Fairness-Guaranteed Task Assignment for Crowdsourced Mobility Services

Yafei Li, Huiling Li, Baolong Mei, Xin Huang, Jianliang Xu, *Senior Member, IEEE*, and Mingliang Xu

***Abstract*—As a new computing paradigm, crowdsourced mobility service is booming with the rapid development of sharing economy. In the typical crowdsourced mobility service, a large number of part-time workers perform the spatial tasks offered by the platform and share the benefits in proportion, thereby, the strategy of task assignment directly affects the level of revenue and fairness among workers. In order to balance the revenue and fairness of workers, in this paper we study a novel type of fairness-aware spatial crowdsourcing problem, namely Fairness-Guaranteed Task Assignment (FGTA), which aims to maximize the total revenue of workers at a certain level of fairness guarantee and that is proved to be NP-hard. To solve this problem, we propose an efficient game-theory based approach for task assignment, which makes use of the best-response framework to iteratively select the best strategy for each worker until a Nash equilibrium is reached. Inspired by the observation that tasks with similar spatial and temporal features can be assigned together to a worker, we propose a spatial-temporal grouping based optimization to further improve the efficiency of task assignment. Furthermore, to improve the quality of Nash equilibrium, we present an effective large neighborhood search based optimization that trains a DQN decision model as destroy operator to accelerate the convergence of optimal task assignment. Finally, extensive experiments conducted on two real-world datasets demonstrate that our proposed approaches achieve better effectiveness and efficiency than the state-of-the-arts.**

***Index Terms*—Crowdsourced mobility services, fairness, game theory, reinforcement learning, task assignment.**

Manuscript received 22 April 2023; revised 24 July 2023; accepted 24 August 2023. Date of publication 31 August 2023; date of current version 4 April 2024. This work was supported in part by the NSFC under Grants 61972362, 62372416, 62325602, and 62036010, in part by YKLBAT under Grant 202105AG070005, in part by the HNSF under Grant 202300410378, in part by CPSF under Grant 2018M630836, in part by HK RGC under Grants C2004-21GF, 12200021, and 12202221, in part by GDNSF under Grant 2019B1515130001, and in part by the Project of Science and Technology Major Project of Yunnan Province under Grant 202102AD080006. Recommended for acceptance by A. Abdrabou. *(Corresponding authors: Jianliang Xu; Mingliang Xu.)*

Yafei Li is with the School of Computer and Artificial Intelligence, Zhengzhou University, Zhengzhou 450001, China, and also with the Yunnan Key Laboratory of Blockchain Application Technology, Beihang Yunnan Innovation Institute, Kunming 650001, China (e-mail: ieyfli@zzu.edu.cn).

Huiling Li, Baolong Mei, and Mingliang Xu are with the School of Computer and Artificial Intelligence, Zhengzhou University, Zhengzhou 450001, China (e-mail: huilingli@gs.zzu.edu.cn; blmeizzu@gs.zzu.edu.cn; iexumingliang@zzu.edu.cn).

Xin Huang and Jianliang Xu are with the Department of Computer Science, Hong Kong Baptist University, Hong Kong, SAR, China (e-mail: xinhuang@comp.hkbu.edu.hk; xujl@comp.hkbu.edu.hk).

This article has supplementary downloadable material available at https://doi.org/10.1109/TMC.2023.3310591, provided by the authors.

Digital Object Identifier 10.1109/TMC.2023.3310591

I. Instruction

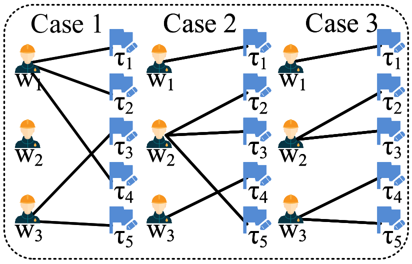
With the rapid development of GPS-enabled mobile devices, Crowdsourced Mobility Services (CMSs), such as Uber [1], Didi [2], Meituan [3], and Gigwalk [4], are becoming increasingly booming. In general, CMSs mainly consist of three parties: service provider, crowd workers, and customers, where crowd workers enter and exit the platform dynamically, customers send in their spatial tasks to the service provider in real time, and the service provider is in charge of assigning the spatial tasks to suitable crowd works. In order to complete these spatial tasks for revenue, crowd workers are always required to travel from one location to another by a specific deadline.

The core issue of CMSs is task assignment [5], [6], [7]. That is, the service provider assigns tasks that dynamically arrive at the platform to appropriate workers. A line of existing works focus on how to efficiently allocate workers and tasks to achieve the best overall utility of the platform [8], [9], [10], [11], [12], [13], [14], [15], [16], [17], [18], [19], [20], [21], e.g., maximizing overall revenue [8], [9], [10], [11], [12], maximizing task completion ratio [13], [15], and minimizing total travel cost [16], [17], [18]. However, considering only the overall utility (e.g., revenue) may lead to a large individual utility difference among workers and inevitably result in part of workers getting less income than they expected, thereby reducing their participation enthusiasm and affecting the service reputation. Furthermore, a recent study [22] reports that individual utility difference among workers could lead to workers leaving the platform, which in turn affects the platform's long-term benefits, i.e., task assignment schemas that fail to achieve a low level of individual utility difference are detrimental to the platform.

To address this issue, several recent works [23], [24], [25] have focused on the fairness of task assignment in crowdsourced services, with the aim of minimizing the individual utility difference among workers or achieving a trade-off between overall utility and individual utility difference. In this case, fairness task assignment in CMSs shares similar ideas with load balancing which always aims to achieve balance in allocating workload to resources [26], [27], but it has its own characteristics such as the requirement of optimizing both fairness and overall utility and the dynamics of both workers and tasks [23]. Also, despite existing efforts [22], [23], [24], [25], [26], [27], [28], [29], [30] on fairness task assignment are greatly insightful, we still observe several counterintuitive results in real-world settings. On the one hand, fairness in CMSs should consider not only workers, but also customers, where workers should be paid fairly based on their online working hours, and customers should pay

Γ \ W	w_1	w_2	w_3
τ_1	25	5	-
τ_2	13	11	-
τ_3	10	9	11
τ_4	21	10	20
τ_5	-	6	19

(a) Revenue matrix



(b) Three assignment cases

three cases	case 1	case 2	case 3
total revenue	89	71	84
total difference	39.3	4	12.7

(c) Total revenue and difference of these three cases

Fig. 1. Running example for fairness task assignment.

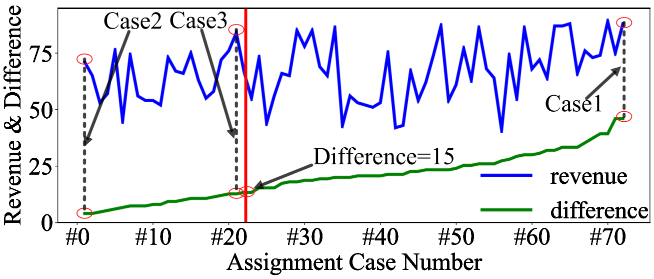


Fig. 2. Total revenue and difference of different task assignment cases in Example 1.

commensurate with the quality of service they receive. However, the existing works fail to address both workers' and customers fairness simultaneously. On the other hand, since it is difficult to justify the correlation between overall revenue and individual revenue difference, there are limitations in integrating revenue and revenue difference in a linear fashion to optimize mixed utility.

Inspired by these observations, in this paper, we present a novel problem of task assignment in crowdsourced mobility services, namely *Fairness-Guaranteed Task Assignment (FGTA)*, where the platform assigns the arrived tasks to suitable workers with the purpose of maximizing the platform's total revenue with ensuring a certain level of individual revenue difference. Next, we further illustrate the FGTA problem with a running example.

Example 1: As shown in Fig. 1, there are three workers $W = \{w_1, w_2, w_3\}$ and five tasks $\Gamma = \{\tau_1, \tau_2, \cdots, \tau_5\}$. Fig. 1(a) shows the revenue of worker w completing task τ and Fig. 1(b) shows three cases of task assignments. For simplicity, we assume the revenue of a worker is the sum of the revenues for the tasks he/she performs, and the total revenue of an assignment case is the sum of the revenues of workers in W and the revenue difference is the average difference between the revenues of any two workers in W. Thus, we have the total revenue and the revenue difference of three cases in Fig. 1(c). Fig. 2 illustrates the distribution of total revenue and individual revenue difference for all 72 cases in Example 1. Note that, all task assignment cases are sorted in ascending order by individual revenue difference. The mentioned three cases in Example 1 are indicated by the dotted lines in Fig. 2. In this example, we require the revenue difference of task assignment to meet a fair level that the revenue difference is less than 15. As such, the main goal becomes to maximize total revenue for the cases to the left of the red line in Fig. 2. Obviously, since case 1 does not meet the fairness requirement, the optimal solution in this example is case 3 which achieves a higher total revenue than case 2.

Despite its practical uses, solving the FGTA problem still requires nontrivial efforts. The main technical challenges are two aspects. First, fairness model is the key to addressing fairness in task assignment, and it is difficult to formulate a fair and reasonable fairness model for both workers and customers. As mentioned above, measuring the fairness of worker revenue is not only related to whether tasks can be completed but also closely related to the quality of task completion. Second, the FGTA problem is NP-hard as proved later, it means that in the worst case the time complexity of finding optimal assignment is exponential to the number of tasks and workers. Therefore, designing efficient algorithms to meet real-time requirements is the other challenge. To address the above two challenges, we first propose a novel fairness model to measure task assignment, which considers both the quality of workers completing tasks, such as the waiting time to start a task and the detour cost during task completion and the online time for workers to obtain the same benefit. On the basis of the proposed fairness model, we propose an effective game theory-based approach and design two improved strategies based on grouping and reinforcement learning optimized large neighborhood search to tackle the FGTA problem efficiently and effectively. The main contributions of this paper can be summarized as follows:

- We present a fairness model to measure the value of workers performing tasks and evaluate the revenue difference among workers, based on which we formally define a novel FGTA problem in crowdsourced mobility services with the aim of maximizing workers' total revenue within a certain level of fairness. We prove that the FGTA problem is NP-hard.
- We formulate the FGTA problem as the multi-round n-player strategy game that is solved by an efficient multi-round task assignment approach. We have proven that each round of task assignment can find a Nash equilibrium solution.
- We propose an efficient spatio-temporal grouping optimization to improve the efficiency of multi-round task assignment approach, and also present an efficient large neighborhood search based optimization that integrates a learning destroy operator to further improve the solution quality.
- We conduct extensive experiments on two real-world datasets to demonstrate the effectiveness and efficiency of our proposed algorithms by comparing them with state-of-the-art methods.

The remainder of this paper is organized as follows. Section II introduces some preliminaries and formally defines the FGTA

TABLE I
SUMMARY OF MAIN NOTATIONS

Notation	**Definition**
W, w	the set of workers, a worker
φ, t^w	a timestamp, the departure time of w
S^w	the schedule of w
$\mathit{\Gamma}_w$	the set of tasks in schedule S^w
v, c^w	the travel speed of w, the capacity of w
$\mathit{\Gamma}, \tau$	the set of tasks, a task
o^τ, e^τ	the origin of τ, the destination of τ
t^c, t^d	the create time of τ, the deadline of τ
$\hat{\mathit{\Gamma}}(w)$	the candidate tasks of w
$n, m, \hat{m}$	the number of workers, tasks, and groups, respectively
s	a moving location
l^w	the location of w
γ	the discount factor
Θ	the revenue difference threshold

problem. Section III details our proposed solutions to solve the FGTA problem. We evaluate the performance of our proposed algorithms in Section IV. Section V reviews the related works. Finally, we conclude this paper and discuss future work in Section VI.

II. Models and Problem Statement

In this section, we first present the system and fairness models. Then, we formally define the FGTA problem, followed by a theorem to establish its hardness. Table I summarizes the main notations used throughout this paper.

A. System Model

Generally, we define our FGTA problem on a road network represented by a graph $G^r = (N^r, E^r)$, where N^r is a node set and E^r is an edge set. Each $n_i^r \in N^r$ represents a road intersection, each $e_{i,j}^r \in E^r$ represents a road segment and is associated with a weight $dis(n_i^r, n_j^r)$ indicating the travel distance between n_i^r and n_j^r through edge $e_{i,j}^r$. For simplicity, our subsequent definitions of locations are all on nodes and we also use $dis(\cdot, \cdot)$ to represent the shortest travel distance between any two nodes.

Definition 1 (Spatial Task): A spatial task $\tau \in \mathit{\Gamma}$ is denoted as a tuple $\tau = (t^c, o^\tau, e^\tau, t^d)$ where τ is created at timestamp t^c and requires a single worker moving from location o^τ to e^τ before deadline t^d. ■

In crowdsourced mobility services, such as ridesharing [1], [2] and takeaway [3], locations o^τ and e^τ are specified by riders or customers as their origin and destination of tasks (i.e., ridesharing requests or takeaway orders), respectively. Meanwhile, the deadline t^d is issued by riders or customers as the expected completion time.

Definition 2 (Worker): A worker $w \in W$ is denoted as a tuple $w = (l^w, t^w, c^w)$ where w is currently located at location l^w with a time t^w of leaving from the platform and a capacity c^w representing the number of tasks that w can perform simultaneously. ■

A worker w comes to the platform randomly, continuously reveals current location l^w of w, and is expected to leave the platform after time t^w.[1] Subject to the constraints, e.g., available vehicle seats and backpack capacity, workers can perform at most c^w tasks at the same time. To simplify, we assume that workers are equally productive, it is reasonable because travel speed is an important indicator to evaluate the efficiency of workers in performing spatial tasks and the speed limitation on road network are the same for workers.

Definition 3 (Worker Schedule): Given a worker $w \in W$, the schedule of w denoted as $S^w = \{s_1, s_2, \ldots, s_{2|\mathit{\Gamma}_w|}\}$ is a sequence of moving events, where $\mathit{\Gamma}_w$ denotes the set of tasks in schedule S^w. Each $s \in S^w$ specifics a location, which may be the origin o^τ or the destination e^τ of a task $\tau \in \mathit{\Gamma}_w$. ■

Note that the schedule S^w for a given worker w is valid if it satisfies the following conditions: i) the terminal time of S^w cannot exceed the leaving time t^w of w, ii) for any task τ in $\mathit{\Gamma}_w$, it should be finished before its deadline t^d, and iii) the number of tasks performed simultaneously by w cannot exceed the capacity c^w of w at any time. Since the worker schedule S^w changes over time when the tasks in $\mathit{\Gamma}_w$ are finished and new tasks are inserted, it only maintains current uncompleted tasks.

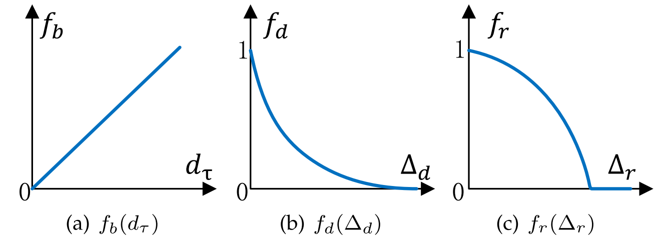
Definition 4 (Task Assignment): In our FGTA problem, tasks and workers arrive at the platform dynamically in the form of streams and the platform periodically assigns tasks to suitable workers. We call each assignment cycle a batch and the tasks not assigned in current batch will be deferred to the next batch. ■

As mentioned above, the platform performs task assignment in a batch fashion, which divides workers and tasks arriving in the form of time-series data streams into batches by fix-size time windows and performs task assignment at the end of each batch [5], [31].

B. Fairness Model

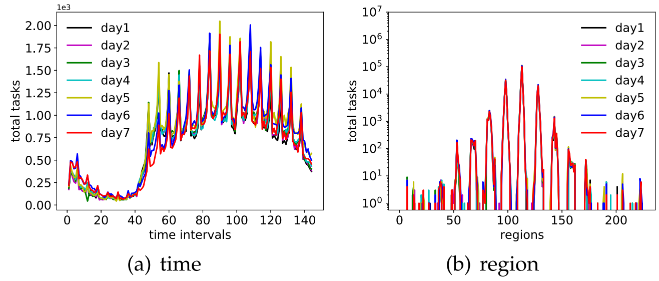
In crowdsourced mobility services, a reasonable pricing model for tasks plays a vital role in measuring fairness. In this section, we propose a novel pricing model that takes into account task difficulty and task completion quality to evaluate the value of workers performing spatial tasks, thereby ensuring that the customers pay a fair price consistent with the quality of service they receive. Specifically, for a task τ, it usually has a default price that reflects task difficulty based on the shortest travel distance $d_\tau \in \mathbb{R}_+$, from location o^τ to e^τ, and it is calculated by an arbitrary function f_b: $\mathbb{R}_+ \rightarrow \$$. For example, in Fig. 3(a), f_b is directly proportional to d_τ. For task completion quality, since one worker can perform multiple tasks at the same time, it may lead to detours (e.g., in the ridesharing scenario, picking up another rider on the way to drop off a rider may increase his/her trip distance, and the added trip distance is the detour for him/her) or slow response (i.e., tasks are performed later than expected, during task execution). Therefore, we evaluate task completion quality in terms of detour ratio and response time. Intuitively, the earlier the task starts and the less detour the task incurs, the higher the quality of the task completes. Here, the detour can be calculated by $\Delta_d = d'_\tau - d_\tau$ where d'_τ is the

[1] In crowdsourced services, many workers are part-time, and they have expected off-hours.



(a) $f_b(d_\tau)$ (b) $f_d(\Delta_d)$ (c) $f_r(\Delta_r)$

Fig. 3. Example of functions $f_b(d_\tau)$, $f_d(\Delta_d)$, and $f_r(\Delta_r)$.



(a) time (b) region

Fig. 4. Spatio-temporal distribution of tasks.

actual travel distance from location o^τ to e^τ when performing τ, and the response time can be calculated by $\Delta_r = t' - t^c$ where t' is the actual time when the worker arrives at origin o^τ, and t^c is the created time of τ. Consequently, considering a task τ and a worker w, we define our generic pricing model as

$$price_w(\tau) = f_b(d_\tau)\big(\alpha f_d(\Delta_d) + (1-\alpha) f_r(\Delta_r)\big), \quad (1)$$

where $f_b(d_\tau)$ is the default price of τ, and $\alpha f_d(\Delta_d) + (1-\alpha) f_r(\Delta_r)$ is the discount function with range [0, 1]. Here, $f_d(\Delta_d)$ and $f_r(\Delta_r)$ representing discounts based on detour and response time are two mapping functions that normalize Δ_d and Δ_r into [0, 1], α is the price balance parameter for balancing the influence of detour and response time on the default price. Fig. 3(b) and (c) show examples of $f_d(\Delta_d)$ and $f_r(\Delta_r)$, where $f_d(\Delta_d)$ decreases as detour Δ_d increases. Also, $f_r(\Delta_r)$ decreases as response time Δ_r increases and the point $f_r(\Delta_r) = 0$ indicates that response time longer than that is unacceptable. Note that only when τ is completed, Δ_d and Δ_r are determined, and its price is finally determined.

Subsequently, the total revenue of worker w from completing tasks in $\varGamma_w$ can be calculated as

$$rev(w, \varGamma_w) = \sum_{\tau \in \varGamma_w} price_w(\tau) - cost(w, \varGamma_w), \quad (2)$$

where $cost(w, \varGamma_w)$ is the total travel cost of worker w following schedule S^w to finish tasks in $\varGamma_w$ and can be calculated as

$$cost(w, \varGamma_w) = dis(l^w, s_1) + \sum_{i=1}^{|S^w|-1} dis(s_i, s_{i+1}), \quad (3)$$

where the cost per unit of travel distance is set to 1.

Based on (2), an effective means of evaluating fairness is the difference in revenue among workers [25]. Given the assumption that workers are equally productive, the fairness in crowdsourced mobility services can be modeled as different workers earning comparable revenues within the same number of working hours. However, the model in [25] ignores the impact of working hours on revenue. As such, we improve it and propose our revenue difference model. Specifically, we take into account workers' working hours over a period of time to increase the rationality of it. Given a set of workers W ($|W| \geq 2$), the total revenue difference in W can be calculated as:

$$D(W) = \sum_{w \in W} \sum_{w' \in W, w \neq w'} \left| \frac{rev(w, \varGamma_w) + hist_w}{t_w + t_{S^w}} - \frac{rev(w', \varGamma_{w'}) + hist_{w'}}{t_{w'} + t_{S^{w'}}} \right| \times \frac{2}{|W|(|W|-1)}, \quad (4)$$

where t_w, t_{S^w} and $hist_w$ is the time that w has been on the platform before performing S^w, the time required to complete S^w and the history revenue w got in a time period, respectively. Here, a time period refers to a working cycle (e.g., a day). Simply put, $D(W)$ is the average of the revenue difference between all pairs of workers in W.

C. Problem Formulation

Based on the above system model and fairness model, we elaborate on the FGTA problem in this subsection. In real application scenarios, since tasks are unevenly distributed in both temporal (Fig. 4(a)) and spatial (Fig. 4(b)), imposing exact averages on workers' revenue is too strict. As such, we aim at performing task assignment within a certain level of fairness among workers and subsequently formulate our FGTA Problem.

Definition 5 (FGTA Problem): Given a set of tasks $\varGamma$, a set of workers W, and a fairness threshold Θ, the goal of the FGTA problem is to schedule workers in W to perform tasks in $\varGamma$ such that the total revenue of all workers is maximized, provided that the fairness threshold Θ is satisfied, i.e.,

$$\max \sum_{w \in W} rev(w, \varGamma_w) \ \text{ s.t., } \ D(W) \leq \Theta. \qquad \blacksquare$$

In what follows, we theoretically analyze the hardness of FGTA problem in detail.

Theorem 1: The FGTA problem is NP-hard. ■

Proof: Please refer to Appendix A.1, available online. □

III. Solutions for FGTA Problem

As stated above, the FGTA is a typical dynamic task assignment problem, and the platform adopts a batch fashion task assignment model to solve this problem. Generally, the platform periodically collects the unassigned tasks and performs task assignment, i.e., finds the optimal assignment plan for each time window. Since FGTA is proven to be NP-hard in Theorem 1, we propose several efficient heuristic algorithms to solve the FGTA problem below.

Following the existing works [8], [31], [32], we model the relationships between workers and tasks within a batch by a weighted bipartite graph $G^b = (W^b, \varGamma^b, E^b, \mathcal{U}^b)$, where W^b is the set of available workers, $\varGamma^b$ is the set of tasks in this batch. For an edge $e^b \in E^b$ linking worker w and task τ, it is associated with a weight $u(w, \tau)$. As such, we propose a solution framework that first calculates the weights $\mathcal{U}^b$ of edges in G^b and then performs task assignment on the basis of G^b. In this paper, we regard the weight $w \in W^b$ of an edge $e^b \in E^b$ between worker w and task τ as the utility, which indicates the potential revenue for w completing τ.

In the following, we first elaborate on how the utility that a given worker performs one task or multiple tasks is calculated according to the revenue function (2). Then we propose the game theory-based task assignment algorithm by modeling the FGTA problem as a multi-round n-player strategy game, proving that it is an exact potential game with Nash equilibrium, and designing the GT algorithm to find the solution with Nash equilibrium. To optimize the efficiency of GT, we propose a grouping-based strategy. While since the Nash equilibrium found by GT is not unique, we adopt Large Neighborhood Search (LNS) to iteratively optimize its quality and propose a reinforcement learning-based destroy operator for accelerating the convergence of LNS.

A. Worker-Task Utility Calculation

As mentioned in our proposed solution framework, the first step is to calculate the utilities of edges in a weighted bipartite graph. In this section, we first give a method to calculate the utility of assigning a task to a worker, and then extend it to compute the utility of assigning multiple tasks to a worker. Finally, we propose an efficient indexing structure to speed up the utility computation, and a discussion is presented to further elaborate its usage.

1) Utility Calculation for a Single Task: According to our pricing model, the price of a task is related to the detour and response time when it is completed, i.e., the price of a task is dominated by the schedule of workers to perform the task. For a worker's schedule, inserting a new task may affect the price of other tasks in the schedule. Therefore, it is unreasonable to simply count the expected price of a new task as the utility of assigning it to a worker. An effective approach is to construct an evaluation metric that measures the utility of inserting a new task. Specifically, we measure the utility by the increase in worker's revenue, which can be calculated as

$$u(w, \tau) = rev(w, \Gamma_w \cup \{\tau\}) - rev(w, \Gamma_w). \quad (5)$$

Next, we discuss how to calculate $rev(w, \Gamma_w \cup \{\tau\})$. According to (2) and (3), we need to get the schedule after assigning τ to w. Generally, we plan a new route for w performing tasks in $\Gamma_w \cup \{\tau\}$ by maximizing the utility $u(w, \tau)$, which is proven to be NP-hard [8]. To solve this issue, we follow the existing works [24], [33], [34], [35], [36] and adopt insertion operation to plan new routes. That is, when assigning a task to a worker, we insert the task into the worker's schedule without reordering the original schedule. For a more detailed description, please refer to Appendix B, available online.

2) Utility Calculation for Multiple Tasks: Based on single task utility calculation, we introduce how to calculate the utility for assigning multiple tasks to a worker. Given a worker w and a task set Γ, the utility of inserting the tasks in Γ into w's schedule S^w can be calculated as

$$u(w, \Gamma) = rev(w, \Gamma_w \cup \Gamma) - rev(w, \Gamma_w). \quad (6)$$

As such, the key issue is to obtain schedule S'^w, we have designed two insertion-based algorithms to solve this issue. Due to space limitation, please find more details of these two algorithms in Appendix C, available online.

3) Distance Estimation Index: Note that, for worker-task utility calculation, it is very time-consuming to frequently invoke the shortest path finding algorithm (e.g., Dijkstra algorithm) to calculate the shortest distance between nodes [37]. In fact, there is no need to frequently calculate the exact shortest distance between nodes, we can make use of distance estimation to reduce lots of calculation cost. Inspired by the observation, we propose a novel index structure, namely *distance estimation* (DSE) index, using grid and clustering techniques to estimate the distance bounds between nodes. Moreover, when constructing the DSE index, we take the task emergence pattern hidden in historical data into account to make distance estimation more precise.

Fig. 5 shows the structure of DSE index. Specifically, the index construction process consists of two phases. In the first phase, we use a probabilistic model presented in [38] to divide the road network (as shown in Fig. 5(a)) into a number of contiguous cells by estimating the computation cost of different cell sizes. In the second phase, we count the number of occurrences of the start and end nodes of tasks (represented as num) in each grid cell based on historical task data and invoke the k-means algorithm [39] to cluster them into $\lceil \frac{num}{\varsigma} \rceil$ regions (as shown in Fig. 5(b)), where ς is a hyperparameter indicating that each region's num is at most ς. If a region's num is greater than ς, we repeat the above strategy until the region's num is less than ς or there is only one node in the region. Note that each region's center is the clustering center of this region, while each grid cell's center is the node closest to the geometric center of this grid cell. As shown in Fig. 5(b), different colors indicate that different regions and highlighted points indicate region centers, where the center of a region and a node are denoted by z and l, respectively. The two matrices shown in Fig. 5(c) and (d) store the distances between region centers and the distances between nodes in region z_2, which can be calculated offline. Therefore, we can quickly estimate the distance between nodes by Lemma 1.

Lemma 1: Considering two locations l_1 and l_2, the distance upper bound between l_1 and l_2 is

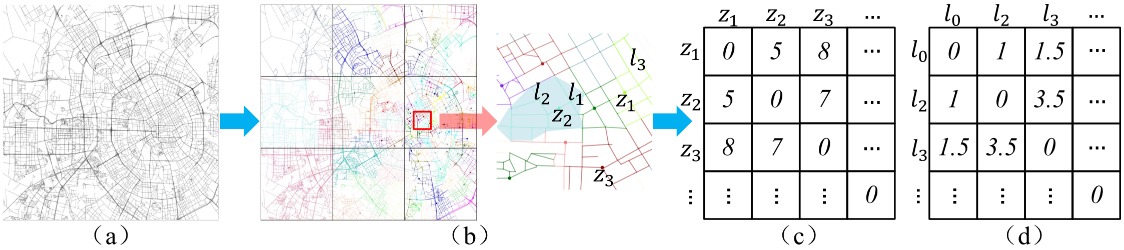
$$dis_{ub}(l_1, l_2) = dis(z_1, z_2) + \max_{z_1} + \max_{z_2}, \quad (7)$$

and the distance lower bound between l_1 and l_2 is

$$dis_{lb}(l_1, l_2) = \mathrm{MAX}(0, dis(z_1, z_2) - \max_{z_1} - \max_{z_2}), \quad (8)$$

where l_1 (l_2) denotes in region z_1 (z_2), $dis(z_1, z_2)$ denotes the shortest distance between the center nodes of z_1 and z_2, $\max_{z_1}$ ($\max_{z_2}$) denotes the maximum distance from the node in z_1 (z_2) to the center node of z_1 (z_2). ■

Proof: The proof is straightforward and thus is omitted. □



	z_1	z_2	z_3	⋯
z_1	0	5	8	⋯
z_2	5	0	7	⋯
z_3	8	7	0	⋯
⋮	⋮	⋮	⋮	0

	l_0	l_2	l_3	⋯
l_0	0	1	1.5	⋯
l_2	1	0	3.5	⋯
l_3	1.5	3.5	0	⋯
⋮	⋮	⋮	⋮	0

Fig. 5. Structure of DSE index.

Discussion: Different from the typical grid index with equal cell size, in this paper we optimize the grid index by considering historical order distribution in each grid cell. Fig. 4(a) and (b) show the task distribution in temporal and spatial of one week on Haikou dataset, respectively. It can be seen that the task distribution has periodicity and uneven quantity features, the number of tasks is very dense in hot spots and peak hours. If the road network is divided into a set of grid cells of equal size, the distance estimation error in hotspots will be large, resulting in a large recomputation cost. That is why we consider the order distribution of the EST index, which provides a good balance between efficiency and accuracy for distance estimation.

B. Game Theory Based Task Assignment

In this subsection, based on the above calculation of utility, we propose our task assignment algorithm. It is natural to solve fairness-aware task assignment by the n-player strategy game model [25], [31], [40] that considers a general game with n players who are allowed to select strategies for maximizing their utilities and achieve a Nash equilibrium. Note that a Nash equilibrium is the state where no players can improve their utilities by varying strategies. In our FGTA problem, we model workers as the players, thus Nash equilibrium means that all workers get the maximum revenues they can make under their current situation, i.e., although the revenues may vary among workers, they all get the maximum revenues they can receive. Based on that, we propose a novel multi-round task assignment approach based on n-player strategy game model to solve the FGTA problem.

The main idea of multi-round task assignment is that workers and tasks are matched in rounds until there are no available tasks for workers, and each worker is assigned at most k tasks in the kth round of task assignment. We model each round of task assignment as a n-player strategy game, which can be denoted as a three-entry tuple $\mathcal{P} = \langle W, \mathcal{ST}, \mathcal{U} \rangle$. Specifically, $W = \{w_1, w_2, \ldots, w_n\}$ is the set of n workers, $\mathcal{ST} = ST_1^k \times ST_2^k \times \ldots \times ST_n^k$ is the joint strategy set for workers in W where ST_i^k is the set of strategies of w_i in the kth round of task assignment. Then, $\vec{st}^k = (st_1^k, st_2^k, \ldots, st_n^k)$ is a joint strategy, where $st_i^k \in ST_i^k$ is the strategy selected by w_i. $\mathcal{U} = \{U_1, U_2, \ldots, U_n\}$ is the set of utility functions for all players and $U_i : \mathcal{ST} \rightarrow R$ is the utility function of w_i. In particular, w_i's strategy st_i^k in the kth round of task assignment specifies a task set $\varGamma_{st_i^k}$ with no more than k tasks, all of which can be inserted into w_i' schedule simultaneously. Considering that in our problem workers perform tasks under the same pricing model and players share the same utility function. Therefore, the utility function of w_i can be represented as:

$$U_i(\vec{st}^k) = u(w_i, \varGamma_{st_i^k}). \tag{9}$$

Next, we show that we can find a pure Nash equilibrium in each round of task assignment. Existing works [25], [41], [42] show that for an exact potential game, a solution converging to a pure Nash equilibrium can be found through the best response framework, which iteratively selects the best strategy for each worker until a Nash equilibrium is reached. Given a worker w, the strategy of w is to select a set of tasks and insert them into the schedule of w. Since the task set is finite, the strategies of workers are obviously countable. As such, we only need to prove that each round of task assignment, i.e., $\mathcal{P}$, is an exact potential game.

Theorem 2: The n-player strategy game $\mathcal{P}$ for the kth round of task assignment is an exact potential game. ■

Proof: Please refer to Appendix A.2, available online. □

The detailed pseudo code for multi-round task assignment is presented in Algorithm 1. It takes a worker set W, workers' schedule set $\mathbb{S}$, and a task set $\varGamma$ as input and outputs the set $\mathbb{S}$ of updated schedules for workers. We first calculate the set of candidate tasks $\hat{\varGamma}(w_i)$ for each worker (lines 1–2) and initialize k to be 1 (line 3). Then, we assign tasks to workers continuously in successive rounds until there are no assignable tasks (lines 4–13). For the assignment in each round, we achieve a Nash equilibrium through best-response framework, i.e., iteratively select the best strategy including at most k tasks for each worker (lines 5–11). Specifically, we first select workers from W in a random order (line 6), and for each selected worker w_i, we choose a subset $\varGamma_{st_i^{k\prime}}$ of k tasks at most from set $\hat{\varGamma}(w_i) \cap \varGamma$, so that the workers can get the maximum utility from completing these tasks (line 7). Here, $\hat{\varGamma}(w_i) \cap \varGamma$ represents the tasks that w_i is currently likely to perform and in fact $\varGamma$ dynamically maintains tasks that are not chosen by workers. Then if strategy $st_i^{k\prime}$ brings w_i a higher utility than the previous strategy st_i^k and the fairness threshold is met, we replace w_i's best strategy st_i^k with $st_i^{k\prime}$ and update $\varGamma$ (lines 8–11). In detail, we delete the tasks in set $\varGamma_{st_i^{k\prime}} \setminus \varGamma_{st_i^k}$ from $\varGamma$ and add the tasks in set $\varGamma_{st_i^k} \setminus \varGamma_{st_i^{k\prime}}$ to $\varGamma$. When a Nash equilibrium in current assignment round is reached, we increment k (line 12). We repeat the above process until there are no more tasks in $\varGamma$ that workers can perform (line 13). Finally, we update workers' schedules and return $\mathbb{S}$ (lines 14–16). Lemma 2 shows that the total utility is monotonically increasing as the number of

Algorithm 1: Game Theory-Based Algorithm.

```
Input: A set $W$ of $n$ workers and their schedule set $\mathbb{S}$, a
       set $\Gamma$ of $m$ tasks
Output: A set $\mathbb{S}$ of updated schedules for $n$ workers
foreach $w_i \in W$ do
    obtain the set $\hat{\Gamma}(w_i)$ of candidate tasks that can be
      inserted into schedule $S_i$;
$k \leftarrow 1$;
repeat
    while not in a pure Nash equilibrium do
        foreach randomly selected $w_i \in W$ do
            $st_i^{k\prime} \leftarrow$ select a subset $\Gamma_{st_i^{k\prime}}$ of up to $k$ tasks
              with the highest utility from set $\hat{\Gamma}(w_i) \cap \Gamma$;
            if $u(w_i, \Gamma_{st_i^k}) < u(w_i, \Gamma_{st_i^{k\prime}})$ and $D(W) \leqslant \Theta$
              then
                replace $st_i^k$ with $st_i^{k\prime}$;
                $\Gamma \leftarrow \Gamma \cup (\Gamma_{st_i^k} \backslash \Gamma_{st_i^{k\prime}})$;
                $\Gamma \leftarrow \Gamma \backslash (\Gamma_{st_i^{k\prime}} \backslash \Gamma_{st_i^k})$;
    $k \leftarrow k + 1$;
until $\bigcup_{i=1}^{n} (\hat{\Gamma}(w_i) \cap \Gamma) = \varnothing$;
foreach $w_i \in W$ do
    update $w_i$'s schedule $S_i$ with its current strategy;
return $\mathbb{S}$;
```

assignment rounds increases, which is consistent with our goal of maximizing total revenue.

Lemma 2: Let $F(\vec{st}_i^{k})$ and $F(\vec{st}_i^{k+1})$ be the total utilities of all workers for batches b_k and b_{k+1}, respectively. Then, we have $F(\vec{st}_i^{k+1}) \geq F(\vec{st}_i^{k})$. ■

Proof: As workers update their best strategies only when the updating leads to an increase in utility, it holds

$$\forall w \in W,\ u(w, \Gamma_{st^{k+1}}) \geq u(w, \Gamma_{st^{k}}). \tag{10}$$

Then, we have

$$F(\vec{st}_i^{k+1}) = \sum_{w \in W} u(w, \Gamma_{st^{k+1}}) \geq F(\vec{st}_i^{k}) = \sum_{w \in W} u(w, \Gamma_{st^{k}}). \tag{11}$$

The proof is completed. □

Since task assignment for workers and tasks are conducted in a batch fashion, we should consider how to deal with the unassigned tasks in the current batch. Generally, if they have not expired in the current batch, they will be sequentially treated as input of the next batch. However, we do not need to treat them as newly arrived tasks, since some computations in the current batch still hold in the next batch. Therefore, we give Lemma 3 to guide task and worker assignment across batches, which can significantly reduce computational cost, especially when the previous batch has too many unassigned tasks.

Lemma 3: Let Γ_i be the set of tasks that cannot be assigned to workers W_i in batch b_i, W' be the arrived workers in batch b_{i+1} but not in batch b_i. For a task $\tau \in \Gamma_i$, τ can only be assigned to the workers W' in batch b_{i+1}. ■

Proof: Obviously, since the worker constraints and task constraints do not change over time, if a task $\tau \in \Gamma_i$ cannot be assigned to workers W_i in batch b_i, it also cannot be assigned to workers W_i in batch b_{i+1}. As such, in batch b_{i+1}, tasks in Γ_i can only be assigned to the newly arrived workers W'. The proof is completed. □

Complexity Analysis: The time complexity of Algorithm 1 is $O(nm|S|^3 + k^2(Pn|ST|k^2|S|^3))$, where $O(|S|^3)$ is the time complexity of Algorithm 5, P is the number of iterations required to achieve a Nash equilibrium using best-response framework, $|ST|$ is the size of the worker's strategy set and k is the largest assignment round.

C. Spatio-Temporal Grouping Based Optimization

As stated in Lemma 3, we can assign tasks batch by batch incrementally, which greatly reduces the computational cost. While, in this subsection, to further improve the efficiency of task assignment, we propose an efficient task grouping based optimization. Specifically, based on this observation that tasks with similar starting/ending points and deadlines can be assigned to the same worker in group, we divide the tasks into groups and compute candidate task groups rather than tasks for task assignment.

Definition 6 (Task Group): A task group $g \in G$ is denoted as a tuple $g = (\Gamma_g, z^o, z^e, t^{\max}, t^{\min})$, where Γ_g is a set of tasks in g, z^o (z^e) is the region covering the starting nodes (ending nodes) of tasks in Γ_g, and $t^{\max}$ ($t^{\min}$) is the latest creation time (earliest deadline) of tasks in Γ_g. ■

We construct task groups based on the feature that tasks with similar spatial and temporal features are likely to be candidate tasks for the same worker, and workers are likely to achieve higher revenue increase when they select tasks in the same group. Specifically, we construct groups according to the following two rules: i) spatial constraint: tasks can be grouped together only if their start nodes and end nodes are both in the same regions; and ii) temporal constraint: the task with the earliest deadline $t^{\min}$ can still be completed under the latest start time $t^{\max}$.

Algorithm 2 gives the pseudo code of constructing task groups, it takes a set of tasks Γ as input and outputs the constructed groups set G. We first initialize the set G (line 1). Then we traverse every task in Γ and classify it into the appropriate group (lines 2–15). In detail, for each task τ, we iterate over each existing group, and if its start node o and end node e are in the start region z^o and end region z^e of group g respectively, which means that the spatial constraint is satisfied (line 4), we check the temporal constraint (lines 5–10). Then, if both the spatial and temporal constraints are satisfied, we add t into set Γ_g and update $t^{\max}$ and $t^{\min}$ (lines 11–12). What's more, if τ is not in any existing groups, we build a new group g' for τ and add g' into G (lines 13-15). Finally, after all the tasks have been traversed, we return the set of constructed groups G (line 16).

Next, we utilize task groups to improve the efficiency of task assignment. In particular, it helps our game theory-based approach in the following two ways: i) considering a task group g and a worker w, if a task τ in Γ_g can be assigned to worker w, all the tasks in Γ_g are the candidate tasks for w, which can improve the efficiency of candidate tasks set calculation; ii) since the tasks Γ_g in a task group g have similar spatial and temporal features, assigning them together to a worker can potentially achieve higher revenue increase. Inspired by this, we can first screen each worker from the group-level to obtain the groups with the most utilities. Then from these groups, we select the

Algorithm 2: Task Group Construction.

```
Input: A set $\Gamma$ of $m$ tasks
Output: A set $G$ of constructed groups
$G \leftarrow \varnothing$;
foreach $\tau \in \Gamma$ do
    foreach $g \in G$ do
        if node $o$ in region $z^o$ & $e$ in region $z^e$ then
            if $t^c > t^{max}$ then
                if $\frac{dis(z^o, z^e)}{v} > (t^{min} - t^c)$ then
                    continue;
            if $t^d < t^{min}$ then
                if $\frac{dis(z^o, z^e)}{v} > (t^d - t^{max})$ then
                    continue;
            $\Gamma_g \leftarrow \Gamma_g \cup \tau$;
            update $t^{max}$, $t^{min}$;
    if $\tau$ not in any groups then
        create a new group $g' \leftarrow (\{\tau\}, z^{o'}, z^{e'}, \varphi, t)$;
        /* $z^{o'}$ and $z^{e'}$ are the regions that
           nodes $o'$ and $e'$ reside                  */
        $G \leftarrow G \cup \{g'\}$;
return $G$;
```

best subset of k tasks for each worker as its best strategy. In this way, since the number of groups is always less than that of tasks, the efficiency of our game theory-based algorithm can be improved.

In addition to improving pruning capability from task-level to group-level for task assignment within a batch, the grouping technique can also make a difference for tasks and workers across batches. Intuitively, if the new tasks arriving in the current batch can be added into unassigned groups in previous batch, we only need to assign them to the workers newly arrived in current batch. As such, we extend Lemma 3 and present Lemma 4.

Lemma 4: Let G_i be the set of groups unassigned in batch b_i, G_{i+1} be the set of groups constructed in batch b_{i+1}, W' be the newly arrived workers in b_{i+1} but not in b_i. For a group $g \in G_i$, if there exists a group $g' \in G_{i+1}$ subject to $\Gamma_g \subseteq \Gamma_{g'}$, g' can only be assigned to the workers W'. ■

Proof: We prove Lemma 4 by proving its inverse negative proposition. Specifically, the inverse negative proposition of Lemma 4 is that, if a group g' can be assigned to workers $W_{i+1} \setminus W'$, then g can also be assigned to workers $W_{i+1} \setminus W'$, where W_{i+1} is all the workers in batch b_{i+1}.

Obviously, for groups g and g', the following equations hold:

$$z^o = z^{o\prime}, z^e = z^{e\prime}, \tag{12}$$

$$t^{\max} \leq t^{\max\prime}, t^{\min} \geq t^{\min\prime}, \tag{13}$$

$$|\Gamma_g| \leq |\Gamma_{g'}|. \tag{14}$$

Here, (12) means that groups g and g' have the same start region and destination region. (13) indicates that the created time of g is earlier than that of g' while g has a looser deadline than g', i.e., workers have more time to complete tasks in g than the tasks in g'. (14) means that g requires less worker capacity than g'. Obviously, if a worker w can complete tasks in g', w is also sure to complete the tasks in g. Thus the inverse negative proposition of Lemma 4 is true. The proof is completed. □

Complexity Analysis: The time complexity of Algorithm 2 is $O(m|G|)$, where $|G|$ is the number of groups. Through the optimization of grouping, the time complexity of Algorithm 1 can be refined as $O(m|G| + n|G||S|^3 + k^2 (Pn|ST|k^2|S|^3))$.

D. Large Neighborhood Search Based Optimization

Despite the above optimization of efficiency, improving the quality of the solution is also necessary. In our game theory-based approach, we use best-response framework to obtain feasible solutions with Nash equilibrium. However, since the solutions with Nash equilibrium are not unique and often not optimal, we should further improve the quality of task assignment.

In mathematical optimization, large neighborhood search (LNS) is a technique that tries to find near-optimal solutions to a combinatorial optimization problem by repeatedly transforming current solution into another solution in the neighborhood of the current solution. Here, the neighborhood of a solution is a set of similar solutions obtained by relatively simple modifications to the original solution. Inspired by the above, we adopt large neighborhood search to optimize our game theory-based approach. The main idea is that we employ two operators, namely *break* and *repair*, to break and repair the solution iteratively until there is a good solution with desired quality. Specifically, as detailed in Algorithm 3, in each assignment round, we first initialize the strategies of all workers through the *break* operator (line 3). Then, we iteratively operate *break* and *repair* operators num times to break and repair the joint strategy of workers (lines 4-8). In each iteration, we first invoke the *break* operator to select a group of workers and break their current strategies (line 5). After that, we invoke the *repair* operator to find a new joint strategy $\overrightarrow{st}^{k'}$ (line 6). If the utility of $\overrightarrow{st}^{k'}$ is greater than the utility of $\overrightarrow{st}^{k}$, i.e., $F_p(\overrightarrow{st}^{k'}) > F_p(\overrightarrow{st}^{k})$, we update $\overrightarrow{st}^{k}$ with $\overrightarrow{st}^{k'}$ (lines 7-8). Finally, we update each worker's schedule, and the refined schedule set $\mathbb{S}$ is returned (line 13).

In Algorithm 3, the neighbor search always uses some greedy approaches to destroy the solution, e.g., breaking the edges with lower utilities or randomly breaking edges, then tries to find a better solution within the range of neighbors. Obviously, the neighborhood search with greedy break operation may be easy to fall into local optimum solution. To address this weakness, we make use of a learning-based approach, i.e., deep reinforcement learning, to learn efficient break strategies in order to quickly find the near-optimal solution. Fig. 6 shows the framework of large neighborhood search combined with reinforcement learning. As a whole, it takes the best-response framework as the repair operator and trains a DQN decision model as the break operator. In what follows, we explain the optimized break and repair operators in detail, respectively.

1) Adaptive Break Operator: In this section, we adopt the model of multi-agent Markov decision process model (MMDP) [43], [44] to formulate the break decision process and adopt the DQN-based method to adaptively determine the break strategy in terms of several combined features. Specifically, we formulate the break decision process as an MMDP

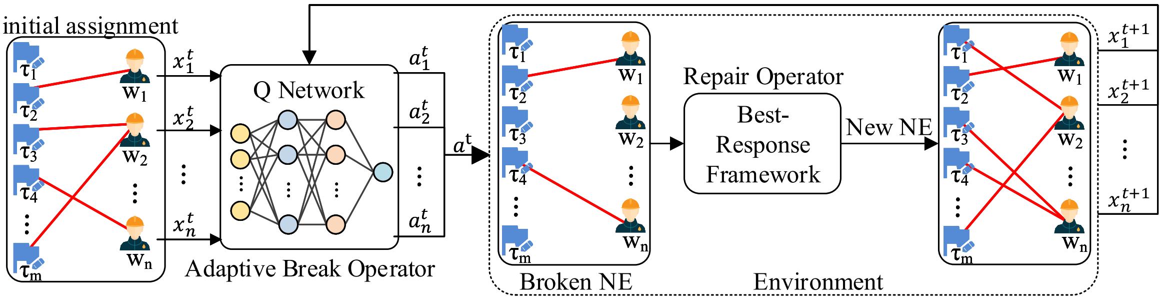


Fig. 6. Framework of LNS combined with reinforcement learning.

```
Algorithm 3: Large Neighborhood Search Based Optimization.
Input: A set $W$ of $n$ workers and their schedule set $\mathbb{S}$, a
       set $\Gamma$ of $m$ tasks
Output: A set $\mathbb{S}$ of updated schedules for $n$ workers
$k \leftarrow 1$;
repeat
    $\vec{st}^{\,k} \leftarrow$ initialize via repair operator;
    for $i \leftarrow 0$ to $num$ do
        call break operator;
        $\vec{st}^{\,k'} \leftarrow$ repair operator;
        if $F_p(\vec{st}^{\,k'}) > F_p(\vec{st}^{\,k})$ then
            $\vec{st}^{\,k} \leftarrow \vec{st}^{\,k'}$;
    $k \leftarrow k + 1$;
until $\bigcup_{i=1}^{n}(\hat{\Gamma}(w_i) \cap \Gamma) = \varnothing$;
foreach $w_i \in W$ do
    update $w_i$'s schedule $S_i$;
return $\mathbb{S}$;
```

game $\mathbb{G} = (W, X, A, P, R, \gamma)$, which can be specified as follows.

Agent $w_i \in W$*:* We treat each worker as an agent that appears and disappears as the worker appears and disappears. Furthermore, we assume that all agents are isomorphic since workers are similar in capability to complete tasks.

State $x^t \in X$*:* At time t, the joint state of all agents is represented as x^t. Then, the state $x_i^t \in x^t$ of an agent w_i is denoted as a four-entry tuple $(\eta_i^c, \eta_i^\tau, \eta_i^w, \eta_i^n)$. In detail, η_i^c represents the proportion of workers whose current strategy utility is higher than w_i. It can be calculated as:

$$\eta_i^c = \frac{|\{w \in W | u(w, \Gamma_{st^k}) > u(w_i, \Gamma_{st_i^k})\}|}{|W|}. \tag{15}$$

Intuitively, η_i^c reflects the position of current strategy utility of w_i among all workers.

η_i^τ represents the proportion of workers who are likely to replace their best strategies when the current strategy of w_i is destroyed. Since a task can only be assigned to one worker, the strategies containing any task in strategy st_i^k are not available for workers in $W \setminus \{w_i\}$. Then, the maximum utility of any worker w's strategies that contain tasks associated with w_i's current strategy st_i^k is

$$u_p(w, st_i^k) = \max_{st^k \in ST^k; \Gamma_{st^k} \cap \Gamma_{st_i^k} \neq \emptyset} u(w, \Gamma_{st^k}). \tag{16}$$

As such, η_i^τ can be calculated as:

$$\eta_i^\tau = \frac{|\{w \in W | u_p(w, st_i^k) > u(w, \Gamma_{st^k})\}|}{|W|}, \tag{17}$$

where $u_p(w, st_i^k) > u(w, \Gamma_{st^k})$ means that if w_i's strategy st_i^k is destroyed, there is a strategy for w with a higher utility than st^k becoming selectable, i.e., w's best strategy has the potential to be updated.

η_i^w represents the proportion of strategies in w_i's strategy set ST_i^k that have higher utility than the strategy st_i^k. Specifically, it can be expressed as:

$$\eta_i^w = \begin{cases} \frac{|\{st^k \in ST_i^k | u(w_i, st^k) > u(w_i, st_i^k)\}|}{|ST_i^k|}, & |ST_i^k| > 0 \\ 0, & |ST_i^k| = 0 \end{cases}. \tag{18}$$

To some extent, it reflects the likelihood that w_i's utility may increase if his/her strategy is destroyed.

η_i^n represents the ratio of the maximum utility of strategies in next round to current strategy's utility, which can be calculated as:

$$\eta_i^n = \frac{\max_{st^{k+1} \in ST_i^{k+1}} u(w_i, \Gamma_{st^{k+1}})}{u(w_i, st_i^k)}. \tag{19}$$

In fact, it suggests the possible increase in utility of w_i when the task assignment round executes from the kth round of task assignment to the next one.

Action $a^t \in A$*:* At each time t, the action of agent w_i is defined as $a_i^t \in \{0, 1\}$, where $a_i^t = 1$ or 0 indicates whether w_i's strategy should be broken or not. Then, the joint action a^t of all agents can uniquely represent a break operation.

Reward $R = X \times A \to \mathbb{R}$*:* We evaluate the break operation based on the utility of solutions optimized by a pair of break and repair operations. As such, we consider the reward for joint action a^t, and give all agents the same reward. Then, the reward is denoted as the probability that the total solution utility obtained by destroy and repair operations is greater than that of the previous solution. In detail, during the training process, after a destroy operation is performed, we perform ϵ repair operations and replace the above probability by counting the frequency of solutions that satisfy the above conditions. Here, we define the set of total utilities of solutions obtained by one break operator and ϵ repair operations as $\hat{U} = \{\hat{u}_1, \hat{u}_2, \ldots, \hat{u}_\epsilon\}$, where $\hat{u} = \sum_{w \in W} u(w, \Gamma_{st^k})$. Then, given the total solution utility $\hat{u}_0$ before breaking, the reward function can be expressed as:

$$R = \frac{|\{\hat{u} \in \hat{U} | \hat{u} - \hat{u}_0 > 0; D(W) \leq \Theta\}|}{\epsilon}. \tag{20}$$

Transition $P = X \times A \times X \rightarrow [0, 1]$: It indicates the transforming probability from state x^t to x^{t+1} by performing action a^t.

Discount Factor $\gamma \in [0, 1]$: γ is the discount factor to balance the current reward with the future reward. The closer it gets to 1, the more important the long-term reward becomes.

We then apply DQN (a value-based deep reinforcement learning algorithm) proposed in [44], [45] to solve the above MMDP model.

2) Best Repair Operator: As described in Algorithm 3, we use the repair operator to adjust workers' strategies after executing the break operator. We can re-establish a new Nash equilibrium from the broken one. Here, we take the best-response framework as the repair operator. Since iteratively adjusting workers' strategies to their best strategies is unaffected by workers' previous strategies, we can find a Nash equilibrium based on any breaking solution. As such, it can serve as the repair operator. For details on how the best response framework works, please see lines 5–11 in Algorithm 1.

Discussion: In general, the LNS continuously improves the quality of Nash equilibrium solutions through break and repair operators. In this way, the increase in the number of iterations may improve the quality of the solution, but will inevitably lead to more time consumption. To this end, we train a DQN to learn break strategies to improve the quality and efficiency of optimization.

IV. Experiments

In this section, we evaluate the effectiveness and efficiency of our proposed approaches on two real-word datasets. All the approaches are implemented in Python3, and experiments were run on a PC with an Intel i9-9900 K 3.6 GHz CPU, NVIDIA GeForce RTX 2070 GPU, and 32 GB of memory.

A. Experiment Settings

Datasets: We assess our proposed approaches in two real datasets released by Didi Chuxing from Chengdu (CD) and Haikou (HK), respectively. Specifically, Chengdu dataset contains 7,065,937 orders from November 1 to November 30, 2016, and Haikou dataset contains 14,160,170 orders from May 1 to October 31, 2017. We regard the origin, destination, start time and end time of orders as the start location, destination, created time and deadline of tasks, respectively. Also, Chengdu dataset contains the trajectories of 1,181,180 workers. We extract the starting location, start time and end time from the trajectories as the initial location, arrival time and leaving time of workers, respectively. Then for the experiments in Haikou dataset, we use simple spatial and temporal mapping to map the workers in the Chengdu dataset to that in Haikou. As for the road networks of Chengdu and Haikou, we extract them from OpenStreetMap.[2] Specifically, there are 36,630 nodes and 50,786 edges on the road network of Chengdu, and 11,644 nodes and 15,398 edges on the road network of Haikou.

TABLE II
Parameter Settings

Parameter	Value
# of tasks	200, 400, **600**, 800, 1000
# of workers	100, 200, **300**, 400, 500
price balance parameter α	0.1, 0.3, **0.5**, 0.7, 0.9
capacity of workers	1, 2, 3, **4**, 5
fairness threshold	0.04, 0.05, **0.06**, 0.07, 0.08, 0.09, 0.1
size of time window	100, 150, **200**, 250, 300
# of iterations of LNS	0, 5, 10, 20, **50**, 100, 200, 500

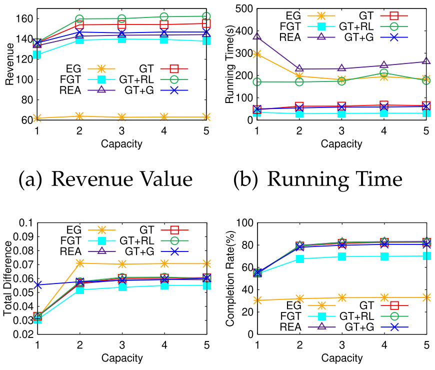
Parameter Settings: Table II describes the main parameter settings used in the experiments. In addition, we specify the clustering parameter ς and the number of repair operations ϵ as 100000 and 10 in our experiment, respectively.

Compared Algorithms: We compare our proposed algorithms against three algorithms.

- GT: it is our game theory-based approach described in Section III-B.
- GT+G: it is GT with the spatio-temporal-grouping based optimization described in Section III-C.
- GT+RL: it is an optimization of GT by large neighborhood search with a trained DQN as the repair operator, which is described in Section III-D.
- GT+LNS: it is an approach that replaces the destroy operator in GT+RL with a greedy destroy policy. In the experiment, we just use it to evaluate the performance of our adaptive destroy operator.
- EG: we implement a greedy algorithm based on the effective greedy algorithm in [46] and make an adjustment as the baseline. The adjusted algorithm iteratively selects a task-worker pair with the highest utility increase and without exceeding the fairness threshold until there are no suitable task-worker pairs. Calculating utilities by (5) and the additional fairness threshold constraint enable it to address our FGTA problem.
- REA [22]: it proposes two formulas to evaluate efficiency and fairness of a matching, respectively. It first computes a fair matching M_{fair} by dichotomy. Then it reassigns one match to another until the given trade-off of fairness and efficiency is reached.
  This is a state-of-the-art matching algorithm in ride-sharing systems.
- FGT [25]: it first calculates all possible candidate task sets for each worker and then obtains Nash equilibrium by best-response method (selecting the best strategy for workers according to the fair calculation formula proposed in [25]), and the solution of Nash equilibrium state as the final matching result. This is a state-of-the-art game theory-based approach to fair task assignment.

Evaluation Metrics: We evaluate the effectiveness and efficiency of our algorithms through four metrics, including total revenue, running time, total difference, and completion rate.

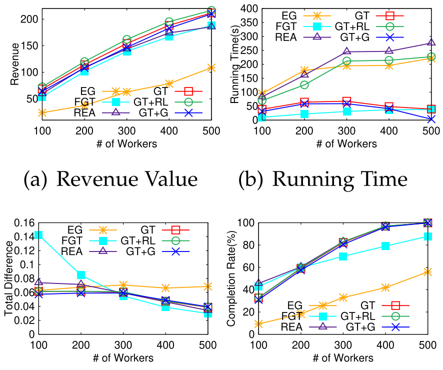
[2] https://www.openstreetmap.org.



(a) Revenue Value (b) Running Time

(c) Total Difference (d) Complete Rate

Fig. 7. Effect of the capacity of workers (CD).



(a) Revenue Value (b) Running Time

(c) Total Difference (d) Complete Rate

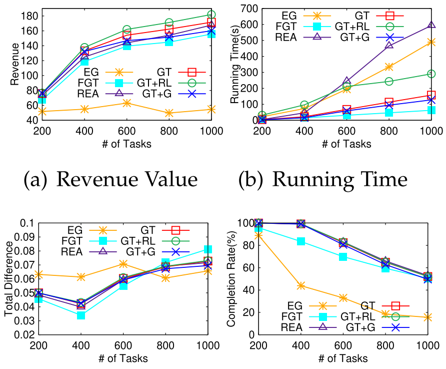
Fig. 8. Effect of the number of workers (CD).

B. Experimental Results

In this section, we analyze the experimental results in depth to validate the performance of our approaches. Since REA [22] can achieve a given trade-off of fairness and efficiency but their fairness indicator is not the same as ours, we adjust it to the same total difference with our GT algorithm under the default parameters. In this way, we can compare the performance of our approaches with REA in other metrics. Then, as the fairness threshold Θ only works on our approaches, we only evaluate the effect of it among our approaches. Also, we show the effectiveness of our RL-based repair operator for LNS framework acceleration by comparing GT+RL and GT+LNS.

Effect of the Capacity of Workers: Fig. 7 shows the experimental results when varying the capacity of workers from 1 to 5. Specifically, as suggested in Fig. 7(a), our proposed algorithms GT+RL, GT, and GT+G are always better than the compared approaches w.r.t., revenue. When c^w changed from 1 to 2, the revenue (Fig. 7(a)) and task completion rate (Fig. 7(d)) of the algorithms except EG increased significantly. However, when c^w is greater than 2, the revenue and completion rate of the algorithms change slightly. The reason is that when c^w is small, workers can participate in fewer tasks at the same time, so the task completion rate and total revenue are low. While with the increase of c^w to a certain value (e.g., in this experiment, 2), the total revenue and task completion rate stay almost constant, which indicates that the capacity is no longer the key factor in limiting workers from completing more tasks and getting more revenue. In terms of running time (Fig. 7(b)), algorithms except REA and EG are insensitive to c^w.

Effect of the Number of Workers: Fig. 8 illustrates the experimental results when varying the numbers of workers in each batch from 100 to 500. Generally, as presented in Fig. 8(a) and (d), as we increase the number of workers, the total revenue and task completion rate increase significantly. The reason is that more workers increase the chances of tasks being completed, and with them the total revenue increases. While as the number of workers increases from 400 to 500, although the increase in task completion rate slows down, the total revenue still increases significantly. It indicates that the increase in the total revenue comes from the improvement of service quality in this case, which further verifies the validity and applicability of our pricing model. In Fig. 8(b), the running time of algorithms except for GT and GT+G increases continuously. After the number of workers exceeds 300, the running time of GT and GT+G even decreases. The reason is that the increase in the number of workers results in fewer matching rounds in GT, GT+G, and GT+RL. While, in GT+RL, since the increase of workers brings additional cost of state calculation and destroy strategy selection, it does not show the decrease in running time like GT and GT+G.

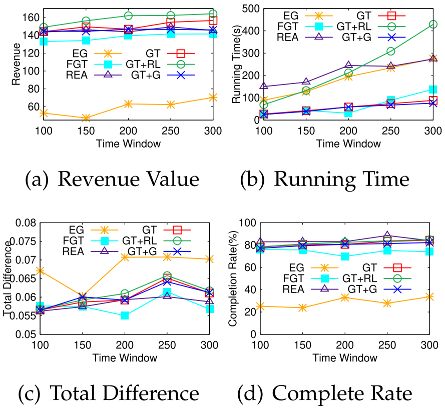


(a) Revenue Value (b) Running Time

(c) Total Difference (d) Complete Rate

Fig. 9. Effect of the number of tasks (CD).

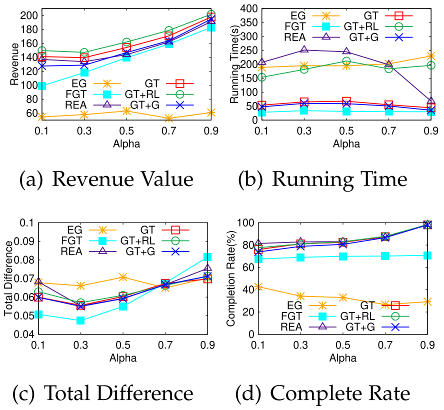
Effect of the Number of Tasks: In Fig. 9, we report the performance of proposed algorithms by varying the number of tasks in each batch from 200 to 1000. Generally, as the number of tasks increases, the total revenue increase obviously, but the rate of increase gradually slows down (Fig. 9(a)). This is because when workers are relatively sufficient, the increase in tasks obviously brings about the increase in total revenue. However, since workers' scheduling ability is limited by the spatio-temporal distribution, capacity and leaving times of workers, the increase in total revenue slows down. Also, as shown in Fig. 9(d) the completion rate of tasks is decreasing, but the total number of tasks completed is increasing. Then, as shown in Fig. 9(b), the running time of all algorithms is increasing across the board. The reason is that there are more candidate tasks per worker, so



(a) Revenue Value (b) Running Time

(c) Total Difference (d) Complete Rate

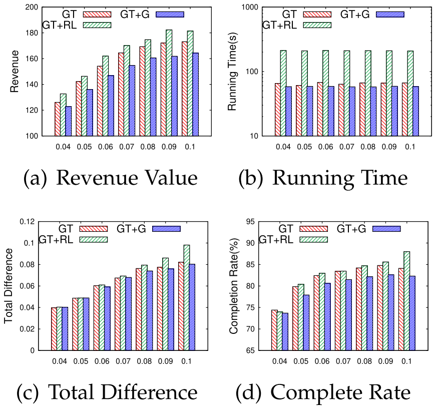
Fig. 10. Effect of the size of time window (CD).



(a) Revenue Value (b) Running Time

(c) Total Difference (d) Complete Rate

Fig. 11. Effect of α (CD).



(a) Revenue Value (b) Running Time

(c) Total Difference (d) Complete Rate

Fig. 12. Effect of the fairness threshold Θ (CD).

we need to deal with more candidate tasks when assigning tasks to workers. The running time of REA and EG increase more rapidly, indicating that they perform worse in large-scale data. Fig. 9(c) suggests that the total difference decreases and then increases as the scale of the task becomes larger. The reason is that when the number of tasks is less than that of workers, differences in revenue among workers are inevitable. While in the case of larger number of tasks, workers in advantageous spatio-temporal conditions (e.g., there are more tasks around) may be assigned more tasks, which is in line with our goal of maximizing revenue. However, despite the introduction of mechanisms to guarantee fairness, the total difference still increases within the range of fairness threshold Θ.

Effect of the Size of Time Window: Fig. 10 illustrates the experimental results with the time window of task assignment (batch size) varying from 100 to 300. Regardless of the size of time window, GT+RL and GT invariably perform better than other algorithms in terms of total revenue. As shown in Fig. 10(a) and (d), there is a slight increase in the total revenue and task completion rate of all algorithms as the time window size increases. The reason is that a larger time window size contributes to search for the better matching plans globally. As such, the running time of all algorithms shows an increasing trend in Fig. 10(b).

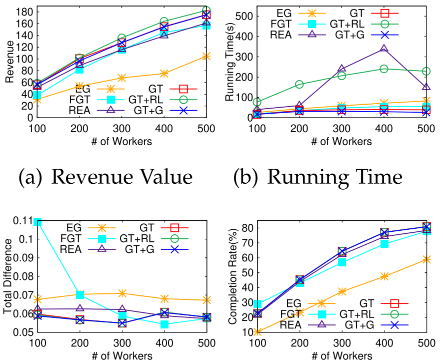
Effect of the Price Balance Parameter α: Fig. 11 illustrates the comparison of all the algorithms under different values of the price balance parameter α. As summarized in Table II, we adjust α from 0.1 to 0.9 at 0.2 intervals. Generally, in Fig. 11(a), the revenue of all algorithms changes significantly as α increases. However, the relative relationship among the performance of all the algorithms in terms of revenue has not changed greatly. For example, GT+RL and GT are always better than other algorithms, w.r.t., total revenue. It indicates that the superior performance of GT and GT+RL in revenue is almost unaffected by α. The reason is that α mainly affects the calculation of revenue, but rarely affects the effort of task strategy on revenue. Furthermore, Our proposed algorithms are insensitive to α w.r.t., the running time. As elaborated in Fig. 11(b), the running time of GT+RL is higher than that of GT because LNS framework optimizes the matching results through continuous iteration of destroy and repair.

TABLE III
EFFECT OF THE ITERATIONS OF LNS

# of Iterations	Running Time		Revenue	
	GT+LNS	GT+RL	GT+LNS	GT+RL
0	67.853	67.853	154.165	154.165
5	80.861	82.741	155.211	157.281
10	94.275	96.639	156.042	158.230
20	118.693	125.376	156.725	158.742
50	194.038	211.794	157.469	159.349
100	322.750	371.596	158.001	159.642
200	521.731	598.100	158.500	160.518
500	1379.176	1493.611	158.663	160.938

Effect of the Fairness Threshold Θ: As illustrated in Fig. 12, we evaluate the performance of our proposed algorithms by varying fairness threshold from 0.04 to 0.1. Specifically, when we continue to expand the fairness threshold, three algorithms have increased in total revenue (Fig. 12(a)) and task completion rate (Fig. 12(d)), and the increase of GT+RL is slightly higher than that of GT and GT+G. As expected, the increase in fairness threshold results in the increase in the total difference (Fig. 12(c)). However, the running time of the algorithm is insensitive to the change in the fairness threshold (Fig. 12(b)).

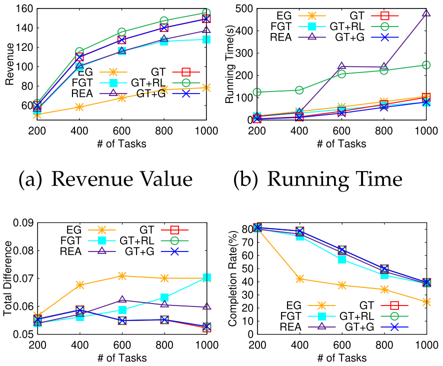
Effect of the Iterations of LNS: Table III illustrates the impact of adaptive destroy operator on total revenue and running time.



(a) Revenue Value (b) Running Time

(c) Total Difference (d) Complete Rate

Fig. 13. Effect of the number of workers (HK).



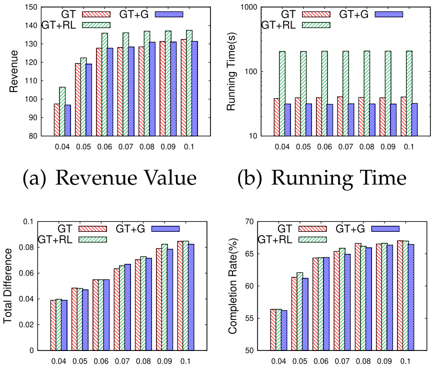
(a) Revenue Value (b) Running Time

(c) Total Difference (d) Complete Rate

Fig. 14. Effect of the number of tasks (HK).

As summarized in Table III, by training a DQN as destroy operator, the GT+RL can achieve higher total revenue with the same number of iterations. The reason is that compared with greedy destroy strategies, RL can select destroy strategies conducive to revenue improvement after reconstruction according to the current environment and state, which can be learned through sufficient training. In addition, with the increase of iterations, the running time of two algorithms increases almost proportionally. The reason is that the time cost is mainly due to the repair operation, and the number of iterations is exactly proportional to the repair operation. Since GT+RL spends extra time on state calculation, the time cost of GT+RL is slightly higher than that of GT+LNS in any case.

Experimental Results on Haikou Dataset: We further evaluate the effectiveness and efficiency of our algorithms when the number of workers, the number of tasks and the fairness threshold vary on Haikou data set. As shown in Figs. 13, 14, and 15, the experimental results on Haikou dataset are similar to those on Chengdu dataset. Specifically, as detailed in Figs. 13(a), 14(a), and 15(a), GT+RL achieves the highest total revenue, and our three game theory-based algorithms outperform other algorithms in terms of total revenue under the condition that the fairness threshold is met. In addition, as the number of workers and the number of tasks increase, the total revenue of all algorithms shows an increasing trend. For running time shown in Figs. 13(b), 14(b), and 15(b), GT+G has the best performance. The increase in the number of workers has less effect on running time than the number of tasks. In terms of the fairness threshold, its effect on running time is slight. While its effect on revenue and completion rate is significant, especially when the fairness threshold is changed from 0.04 to 0.05. This suggests that the excessive demand for fairness does damage the total revenue.



(a) Revenue Value (b) Running Time

(c) Total Difference (d) Complete Rate

Fig. 15. Effect of the fairness threshold Θ (HK).

Experimental Summary: In summary, the experimental results on the two datasets show a similar pattern. Specifically, GT+RL always performs best in terms of total revenue, followed by GT, GT+G, REA, FGT, and EG. All six algorithms perform similarly on completion rate as they do on total revenue, but GT+RL, GT+G, GT, and REA differ less in completion rate than in revenue. Because the complete rate is also heavily influenced by the spatial and temporal constraints of workers and tasks. In terms of running time, REA, EG, and GT+RL obviously take much of running time than that of GT, GT+G, and FGT. Since the total difference is directly affected by the fairness threshold, GT, GT+G, and GT+RL differ less in total difference. In terms of the effect of parameter settings, the revenue, complete rate, and running time are most sensitive to the changes in the number of tasks and workers but less sensitive to the variation in the size of the time window. Then only the total revenue is slightly affected by the price balance parameter α. While the fairness threshold has an obvious influence on total revenue, a slight influence on complete rate, but almost no influence on running time.

V. Related Work

In this section, we review literatures on task assignment and fairness in mobility services.

A. *Task Assignment in Mobility Services*

Task assignment has attracted widespread attention in crowdsourced mobility services. Existing efforts on task assignment focus on different goals, such as maximizing platform's total revenue [8], [9], [10], [48] and task complete rate [13], [15], or minimizing the total travel cost [16], [17].

TABLE IV
COMPARISON OF EXISTING FAIRNESS MODELS IN TASK ASSIGNMENT

Reference	Fairness Metric	Factors of Concern		
		accumulative fairness	working hours	dynamic
[22]	minimal utility of workers	✓	✓	✗
[23]	entropy variant of amortized fairness among workers	✓	✓	✗
[24]	Fagin-Williams-share worker proportion	✓	✗	✗
[25]	average utility difference among workers	✗	✗	✗
[28]	Jain's fairness index	✗	✗	✓
[29]	maximal proportional utility difference among workers	✓	✓	✓
[30]	distributive fairness	✗	✗	✗
[47]	workers' average chance to access task	✗	✗	✗
ours	average unit utility difference among workers	✓	✓	✓

[1] Here "accumulative fairness" indicates whether the fairness considered is accumulative fairness, "working hours" indicates whether the effect of worker's working hours on fairness has been paid attention to, and "dynamic" means the utility in existing models is dynamic or static.
[2] For the convenience of comparison, we use utility to represent the profitability indicator in existing works.

For instance, in [8], the authors studied the taxi order dispatching problem, and to address it in large-scale, they proposed a systematic solution combining reinforcement learning and combinatorial optimization algorithms. With the aim of maximizing global assignment quality, Cheng et al. [9] took both present and future workers/tasks into account by resorting to predictive techniques. In [13], Zhao et al. addressed a destination-aware task assignment, where workers are assigned appropriate tasks on the premise of reaching their destinations before deadlines, so as to maximize the task completion rate. To settle this problem, they decomposed workers into smaller worker clusters and developed the depth-first search algorithm with progressive bounds. In [17], the authors first focused on optimizing the worst task performance which is modeled as the task delay cost (travel cost). Further, they discussed the deterministic boundary of the problem and proposed algorithms to solve it. Since our problem involves ensuring a certain level of fairness while maximizing the overall revenue, the approaches of the above works are difficult to directly solve our problem.

B. Fairness in Mobility Services

The guarantee of fairness has gradually played an important role in the task assignment of crowdsourced mobility services, which is primarily aimed at ensuring income fair between crowdsourcing participants [47]. In general, recent efforts mainly focused on balancing effectiveness and fairness [22], [23], [24], [25], [28], [29], [30]. Also, as shown in Table IV, we present a summary table for comparison of fairness models in existing works.

A line of works take fairness and effectiveness (performance indicators, such as task completion rate [30], workers' income [29], total utility [23], [24]) as the common optimization objective and optimize them simultaneously. In [30], the authors proposed a two-stage assignment model to ensure that workers perform crowdsourced delivery tasks reliably and assign each worker a task that minimized unfairness. In online ride-hailing system, Sühr et al. studied the fairness of drivers' income being proportional to their time on the platform, by considering the fairness of drivers' revenue over time rather than the fairness of each match [29]. In [24], Chen et al. studied the problem of worker fairness problem and its variants in spatial crowdsourcing. Specifically, they solved their dual-objective matching problem of fairness and effectiveness by optimizing the joint function of fair and utility. Shi et al. [23] proposed an effective and efficient assignment scheme based on reinforcement learning, avoiding the short-sightedness of traditional fairness assignment algorithms. Also, work [25] maximized the overall fairness of task assignment on spatial crowdsourcing platforms by minimizing the income differences among workers. Specifically, they adopted game theory-based approaches to find the Nash equilibrium matching results by choosing the best strategies for each worker that favors fairness. While different from their work, our FGTA problem maximizes the workers' total revenue with a certain level of fairness.

VI. CONCLUSION AND FUTURE WORK

In this paper, we propose a novel fairness-guaranteed task assignment problem in crowdsourced mobility services, where the total revenue of workers is maximized under the premise of a certain degree of fairness guarantee. To solve this problem, we propose an effective multi-round task assignment approach and optimize its efficiency via a spatio-temporal grouping strategy. Moreover, we present a novel large neighborhood search optimization combined with reinforcement learning to improve the solution quality. Finally, we have conducted extensive experiments on two real-world datasets to show the efficiency and effectiveness of our proposed solutions by comparing them with state-of-the-art ones.

As for future work, we attempt to extend this work in two directions. First, some complex tasks, in reality, need to be completed by multiple workers, so we intend to study the problem of collaborative task assignment. Second, we plan to study the problem of task assignment among different platforms, which can integrate more workers and tasks in a wide range and achieve a great utilization increase of social resources.

REFERENCES

[1] "Uber," [Online]. Available: https://www.uber.com

[2] "Didi," [Online]. Available: https://www.didiglobal.com

[3] "Meituan," [Online]. Available: http://www.meituan.com

[4] "Gigwalk," [Online]. Available: https://www.gigwalk.com

[5] Y. Tong et al., "Spatial crowdsourcing: A survey," VLDB *J.*, vol. 29, no. 1, pp. 217–250, 2020.

[6] Y. Tong, L. Chen, and C. Shahabi, "Spatial crowdsourcing: Challenges, techniques, and applications," in *Proc. VLDB Endow.*, vol. 10, no. 12, pp. 1988–1991, 2017.

[7] B. Guo, Y. Liu, L. Wang, V. O. K. Li, J. C. K. Lam, and Z. Yu, "Task allocation in spatial crowdsourcing: Current state and future directions," *IEEE Internet Things J.*, vol. 5, no. 3, pp. 1749–1764, Jun. 2018.

[8] Y. Tong, D. Shi, Y. Xu, W. Lv, Z. Qin, and X. Tang, "Combinatorial optimization meets reinforcement learning: Effective taxi order dispatching at large-scale," *IEEE Trans. Knowl. Data Eng.*, to be published, doi: 10.1109/TKDE.2021.3127077.

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

[10] Y. Zhao, K. Zheng, Y. Cui, H. Su, F. Zhu, and X. Zhou, "Profit-driven task assignment in spatial crowdsourcing," in *Proc. Int. Joint Conf. Artif. Intell.*, 2019, pp. 1914–1920.

[11] Y. Li, H. Li, X. Huang, J. Xu, Y. Han, and M. Xu, "Utility-aware dynamic ridesharing in spatial crowdsourcing," *IEEE Trans. Mobile Comput.*, to be published, doi: 10.1109/TMC.2022.3232215.

[12] L. Zheng, L. Chen, and J. Ye, "Order dispatch in price-aware ridesharing," in *Proc. VLDB Endow.*, vol. 11, no. 8, pp. 853–865, 2018.

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

[14] L. Wang, Z. Yu, Q. Han, B. Guo, and H. Xiong, "Multi-objective optimization based allocation of heterogeneous spatial crowdsourcing tasks," *IEEE Trans. Mobile Comput.*, vol. 17, no. 7, pp. 1637–1650, Jul. 2018.

[15] Y. Zhao, K. Zheng, Y. Cui, H. Su, F. Zhu, and X. Zhou, "Predictive task assignment in spatial crowdsourcing: A data-driven approach," in *Proc. IEEE Int. Conf. Data Eng.*, 2020, pp. 13–24.

[16] Z. Chen, P. Cheng, Y. Zeng, and L. Chen, "Minimizing maximum delay of task assignment in spatial crowdsourcing," in *Proc. IEEE Int. Conf. Data Eng.*, 2019, pp. 1454–1465.

[17] P. Cheng, X. Jian, and L. Chen, "An experimental evaluation of task assignment in spatial crowdsourcing," in *Proc. VLDB Endow.*, vol. 11, no. 11, pp. 1428–1440, 2018.

[18] Y. Zhao, K. Zheng, H. Yin, G. Liu, J. Fang, and X. Zhou, "Preference-aware task assignment in spatial crowdsourcing: From individuals to groups," *IEEE Trans. Knowl. Data Eng.*, vol. 34, no. 7, pp. 3461–3477, Jul. 2022.

[19] L. Zhao, W. Tan, B. Li, L. Xu, and Y. Yang, "Multiple cooperative task assignment on reliability-oriented social crowdsourcing," *IEEE Trans. Serv. Comput.*, vol. 15, no. 6, pp. 3402–3416, Nov./Dec. 2022.

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

[21] M. Asghari et al., "Price-aware real-time ride-sharing at scale: An auction-based approach," in *Proc. 24th ACM SIGSPATIAL Int. Conf. Adv. Geographic Inf. Syst.*, 2016, pp. 1–10.

[22] N. S. Lesmana, X. Zhang, and X. Bei, "Balancing efficiency and fairness in on-demand ridesourcing," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 5309–5319.

[23] D. Shi et al., "Learning to assign: Towards fair task assignment in large-scale ride hailing," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2021, pp. 3549–3557.

[24] Z. Chen et al., "Fair task assignment in spatial crowdsourcing," in *Proc. VLDB Endowment*, vol. 13, no. 11, pp. 2479–2492, 2020.

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

[26] M. H. Kashani and E. Mahdipour, "Load balancing algorithms in fog computing," *IEEE Trans. Serv. Comput.*, vol. 16, no. 2, pp. 1505–1521, Mar./Apr. 2023.

[27] S. Sthapit, J. Thompson, N. M. Robertson, and J. R. Hopgood, "Computational load balancing on the edge in absence of cloud and fog," *IEEE Trans. Mobile Comput.*, vol. 18, no. 7, pp. 1499–1512, Jul. 2019.

[28] J. Zhang, T. Jiang, X. Gao, and G. Chen, "An online fairness-aware task planning approach for spatial crowdsourcing," *IEEE Trans. Mobile. Comput.*, to be published, doi: 10.1109/TMC.2022.3229112.

[29] T. Sühr et al., "Two-sided fairness for repeated matchings in two-sided markets: A case study of a ride-hailing platform," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2019, pp. 3082–3092.

[30] F. Basik, B. Gedik, H. Ferhatosmanoglu, and K. -L. Wu, "Fair task allocation in crowdsourced delivery," *IEEE Trans. Serv. Comput.*, vol. 14, no. 4, pp. 1040–1053, Jul./Aug. 2021.

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

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

[33] Y. Xu, Y. Tong, Y. Shi, Q. Tao, K. Xu, and W. Li, "An efficient insertion operator in dynamic ridesharing services," *IEEE Trans. Knowl. Data Eng.*, 2020, pp. 1022–1033.

[34] S. Ma, Y. Zheng, and O. Wolfson, "Real-time city-scale taxi ridesharing," *IEEE Trans. Knowl. Data Eng.*, vol. 27, no. 7, pp. 1782–1795, Jul. 2015.

[35] Y. Tong et al., "Unified route planning for shared mobility: An insertion-based framework," *ACM Trans. Database Syst.*, vol. 47, no. 1, pp. 1–48, 2022.

[36] Y. Tong et al., "A unified approach to route planning for shared mobility," in *Proc. VLDB Endow.*, vol. 11, no. 11, pp. 1633–1646, 2018.

[37] S. W. AbuSalim et al., "Comparative analysis between Dijkstra and Bellman-Ford algorithms in shortest path optimization," in *Proc. IOP Conf. Ser. Mater. Sci. Eng.*, 2020, Art. no. 012077.

[38] Y. Li et al., "Efficient top-k matching for publish/subscribe ride hitching," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 4, pp. 3808–3821, Apr. 2021.

[39] A. K. Jain, "Data clustering: 50 years beyond k-means," *Pattern Recognit. Lett.*, vol. 31, no. 8, pp. 651–666, 2010.

[40] Y. Zhao et al., "Coalition-based task assignment in spatial crowdsourcing," in *Proc. IEEE Int. Conf. Data Eng.*, 2021, pp. 241–252.

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

[42] L. Du et al., "Dynamic private task assignment under differential privacy," in *Proc. IEEE Int. Conf. Data Eng*., 2023, pp. 2740–2752.

[43] Z. Zhang et al., "Reinforcement learning under a multi-agent predictive state representation model: Method and theory," in *Proc. Int. Conf. Learn. Representations*, 2021, pp. 1–12.

[44] J. Ke, F. Xiao, H. Yang, and J. Ye, "Learning to delay in ride-sourcing systems: A multi-agent deep reinforcement learning framework," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 5, pp. 2280–2292, May 2020.

[45] M. Volodymyr et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 1, pp. 529–533, 2015.

[46] P. Cheng, H. Xin, and L. Chen, "Utility-aware ridesharing on road networks," in *Proc. SIGMOD Int. Conf. Manage. Data*, 2017, pp. 1197–1210.

[47] R. Borromeo et al., "Fairness and transparency in crowdsourcing," in *Proc. Int. Conf. Extending Database Technol*., 2017, pp. 466–469.

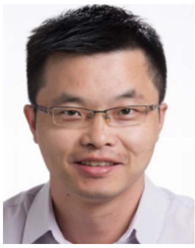
[48] B. Zhao et al., "Preference-aware task assignment in on-demand taxi dispatching: An online stable matching approach," in *Proc. AAAI Conf. Artif. Intell.*, 2019, pp. 2245–2252.

Yafei Li received the PhD degree in computer science from Hong Kong Baptist University, in 2015. He is currently a professor in the School of Computer and Artificial Intelligence, Zhengzhou University, China. His research interests span mobile and spatial data management, location-based services, and urban computing. He has authored more than 20 journal and conference papers in these areas, including *IEEE Transactions on Knowledge and Data Engineering*, *IEEE Transactions on Mobile Computing*, *IEEE Transactions on Services Computing*, *ACM Transactions on the Web*, *ACM Transactions on Intelligent Systems and Technology*, PVLDB, IEEE ICDE, WWW, etc.

Huiling Li received the BEng degree in software engineering from Zhengzhou University, in 2021. He is currently working toward the MEng degree in the School of Computer and Artificial Intelligence of Zhengzhou University. His research interests include multi-agent computing, deep learning, and spatiotemporal data processing.

Baolong Mei received the BEng degree in computer science and technology from Zhengzhou University, China, in 2022. He is currently working toward the MEng degree at the School of Computer and Artificial Intelligence, Zhengzhou University. His research interests mainly focus on location-based services, multi-agent computing, and spatiotemporal data management.

Xin Huang received the PhD degree from the Chinese University of Hong Kong (CUHK), in 2014. He is currently an assistant professor with Hong Kong Baptist University. His research interests mainly focus on graph data management, and mining.



Jianliang Xu (Senior Member, IEEE) received the BEng degree in computer science and engineering from Zhejiang University, Hangzhou, China, and the PhD degree in computer science from the Hong Kong University of Science and Technology. He is a professor in the Department of Computer Science, Hong Kong Baptist University. He held visiting positions with Pennsylvania State University and Fudan University. His research interests include Big Data management, mobile computing, and data security and privacy. He has published more than 200 technical papers in these areas. He has served as a program cochair/vice chair for a number of major international conferences including IEEE ICDCS 2012, IEEE CPSNA 2015, and APWeb-WAIM 2018. He is an associate editor of *IEEE Transactions on Knowledge and Data Engineering* and the *Proceedings of the VLDB Endowment* 2018.

Mingliang Xu received the PhD degree from the State Key Lab of CAD&CG, Zhejiang University, China. He is a professor in the School of Computer and Artificial Intelligence of Zhengzhou University, China. His current research interests include computer graphics, multimedia and artificial intelligence. He has authored more than 60 journal and conference papers in these areas, including *ACM Transactions on Graphics*, *IEEE Transactions on Pattern Analysis and Machine Intelligence*/*IEEE Transactions on Image Processing*/*IEEE Transactions on Circuits and Systems for Video Technology*, ACM SIGGRAPH (Asia)/MM, ICCV, etc.