# Fair Task Assignment in Spatial Crowdsourcing

Zhao Chen [†], Peng Cheng [*], Lei Chen [†], Xuemin Lin [*,#], Cyrus Shahabi [‡]
[†] *The Hong Kong University of Science and Technology, Hong Kong, China*
{zchenah, leichen}@cse.ust.hk
[*] *Shanghai Key Laboratory of Trustworthy Computing, East China Normal University, Shanghai, China*
pcheng@sei.ecnu.edu.cn
[#] *The University of New South Wales, Australia*
lxue@cse.unsw.edu.au
[‡] *University of Southern California, California, USA*
shahabi@usc.edu

## ABSTRACT

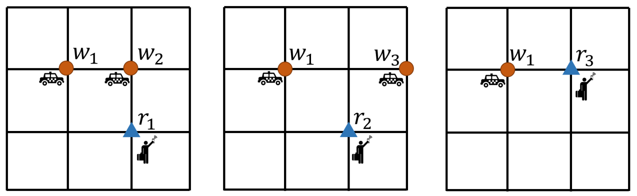
With the pervasiveness of mobile devices, wireless broadband and sharing economy, spatial crowdsourcing is becoming part of our daily life. Existing studies on spatial crowdsourcing usually focus on enhancing the platform interests and customer experiences. In this work, however, we study the fair assignment of tasks to workers in spatial crowdsourcing. That is, we aim to assign tasks, considered as a resource in short supply, to individual spatial workers in a fair manner. In this paper, we first formally define an online bi-objective matching problem, namely the Fair and Effective Task Assignment (FETA) problem, with its special cases/variants of it to capture most typical spatial crowdsourcing scenarios. We propose corresponding solutions for each variant of FETA. Particularly, we show that the dynamic sequential variant, which is a generalization of an existing fairness scheduling problem, can be solved with an $O(n)$ fairness cost bound ($n$ is the total number of workers), and give an $O(\frac{n}{m})$ fairness cost bound for the $m$-sized general batch case ($m$ is the minimum batch size). Finally, we evaluate the effectiveness and efficiency of our algorithm on both synthetic and real data sets.

**PVLDB Reference Format:**
Zhao Chen, Peng Cheng, Lei Chen, Xuemin Lin and Cyrus Shahabi. Fair Task Assignment in Spatial Crowdsourcing. *PVLDB*, 13(11): 2479-2492, 2020.
DOI: https://doi.org/10.14778/3407790.3407839

## 1. INTRODUCTION

Recently, with the rise of offline-to-online (O2O) and sharing economy applications, spatial crowdsourcing has become a popular business model with plenty of applications emerging (e.g., Task Rabbit [3], Seamless [5] and Eleme [4]). In these applications, spatial crowdsourcing platforms assign workers to suitable tasks, then the workers physically move to the target locations to perform the tasks. Although there has been a lot of existing studies on spatial crowdsourcing [13, 22, 28, 35], the fairness of task assignment from the workers' perspective (i.e., whether the workers are assigned with the same number of tasks if they work for the same amount of time), has not been well studied. The motivation of spatial workers is usually to receive a monetary reward for performing their assigned tasks. From the perspective of workers, tasks are one type of resource and the distributions of tasks may change dramatically at different locations/times. When many workers are competing for the short-supplied tasks during a time period, there should be an appropriate approach to allocate tasks fairly to workers.

*Proceedings of the VLDB Endowment,* Vol. 13, No. 11
ISSN 2150-8097.
DOI: https://doi.org/10.14778/3407790.3407839



(a) Round 1 (b) Round 2 (c) Round 3

Figure 1: A Motivation Example.

For example, on the online ride hailing platforms (e.g., Uber [7] and Didi Chuxing [1]), passengers' requests arrive dynamically within the entire city and the platform assigns them to nearby drivers (workers) round by round (e.g., every 2 seconds as a round [8]). When there are fewer passengers than drivers in some region in a given round, some drivers have to wait for the next round to be assigned a task. Task assignments are usually determined according to some global objectives such as maximizing the throughput of platforms (i.e., the number of matched task-worker pairs) [22] or minimizing the total moving distance of all vehicles [36]. However, the interest of individual worker is usually ignored during assignment, resulting in some workers to wait for many rounds without any tasks assigned in some extreme cases. It is an unfair situation that workers spending similar hours on the platform receive inequitable incomes. What is worse, unfairly treated workers may reduce working hours or permanently leave the platform, eventually harming the platform.

To illustrate, consider the following example.

**Example 1** (An Example of Fair Task Assignment in Spatial Crowdsourcing)**.** *As shown in Figure 1, in the example of the fair task assignment in spatial crowdsourcing, three workers (drivers),* $w_1 \sim w_3$*, and three tasks (requests)* $r_1 \sim r_3$ *are arriving at the platform in different rounds. Suppose the platform, to minimize the total travel cost, utilizes the greedy strategy to assign each request to its nearest driver. Specifically, as shown in Figure 1(a), driver* $w_2$ *is the closest worker to request* $r_1$*, thus the platform assigns driver* $w_2$ *to* $r_1$*. In round 2 as shown in Figure 1(b), one new driver* $w_3$ *and one new request* $r_2$ *appear, since* $w_1$ *and* $w_3$ *have the same distance from* $r_2$*, the platform randomly assigns* $w_3$ *to* $r_2$*. Sub-*

*sequently, in round 3 as shown in Figure 1(c), the platform sends $w_1$ to a newly arrived request $r_3$. We notice that $w_1$ waits for 2 rounds to receive a task while other workers can serve riders in one round from the time they join the platform, which is unfair for $w_1$. The platform can be more equitable by assigning $r_2$ to $w_1$ at round 2 and $r_3$ to $w_3$ at round 3. In this case, no worker observes a significantly longer waiting time than others.*

In Example 1, we observe that worker $w_1$ has chances to serve all the three tasks $r_1 \sim r_3$. However, as the greedy strategy is not fair, $w_1$ waits for 2 rounds until $r_3$ arrives.

There are several challenges on how to perform fair assignment. First, we need to formally define what is *fairness* in spatial crowdsourcing scenarios. Although the straightforward definition of *fairness* is to treat everyone equally, the rigorous definition could be quite diverse in different contexts. In this work we define the worker fairness by extending a common concept, Fagin-Williams share (FW-share), proposed in the existing study of *Carpool Problem* [18]. Specifically, FW-share is designed for one-to-many matching with equal share, and we extend it to a generalized definition for many-to-many matching with variable share.

With the definition of *fairness*, the next problem is how to combine it with existing objectives such as minimizing total travel cost or maximizing total revenue. In other words, the fair assignment should optimize for the fairness cost without sacrificing other objectives. To address this, we formally define the Fair and Effective Task Assignment problem (FETA) with the optimization goal of maximizing a linear combination of the minimum individual worker fairness and the total utility. FETA is a bi-objective online matching problem, and we show that its static version is NP-hard.

We further define several special cases of FETA to capture various spatial crowdsourcing scenarios. Specifically, the single batch case of FETA is a static bi-objective matching problem for which we design a novel polynomial time exact algorithm for it. The dynamic sequential case of FETA is an online one-to-many matching problem which generalizes the common Carpool Problem, and we improve the existing Carpool algorithm to solve it with a similar $O(n)$ bound of fairness cost. Finally, for the general FETA with dynamic many-to-many batches, we develop an algorithm by integrating the previous techniques, which can achieve a bound of $O(\frac{n}{m})$ for $m$-sized batches.

To summarize, we have the following contributions:

- We study the worker fairness problem in spatial crowdsourcing and offer a formal problem definition in Section 2.
- We discuss the formal measurement of fairness cost in details and give an example of fairness measure function for many-to-many matching in Section 3.
- We study variants of FETA and design corresponding algorithms for each variant in Sections 4, 5, 6 and 7.
- We conduct extensive experiments on real and synthetic data sets to show the effectiveness and efficiency of our proposed algorithm in Section 8.

Finally, Section 9 summarizes related works. Some supplementary proofs are included in the Appendix.

## 2. PROBLEM DEFINITION

In this section we introduce the basic concepts of spatial crowdsourcing, then give the formal definition and evaluation metric.

Table 1: Symbols and Descriptions

| *Symbol* | *Description* |
|---|---|
| $G = \langle W, R, E \rangle$ | a bipartite graph representing one batch of connected requests and workers. |
| $R_{w_i}$ | the valid request set of worker $w_i$ |
| $W_{r_j}$ | the valid worker set of request $r_j$ |
| $F(G, w_i)$ | the deserved bonus of $w_i$ in $G$ |
| $c^x_{w_i}$ | the fairness cost of $w_i$ in batch $x$ |
| $\lambda^x_w$ | the cumulative fairness cost of $w_i$ at the $x$th batch |
| $u_{ij}$ | the utility score of $i, j$ being matched |
| $\mu^x_{w_i}$ | the cumulative utility of worker $w_i$ at the $x$th batch |

### 2.1 Basic Concepts of Spatial Crowdsourcing

**Definition 1** (Spatial Workers)**.** Let $w_i$ denote a worker, and he/she is active at location $l_i$ on time $bt_i$.

**Definition 2** (Spatial Requests/Tasks)**.** A spatial request $r_j$ is a three-tuple $\langle l_j, t_j, b_j \rangle$, where $t_j$ is the creation time, $l_j$ is the request location, and $b_j \in [0, B_{max}]$ is the request bonus.

Following existing studies [37], we also assume that each task requires exactly one worker to accomplish and has no failure or partial completion status. Spatial crowdsourcing usually has three matching modes: the *static mode*, the *online mode* and the *batched mode* [32, 33]. Without loss of generality, our definitions are based on the *batched mode* settings, where available workers and unassigned tasks are matched for each successive time period. The *static mode*, where all workers and tasks are given in advance, can be considered as one huge batch with all the workers and tasks. The *online mode*, where workers/tasks comes one-by-one, can be considered as a special case of batched mode with all 1-size matchings. We use a weighted bipartite graph, namely *worker-task graph*, to represent a batch.

**Definition 3** (Worker-Task Graph)**.** A graph $G = \langle W, R, E \rangle$ is given at each spatial crowdsourcing batch, where the worker set $W$ and the request set $R$ are the bipartite nodes at each side, and every worker-and-task pair $\langle w_i, r_j \rangle \in E$ has a utility $u_{ij} \in [0, U_{max}]$.

The utility is supposed to be a general indicator which represents overall interests of the platform/system. It can be any performance representations that differ between assignments, such as the task suitability and worker travel cost [14, 32], or a combination of them. The cost of a worker to finish a task is not explicitly modeled in Definition 2 and 3 because either they are negligible or can be modeled as the utility penalty.

Due to the constraints of spatial distance and the worker ability (such as the seat limit for ridesharing), not all workers and requests can be matched together. If a worker $w_i$ and a request $r_j$ satisfies all constraints, they are valid to be matched together with a certain amount of utility. The utility can be any performance representation such as the task suitability and worker travel cost [14, 32]. Without loss of generality, worker-task graphs are assumed to be complete, as an invalid pair can be considered as one pair with 0 utility.

### 2.2 The Fair and Effective Task Assignment Problem

**Definition 4** (Worker Fairness Measure Function)**.** Given a worker-task graph $G = \langle W, R, E \rangle$, a worker fairness measure function (measure function for short) $F(\cdot)$ is a mapping from each worker $w_i \in W$ to their *deserved bonus* in $G$ s.t.

$$\sum_{w_i \in W} F(G, w_i) = \sum_{r_j \in R} b_j$$

The *deserved bonus* of $w_i$ in $G$ is the bonus of his/her assigned tasks in the optimally fair assignment, which will be introduced in Section 3. Following the convention in existing works [10, 18, 26], we define the fairness cost of a worker as the difference between his/her deserved bonus proportion and his/her actual allocated proportion.

**Definition 5** (Worker Fairness Cost)**.** Given a worker-task graph $G_x = \langle W_x, R_x, E_x \rangle$, a matching $M_x \subseteq E_x$ on $G_x$ and a measure function $F$, the fairness cost $c^x_{w_i}$ of each $w_i \in W_x$ is:

$$c^x_{w_i} = \begin{cases} F(G_x, w_i) - b_j, & \exists r_j \quad s.t. \quad \langle w_i, r_j \rangle \in M_x \\ F(G_x, w_i), & \forall r_j \quad s.t. \quad \langle w_i, r_j \rangle \notin M_x \end{cases} \tag{1}$$

The intuition of Definitions 4 and 5 is that the discrete request bonus can be divided arbitrarily into any amount of shares and each worker deserves some portion of it. A worker $w_i$ is not matched means that the allocated proportion to him/her is 0, then the fairness cost should be his/her deserved bonus proportion, i.e., $F(G, w_i)$. $w_i$ is matched with a task $r_j$ indicates that the allocated proportion of $w_i$ is $b_j$, then the fairness cost is $F(G, w_i) - b_j$ (can be negative). More details about fairness principles are introduced in Section 3.

With the formal definition of worker fairness cost, we define the fair and effective task assignment problem as below.

**Definition 6** (Fair and Effective Task Assignment Problem, FETA)**.** Given a worker fairness measure function $F$ and a series of $X$ worker-task graphs $\mathbb{G} = \{G_1, G_2, ..., G_X\}$ arriving one by one, where $G_x = \langle W_x, R_x, E_x \rangle$, FETA is to find a matching $M_x$ for each $G_x$ such that the following objective is maximized

$$(1 - \alpha) \sum_{x=1..X} \frac{\mu^x}{X} - \alpha \max_{w_i \in W} \lambda^X_{w_i} \tag{2}$$

where:
$\mu^x = \sum_{\langle i,j \rangle \in M_x} \frac{u_{ij}}{|R_x|}$ is the $x$-th batch utility; and
$\lambda^X_{w_i} = \sum_{x=1..X} c^x_{w_i}$ is the cumulative fairness cost of $w_i$; and
$\alpha \in [0, 1]$ is the fairness importance parameter.

There are two separated components in the goal of FETA: maximizing of the total utility and minimizing of the maximum fairness cost. The weight parameter $\alpha$ determines how important fairness cost is compared with utility. FETA can be categorized as a *bi-objective* batch-based matching problem and its combined goal is the linear weighted form of bi-goals [16]. A brief introduction of multi-objective optimization is given in Section 4.1.

The general case of FETA is designed for the spatial crowdsourcing in batch mode. When $|R_x| = 1$ and $\forall x \in [1, 2, \cdots, X]$, it becomes a dynamic sequential case FETA fitted for the online mode of spatial crowdsourcing. When $X = 1$, it is a single batch case FETA. When all batches are given in advance, it is the static case FETA. In addition, FETA is called $m$-sized when each $G_x \in \mathbb{G}$ has at least $m$ tasks.

## 2.3 Performance Evaluation Metric of FETA

FETA is essentially an online problem because the worker-task graphs are given one by one dynamically. Usually the performance of online algorithms are evaluated by the competitive ratio [11], which represents how much the online result is worse than the static optimal result. While the static version of FETA, with the whole graph series given in advance, is an NP-hard problem (shown in Section 5). Therefore, the competitive ratio is not appropriate for FETA, thus we use the **actual online result** instead. In fact, using the online result directly for performance evaluation is also the convention in most existing fairness scheduling works [15, 18, 26].

In our analysis, we assume that the input of FETA is given by an **adaptive adversary** following the convention in the existing studies [10, 15]. Briefly speaking, an adaptive adversary knows all information about how the algorithm runs and can adjust its input of future rounds accordingly. Adaptive adversaries can give those worst case problem instances and thus are widely used to analyze the upper bound of online algorithms. For a detailed introduction of adversary types and the results of fairness scheduling with different types of adversaries, please refer to the related works [10, 11].

## 3. WORKER FAIRNESS MEASUREMENT

In this section, we first review some existing works about fairness measurement, i.e., FW-share proposed for the Carpool problem. The existing FW-share method, based on some intuitive fairness principles, can handle one-to-many matching but is not suitable for many-to-many matching. We then discuss some fairness measurement principles for the many-to-many matching scenario and propose a new fairness measurement function.

## 3.1 Existing Studies for One-to-Many Matching

Fagin and Williams proposed a simply defined fairness scheduling problem named the *Carpool Problem* [18]. Given a carpool with total $n$ persons to go to work together in $N$ days and only a subset of them may appear. On each day, they need to choose a driver as fairly as possible. They first designed a fairness measurement, namely Fagin-Williams share (FW-share), and then formally define the goal of minimizing the largest owed credit amount (i.e., fairness cost). We introduce their definition of FW-share here.

**Definition 7** (FW-Share Fairness Measurement [18])**.** For an $m$-sized subset among a $n$ person carpool, everyone owes the same credits as: $FW(n, m) = \frac{lcm(1,2,...,n)}{m}$, and the driver earns $lcm(1, 2, ..., n)$ credits ($lcm$ means the least common multiple).

For example, if there are 5 persons in a carpool, for convenience, we set the cost of each drive as $lcm(1, 2, 3, 4, 5) = 60$. For a day with 3 persons, each of them share $60/3 = 20$ of the cost and the select driver earn 60 credits. After the day, the driver's credit will increase by $60 - 20 = 40$, and the two passengers' credits will decrease by 20. Other people are not affected. The cumulative FW-share based measurement indicates the ideal credit of each people in a carpool.

The $lcm$ in the original FW-share definition is used to convert all shares to integers for easy calculation. Thus, for simplicity, we remove the $lcm$ and rephrase it within our problem setting as below.

**Definition 8** (FW-Share Worker Portion)**.** Given one request $r_j$ and its valid worker set $W_{r_j}$ in a one-to-many spatial crowdsourcing matching problem, the deserved proportion of $r_j$ for each $w_i \in W_{r_j}$ is $m^{w_i}_{r_j} = \frac{b_j}{|W_{r_j}|}$.

The intuition of FW-share worker proportion is to find the deserved amount of requests that each driver/worker should have. Specifically, under the ideal fairness-guaranteed setting, each valid worker should share a portion of the task. The difference between

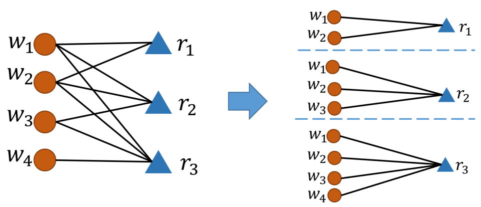


Figure 2: Example of one batch.

the ideal fair proportion and the actual matched result leads to the fairness cost in a matching. To find the ideal proportion, we can assume that one task can be split into small parts and finished by multiple workers together. For example, if the given batch has one task $r$ and $n$ workers, each worker shares the $1/n$ proportion of the task $r$. In online spatial crowdsourcing, as it is a one-to-many matching mode, we can apply spatial crowdsourcing FW-share worker proportion to measure the fairness for workers.

D. Coppersmith *et al.* proposed four simple principles [15] that a fairness measurement for the Carpool problem should follow and proved that FW-share, surprisingly, is the only one that satisfies all of them.

**Definition 9** (FW-share principles [15])**.** A fairness measurement for the Carpool problem should satisfy the following principles:

- **Full Coverage**: The total shares of all persons in the whole schedule should equal to the total number of trips (i.e., the total times of driving).
- **Symmetry**: People with the same schedule have the same shares.
- **Dummy**: Unscheduled people should have 0 share.
- **Concatenation**: The share of every person should remain the same in either two separated schedules or the concatenated one.

## 3.2 Extension to Many-to-Many Matching

As discussed in Section 2, tasks and workers in spatial crowdsourcing can arrive batch by batch, and each batch is a many-to-many matching problem. Although a many-to-many batch can be separated into several one-to-many batches (as shown in Example 2), we cannot simply apply the FW-share method on the one-to-many batches as the constraint of worker capacity may be violated.

**Example 2.** *There are three tasks $\{r_1, r_2, r_3\}$ with corresponding valid worker set $W_{r_1} = \{w_1, w_2\}, W_{r_2} = \{w_1, w_2, w_3\}$ and $W_{r_3} = \{w_1, w_2, w_3, w_4\}$. For simplicity, we assume that all task payments are 1 in this example. A direct method to calculate the proportional share of each worker is to separate the many-to-many graph into three one-to-many sub-graphs as shown in Figure 2, and then sum up the single task proportions as defined Definition 8. We have the result that the ideal share of $w_1$ is $\frac{13}{12} = \frac{1}{2} + \frac{1}{3} + \frac{1}{4}$ and the ideal share of $w_3$ is $\frac{7}{12} = \frac{1}{3} + \frac{1}{4}$.*

In Example 2, the share of $w_1$ is $\frac{13}{12} > 1$, which is not a reasonable share. In most spatial crowdsourcing applications (e.g., online car-hailing and food delivery), Once a worker is assigned with a task, he/she will dedicate to the task for a while. Therefore, even a worker is valid for more than one task in one batch, he/she can only be assigned to at most one task. If we calculate the ideal share of each worker in a batch through simply summing up the ideal shares of all his/her valid tasks, some workers may have ideal shares larger than their workload limitation. Thus, the principles for share measurement in the many-to-many matching need to be revised.

We extend the four principles [19] to the many-to-many matching in spatial crowdsourcing scenario as follows.

**Definition 10.** Principles for Many-to-Many Matching Fairness

- **Batch Coverage**: For each batch, the total ideal share of all workers should equal to the total bonus of all tasks.
- **Interchangeability**: For any two workers $w_1, w_2$, if their assignments are always interchangeable with the same bonus, they should have equal share.
- **Limited Workload:** The share of each worker should not exceed the maximum bonus of the batch.

Specifically, *Batch Coverage* and *Interchangeability* are the naturally extended versions of *Full Coverage* and *Symmetry* [19]. *Limited Workload* is a new constraint brought by the many-to-many matching scenario. For example, in Example 2, the maximum bonus is 1, thus the share of $w_1$ violates the Limited Workload principle (i.e., $\frac{13}{12} > 1$).

The principles of many-to-many matching is not easy to be fully satisfied at the same time. For example the fairness calculation in Example 2 violates the Limited Workload principle. If we adjust it by scaling down all share to make the largest share equal to 1, then it violates the Batch Coverage.

We propose a novel total matching count based measurement, namely Matching Count Share (MC-share), for the many-to-many spatial crowdsourcing scenario. MC-share satisfies all fairness principles discussed in Definition 10. The detailed proof of its satisfaction of the principles for many-to-many matching fairness is presented in the Appendix B.

**Definition 11** (MC-Share)**.** Given a worker-task graph $G = \langle W, R, E\rangle$, for any $w_i \in W$, let $\mathbb{M}$ be the set of all maximum matching over $G$ and $\mathbb{M}_{w_i} \subseteq \mathbb{M}$ be the subset of matchings including $w_i$, then we define the MC-share fairness measure function for $G$ as

$$MCF_G(w_i) = \frac{\sum\limits_{w_i, r_j \in \mathbb{M}_{w_i}} b_j}{|\mathbb{M}|}$$

For the bipartite graph in Figure 2, there are total 8 different maximum matchings. $w_1, w_2$ and $w_3$ are matched in 7 matchings and $w_4$ is matched in 3 cases. Assuming all tasks have bonus 1, we have $MCF(w_1) = MCF(w_2) = MCF(w_3) = \frac{7}{8}$ and $MCF(w_4) = \frac{3}{8}$.

Note that, in one-to-many cases, MC-share is same with FW-share. Thus, in the rest of this paper, we use MC-share as the fairness measure function consistently.

Next, we analyze several special cases of the FETA problem, namely the *single-batch* case of FETA (SBC-FETA), the *static version* of FETA (SV-FETA), *the dynamic sequential case* of FETA (DS-FETA) and the general case of FETA. The definitions of different cases are given in the corresponding sections respectively. A summarized comparison of these cases is given in Table 2.

## 4. THE SINGLE-BATCH CASE OF FETA

We propose an exact matching algorithm with detailed analyses for SBC-FETA, which is the foundation of other cases of FETA.

In SBC-FETA, there is one single batch of tasks and workers only. We will first introduce some basis of bi-objective problem and then present our solution for SBC-FETA.

## 4.1 Bi-objective Optimization Problems

Multi-objective problems are the ones having multiple optimization goals [16]. The key challenge of multi-objective optimization problem is to determine the superiority of solutions.

Table 2: Comparison of Different Cases of FETA

| | *Batch No.* | *Batch Size* | *Online* |
|---|---|---|---|
| *Single-Batch Case (SBC-FETA)* | 1 | unlimited | No |
| *Static Version (SV-FETA)* | unlimited | unlimited | No |
| *Dynamic Sequential (DS-FETA)* | unlimited | 1 task | Yes |
| *General Case FETA* | unlimited | unlimited | Yes |

For a single-objective problem, a solution $s_1$ is better than another one $s_2$ simply means that the objective function value of $s_1$ is larger/smaller than that of $s_2$. However, in multi-objective optimization problems, since there are more than one objective function values for each solution, the *dominance* relationship determines the goodness of solutions. A multi-objective solution $m_1$ *dominates* another one $m_2$ means that $m_1$ is not worse than $m_2$ in all objectives and better than $m_2$ in at least one objective. If $m_1$ is not dominated by any other solutions, it is a *non-dominated* solution.

All non-dominated solutions constitute the non-dominated set (a.k.a the Pareto-optimal set), which must contain the optimal solutions of any linear weighted goals. Therefore, the ideal result for a multi-objective problem is to find the exact non-dominated set. However, it is not easy (usually impossible in polynomial time), since objectives are not related with each other. We need to traverse through the whole solution space.

SBC-FETA is a bi-objective matching problem because it has two objective components: maximizing the utility goal and minimizing the fairness cost. Although the two goals are not related and the size of its solution space is $O(n!)$ (all possible matchings), we find that the problem can still be solved in polynomial time by utilizing the min-max essence of fairness definition. In the next part we present our method for SBC-FETA. Algorithm 1 returns the whole non-dominated solution set, with which our $\alpha$-balanced goal in Definition 6 can be achieved by one-pass iterating.

## 4.2 A Polynomial Time Exact Algorithm

The key idea of Algorithm 1 is to find the maximum utility matching for each min-max fairness cost matching. This can be done in polynomial time mainly because the amount of all possible min-max fairness cost values (at most $n\cdot(m+1)$, where $n$ is the number of workers and $m$ is the number of requests) is much smaller the size of the whole solution space. This amount is represented by the number of edges in the fairness graph $G_F$ constructed by the procedure *buildFairnessGraph*, which takes the original task-worker graph as the base structure and adds a dummy task for each worker to represent the situation when the worker is left unmatched.

Next, we explain the main algorithm, namely *singleBatchMatching*. First we find a min max matching $M_F$ of $G_F$, which can be solved by any min max matching algorithm (e.g., Threshold [12]). Note that, $M_F S$ prefers real tasks to dummy tasks because a min max matching must choose an edge with a smaller weight and a real task always has a lower weight than a dummy task. Then the max edge weight, $L_{(0)}$, of $M_F$ is used to generate the graph $G_U$, which consists of only edges with weights lower than $L_{(0)}$. With $G_U$ we can find a max sum matching $M_U$ (may be not unique) of it, which must be the matching with the largest total utility and also with maximum fairness cost $L_{(0)}$. Because there is no perfect matching with fairness cost value smaller than $L_{(0)}$, we only check fairness cost values larger than it. For each such value we repeat a similar process as above to find the maximum utility matching accordingly. The trick here is that we do not need to find the min max matchings for these values because such a matching must contains the edge with the max fairness cost and does not care how other edges with smaller weights are picked. Thus, in Line 10, the matching $M_U^{(l)}$ is

**Algorithm 1:** The solution for Single-Batch FETA

```
Data: a task-worker graphs $G = \langle W, R, E_F, E_U\rangle$
Result: the whole non-dominated set of matchings for $G$
1  Algorithm singleBatchMatching()
2      let $G_F$ = buildFairnessGraph() ;
3      find $G_F$'s min-max matching $M_F$ ;
4      let $L_{(0)}$ be the largest edge weight of $M_F$;
5      sort $G_F$ edges weights $L_{i,j} > L_{(0)}$ in ascending order as
         $\mathbb{L} = L_{(1)}, L_{(2)}, \cdots, L_{(n)}$ ;
6      let $G_U = \langle W, R, E_U - \{\langle w_i, r_j\rangle | L_{i,j} > L_{(0)}\}\rangle$ ;
7      find $G_U$'s max sum matching $M_U^{(0)}$ ;
8      for each $L_{(l)} = L_{i,j}$ in $\mathbb{L}$ do
9          remove $w_i, r_j$ and their incident edges $E_i, E_j$ from $G_U$ ;
10         find $G_U$'s max sum matching $M_U^{(l)}$ ;
11         add $w_i, r_j$ and $E_i, E_j$ back to $G_U$ ;
12         add $\langle w_i, r_j\rangle$ to $G_U$ and $M_U^{(l)}$ ;
13     return all $\langle M_U^{(l)}, L_{(l)}\rangle$ ;

1  Procedure buildFairnessGraph()
2      let $G_F = \langle G.W, G.R, G.E_F\rangle$ ;
3      for each worker $w_i$ do
4          let $L_{i,0} = F(G, w_i)$;
5          add a dummy task node $r_{w_i}$;
6          add a dummy edge $\langle w_i, r_{w_i}\rangle$ with weight $L_{i,0}$;
7          for each task $r_j$ do
8              update edge weight of $\langle w_i, r_j\rangle$ as
                 $L_{i,j} = F(G, w_i) - B_j$;
9      return $G_F$ ;
```

actually obtained with the edge $\langle w_i, r_j\rangle$ removed. The corresponding perfect max utility matching can be constructed by adding the edge $\langle w_i, r_j\rangle$ to $M_U^{(l)}$ and its fairness cost is $L_{i,j}$.

With the traditional notations used for graph matching problem, the time complexity of Algorithm 1 is $O(V^3E)$ if the max sum matching in Line 10 of *singleBatchMatching* is done by Hungarian Algorithm which costs $O(V^3)$.

## 5. THE STATIC VERSION OF FETA

For the static version of FETA (SV-FETA), the word *static* represents the opposite of the intrinsic *online* property of FETA. The only difference of SV-FETA from Definition 6 is that all worker-task graphs (batches) are given in advance. For most works of online problems, the static version is studied to be a comparison with the original online version. In this part, we give a brief analysis of SV-FETA. Particularly, we first show the NP-hardness of it and then introduce a min cost flow based algorithm which can solve SV-FETA with a constant number of batches in polynomial time.

## 5.1 The NP-hardness of SV-FETA

SV-FETA is also a bi-objective optimization problem. However, SV-FETA has a more complex problem space than SBC-FETA. In Section 4, the key property we utilized is that the amount of all possible fairness cost values is at most equal to the number of edges in the fairness graph of one single batch. While this property does not hold for multiple batches because the fairness cost is a cumulative value from all matching results. Actually, we find that even to find one non-dominated solution of a required fairness cost is NP-complete. The formal result is given in the below theorem.

**Theorem 5.1.** *Given $N$ batches of worker-task graphs and a required fairness cost $\lambda$, the problem of whether there is a matching can give the fairness cost of $\lambda$ is NP-complete.*

*Proof.* Our proof is achieved through a reduction from the subset sum problem [20] to our problem. The subset sum problem is: given a set $S$ of $n$ integers, is there a non-empty subset $S' \subseteq S$ whose sum is equal to $T$ ($S' \neq \emptyset$ and $\sum_{s\in S'} s = T$)?

Given a subset sum problem instance $I$ with a set $S$ of $n$ integers and a target sum $T$, we can transform it into our problem with the following steps:

1. Let $s_{max}$ be the maximum integer in $S$. For each $s \in S$, we transfer it to $s' = \frac{s}{s_{max}}$. The new number set is noted as $S'$. In addition, we update the target summation value $T$ to $T' = \frac{T}{s_{max}}$;
2. Add worker $w_i$ and $\langle w_i, r_i\rangle$ for each $s_i \in S$;
3. For each $v_i$, add a graph $G_i$ with $w$, $w_i$ and $r_i$, and let $F(G,w) = v_i$, $F(G, w_i) = 0$ and $B_i = v_i$;
4. let the required fairness cost $\lambda = T$.

For each graph $G_i$, if a matching matches $r_i$ to $w$, the fairness cost remain unchanged; if it matches $r_i$ to $w_i$, the fairness cost is increased by $v_i$. If we can find such a matching s.t. fairness cost is $\lambda$, we then find a subset $V \subseteq \{v_i | i = 1..n\}$ s.t. $\sum_{v\in V} v = T$, which represents a solution for the original subset sum problem. □

Due to the NP-hardness of the above problem, we cannot determine whether there is a non-dominated solution in polynomial time, thus cannot achieve the exact non-dominated set for SV-FETA or confirm the optimality of a linearly scaling result. However, the batch size is the key parameter for the complexity, and we will show an exact algorithm for those simple instances of SV-FETA with limited batch size.

## 5.2 A Solution for SV-FETA

Inspired by the solution for the static Carpool Problem [25], we design a min cost flow based algorithm for SV-FETA which can retrieve the whole exact non-dominated set in polynomial time if the number of batches is constant. The pseudocode is shown in Algorithm 2.

First let's check the sub procedure *buildCostNetwork*. The structure of the network is based on the existing approach for the static Carpool Problem [25,44]. Please refer to [43] for an illustration of the network structure. The Carpool Problem only cares about the fairness result but not any other properties such as the total utility. A maximum flow over such a graph contains a matching with the best fairness result, which has been proved to be a constant smaller than 1 ( [25]). Although the fairness definition in our problem generalizes that in the Carpool Problem, this does not affect the correctness of using this kind of networks. Because the capacity in Line 8 of *buildCostNetwork* only cares about the total fairness allocation but not those of individual edge weights. To handle the utility goal while maintaining the fairness result, we use the same network structure and add the flow costs to represent the reversed utility of each work-task pair. Thus, a min cost flow over our network will give a matching with the same minimum fairness cost and with a maximum total utility.

The min cost flow over one network gives one non-dominated solution only. To get the exact non-dominated set, we still need a method to traverse the whole solution space. The key observation here is that the total matched times of a worker $w_i$ is limited by the capacity of edge $\langle s, v_{w_i}\rangle$. Thus, if we adjust these capacity values 1 by 1, the corresponding min cost flow will give the matching also with maximum total utility but a *progressively released* minimum fairness cost. The phrase *progressively released* means that this process will finally enumerate all possible fairness costs. This is because a fairness cost must be caused by a worker be matched less

**Algorithm 2:** The min cost flow based algorithm for SV-FETA

```
Data: task-worker graphs $\mathbb{G} = \langle G_1, G_2 ..., G_k\rangle$
Result: the non-dominated set of matchings for $\mathbb{G}$
1  Algorithm staticMatching()
2      let $N$ = buildCostNetwork() ;
3      find $N$'s min cost flow $F_0$ from $s$ to $t$ ;
4      all $\langle w_i^x, r_j^x\rangle$ edges in $F_0$ forms a matching $M_0$;
5      for each worker $w_i$ do
6          let $a_i$ be the total appearance time of $w_i$ ;
7          let $C_i = [0, a_i]$ be the possible capacity set of $w_i$ ;
8      for each capacity combination $cc_y$ in $\bigtimes_{i=1..n} C_i$ do
9          update capacities from $s$ to $v_{w_i}$ for each $w_i$ according to $cc_y$;
10         find the min cost flow $F_y$ of update capacities ;
11         form the matching $M_y$ with $F_y$ if $M_y$ is perfect;
12     return $M_0$ and all $M_y$ ;
1  Procedure buildCostNetwork()
       Data: task-worker graphs $G_1, G_2..., G_k$
2      init the cost network $N$ with a source $s$ and a sink $t$ ;
3      for each $G_x = \langle W_x, R_x, E_x\rangle$ do
4          add a subnetwork $N_x$ with the same strcture as $G_x$ ;
5          add edge $\langle w_i^x, r_j^x\rangle$ in $N_x$ be and with capacity as 1 and cost
             as $U_{max} - u_{ij}$ ;
6      for each worker $w_i$ do
7          create a worker node $v_{w_i}$ ;
8          add an edge from the source $s$ to $v_{w_i}$ with capacity as
             $\sum_{x\in[1,k]} F(G_x, w_i)$ and cost as 0 ;
9          add an edge from $v_{w_i}$ to its every appearance in
             $N_x, x \in [1,k]$ with capacity 1 and cost 0 ;
10     add an edge from each task to the sink $t$ with capacity 1 and cost
         0 ;
11     return $N$ ;
```

than the time he/she deserves. Decreasing a worker's capacity by 1 in the network while increasing the other's by 1 will force a matching belong to him/her goes to the other one. In the loop at Line 8 to Line 11 of *staticMatching*, the algorithm actually tries all possible fairness costs by enumerating over all capacity combinations of all workers. The maximum matched times of a worker cannot be larger than the times he appears in different batches and the total batches is supposed to be a constant number $k$. In the mean time, some such combinations have more decreases than increases, thus do not satisfy the perfect matching constraint. Then, we know the total amount of all capacity combination $y \leq \binom{n}{k}$.

With the notations for network flows, the time complexity of Algorithm 2 is $O(V^{k+1} E \log V \log(VC))$, where $C$ is the largest cost, if the min cost flow in done by the network simplex algorithm [29,44].

## 6. THE DYNAMIC SEQUENTIAL CASE OF FETA

The Dynamic Sequential FETA problem (DS-FETA) is a special case of FETA, where batches are all one-to-many graphs. DS-FETA is a bi-objective online matching problem. Furthermore, as discussed in Section 2, DS-FETA also has its applications for those spatial crowdsourcing scenarios with instant assignments. Therefore, we present the analysis and solution for it independently in this part. We first briefly review some existing results, then propose our algorithm for DS-FETA with performance analysis.

## 6.1 Review of Existing Studies

DS-FETA can be considered as a generalization of the Carpool Problem. When all utilities are equal or the fairness importance

**Algorithm 3:** A Greedy Method for the DS-FETA

```
Data: the $x$-th batch $G_x = \langle W_x, R_x = \{r_x\}, E_x\rangle$
Data: the current cumulative fairness cost of each worker $\lambda_{w_i}^{(x-1)}$
      (the initial fairness cost $\lambda_{w_i}^{(0)} = 0$)
Result: a worker $w_i \in W_x$ to be assigned
1  Algorithm ds-greedy()
2      for each $w_i \in W_x$ do
3          let $\lambda_{w_i}^{x'} = \lambda_{w_i}^{x-1} + F(G_x, w_i)$ ;
4          let $\mu_{w_i}^{x'} = u_{ix}$ ;
5      let $w_{max} = \arg\max_{w_i \in W_x} \lambda_{w_i}^{x'}$ ;
6      if $\lambda_{w_{max}}^{x'}$ breaks Inequation (3) then
7          return $w_{max}$ and update all $\lambda_{w_i}^{x}$ ;
8      else
9          let $w_{max} = \arg\max_{w_i \in W_x} \mu_{w_i}^{x'}$ ;
10         return $w_{max}$ and update all $\lambda_{w_i}^{x}$ ;
```

parameter $\alpha$ is 1, DS-FETA with FW-share as the measure function is exactly the same as the Carpool problem. In existing studies, a greedy method, namely *FW-greedy* [18], is proved to be a quite effective online algorithm on solving the carpool problem [10, 15, 18]. Briefly, *FW-greedy* just choose the one with the largest owned credit to drive at each day. It is proved that the performance of *FW-greedy* is $O(n)$, where $n$ is the size of the carpool. It shows an important insight that the result is only affected by the total number of people involved but not the number of days. In addition, FW-greedy nearly reaches the known lower bound of the problem [15].

Some other existing results of the Carpool Problem and *FW-greedy* are reviewed in Section 9.

In the following, we will discuss how the generalization makes differences between the carpool problem and DS-FETA, and introduce our algorithm for DS-FETA.

## 6.2 A Greedy Method for the DS-FETA

DS-FETA has two major differences from the Carpool Problem. First, the carpool fairness requires that 1 workload equally shared by all workers and it is generalized in FETA as that tasks may have different bonus and workers may have different share. Second, other than to minimize the fairness cost, DS-FETA has the additional total utility part to be considered. To handle these two differences, we give Algorithm 3 which improves the greedy mechanism in FW-greedy and achieves a similar bound for the generalized problem.

Algorithm 3 is based on the same "greedy in each round" idea of FW-greedy. The idea is simple yet effective because after all we do not have any information other than the given batch and cumulative result of previous batch. The major difference between our algorithm and FW-greedy is that it does not always greedily assign the task to the most *unfair* worker. Particularly, it prefers to the largest utility (Line 9) instead of the local optimal fairness goal (Line 5) in those batches when the maximum fairness cost may be affected by the matching result.

The contribution of Algorithm 3 compared with existing works of the Carpool Problem is two-folds. First, a similar bound is obtained with the generalized fairness cost and, in the meantime, the additional utility part is considered heuristically. Second, with the Carpool Problem as a special case of DS-FETA, it shows that the previous always-greedy mechanism used by FW-greedy is actually not necessary here.

Next we show how the largest fairness cost is bounded in Algorithm 3. Note that the utility goal can be as bad as possible under the adaptive adversary setting (proof given in the Appendix part), thus we focus on the fairness goal performance only.

**Lemma 6.1.** *For any given batch of a DS-FETA, at least one of the following situation happens with Algorithm 3:*

1. *The largest fairness cost remains unchanged;*
2. *The largest fairness cost decreases and the difference between the largest fairness cost and the second largest one also decreases;*
3. *The difference remains smaller than $B_{max}$.*

*Proof.* Let the current worker with the largest fairness cost be $w$. If $w$ is not in the given batch, situation 1 happens. If $w$ is in but does not get assigned, supposing $w'$ gets assigned, we know $F(w) + \lambda_w < F(w') + \lambda_{w'}$, so the largest fairness cost remains and the new difference is $|F(w')+\lambda_{w'}-b_j-(F(w)+\lambda_w)| < b_j < B_{max}$. If $w$ is in and gets assigned, we know that $F(w) + \lambda_w > F(w') + \lambda_{w'}$, so the largest fairness cost decreases and the new difference is $|(F(w) + \lambda_w - b_j) - (F(w') + \lambda_{w'})|$ which is either smaller than the previous difference $F(w) - F(w')$ or smaller than $B_{max}$. □

Lemma 6.1 shows that the largest fairness cost only increases with the bounded difference. Based on this result, we have the performance bound as below.

**Theorem 6.1.** *For any DS-FETA instance with $n$ workers, Algorithm 3 achieves the upper bound $O(n)$.*

*Proof.* For simplicity, we omit the constant factor $B_{max}$ and assume workers are always sorted by their fairness costs, then the largest/first worker means the one with the largest fairness cost and so for the 2nd, 3rd etc. We prove a stronger result as below.

For any $n$-worker DS-FETA, let $w_{(k)}$ be the $k$th worker with fairness cost $\lambda_{(k)}$ and $W_{(k)}$ be the set of top $k$ workers, Algorithm 3 ensures that the following relation holds at all batches:

$$\lambda_{(1)} + \lambda_{(2)} + ... + \lambda_{(n-k)} \leq k(n-k) \quad \forall k \in \{0..n-1\} \quad (3)$$

When $k = 0$, Inequality (3) is actually $\sum_{i=1..n} \lambda_{(i)} \leq 0$. With all initial fairness costs being 0, this is always true for any matching results because the sum of all fairness costs is constant. In addition, at the initial status, Inequality (3) is obviously true for $k > 0$.

Suppose the $x$th batch with $r_j$ s.t. (3) breaks for the first time with $k = K$, which means $\lambda_{(1)}^{(x)} + \lambda_{(2)}^{(x)} + ... + \lambda_{(n-K)}^{(x)} > (K)(n-K)$, i.e., $\lambda_{(n-K)}^{(x)} > K$ after matching.

Because (3) holds before $x$th batch, we know $\lambda_{(1)} + \lambda_{(2)} + ... + \lambda_{(n-K+1)} \leq (K-1)(n-K+1)$, i.e., $\lambda_{(n-K+1)} \leq K-1$. This means $w_{(n-K+1)}$, as well as other workers afterward, cannot be in $W_{(n-K)}^{x}$ because of the increasing limit from Lemma 6.1. So we know the largest $n-K$ workers remains the same, i.e., $W_{(n-K)}^{x} = W_{(n-K)}$. From the definition, the sum of all fairness costs involved in one batch keeps the same, so there must be some other worker be in the batch and get assigned. While, this is not possible under our greedy matching so the assumption of such a batch existing is wrong. □

## 7. THE GENERAL CASE OF FETA

In this part we focus on the general case of FETA. Specifically, we first present our solution which adopts the ideas from the previous section. And then give its correctness proof and performance analysis.

**Algorithm 4:** The Algorithm for General Case FETA

```
Data: the $x$-th batch $G_x = \langle W_x, R_x, E_x \rangle$
Data: fairness costs $\lambda_{w_i}^{x-1}$, total utility $\mu^{x-1}$
Result: a matching $M_x$ for $G_x$
1  Algorithm matchTogether()
2      let $G_F$ = buildCumulativeFairnessGraph() ;
3      initiate $M_F, L_0, G_U, \mathbb{L}$ as in Algorithm 1;
4      let $\mathbb{L}_b = \{L | L \in \mathbb{L}, L > \lambda_{max} - B_{max}\}$ ;
5      if $\mathbb{L}_b = \emptyset$ then
6          let $M_x$ be $G_x$'s max-sum matching ;
7          return $M_x$ and update all $\lambda_{w_i}^x$ and $\mu^x$ ;
8      for each $L_{(l)} = L_{i,j}$ in $\mathbb{L}_b$ do
9          obtain $M_U^{(l)}$ as in Algorithm 1 ;
10     add $\langle w_i, r_j \rangle$ to $G_U$ for all $L_{i,j} \in \mathbb{L} - \mathbb{L}_b$ ;
11     find $G_U$'s max-sum matching $M_U^b$ ;
12     pick $M_x$ with the largest goal in all $M_U^{(l)}$ and $M_U^b$ ;
13     return $M_x$ and update all $\lambda_{w_i}^x$ and $\mu^x$ ;
1  Procedure buildCumulativeFairnessGraph()
2      let $G_F = \langle G.W, G.R, G.E_F \rangle$ ;
3      for each worker $w_i$ do
4          let $L_{i,0} = \lambda_{w_i}^{x-1} + F(G_x, w_i)$;
5          add a dummy task node $r_{w_i}$;
6          add a dummy edge $\langle w_i, r_{w_i} \rangle$ with weight $L_{i,0}$;
7          for each task $r_j$ do
8              update edge weight of $\langle w_i, r_j \rangle$ as
                   $L_{i,j} = \lambda_{w_i}^{x-1} + F(G, w_i) - b_j$;
9      return $G_F$ ;
```

## 7.1 A General Solution for FETA

The general case of FETA can be considered as several single-batches come dynamically. So, we utilize the solutions of the single-batch case (Algorithm 1) and dynamic sequential case (Algorithm 3) in Section 6 and propose a combined algorithm, named Match-Together (MT), for the general case as in Algorithm 4.

We summarize the steps of MT and explain some background ideas first. The algorithm runs at every batch to do online spatial crowdsourcing assignment. For each given task-work graph, MT creates a corresponding fairness graph similar as Algorithm 1 does. While the only difference is that the fairness graph in MT is based on cumulative fairness costs (Line 4 and Line 8 of the sub-procedure) but not just fairness shares of the current graph. The key step of Algorithm 4 is how to adopt the single-batch static method Algorithm 1 dynamically in the greedy style of Algorithm 3. Unlike the simple one-to-many graphs in the dynamic sequential case, the graphs in the general case is many-to-many. Therefore, all possible fairness costs ($\mathbb{L}$ in Line 4) need to be checked to see whether the largest fairness cost is *safe* from the current batch matching result. If so, MT will return the matching with utility maximized (Line 7). If not, i.e., either the worker with the largest fairness cost is in the given batch or some other worker's fairness cost may approach the largest one, MT will conduct similar steps as in Algorithm 1 to return the matching that maximizes the $\alpha$-parameterized goal by iterating over the whole non-dominated solution set. The time complexity of MT, the same as Algorithm 1, is $O(V^3E)$.

## 7.2 Algorithm Analysis

In this part we show the analysis of the $O(\frac{n}{m})$ performance bound of fairness cost for FETA.

**Theorem 7.1.** *For any $n$-worker FETA instance which is $m$-sized and $m \geq 3$, Algorithm 4 can achieve a result not worse than $\frac{n}{m-1}$.*

*Proof.* Similar as for Theorem 6.1, we prove a stronger result as below:

$$\lambda_{(1)} + \lambda_{(2)} + ... + \lambda_{(n-k)} \leq \frac{k(n-k)}{m-1} \quad \forall k \in \{3..n-1\} \quad (4)$$

We suppose the $x$th batch with $r_j$ s.t. (4) breaks for the first time with $k = K$, which means $\lambda_{(1)}^{(x)} + \lambda_{(2)}^{(x)} + ... + \lambda_{(n-K)}^{(x)} > \frac{k(n-k)}{m-1}$, i.e., $\lambda_{(n-K)}^{(x)} > \frac{k}{m-1}$ after matching.

Because (4) holds before $x$th batch, we know $\lambda_{(1)} + \lambda_{(2)} + ... + \lambda_{(n-K+1)} \leq (K-1)(n-K+1)/m$. In addition, the batch must have at least $m$ tasks. Thus, we have $\lambda_{(n-K+1)} \leq (K-1)/m$.

This means $w_{(n-K+1)}$, as well as other workers afterward, cannot be in $W_{(n-K)}^x$ because of the increasing limitation (the general situation similar as Lemma 6.1 for dynamic sequential case). So we know the largest $n - K$ workers remains the same, i.e., $W_{(n-K)}^x = W_{(n-K)}$. From the definition, the sum of all fairness costs involved in one batch keeps the same, so there must be some other worker be in the batch and get assigned. While, this is not possible under our greedy matching so the assumption of such a batch existing is wrong. ☐

We can see that Theorem 6.1 is actually a special case of Theorem 7.1 with $m = 2$.

## 8. EXPERIMENTS

In this section, we study the performance of all algorithms proposed for the worker fairness aware assignment problem on a real world taxi trip dataset and some synthetic data.

## 8.1 Experiment Setup

In this part, we first present the detailed setting of the synthetic dataset and the real world dataset, then give the evaluation metric and the implementation.

### 8.1.1 Datasets

**Synthetic Datasets.** To generate synthetic datasets, we first initiate a spatial space and a time range, then generate data with specified distributions. Spatial and temporal parameters are given in Table 3 and 4. Specifically, *Grid* is a $100 * 100$ Manhattan space with tasks and workers generated uniformly on each intersection point. *Euclidean* is a $1000 * 1000$ continuous 2-dimensional Euclidean space. Tasks and workers are generated following a Normal distribution centered at the point $(500, 500)$ with the variance of $100^2$. Arriving timestamps are generated following distributions in Table 4 and rounded into discrete values in $\{0, 1, ..., 9999\}$. To reduce the randomness of sampling, experiments with each space setting are repeated for 10 times and the average results are reported.

The utility and task bonus distributions as well as other parameters are given in Table 5. For simplicity, we use MC-Share in Definition 11 as the fairness measure function.

**Real Datasets.** We use the real taxi location and timestamp data set from the widely used public taxi trip data in New York city provided by NYC Taxi and Limousine Commission [6]. Specifically, we use the yellow taxi (one type of NYC taxis) data on Jan 2017 and Feb 2017. There are 18,878,953 taxi trip records in the dataset. A taxi trip includes the pick-up and drop-off locations and their timestamps. All locations are aligned to the road network provided by OpenStreetMap. The whole city is separated into $(2km * 2km)$-size grids as the spatial constraint (matchings are allowed only inside the same grid). The locations and timestamps of taxis are utilized to initialize the locations and online timestamps of crowd workers. The locations and timestamps of pick-ups of taxi trips are used to configure the locations and timestamps of tasks. Once a

Table 3: Synthetic data generation setting (locations)

| *space* | *distribution* | *parameters* | *size* |
|---|---|---|---|
| Grid | Uniform | None | $100 * 100$ |
| Euclidean | Normal | $\mu = 500, \sigma = 100$ | $1000 * 1000$ |

Table 4: Synthetic data generation setting (timestamps)

| *name* | *distribution* | *settings* |
|---|---|---|
| T1 | Uniform | $[0, 9999]$ |
| T2 | Exponential | $\lambda = 1$, accumulated to 9999 |
| T3 | Normal | $\mu = 5000, \sigma = 100$, rounded to $[0, 9999]$ |

worker $w$ finishes a task $t$, we assume that the worker $w$ will be available again at the drop-off location of $t$. Once $k$ tasks appear in a grid, a new batch is generated in the grid. The bonus of a task is configured as the actual fare of its corresponding taxi trip in the real dataset. Utility of matching between a task and a worker is determined by the bonus minus the estimated pick-up price (the price for the worker moving to the origin location of the task) according to the NYC taxi price table [2]. MC-Share is used as the fairness measure function.

### 8.1.2 *Goals and Evaluation Methods*

**Compared Algorithms.** We evaluate algorithms for the dynamic sequential case and those for the general batched case separately because they apply to different types of spatial crowdsourcing scenarios.

For the dynamic sequential case, we compare our DS-Greedy (DSG) with the following baseline algorithms.

- FW-Greedy (FWG), the original greedy method proposed in [18], which always picks the worker to minimize the current fairness cost.
- Utility-Oriented Greedy (UOG), the greedy method that always chooses the matching pair with a maximum utility.

For the general batched case, our algorithm Match-Together (MT) is compared with:

- Single Batch Greedy (SBG), the method utilizes the fairness graph in Algorithm 1 to minimize the current fairness cost for each batch.
- Utility-Oriented Bipartite Matching (UOM), the method uses min-sum bipartite matching algorithm [12] to maximize the total utility for each batch.

**Evaluation Metrics.** For both DSG and MT, the cumulative fairness cost part shows whether the maximum fairness cost follows the theory guarantee. Furthermore, we compare the actual utility performance with the best utility result from UOG and UOM. For the efficiency part, the time complexity of DSG is rather trivial and therefore we evaluate the time cost of MT only. All programs are implemented in Python 3.7, and run on a machine with a 6 core CPU at 4.3GHz, 32G memory and Ubuntu 18.04.

## 8.2 Results on Synthetic Datasets

### 8.2.1 *Overall Evaluation*

**The general case FETA.** First we conduct the overall evaluation for the general case on data generated with Grid, T3, and the default settings are shown in Table 5 in italic. Figure 3(a) shows the overall effectiveness result. The x-axis is the progress of the whole matching process (i.e., the proportion of finished batches) and the y-axis is the linear weighted goal of FETA with $\alpha = 0.5$. At the beginning of the process, some workers may not arrive, thus they do not have any effect on the result. Usually most workers are involved before the process goes to 20%. After the startup stage (the first 20% progress), the performances of all algorithms become stable, and we can see that MT achieves much better results than the two baseline methods during the whole process. The results of UOM decrease slightly till the end because it does not consider the fairness issue. After more batches are finished, unassigned workers always have chances to keep unassigned under UOM. Figure 3(b) shows how the overall effectiveness (the linear weighted goal with $\alpha = 0.5$) varies with different batch sizes. The result shows that MT outperforms the two baseline methods in experiments with different batch sizes. Figure 3(c) shows the running time of the compared approaches. The y-axis is the average time cost of each 1000 batches. MT is the slowest one among all tested methods. The time cost of MT and UOM is similar and both are much higher than SBG. The time costs are acceptable for real world scenarios, since spatial crowdsourcing applications usually do not have a burst of tasks in a short time and from the same region (e.g., more than 1000 people calling for a ride in the same block in one second is nearly impossible). Figure 3(d) shows the effectiveness result under different data distribution settings. MT outperforms baseline methods in all settings, especially for those with $T1$ (the uniform distribution in Table 4). Uniformly distributed tasks are more sparse than others, thus there are less chance to match unassigned workers in following batches. Therefore, the fairness issue for $T1$ is more severe and MT can perform better.

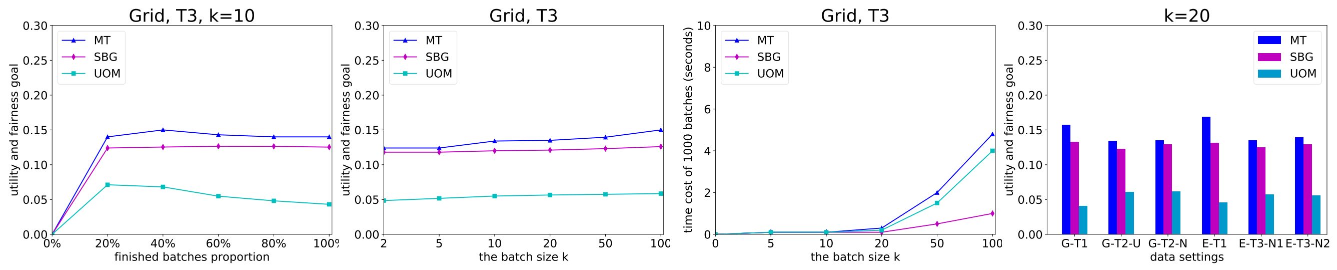
Table 5: Synthetic data parameters

| *factor* | *settings* |
|---|---|
| batch size $\lvert R \rvert$ | $2, 5, 10, \mathit{20}, 50, 100$ |
| worker task ratio $\lvert W \rvert : \lvert R \rvert$ | $1:1, 2:1, \mathit{5{:}1}, 10:1, 20:1$ |
| fairness importance $\alpha$ | $0.1, 0.3, \mathit{0.5}, 0.7, 0.9$ |
| utility | U: Uniform $[0, 2]$<br>*N: Normal*($\mu = 1, \sigma = 0.1$) |
| task bonus | *N1: Normal*($\mu = 1, \sigma = 0.1$)<br>N2: Normal($\mu = 1, \sigma = 0.5$) |

**DS-FETA.** The overall evaluation result for the dynamic sequential case with Grid, T3, and the default settings (italic font in Table 5) is given in Figure 4. The result is similar to the general case's. The effective result in Figure 4(a), 4(b) and 4(c) show that DSG outperforms the two baseline methods. We can see that although DSG cannot achieve the optimal fairness or utility result as shown in Figure 4(b) and 4(c), its combined result shown in Figure 4(a) is always much better than the baseline methods. The reason is that when DSG compromises for fairness, it always achieves a better utility as return. In addition, DSG trades utility for better fairness compared with UOM as well. Figure 4(c) shows the running time of all three algorithms. UOG is the fastest because it is a simple greedy algorithm. Our maximum matching based algorithms, DSG and FWG, are slower but still efficient.

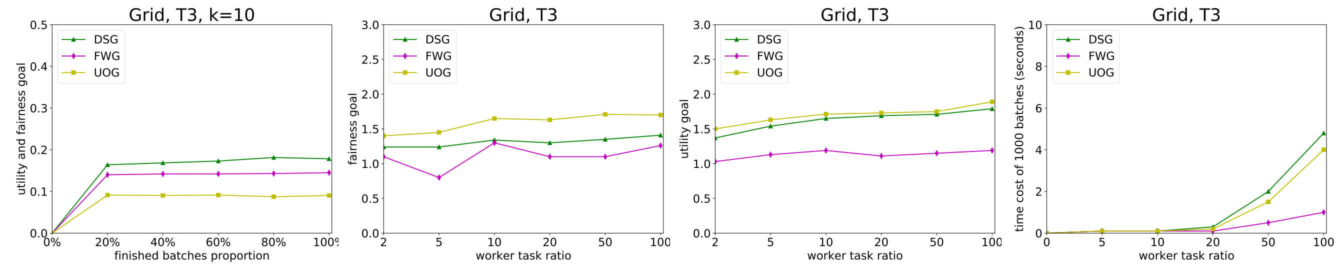
### 8.2.2 *Effects of Factors*

**Effect of relative sparsity of tasks and workers.** Relative sparsity represents the ratio between the numbers of workers and requests within a fixed area and time period. Spatial crowdsourcing tasks in different areas or different time periods may have quite different relative sparsity. For example, the taxi trip requests in a metropolis highly fluctuates in one day. Usually the performance of matching problems is stable when this ratio changes (e.g., the total travel cost in [36]). To evaluate the effect caused by the relative sparsity, we check the utility and fairness parts separately for different worker task ratios: $[1:1, 2:1, 5:1, 10:1, 20:1]$. The result is shown in Figures 5(a) and 5(b). We can see that the fairness cost decreases slightly as the ratio increases because there



(a) Result In Progress (b) Overall Result (c) Time Cost (d) Distributions

Figure 3: Overall result on synthetic data of the general case.



(a) Result In Progress (b) Fairness (c) Utility (d) Time Cost

Figure 4: Overall result on synthetic data of the sequential case.

is more chance for a worker to be left unmatched. While the utility result is better for a larger ratio because it has more matching choices with a larger utility. This result also shows the importance of such fairness aware algorithms.

**Effect of batch size $k$.** We want to check if the batch size $k$ affects the fairness performance as in Theorem 7.1. By fixing task worker ratio to $1 : 5$ and varying $k$ from 2 to 100, we give the results from MT and baseline methods. We expect that the result of MT should be better when the batch size become larger. As shown in Figures 5(c) and 5(d), the batch size effect is hard to see for smaller $k$s and there is an obvious negative correlation relationship between $k$ and the fairness result for $k \geq 20$. The reason is that smaller batches have less chance to cover the unfair workers and thus have less change to affect the final min-max fairness cost.

**Effect of fairness importance parameter $\alpha$.** Because FETA is a bi-objective problem, MT needs $\alpha$ to determine how to balance the two goals. We show the results by varying $\alpha$ from 0.1 to 0.9 in Figures 5(e) and 5(f). Note that, the two baseline methods do not have this parameter, thus their results do not change. For the results of fairness cost in Figure 5(e), we can see that the result of MT is similar to the result of SBG when $\alpha \leq 0.3$ and similar to result of UOM when $\alpha \geq 0.7$. Because changing $\alpha$ is supposed to regulate the fairness importance for matching. For the utility result in Figure 5(e), MT performs better as $\alpha$ increases. The reason is that, as $\alpha$ increases, it can always find larger utility matchings without hurting the fairness goal a lot.

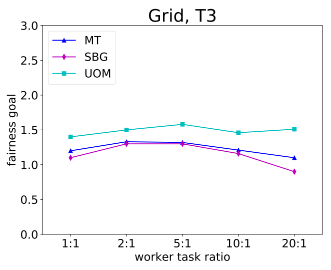
**Effect of distribution of workers and requests.** We check the effects of different distributions for task bonus and matching utility as shown in Table 5. For example, N-N1 means the data is generated with normal distributed utility and normal distributed ($\sigma = 0.1$) bonus. As shown in Figures 5(g) and 5(h), MT achieves both good fairness and utility results on datasets with all distributions. All distributions lead to similar results, but we can still see a slight difference between the normal distributed utility and uniform distributed utility. This is because uniform distribution provides more worker-task pairs with high utilities for algorithm to choose.

## 8.3 Results on Real Datasets

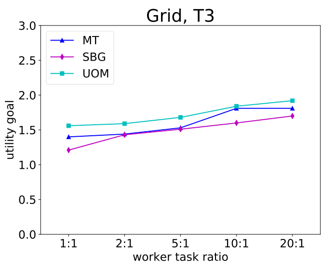
In the experiments on real data we mainly focus on the difference from results on the synthetic result and the real data. We evaluate the proposed algorithms on each daily data and use the average result as the final result. For each daily data, we group them by hours and show the progress result of the whole day in Figure 6(a). The taxi trip dataset does not have any obvious patterns or distributions. The major difference between different hours is the data density. We can see that MT has a better result on 12pm and 8pm. The task density is higher during these hours due to the same reason as its better performance on the synthetic data.

In Figures 6(b) and 6(c), we vary the batch size and separately show the results of fairness cost and utility. In Figure 6(b), we can see that all three methods have similar utility performance and the result of MT is closer to the optimal result compared with UOM. Compared with the result of synthetic data, the gap between UOM and MT is more obvious. In Figure 6(c), UOM performs much worse than SBG and MT for most batch sizes. The reason is that the tasks in real dataset is relatively sparse in most time. Thus, its fairness issue should be more serious. This is similar to the synthetic result in Figure 3(d) where MT performs better for sparse data distributions.

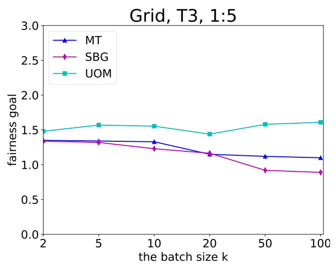
In addition, we compare the time costs of three methods and give the result in Figure 6(d). We group tasks be different time spans (2 seconds, 10 seconds, ...) as shown in the x-axis and record the max time cost among all batches. The time cost of MT is at most around 5 seconds for the 100 seconds batch with 623 tasks. For normal batch timespans, such as 2 seconds and 10 seconds, the time cost is always smaller than 100 milliseconds. Thus, its efficiency is good enough for industry level spatial crowdsourcing applications.



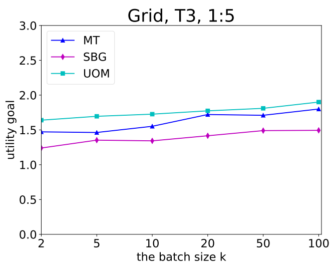
(a) Fairness of varying $|W| : |R|$



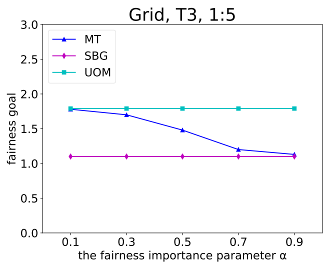
(b) Utility of varying $|W| : |R|$



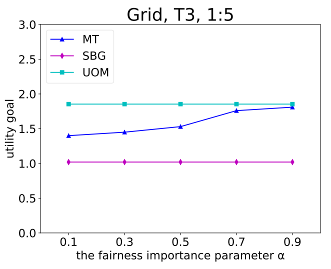
(c) Fairness of varying $k$



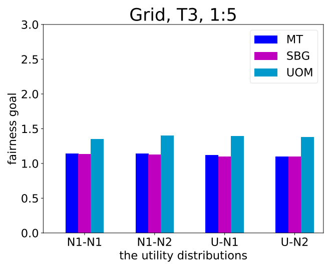
(d) Utility of varying $k$



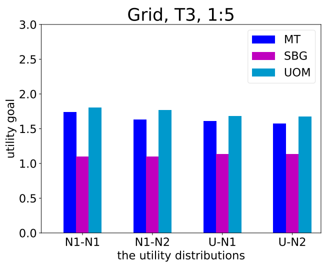
(e) Fairness result, vary $\alpha$



(f) Utility result, vary $\alpha$

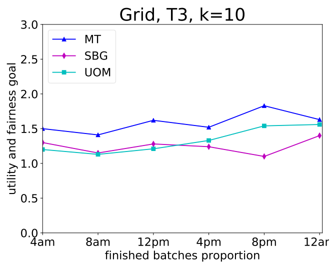


(g) Fairness of varying distribution

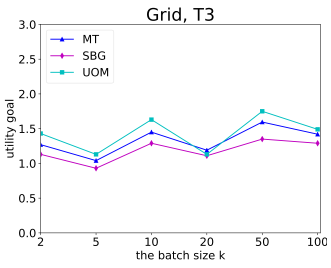


(h) Utility of, varying distribution

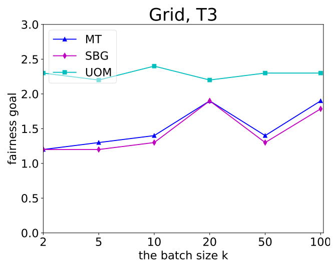
Figure 5: Result on synthetic data with different factors.



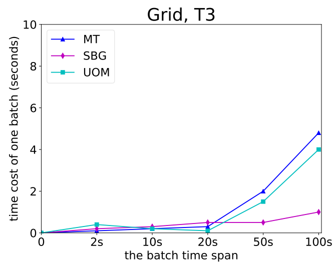
(a) Result In Progress



(b) Utility



(c) Fairness



(d) Time Cost

Figure 6: Overall result on real data.

## 8.4 Summary of Experiment Results

We summarize our major findings as follows:

- Compared to all baseline methods, MT has a better and more stable effectiveness result as well as a competitable efficiency result.
- The batch size does not affect MT performance a lot as expected in most cases. The reason is that the extreme unfair cases that lead to the lower bound in Theorem 7.1 is very rare or nearly impossible in most datasets.
- The relative worker task ratio in a batch can slightly affect both the fairness and the utility.

## 9. RELATED WORKS

**Task Assignment in Spatial Crowdsourcing.** In recent years, with the fast development of smart phones and other mobile devices, spatial crowdsourcing becomes more and more popular in various applications such as Offline-to-Online (O2O) service and car-hailing service. Task assignment is the core problems in spatial crowdsourcing [28,30,32–34,36,38–41,45,46]. The task allocation problem on spatial crowdsourcing is first proposed in [22]. They try to maximize the platform's throughput with batch-based algorithms, i.e., the total number of assigned tasks. Some follow up studies also focused on how to do better batch-based assignment for spatial tasks and propose additional constraints and goals, such as the crowd worker reliability [23], the maximum assigned utility goal [32] and the additional spatial temporal diversity goal [13]. Privacy issues in spatial task assignment are also studied in [27,31]. All of them consider the task assignment as a batch based matching problem aiming to maximize the total throughput.

In the real world scenario, both workers and tasks come dynamically. Thus, the spatial task assignment is essentially an online problem. An online model is first used in [21] to describe the assignment process, and they also proposed a novel method for one-side online task assignment. A two-side online matching problem with the total utility maximization goal and a well-performed online algorithm under the random order evaluation model is introduced in [35]. Then, a further generalized online model as well as an assignment algorithm with better performance under i.i.d evaluation model (both tasks and workers) is given in [36]. A very recent work [47] also proposed a stable marriage matching based optimization goal in the online manner.

**Fairness Scheduling.** The first related fairness aware scheduling problem named the "Carpool Problem" is proposed in [18]. They proposed an intuitive fairness measurement (named FW share afterward) and an online greedy algorithm (named FW-greedy) based on it. They gave a linear $N/3$ fairness lower bound for FW-greedy.

Several related fairness scheduling problems, including the edging orientation problem (EOP) and the vector rounding problem (VRP) is studied in [10]. They proved that VRP can be transformed to EOP with double expected cost and the Carpool Problem is a special case of VRP. They showed that a randomized algorithm, named local-greedy, achieved upper bound $O(\sqrt{n \log n})$ and lower bound $\Omega(\sqrt[3]{\log n})$ for the Carpool Problem.

A short yet comprehensive work [26] showed that the bound of local-greedy holds for the group of people interact with each other but no need for all people in the carpool. They proposed four self-evident principles that a fairness measure should follow and proved that FW-share is the only valid measurement for the principles.

The Carpool Problem is a special case of our FETA problem. To be specific, it is a DS-FETA without the utility goal and with all equal-bonus task and equal-share fairness measure. VRP is also a DS-FETA special case without the utility goal and with all equal-bonus task and arbitrary fairness measure.

**Bi-objective Online Matching.** Traditionally, the online property of online matching problems means that (only one side of) nodes of the graph for matching is revealed gradually one by one. Our problem is different from this kind of problems because we assume that graphs not node are revealed one by one. While, from another perspective, nodes come batch by batch can be considered as a special case of one by one. Thus, our problem can be considered as a variant of online matching problems.

Among these works about online matching, [9,17,24] studied the bi-objective problems. Several types of objective functions have been studied. [9] proposed the online bi-objective problem of maximizing both weight and cardinality. [17, 24], with a more popular setting in advertising applications, assume there are two types of edges and aims to find a matching that maximize either the cardinality or the total matched weight. To our best knowledge, there is no existing online bi-objective matching works about both min-max and min-sum goals.

## 10. CONCLUSION

In this paper, we proposed and studied the worker fairness issue in spatial crowdsourcing. We first define the fairness of workers in many-to-many bipartite graphs and proposed the fair and effective task assignment problem formally. We design well-bounded algorithms for the different cases of FETA for different scenarios in spatial crowdsourcing. Our work shows that the fairness issue brings some interesting problems, and we believe that these problems deserve more studies in the future.

## Acknowledgment

Zhao Chen and Lei Chen's work is partially supported by the Hong Kong RGC GRF Project 16209519, CRF Project C6030-18G, C10 31-18G, C5026-18G, AOE Project AoE/E-603/18, China NSFC No. 61729201, Guangdong Basic and Applied Basic Research Foundation 2019B151530001, Hong Kong ITC ITF grants ITS/044/18FX and ITS/470/18FX, Microsoft Research Asia Collaborative Research Grant, Wechat and Webank Research Grants, and Didi-HKUST joint research lab project. Peng Cheng's work is supported by Shanghai Pujiang Program 19PJ1403300. Xuemin Lin's work is supported by National Key R&D Program of China 2018AAA0102502, NSFC61672235, 2018YFB1003504, ARC DP180103096 and DP200101338. Cyrus Shahabi's work is funded in part by NSF grants IIS-1910950 and CNS-2027794. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the sponsors. Corresponding Author: Peng Cheng.

## APPENDIX

## A. LOWER BOUNDS OF FETA

We first give a worst-case adversary example inspired by the adversary firstly described in [10], then propose a proof for the $O(n)$ lower bound for fairness goal and arbitrary worst bound for the utility goal.

**Example 3** (Adaptive adversarial worst case in sequential model)**.** *The adversary acts at each round as: (a) if there exists workers $w_a, w_b$ with $\lambda_{w_a} = \lambda_{w_b}$, give the next task $r_j$ to $w_a$ and $w_b$ with equal share; (b) if not, give $r_j$ to the most unfair worker $w$ with* 1 *share and* 0 *utility, and to arbitrary other worker with* 0 *share and $U_{max}$ utility.*

**Theorem A.1** (Lower Bound of deterministic algorithms for adapative adversary)**.** *In (the sequential case of) FETA, no algorithm can achieve result better than $O(n)$, where $n$ is the number of all workers.*

*Proof.* For the adversary in Example 3, no matter which $w_a$ or $w_b$ gets matched, the new fairness scores will be $F+1/2$ and $F-1/2$. Since this is the only possible score changes, at any round the distance between different scores must be times of $1/2$. In addition, for any continuous rounds with such score changes, the maximum score among all affected workers must increase. The adversary must stop at some round otherwise the process will lead to unlimited maximum score. When the adversary stops, there are not any two same scores. Then there will not be two neighbor scores having a difference larger than $1/2$ because such scores pattern cannot get from the $-1/2, +1/2$ score change. Therefore, all workers are with different scores and their gaps are all $1/2$. In addition, the summation of all scores is 0, thus the largest score is $\frac{\lfloor n \rfloor}{4}$. When the fairness bound is reached, the utility can be any worse by given repeated such tasks. □

## B. PROPERTY OF MC-SHARE

**Lemma B.1.** *Given any work-task graph $G$, $MCF_G(w_i)$ in Definition 11 follows 1) Limited Workload, 2) Interchangeability, and 3) Batch Coverage, for any $w_i$.*

*Proof.* Because $|\mathbb{M}_{w_i}| \le |\mathbb{M}|$, $MCF_G(w_i)$ cannot exceed the largest bonus in batch $B_{max}$. So 1) is satisfied.

2) implies that $|\mathbb{M}_{w_1}| = |\mathbb{M}_{w_2}|$, they are always be matched together, and for any matchings they have the same bonus Thus, two interchangeable workers must have equal MC-share.

For 3), similar to the definition of $\mathbb{M}_{w_i}$, for each $r_j \in R$, we define $\mathbb{M}_{r_j}$ as the set of maximum matchings with $r_j$ matched and $\mathbb{M}_{w_i,r_j}$ be the set of those with both $w_i$ and $r_j$ matched. Wlog, we assume all $b_j = 1$ for simplicity. For each $w_i$, let $R_{w_i}$ be the set of tasks $w_i$ is assigned with in $\mathbb{M}_{w_i}$, then, because $|\mathbb{M}_{w_i}| = \sum_{r_j \in R_{w_i}} |\mathbb{M}_{w_i,r_j}|$, we have

$$\sum_{w_i} |\mathbb{M}_{w_i}| = \sum_{w_i,r_j} |\mathbb{M}_{w_i,r_j}| = |\mathbb{M}||\mathbb{M}|$$

$$\sum_{w_i \in W} f(w_i) = \frac{\sum_{w_i \in W} |\mathbb{M}_{w_i}|}{|\mathbb{M}|} = |\mathbb{M}|$$

□

Calculation of MC-share values involve enumerating over all maximum matchings in a graph. An efficient enough enumeration method for small bipartite graphs (e.g., with less than 100 nodes in each side) is introduced in [42].

## C. REFERENCES

[1] [online] didi chuxing. `https://www.didichuxing.com`.

[2] [online] nyc taxi fare. `https://www1.nyc.gov/site/tlc/passengers/taxi-fare.page`.

[3] [online] taskrabbit. `https://www.taskrabbit.com`.

[4] [online] Eleme. `https://www.ele.me`.

[5] [online] Seamless. `https://www.seamless.com/`.

[6] [online] trip record data from nyc taxi and limousine commission. `http://www.nyc.gov/html/tlc/html/about/trip_record_data.shtml`.

[7] [online] uber. `https://www.uber.com`.

[8] [online] uber: How does uber match riders with drivers? `https://marketplace.uber.com/matching`.

[9] G. Aggarwal, Y. Cai, A. Mehta, and G. Pierrakos. Biobjective online bipartite matching. In *International Conference on Web and Internet Economics*, pages 218–231. Springer, 2014.

[10] M. Ajtai, J. Aspnes, M. Naor, Y. Rabani, L. J. Schulman, and O. Waarts. Fairness in scheduling. *Journal of Algorithms*, 29(2):306–357, 1998.

[11] A. Borodin and R. El-Yaniv. *Online computation and competitive analysis*. cambridge university press, 2005.

[12] R. E. Burkard, M. Dell'Amico, and S. Martello. *Assignment problems, revised reprint*, volume 125. Siam, 2009.

[13] P. Cheng, X. Lian, Z. Chen, et al. Reliable diversity-based spatial crowdsourcing by moving workers. *PVLDB*, 8(10):1022–1033, 2015.

[14] P. Cheng, H. Xin, and L. Chen. Utility-aware ridesharing on road networks. In *SIGMOD*, pages 1197–1210, 2017.

[15] D. Coppersmith, T. Nowicki, G. Paleologo, C. Tresser, and C. W. Wu. The optimality of the online greedy algorithm in carpool and chairman assignment problems. *ACM Transactions on Algorithms (TALG)*, 7(3):37, 2011.

[16] M. T. Emmerich and A. H. Deutz. A tutorial on multiobjective optimization: fundamentals and evolutionary methods. *Natural computing*, 17(3):585–609, 2018.

[17] H. Esfandiari, N. Korula, and V. Mirrokni. Bi-objective online matching and submodular allocations. In *Advances in Neural Information Processing Systems*, pages 2739–2747, 2016.

[18] R. Fagin and J. H. Williams. A fair carpool scheduling algorithm. *IBM Journal of Research and development*, 27(2):133–139, 1983.

[19] M. Furuhata, M. Dessouky, F. Ordóñez, M. E. Brunet, X. Wang, and S. Koenig. Ridesharing: The state-of-the-art and future directions. *Transportation Research Part B: Methodological*, 57:28–46, 2013.

[20] M. R. Garey and D. S. Johnson. *Computers and intractability*, volume 174. freeman San Francisco, 1979.

[21] U. U. Hassan and E. Curry. A multi-armed bandit approach to online spatial task assignment. In *UTC-ATC-ScalCom*. IEEE, 2014.

[22] L. Kazemi and C. Shahabi. Geocrowd: enabling query answering with spatial crowdsourcing. In *SIGSPATIAL*. ACM, 2012.

[23] L. Kazemi, C. Shahabi, and L. Chen. Geotrucrowd: trustworthy query answering with spatial crowdsourcing. In *SIGSPATIAL*. ACM, 2013.

[24] N. Korula, V. S. Mirrokni, and M. Zadimoghaddam. Bicriteria online matching: Maximizing weight and cardinality. In *International conference on web and internet economics*, pages 305–318. Springer, 2013.

[25] S. Mneimneh and S. Farhat. The offline carpool problem revisited. In *International Symposium on Mathematical Foundations of Computer Science*, pages 483–492. Springer, 2015.

[26] M. Naor. On fairness in the carpool problem. *Journal of Algorithms*, 55(1):93–98, 2005.

[27] L. Pournajaf, L. Xiong, V. Sunderam, and S. Goryczka. Spatial task assignment for crowd sensing with cloaked locations. In *MDM*, volume 1, pages 73–82. IEEE, 2014.

[28] T. Song, Y. Tong, and et al. Trichromatic online matching in real-time spatial crowdsourcing. *ICDE*, pages 1009–1020, 2017.

[29] R. E. Tarjan. Dynamic trees as search trees via euler tours, applied to the network simplex algorithm. *Mathematical Programming*, 78(2):169–177, 1997.

[30] H.-F. Ting and X. Xiang. Near optimal algorithms for online maximum edge-weighted b-matching and two-sided vertex-weighted b-matching. *Theoretical Computer Science*, 607:247–256, 2015.

[31] H. To, G. Ghinita, and C. Shahabi. A framework for protecting worker location privacy in spatial crowdsourcing. *PVLDB*, 7(10):919–930, 2014.

[32] H. To, C. Shahabi, and L. Kazemi. A server-assigned spatial crowdsourcing framework. *ACM TSAS*, 1(1):2, 2015.

[33] Y. Tong, L. Chen, and C. Shahabi. Spatial crowdsourcing: Challenges, techniques, and applications. *PVLDB*, 10:1988–1991, 2017.

[34] Y. Tong, L. Chen, Z. Zhou, H. V. Jagadish, L. Shou, and W. Lv. SLADE: A smart large-scale task decomposer in crowdsourcing. *IEEE Trans. Knowl. Data Eng.*, 30(8):1588–1601, 2018.

[35] Y. Tong, J. She, and et al. Online mobile micro-task allocation in spatial crowdsourcing. *ICDE*, pages 49–60, 2016.

[36] Y. Tong, L. Wang, and et al. Flexible online task assignment in real-time spatial data. *PVLDB*, 10:1334–1345, 2017.

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

[38] Y. Tong, Y. Zeng, B. Ding, L. Wang, and L. Chen. Two-sided online micro-task assignment in spatial crowdsourcing. *IEEE Transactions on Knowledge and Data Engineering*, 2019.

[39] Y. Tong, Y. Zeng, Z. Zhou, L. Chen, J. Ye, and K. Xu. A unified approach to route planning for shared mobility. *PVLDB*, 11(11):1633–1646, 2018.

[40] Y. Tong and Z. Zhou. Dynamic task assignment in spatial crowdsourcing. *SIGSPATIAL Special*, 10(2):18–25, 2018.

[41] Y. Tong, Z. Zhou, Y. Zeng, L. Chen, and C. Shahabi. Spatial crowdsourcing: a survey. *The VLDB Journal*, 29(1):217–250, 2020.

[42] T. Uno. Algorithms for enumerating all perfect, maximum and maximal matchings in bipartite graphs. In *International Symposium on Algorithms and Computation*, pages 92–101. Springer, 1997.

[43] D. P. Williamson. [online] section 3.1.1, lecture notes on network flow algorithms. `https://people.orie.cornell.edu/dpw/techreports/cornell-flow.pdf`, 2004.

[44] D. P. Williamson. *Network Flow Algorithms*. Cambridge University Press, 2019.

[45] Y. Zeng, Y. Tong, and L. Chen. Last-mile delivery made practical: An efficient route planning framework with theoretical guarantees. *PVLDB*, 13(3):320–333, 2019.

[46] Y. Zeng, Y. Tong, L. Chen, and Z. Zhou. Latency-oriented task completion via spatial crowdsourcing. In *ICDE*, pages 317–328, 2018.

[47] 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 Conference on Artificial Intelligence (AAAI 2019)*, 2019.