs to workers are likely to be abandoned by workers.

Algorithm 3: Improved Evolutionary Game-Theoretic (IEGT) Approach

```
Input: A population $G_k$, delivery point set $DP$, task set $S$
Output: Task assignment: $A$
1  $A \leftarrow \emptyset$;
2  $G'_k \leftarrow G_k$;
3  for each worker $w \in G_k$ do
4      Obtain the VDPS set, $\mathbb{VDPS}(w)$, from $DP$ (according to Section IV);
5      $\mathbb{VDPS}'(w) \leftarrow \mathbb{VDPS}(w)$;
6  for each worker $w_i \in G'_k$ do
7      if $w_i$ has VDPSs then
8          Randomly assign a $VDPS(w_i) \in \mathbb{VDPS}'(w_i)$ (where $|VDPS(w_i)| = 1$) to $w_i$;
9          $w_i.st \leftarrow VDPS(w_i)$;
10         Compute $w_i$'s utility based on the payoff (Equation 1);
11         $G'_k \leftarrow G'_k - w_i$;
12         for each worker $w_j \in G'_k$ do
13             $\mathbb{VDPS}'(w_j) \leftarrow \mathbb{VDPS}'(w_j) - VDPS(w_i)$;
14     else
15         $w_i.st \leftarrow null$;
16         $G'_k \leftarrow G'_k - w_i$;
17 $t \leftarrow 1$;
18 $p \leftarrow 0$;
19 repeat
20     for each worker $w_i \in G_k$ do
21         Compute the replicator dynamics $\dot{\sigma}_{km}(t)$ according to Equation 11;
22         if $\dot{\sigma}_{km}(t) < 0$ then
23             if $w_i$ has another VDPSs that can bring a higher payoff then
24                 Randomly assign $VDPS'(w_i)$ with a higher payoff to $w_i$;
25                 $w_i.st \leftarrow VDPS'(w_i)$;
26     $t \leftarrow t + 1$;
27 until $\dot{\sigma}_k(t) = 0$ or $G_k.\vec{st}^t = G_k.\vec{st}^{t-1}$;
28 /*$\dot{\sigma}_k(t)$ denotes the replicator dynamics of all the workers in the $t$th iteration,
   and $G_k.\vec{st}^t$ denotes the strategies of all the workers in the $t$th iteration*/
29 update $A$;
30 return $A$;
```

As we can see from Algorithm 3, the task assignment game is played repeatedly and evolves over iterations until the utilities for all the workers in the population are equal (i.e., $\dot{\sigma}_k(t) = 0$). However, in our problem, the strategies adopted by workers differ, which may lead to different payoffs. Therefore, we improve the iteration termination by adding the condition that no one changes their strategy (i.e., $G_k.\vec{st}^{\,t} = G_k.\vec{st}^{\,t-1}$) (line 27). It is easy to see that once either condition is met, an IESS (see Definition 10) is achieved. The time complexity of Algorithm 3 is $O(2^{|dc.DP|} \cdot |dc.DP|^3 + |W|^2 \cdot |maxVDPS| + \mathcal{T} \cdot |W| \cdot |maxVDPS|)$, where the first term is the complexity of Algorithm 1, $|W|$ is the number of workers, $|maxVDPS|$ is the maximum number of valid delivery points among all workers, and $\mathcal{T}$ is the number of iterations.

VII. Experimental Evaluation

A. Experimental Setup

Datasets. The experiments are carried out on two datasets: *gMission* (GM) and *synthetic* (SYN). gMission is an open source SC dataset [29]. Each task is associated with a location, an expiration time, and a reward, and each worker is associated with a location. As the data has no associated distribution center (that distributes tasks), we compute the centroid of all the tasks as the center, i.e., $dc.l = (\bar{x}, \bar{y}) = (\sum_{i=1}^{|S|} \frac{x_i}{|S|}, \sum_{i=1}^{|S|} \frac{y_i}{|S|})$, where $|S|$ is the number of tasks and (x_i, y_i) is the location of task $s_i \in S$. We adopt k-Means clustering to obtain x clusters ($x = 20, 40, 60, 80, 100$), whose centroids are regarded as delivery points, and tasks in each cluster are then to be delivered to the corresponding delivery point.

To obtain synthetic datasets, we generate locations of workers and delivery points following a uniform distribution within the $2D$ space $[0, 100]^2$ based on observations from real datasets (e.g., gMission). It is common practice in experimental studies of SC platforms to use uniformly (and randomly) distributed attribute values [7], the argument being that this captures the effects of the attributes on a more fair basis. We randomly generate 50 distribution centers in this space. Each worker and delivery point are associated with a distribution center at random, which means that a worker works for a particular distribution center and a distribution center distributes tasks to a particular set of delivery points. Delivery tasks are associated at random with a delivery point. The reward of each task is set to 1, and the speed of workers in both datasets is set to 5 km/h. Since task assignment across distribution centers is independent, we can perform task assignment for different distribution centers in parallel.

Evaluation Methods. Among the few existing studies of fair task assignment, no study considers the same problem (with the goal of minimizing the payoff difference among workers) or gives a method that can solve the proposed problem, and thus we were unable to identify a baseline. Instead, we use two typical task assignment algorithms that do not take fairness into consideration as baselines. We study the following algorithms.

i) MPTA: The Maximal Payoff based Task Assignment (MPTA) algorithm uses a tree-decomposition technique [30], [31]. After finding VDPSs for each worker, MPTA applies a tree-decomposition-based algorithm to identify the task assignment with maximal total payoffs.

TABLE I
Experiment Parameters

Parameter	Value
Distance threshold ϵ (km) (GM)	0.2, 0.4, 0.6, 0.8, 1
Distance threshold ϵ (km) (SYN)	0.5, 1, 1.5, 2, 2.5, 3, 3.5, 4
Number of tasks $\lvert S\rvert$ (GM)	100, 200, 300, 400, 500,
Number of tasks $\lvert S\rvert$ (SYN)	25K, 50K, 75K, 100K, 125K
Number of workers $\lvert W\rvert$ (GM)	20, 40, 60, 80, 100
Number of workers $\lvert W\rvert$ (SYN)	1K, 2K, 3K, 4K, 5K
Number of delivery points $\lvert DP\rvert$ (GM)	20, 40, 60, 80, 100
Number of delivery points $\lvert DP\rvert$ (SYN)	3K, 3.5K, 4K, 4.5K, 5K
Expiration time of tasks e (SYN)	0.5h, 1h, 1.5h, 2h, 2.5h
Maximum acceptable delivery point number $maxDP$ (SYN)	1, 2, 3, 4

ii) GTA: The Greedy Task Assignment (GTA) algorithm assigns each worker the VDPS with the maximal payoff from the unassigned tasks. This proceeds until all tasks are assigned or all workers are exhausted.

iii) FGT: Our Fairness-aware Game-Theoretic (FGT) approach. We set α and β to 0.5 (when computing the IAU) so that the effects of MP (the total extra payoffs a worker can gain compared with those of the other workers) and LP (the total extra payoffs other workers can gain compared with that of the current worker) are weighted equally. We have tried other settings for α and β and have found that FGT works well when they are set to 0.5.

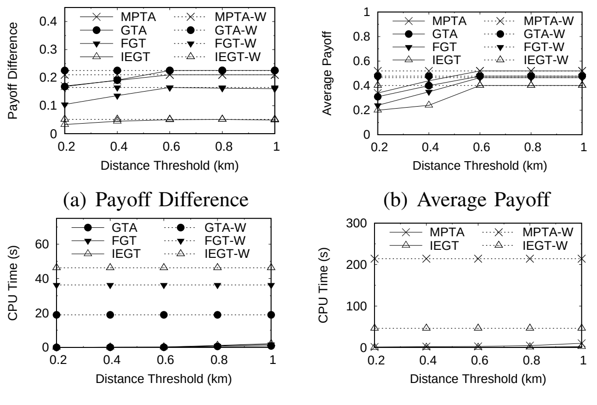
iv) IEGT: Our Improved Evolutionary Game-Theoretic (IEGT) approach.

Metrics. Three main metrics are compared for the above algorithms, i.e., *Payoff Difference* (see Equation 2), *Average Payoff*, and *CPU Time*, for finding task assignments. A small *Payoff Difference* value implies that the task assignment is fair, and a large *Average Payoff* value means that the task assignment offers high total utility.

Table I shows our experimental settings, where the default values of all parameters are underlined. All experiments are run on a 2*Intel Xeon Gold 6148 @ 2.40 GHz, 192 GB RAM.

B. Experimental Results

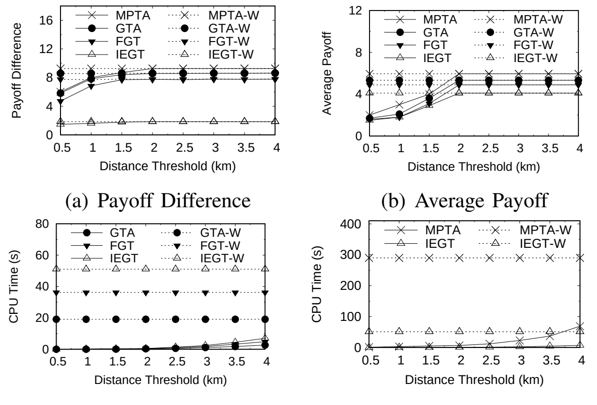
a) Effect of ϵ: We first study the effect of distance threshold ϵ, which is used to prune distant delivery points when finding delivery point sets (see Section IV). In this set of experiments, we also evaluate the performance of the dynamic programming algorithm without pruning by applying it into the task assignment algorithms, i.e., MPTA, GTA, FGT, and IEGT, denoted as MPTA-W, GTA-W, FGT-W, and IEGT-W, respectively. As shown in Figures 2 and 3, the task assignment methods (i.e., MPTA, GTA, FGT, and IEGT) with ϵ-constrained pruning can achieve the same effectiveness (including *payoff difference* and *average payoff*) as those (i.e., MPTA-W, GTA-W, FGT-W, and IEGT-W) without pruning when $\epsilon \geq 0.6$ ($\epsilon \geq 2$) in the GM (SYN) dataset, while improving the CPU time substantially. This demonstrates the superiority of the pruning strategy for solving the FTA problem. In Figures 2(a) and 3(a), the payoff differences for all approaches (with pruning) naturally increase as ϵ gets larger. However, we also notice that the increase becomes



(a) Payoff Difference (b) Average Payoff

(c) CPU Time (d) CPU Time

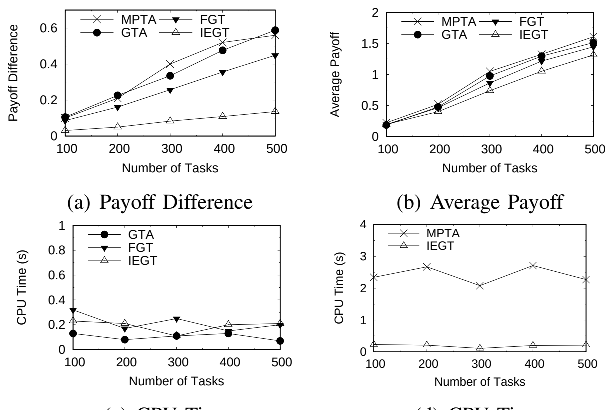
Fig. 2. Effect of ϵ on GM



(a) Payoff Difference (b) Average Payoff

(c) CPU Time (d) CPU Time

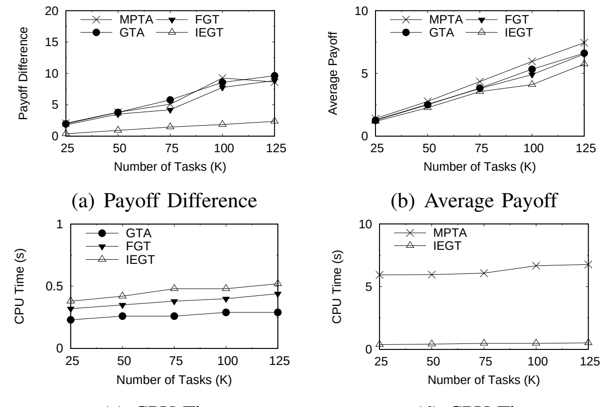
Fig. 3. Effect of ϵ on SYN



(a) Payoff Difference (b) Average Payoff

(c) CPU Time (d) CPU Time

Fig. 4. Effect of $|S|$ on GM



(a) Payoff Difference (b) Average Payoff

(c) CPU Time (d) CPU Time

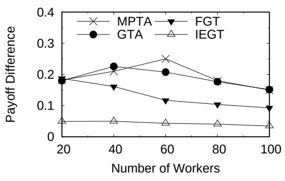
Fig. 5. Effect of $|S|$ on SYN

less pronounced when $\epsilon \geq 0.6$ in the GM dataset and when $\epsilon \geq 2$ in the SYN dataset. This is due to the fact that with larger ϵ, the distances between adjacent delivery points in generated delivery point sequences increase, which means that the sequences bring a lower payoff to the workers and will be abandoned by the approaches. In other words, the different approaches focus on delivery point sequences with inter-point distances that are less than ϵ ($\epsilon \geq 0.6$ in GM and $\epsilon \geq 2$ in SYN). In Figures 2(b) and 3(b), the average payoff is not affected substantially when varying $\epsilon \geq 0.6$ in GM and $\epsilon \geq 2$ in SYN. Although IEGT generates a smaller average payoff than the others, it is able to achieve the most fair task assignment, as shown in Figures 2(a) and 3(a). In Figures 2(c), 2(d), 3(c), and 3(d), although the CPU time of all algorithms with pruning increases as ϵ increases, our proposed algorithms clearly outperform MPTA.

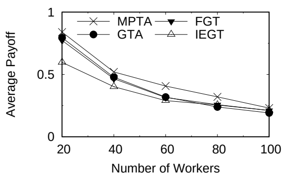
b) Effect of $|S|$: Next we study the effect of $|S|$. From Figures 4(a), 4(b), 5(a), and 5(b), we can see that the payoff differences and average payoffs of all methods exhibit a similar increasing trend when $|S|$ grows. The reason is that a larger $|S|$ means that workers can obtain higher payoffs, which also results in more unfair assignments. We see that MPTA generates the highest average payoff, followed by GTA, FGT, and IEGT. However, the payoff difference of IEGT is consistently lower than those of the others, which demonstrates the superiority of the evolutionary game strategy. In particular, the payoff difference of IEGT is only 18.0%–27.3% of that of MPTA, 19.0%–29.2% of that of GTA, and 20.8%–34.6% of that of FGT. Considering CPU time, Figures 4(c), 4(d), 5(c), and 5(d) show that the methods are almost unaffected by variations in $|S|$. The reason is that we only assign a suitable delivery point sequence to each worker so that the worker can perform all the tasks associated with each delivery point; we ignore the order and time of processing tasks.

c) Effect of $|W|$: Next, we study the effect of $|W|$. As can be seen in Figures 6 and 7, IEGT and FGT can achieve a more fair task assignment than GTA while sacrificing some efficiency. However, the computational efficiency of IEGT and FGT are acceptable. The payoff differences of all methods except IEGT decrease with growing $|W|$ on GM (when $|W| \geq 60$) and SYN, which means that workers tend to be treated fairly when more workers are involved. IEGT is relatively stable when varying $|W|$, which highlights the evolutionary stability in the game, i.e., once an evolutionary equilibrium is achieved, it will be stable even when more workers are involved. Although MPTA generates the highest average payoff, it assigns tasks more unfairly than the game-theoretic approaches, and it is the most time-consuming.

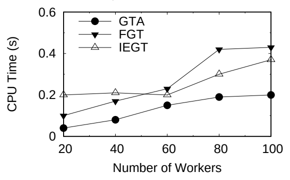
d) Effect of $|DP|$: As expected, the payoff differences of all the algorithms decline gradually as $|DP|$ grows (see Figures 8(a) and 9(a)). This is due to the fact that with more delivery points, workers tend to have more strategies to choose among to reduce the payoff difference among them. Figures 8(b) and 9(b) show that the average payoffs of all methods also exhibit a downward trend when increasing $|DP|$ because this has the effect that fewer tasks are located at



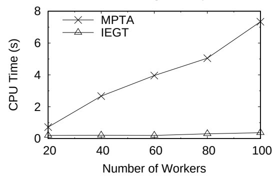
(a) Payoff Difference



(b) Average Payoff

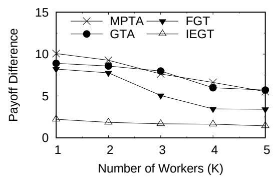


(c) CPU Time

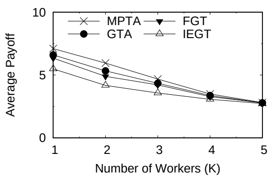


(d) CPU Time

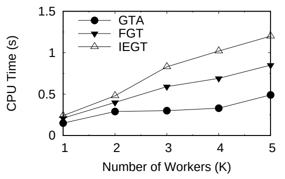
Fig. 6. Effect of $|W|$ on GM



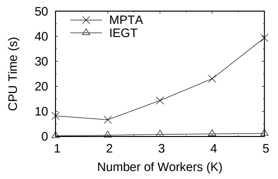
(a) Payoff Difference



(b) Average Payoff



(c) CPU Time

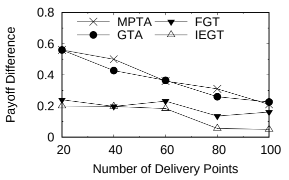


(d) CPU Time

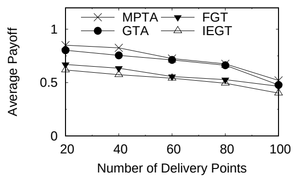
Fig. 7. Effect of $|W|$ on SYN

each delivery point, which yields lower payoffs. Although the average payoff of MPTA is the highest, it is the most time-consuming, as shown in Figures 8(d) and 9(d). The CPU costs of the other methods are almost negligible compared to that of MPTA. To save space, in the following experiments, we disregard the uncompetitive CPU time of MPTA, and we do not report results for the GM dataset, as these are similar to those obtained for the SYN dataset.

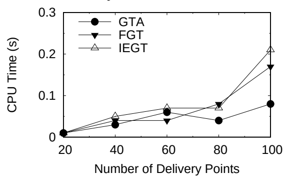
e) Effect of e: As illustrated in Figure 10(a), the payoff differences of the methods first increase and then remain stable. This may be due to the fact that when e is first increased, there will be more reachable delivery points for each worker. Workers then have more strategy choices, which may lead to the increasing payoff differences. After e reaches a



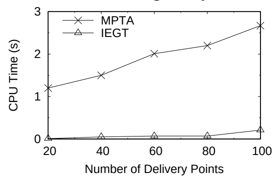
(a) Payoff Difference



(b) Average Payoff

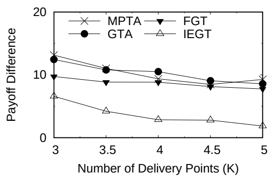


(c) CPU Time

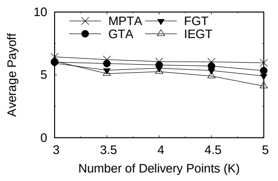


(d) CPU Time

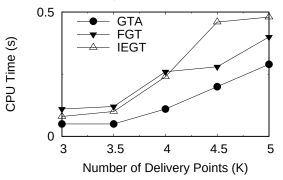
Fig. 8. Effect of $|DP|$ on GM



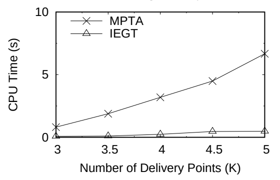
(a) Payoff Difference



(b) Average Payoff

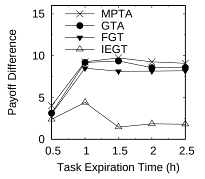


(c) CPU Time

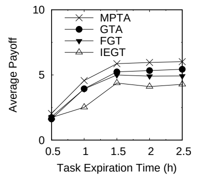


(d) CPU Time

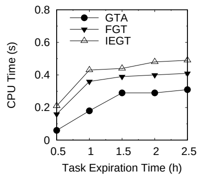
Fig. 9. Effect of $|DP|$ on SYN



(a) Payoff Difference



(b) Average Payoff



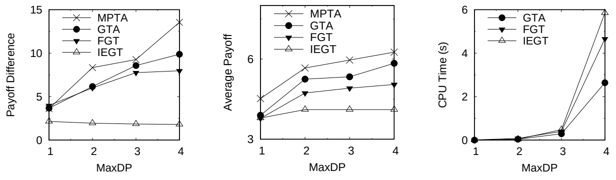
(c) CPU Time

Fig. 10. Effect of e on SYN

certain level (i.e., $e \geq 1.5$), the reachable delivery point set of each worker remains unchanged, thus the payoff differences keep stable. Not surprisingly, Figures 10(b) and 10(c) show that all the approaches can initially achieve higher average payoffs with more CPU time when the deadlines are relaxed. A larger e means that each worker on average has more reachable delivery points, which contributes to achieving higher average payoffs and incurs more CPU time. When $e \geq 1.5$, the average payoffs and CPU times of the methods remain stable for the same reason that the changes in payoff differences, i.e., the reachable delivery point set for each worker is unchanged, remain stable when $e \geq 1.5$.

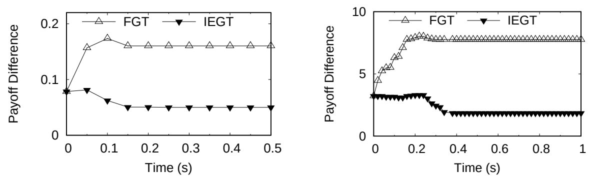
f) Effect of $maxDP$: We proceed to consider the effect of $maxDP$, the maximum acceptable number of delivery points for workers. In Figure 11(a), the payoff differences of MPTA, GTA, and FGT increase with $maxDP$. We also see that the payoff difference gap between IEGT and other approaches increases. This is due to the fact that when applying IEGT, the evolutionary point is relatively stable, leading to a satisfied fair task assignment, in which workers obtain similar payoffs regardless of $maxDP$. It is noteworthy that the payoff difference generated by IEGT is only 13.3%–59.3% of those achieved by the other algorithms. With the increase of $maxDP$, workers have more reachable delivery points and tasks, which offers more opportunities to select tasks with higher rewards, explaining the rising average payoff in Figure 11(b). Considering CPU time in Figure 11(c), IEGT and FGT are more time-consuming than GTA, as they select strategies for workers iteratively, while GTA performs strategy selection only once.

g) Convergence of Game-Theoretic Approaches: The question of convergence to an equilibrium has received significant attention in the game theory field. Therefore, we illustrate the convergence of our algorithms to an equilibrium in Figure 12, enabling us to conclude that the algorithms are



(a) Payoff Difference (b) Average Payoff (c) CPU Time

Fig. 11. Effect of $maxDP$ on SYN



(a) Convergence on GM (b) Convergence on SYN

Fig. 12. Convergence of FGT and IEGT

convergent.

VIII. Conclusion

We propose and offer solutions to a problem termed Fairness-aware Task Assignment, that aims to achieve assignment fairness among workers. In order to achieve high fairness, high average worker payoff, and high running time efficiency, we propose a dynamic programming algorithm along with a pruning strategy that generates valid delivery point sets for workers, and we design two game-theoretic task assignment methods, i.e., a Fairness-aware Game-Theoretic method and an Improved Evolutionary Game-Theoretic method, that target fair task assignments with good average worker payoff. To the best of our knowledge, this is the first study in spatial crowdsourcing that considers fairness by building on concepts from game theory. An empirical study with real and synthetic datasets offers evidence that the paper's proposals improve on the state of the art in terms of fairness and computational efficiency while offering acceptable average worker payoffs. One interesting research direction is to introduce additional descriptive models of fairness, e.g., priority-aware fairness, into spatial crowdsourcing task assignment. Other directions include to improve the game-theoretic algorithm's efficiency by enabling early termination of iterations, and to explore task assignment that redefines worker payoff by considering workers with different contributions to tasks.

Acknowledgment

This work is partially supported by Natural Science Foundation of China (No. 61972069, 61836007 and 61832017).

References

[1] P. Cheng, L. Chen, and J. Ye, "Cooperation-aware task assignment in spatial crowdsourcing," in *ICDE*, 2019, pp. 1442–1453.

[2] H. Gintis, *Game theory evolving: a problem-centered introduction to modeling strategic interaction*. Princeton University Press, 2001.

[3] S. P. Schappe, "Understanding employee job satisfaction: the importance of procedural and distributive justice," *Journal of Business and Psychology*, vol. 12, no. 4, pp. 493–503, 1998.

[4] R. Borromeo, T. Laurent, M. Toyama, and S. Amer-Yahia, "Fairness and transparency in crowdsourcing," in *EDBT*, 2017, pp. 466–469.

[5] D. Durward, I. Blohm, and J. M. Leimeister, "Is there papa in crowd work?: a literature review on ethical dimensions in crowdsourcing," in *UIC*, 2016, pp. 823–832.

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

[7] Y. Tong, J. She, B. Ding, and L. Wang, "Online mobile micro-task allocation in spatial crowdsourcing," in *ICDE*, 2016, pp. 49–60.

[8] Y. Tong, L. Wang, Z. Zhou, L. Chen, B. Du, and J. Ye, "Dynamic pricing in spatial crowdsourcing: a matching-based approach," in *SIGMOD*, 2018, pp. 773–788.

[9] Y. Tong, L. Wang, Z. Zhou, B. Ding, L. Chen, J. Ye, and K. Xu, "Flexible online task assignment in real-time spatial data," *PVLDB*, vol. 10, no. 11, pp. 1334–1345, 2017.

[10] Y. Zhao, J. Xia, G. Liu, H. Su, D. Lian, S. Shang, and K. Zheng, "Preference-aware task assignment in spatial crowdsourcing," in *AAAI*, 2019, pp. 2629–2636.

[11] Y. Zhao, K. Zheng, H. Yin, G. Liu, J. Fang, and X. Zhou, "Preference-aware task assignment in spatial crowdsourcing: from individuals to groups," *TKDE*, 2020.

[12] Y. Zhao, K. Zheng, Y. Cui, H. Su, F. Zhu, and X. Zhou, "Predictive task assignment in spatial crowdsourcing: a data-driven approach," in *ICDE*, 2020, pp. 13–24.

[13] X. Li, Y. Zhao, J. Guo, and K. Zheng, "Group task assignment with social impact-based preference in spatial crowdsourcing," in *DASFAA*, 2020, pp. 677–693.

[14] X. Li, Y. Zhao, X. Zhou, and K. Zheng, "Consensus-based group task assignment with social impact in spatial crowdsourcing," *DSE*, vol. 5, no. 4, pp. 375–390, 2020.

[15] Y. Cui, L. Deng, Y. Zhao, B. Yao, V. W. Zheng, and K. Zheng, "Hidden poi ranking with spatial crowdsourcing," in *SIGKDD*, 2019, pp. 814–824.

[16] Y. Zhao, J. Guo, X. Chen, J. Hao, X. Zhou, and K. Zheng, "Coalition-based task assignment in spatial crowdsourcing," in *ICDE*, 2021, pp. 1–12.

[17] J. Xia, Y. Zhao, G. Liu, J. Xu, M. Zhang, and K. Zheng, "Profit-driven task assignment in spatial crowdsourcing," in *IJCAI*, 2019, pp. 1914–1920.

[18] J. She, Y. Tong, L. Chen, and C. C. Cao, "Conflict-aware event-participant arrangement and its variant for online setting," *TKDE*, vol. 28, no. 9, pp. 2281–2295, 2016.

[19] S. P. Schappe, "Understanding employee job satisfaction: the importance of procedural and distributive justice," *Journal of Business and Psychology*, vol. 12, no. 4, pp. 439–503, 1998.

[20] Q. C. Ye, Y. Zhang, and R. Dekker, "Fair task allocation in transportation," *OMEGA*, vol. 68, pp. 1–16, 2017.

[21] 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 *AAAI*, vol. 33, no. 01, 2019, pp. 2245–2252.

[22] B. Fuat, G. Bugra, F. Hakan, and W. Kun-Lung, "Fair task allocation in crowdsourced delivery," *TSC*, vol. PP, pp. 1–14, 2018.

[23] V. V. Vazirani, *Approximation algorithms*. Springer Science and Business Media, 2013.

[24] D. Fudenberg and J. Tirole, *Game theory*. MIT Press, 1991.

[25] E. Fehr and K. M. Schmidt, "A theory of fairness, competition, and cooperation," *The Quarterly Journal of Economics*, vol. 114, no. 3, pp. 817–868, 1999.

[26] S. De Jong, K. Tuyls, K. Verbeeck, and N. Roos, "Priority awareness: towards a computational model of human fairness for multi-agent systems," in *AAMAS*, 2005, pp. 117–128.

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

[28] Z. J. Li and C.-T. Cheng, "An evolutionary game algorithm for grid resource allocation under bounded rationality," *Concurrency and Computation: Practice and Experience*, vol. 21, no. 9, pp. 1205–1223, 2009.

[29] 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," *PVLDB*, vol. 7, no. 13, pp. 1629–1632, 2014.

[30] Y. Zhao, Y. Li, Y. Wang, H. Su, and K. Zheng, "Destination-aware task assignment in spatial crowdsourcing," in *CIKM*, 2017, pp. 297–306.

[31] Y. Zhao, K. Zheng, Y. Li, H. Su, J. Liu, and X. Zhou, "Destination-aware task assignment in spatial crowdsourcing: A worker decomposition approach," *TKDE*, vol. 32, no. 12, pp. 2336–2350, 2020.