Optimizing Dynamic Task Assignment in Spatial Crowdsourcing: Bilateral Preference-Aware Approaches

Yang Huang, Yumeng Liu, Xu Zhou, Tianyue Ren, Zhibang Yang, Keqin Li, *Fellow, IEEE*, and Kenli Li, *Senior Member, IEEE*

***Abstract*—Task assignment is a crucial challenge in spatial crowdsourcing (SC). Most existing studies have two limitations: First, only one-sided preferences of workers or tasks are taken into account, and the satisfaction of workers or tasks could be improved; Second, tasks are always assigned based on the current locations of workers, which is no. suitable for many real-life applications, such as carpooling, where the trajectories of workers require to be taken into account. To this end, we investigate a new problem of Bilateral Preference-aware Dynamic Task Assignment (BDTA), which is proven to be NP-hard, to maximize overall satisfaction by incorporating worker-task bilateral preferences and assigns tasks using the trajectories of workers. For the BDTA problem, we first propose a hybrid batch processing framework to address uneven data distribution. After that, a task-initiated bidirectional select algorithm is proposed to mitigates the impact of task order on the matching results. Furthermore, we propose an α-approximate task-initiated generalized deferred-acceptance algorithm and a reverse generalized deferred-acceptance algorithm to enhance the stability and overall satisfaction of task assignment results. Extensive experiments are conducted on both real and synthetic datasets to validate the effectiveness and efficiency of the proposed algorithms. Code is available at (https://github.com/good-hy/BPTA).**

***Index Terms*—Bilateral preference, spatial crowdsourcing, stable task assignment.**

I. Introduction

CROWDSOURCING, operating within the internet paradigm, leverages platforms to delegate tasks that were once reliant on specific individuals or organizations to a pool of workers, fostering a mutually beneficial arrangement. With advancements in mobile internet and crowdsourcing platforms, traditional crowdsourcing services have become extensively integrated into people's daily lives, enhancing convenience.

Unlike traditional crowdsourcing, spatial crowdsourcing (SC) relies on geographic locations and time. Well-known applications, such as Uber[1], Streetbees,[2] DiDi Chuxing[3], have become essential to daily life, offering significant convenience.

Task assignment is a critical challenge for SC platforms [1], [2], [3], [4], [5]. Although there has been extensive research on this topic, existing approaches still face several limitations.

- *Task assignment often lacks stability due to an imbalanced consideration of bilateral preferences:* User satisfaction in task allocation is crucial for the sustainability and growth of SC platforms [6]. Previous studies [4], [7], [8], [9], [10] mainly focus on unilateral preferences such as worker preferences (e.g., income [11], task difficulty [12]) or task preferences (e.g., reputation [2], skill [13], time availability [14]) to achieve single-objective optimization, such as maximizing the task completion number [15]. Task-based methods [14] prioritize task quality but limit worker agency and reduce motivation. Instead, worker-based methods [7] may lead to over-assignment and dissatisfaction due to workload imbalance. These approaches consider only the needs of a single party and fail to meet bilateral needs simultaneously, affecting user satisfaction and potentially leading to worker turnover [16].
- *Tasks are assigned based on the current locations of workers:* Existing SC problems are primarily focused on full-time workers, implying that tasks are assigned based on their proximity to the current location of these workers, as in [12], [17], [18], [19]. While location-based task allocation reduces the distance between workers and tasks, enabling faster completion and less delay, it may no. align with the routes workers prefer to take [20], [21]. With the rise of the sharing economy, some users are willing to accept simple crowdsourced tasks as part-timers during their free time or commuting hours to earn extra income. In our daily life, there are tasks such as

Received 19 November 2024; revised 24 July 2025; accepted 19 August 2025. Date of publication 12 September 2025; date of current version 9 January 2026. This work was supported in part by the Creative Research Groups Program of the National Natural Science Foundation of China under Grant 62321003, in part by the National Natural Science Foundation of China under Grant U23A20317, Grant 62572182, Grant 62402481, and Grant 62576051, and in part by the Natural Science Foundation of Hunan Province under Grant 2023JJ30083. Recommended for acceptance by Y. Che. *(Corresponding authors: Yumeng Liu; Tianyue Ren.)*

Yang Huang, Xu Zhou, Tianyue Ren, Zhibang Yang, and Kenli Li are with the College of Computer Science and Electronic Engineering, Hunan University, Changsha, Hunan 410000, China (e-mail: hy19@hnu.edu.cn; zhxu@hnu.edu.cn; hnurty@hnu.edu.cn; yangzb@hnu.edu.cn; lkl@hnu.edu.cn).

Yumeng Liu is with the Institute of Software, Chinese Academy of Sciences, Beijing 100045, China (e-mail: yumeng@iscas.ac.cn).

Keqin Li is with the Department of Computer Science, State University of New York, New Paltz, NY 12651 USA (e-mail: lik@newpaltz.edu).

This article has supplementary downloadable material available at https://doi.org/10.1109/TMC.2025.3603833, provided by the authors.

Digital Object Identifier 10.1109/TMC.2025.3603833

[1]https://www.uber.com
[2]https://www.streetbees.com
[3]https://www.didiglobal.com

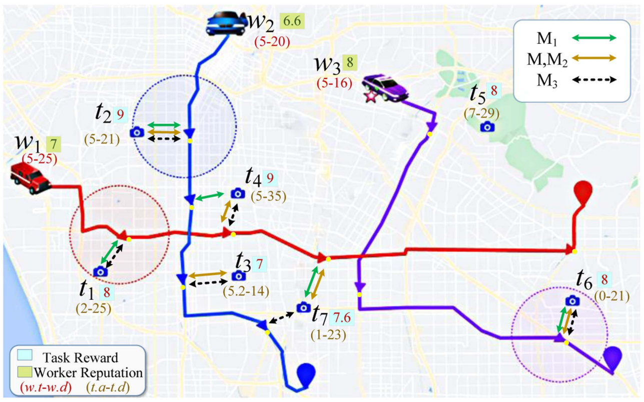

Fig. 1. An example of the BDTA problem.

TABLE I
INFORMATION OF WORKERS

Worker	$w.p$	$w.t$	$w.d$	$w.s$	$w.r$	$w.c$
w_1	80	5	25	7	3	2
w_2	60	5	20	6.6	3	2
w_3	45	5	16	8	3	2

TABLE II
INFORMATION OF TASKS

Task	t_1	t_2	t_3	t_4	t_5	t_6	t_7
$t.a$	2	5	5.2	5	7	0	1
$t.d$	25	21	14	35	29	21	23
$t.R$	8	9	7	9	8	8	7.6
$t.minR$	4	4	6	6	6	5	5

TABLE III
SHORTEST DISTANCE BETWEEN EACH WORKER-TASK

Worker\Task	t_1	t_2	t_3	t_4	t_5	t_6	t_7
w_1	2.8	4.12	2.5	2.35	5.61	10.61	1.7
w_2	3.11	2.35	1.75	2.2	7.77	4.33	1.8
w_3	9.99	8.86	3.88	4.12	2.9	2.5	2.4

TABLE IV
THE DISTANCE FROM THE WORKER TO THE DETOUR POINT

Task	t_1	t_2	t_3	t_4	t_5	t_6	t_7
w_1	12	∞	25	25	∞	∞	30
w_2	∞	15	35	25	∞	∞	45
w_3	∞	∞	∞	∞	5	40	30

TABLE V
DATASETS

Dataset	$\|T\|$	$\|W\|$	$\|w.p\|$
Berlin [34]	3,083	236	10$\sim$60
T-Drive [35], [36]	1,000$\sim$20,000	10,357	40

photographing landmarks, water quality testing, etc., which can be completed on the way home from work. For example, services like Amazon Flex[4] provide similar opportunities. In this case, trajectory-based task allocation is more meaningful.

To address these issues, our previous work [1] studied the bilateral preference-aware task assignment (BPTA) problem, incorporating the routine trajectories of workers and the preferences of both workers and tasks. It focused on assignment in a static scenario, assuming that the spatio-temporal information of tasks and workers is known in advance [22], [23]. *However, tasks and workers interact dynamically in practice, and the spatio-temporal data of either cannot be known in advance [4].* The algorithms in [1] cannot be directly applied to the online problem in this paper. Inspired by this, we propose the Bilateral Preference-aware Dynamic Task Assignment (BDTA) problem, which aims to assign tasks online while considering the preferences of both tasks and workers.

Example 1: Fig. 1 illustrates a BDTA example with tasks t_1 to t_7 and three workers w_1 to w_3, arriving in the order $t_6, t_7, t_1, t_2, t_4, w_1, w_2, w_3, t_3, t_5$. Tables I and II present the key attributes of workers (Definition 1) and tasks (Definition 2), respectively. Tables III and IV report the shortest distances from workers' trajectories to task locations, where ∞ indicates a task beyond the worker's service radius. All distances assume a uniform worker speed of 5 units per unit time. Specifically, worker preferences depend on the benefits of completing the task, while task preferences depend on the reputation scores of workers. Overall satisfaction is computed based on both worker and task satisfaction with the final matching.

In reality, most SC platforms allocate tasks to the closest workers to increase the assigned task count while minimizing the travel costs for workers [24]. By this, based on the time of appearance of tasks/workers in Tables I and II, we can obtain a feasible assignment $M_1 = \{(w_1, t_1), (w_2, t_2), (w_2, t_4), (w_3, t_6), (w_1, t_7)\}$ with an overall satisfaction of 0.687 by Definition 11. t_3, t_5 are no. matched because there is no. enough capacity or time to be completed. However, this approach is unfair to tasks as it only considers the preferences of workers, assigning the nearest tasks to them while ignoring the preferences of the tasks.

Generally, workers prefer to complete high-profit tasks, while tasks tend to choose workers with high reputation. Take task t_4 as an example. In M_1, it is assigned to worker w_2, but it is actually more suitable for worker w_1, because t_4 prefers w_1, and w_1 also prefers t_4 the most among the 7 tasks.

For the BDTA problem outlined herein, which introduces the bilateral preferences of workers and tasks, the matching result is $M_2 = \{(w_2, t_2), (w_2, t_3), (w_1, t_4), (w_3, t_6), (w_1, t_7)\}$, with an overall satisfaction of 0.712, as shown by the orange arrows in Fig. 1. Compared to M_1, M_2 is more stable and has a higher satisfaction. Here, at timestamps 5, for the work-task match pairs (w_1, t_4), (w_3, t_6) and (w_1, t_7), tasks are assigned to their most preferred workers, and workers are also matched to their favorite tasks. Task t_4 is the favorite task of both workers w_1 and w_2. Considering the preferences of task t_4, since the

[4]https://flex.amazon.com/

reputation of w_2 is lower than that of w_1, t_4 is matched with w_1, and it is fair and reasonable for w_2 to be matched with the second preferred task t_2. Likewise, we obtain the worker-task pair (w_2, t_3) at timestamp 5.2. For task t_5, only w_3 satisfies the distance constraint. However, w_3 cannot reach the destination on time if it accepts t_5, so t_5 is no. assigned and waits for new workers.

Challenges: It mainly faces four challenges in the BDTA problem. First, handling unevenly distributed tasks in real-time can lead to resource waste and missed opportunities, necessitating efficient batch processing to balance task allocation. Second, assigning tasks based on workers' trajectories, rather than their current locations, increases both complexity and computation time. Evaluating all potential detour points (i.e., trajectory points) to calculate the shortest detour distances between workers and tasks exacerbates this challenge further. Third, bilateral task distributions driven solely by unilateral preferences can result in unstable allocations, diminishing the overall satisfaction of both workers and tasks. Fourth, as formally proved in Section II, the Bilateral Preference-aware Dynamic Task Assignment problem is NP-hard, which makes efficient task allocation even more challenging.

Contributions: Compared to the preliminary version in [1], we make the following new extensions: First, considering that task peaks and waiting times affect user experience, we propose a new hybrid batch framework for the BDTA problem. Second, tasks and workers face time constraints, which can lead to resource waste when strict preferences leave feasible tasks unmatched. Therefore, we propose a dynamic preference update strategy to optimize the Boston mechanism-based greedy matching. Third, stable matching solutions are unattainable in online multi-constraint task distributions, so we design two new algorithms based on the i.e. of generalized deferred acceptance to balance satisfaction and solution quality.

- We formalize a new problem of BDTA in SC, which is based on the routine trajectories of workers and the preferences of both workers and tasks, and prove it to be NP-hard (Section II).
- We propose a hybrid batch processing framework that jointly considers time and task volume to trade off latency and efficiency (Section III).
- We propose the TIB algorithm to dynamically update task preferences based on the Boston mechanism, increasing the number of worker-task pairings for a task assignment result with good satisfaction (Section IV).
- We design the TIDA algorithm as an α-approximate stable matching based on the generalized deferred acceptance i.e., enhancing the stability of assignment results and overall satisfaction (Section V).
- We design the RGDA algorithm, dynamically selecting tasks or workers to initiate the Generalized Deferred-Acceptance algorithm based on the overall task count and the remaining available capacity of all workers, to improve matching efficiency and quality (Section VI).
- Extensive experiments conducted on real and synthetic datasets validate the effectiveness and efficiency of the proposed algorithms.(Section VII).

Besides, Section VIII reviews the related work, and Section IX concludes the paper and outlines a possible future work.

II. Preliminaries

This section defines the BDTA problem and concepts.

Definition 1 (Spatial Worker): A spatial worker $w = \langle w.p, w.t, w.d, w.r, w.s, w.c\rangle$ refers to an individual who voluntarily deviates from the routine trajectory $w.p$ to undertake spatial tasks. $w.t$, $w.d$, and $w.r$ define the planned departure time, deadline, and the service radius around at point on $w.p$, within which a worker can engage in tasks. Additionally, the reputation score $w.s$ and the maximum task capacity $w.c$ of w are also specified.

The routine trajectory $w.p$ is a sequence of points $\{s, o_1, o_2, \ldots, o_j, d\}$, with s as the source and d the target. Each o_i (for $1 \leq i \leq j$) is a detour point, allowing the worker to deviate from the routine and perform task-related activities.

Definition 2 (Spatial Task): A spatial task $t{=}{<}t.l, t.a, t.d, t.R, t.\text{minR}{>}$ is provided to the platform by the task requester. Specifically, $t.l$ represents the spatial coordinates of task t in 2D space, $t.a$ is the appearance time of the task, $t.d$ is the deadline by which workers must arrive at the task location to complete it, $t.R$ denotes the reward generated upon successful completion of t, and $t.\text{minR}$ denotes the minimum required reputation for a worker to be eligible for t. Similar to [25], we presume negligible task processing time.

Definition 3 (Detour Distance): Given a worker w and a task t, the detour distance $\hat{d}(w.p, t){=}d(o_i, t.l){+}d(t.l, o_j)$ represents the distance a worker deviates from their routine trajectory at o_i to reach the task location $t.l$ and then returns to the trajectory at o_j.

To mitigate the computational complexity of calculating the detour distances of workers, we transform both o_i and o_j into the same point o_k, where o_k represents the trajectory point of worker w nearest to the task location. Thus, the detour distance is computed as $\hat{d}(w.p, t){=}\ 2{\times}d(w, t){=}2{\times}\ \min\{d(o_k, t.l), o_k{\in}w.p\}$.

Definition 4 (Work Available Detour Distance): Given a worker w with assigned tasks $M(w)$, the available detour distance AD_w is calculated as $AD_w = (w.d - ct) \cdot \text{speed} - L_{w.p} - \sum_{t_i \in M(w)} \hat{d}(w.p, t_i)$. Here, ct is the current time, $L_{w.p}$ is the trajectory length of $w.p$, and speed is the average moving speed of worker w.

The term $(w.d - ct) \cdot speed - L_{w.p}$ represents the maximum detour distance worker w can travel before $w.d$. To complete all assigned tasks, worker w must ensure that $AD_w \geq 0$ and can reach the destination earlier than $w.d$.

Definition 5 (Task Available Detour Distance): Given a task t and a worker w, the maximum available detour distance for task t, denoted as AD_t, can be calculated as $AD_t = (t.d - ct) \cdot \text{speed} - L_{wt} - \sum_{t_k \in M(w_j) | o_{t_k} \prec o_j} \hat{d}(w, t_k)$, where ct denotes the current time. L_{wt} represents the trajectory length from the source point $w.p.s$ of worker w to the detour point o_j, the closest to task t, and is given by $L_{wt} = d(w.p.s, o_j)$. Here, $\{t_k \in M(w_j) \mid o_{t_k} \prec o_j\}$ restricts t_k to tasks in $M(w_j)$ whose detour points o_{t_k} precede o_j along the $w.p$.

The first component denotes the detour distance before worker w_j completes task t_i, while the remaining components indicate the distance between the source of w_j and t_j. If $AD_t < 0$, task t cannot be completed on time by w_j.

Example 2. Assume that at timestamp 5.2, worker w_1 initially accepts task t_4 before task t_1. After accepting t_4, the available detour distance for w_1 is calculated as $AD_{w_1} = (25 - 5.2) \times 5 - 80 - 2.8 \times 2 = 13.4$, and for t_4, it holds that $AD_{t_4} = (35 - 5.2) \times 5 - 25 - 2.8 = 121.2$.

Definition 6 (Worker Preference List). The preference list of a worker w is denoted as $PW(w) = \{t_1, \ldots, t_m\}$, where each task $t \in PW(w)$ satisfies the following conditions:

i) *Preference constraint:* Worker preference value $P_t(w){=}t.R{-}\hat{d}(w.p,t){\cdot}c > 0$, where c represents the cost per unit distance traveled, and $P_{t_i}(w) > P_{t_j}(w)$ for $i < j$.

ii) *Radius constraint:* $\hat{d}(w.p,t){\le}w.r$.

iii) *Time constraint:* $(w.d - ct)$ $\cdot$speed $-L_{w.p}$ $-\hat{d}(w.p,t) > 0$ and $(t.d - ct)$ $\cdot$speed $-L_{wt} - \hat{d}(w.p,t) > 0$, where L_{wt} is the trajectory length of w from origin to detour points of t.

Obviously, Cond. ***(i)*** means workers that are assigned tasks with positive earnings; Cond. ***(ii)*** ensures that workers accept tasks within their radius; and Cond. ***(iii)*** stipulates that task completion requires adequate time for both worker and task.

Definition 7 (Task Preference List). The preference list of each task t is denoted as $PT(t) = \{w_1, \ldots, w_m\}$, where the workers $w \in PT(t)$ satisfy the following conditions:

i) *Preference constraint:* Task preference value $P_w(t) = w.s$, where $w.s > t.minR$ and $P_{w_i}(t) > P_{w_j}(t)$ for $i < j$.

ii) *Radius and Time constraints:* Each worker satisfies the same radius and time constraints as in Definition 6.

Example 3. Assume that at timestamp 5.2, the current sets of tasks and workers, as shown in Fig. 1, are $\{t_6, t_7, t_1, t_2, t_4, t_3\}$ and $\{w_1, w_2, w_3\}$, respectively. According to Definition 6, for worker w_3, tasks t_6 and t_7 satisfy both the radius and time constraints. The corresponding preference values are computed as $P_{t_6}(w_3) = 8 - 2.5 \times 2 = 3$ and $P_{t_7}(w_3) = 7.6 - 2.4 \times 2 = 2.8$, resulting in the preference list $PW(w_3) = \{t_6, t_7\}$. Similarly, we obtain the following task preference lists: $PT(t_1) = \{w_1\}$, $PT(t_2) = \{w_2\}$, $PT(t_4) = \{w_1, w_2\}$, $PT(t_6) = \{w_3\}$, and $PT(t_7) = \{w_3, w_1, w_2\}$.

For task t_3, based on Definition 7, workers w_1 and w_2 satisfy the required preference constraints, with $P_{w_1}(t_3) = 7$ and $P_{w_2}(t_3) = 6.6$, yielding $PT(t_3) = \{w_1, w_2\}$. Similarly, the preference lists of other workers are $PW(w_1) = \{t_4, t_7, t_1, t_3\}$ and $PW(w_2) = \{t_4, t_2, t_7, t_3\}$.

If we consider only the mutual preferences of tasks and workers, task t_6 will be assigned to worker w_3, as both $PT(t_6)$ and $PW(w_3)$ indicate a mutual match.

Definition 8 (Blocking Pair). Given a worker w, a task t, and a matching set M, $(w,t) \notin M$ is a blocking pair if $\exists t' \in M(w)$ that meets: **(** *i* **)** $P_w(t) > P_{M(t)}(t)$ and $P_t(w) > P_{t'}(w)$. **(** *ii* **)** For the new task set $M'(w)$, it holds that $AD_{t''} \geq 0$ for each task $t'' \in M'(w)$, $AD_w \geq 0$, and $w.\bar{c} \leq w.c$.

Here, $M(w)$ is the set of tasks assigned to w, $M(t)$ is the worker assigned to t, $M'(w)$ is derived from $M(w)$ by either adding t or replacing a task $t' \in M(w)$ with t, and $w.\bar{c} = |M'(w)|$. Additionally, if $M(w) = \varnothing$, then $P_{M(w)}(w) = 0$; similarly, if $M(t) = \varnothing$, then $P_{M(t)}(t) = 0$.

Definition 9 (Stable Matching). A matching set M is stable if it has no.blocking pairs.

Example 4. Going back to Example 1, we have $M_1(t_4) = \{w_2\}$ and $M_1(w_1) = \{t_1, t_7\}$ for the feasible assignment M_1. However, worker w_1 prefers task t_4 to t_1 ($P_{t_4}(w_1) > P_{t_1}(w_1)$), and task t_4 also prefers worker w_1 to w_2 ($P_{w_1}(t_4) > P_{w_2}(t_4)$). If task t_4 is directly added to $M_1(w_1)$, the capacity constraint in (2) would be violated, i.e., $|M_1(w_1)| = 3 > w_1.c = 2$. If task t_4 replaces task t_1 for worker w_1, we have $AD_{w_1}{=}(25{-}5){\times}5{-}80{-}1.7{\times}2{+}2.8{\times}2{=}22.2{>}0$, $AD_{t_7}{=}(25{-}5){\times}5{-}30{-}2.35{\times}2{-}1.7{=}63.6{>}0$, and $AD_{t_4}{=}122.65{>}0$. Therefore, (w_1, t_4) forms a blocking Pair. In contrast, for M_2, there are no.blocking pairs that prefer each other but cannot be matched, so a stable matching is formed.

Definition 10 (Overall Satisfaction). Given a dynamically arriving task set T and a worker set W, let M denote the set of feasible worker-task pairs for all $t \in T$ and $w \in W$ in the BDTA problem. The overall satisfaction of M is a weighted combination of the average satisfaction of tasks and workers, defined as:

$$\check{S}_M{=}\mu{\cdot}\sum_{i=1}^{|T|} S_t(t_i){\cdot}\frac{1}{|T|}{+}(1{-}\mu){\cdot}\sum_{j=1}^{|W|} S_w(w_j){\cdot}\frac{1}{|W|}, \tag{1}$$

where $|T|$ and $|W|$ refer to the task count and worker count, respectively. The satisfaction of task t_i is given by $S_t(t_i){=}\frac{P_{w'}(t_i)}{P_{w^*}(t_i)}$, where w' denotes the worker assigned to task t_i, and w^* denotes the most preferred worker of t_i in its preference list. For unmatched tasks, $S_t(t_i) = 0$. For a worker w_j, $S_w(w_j) = 0$ if w_j is unmatched; otherwise,

$$S_w(w_j){=}\frac{1}{|M(w_j)|} \cdot \sum_{t' \in M(w_j)} \frac{P_{t'_i}(w_j)}{P_{t^*_i}(w')},$$

where $M(w_j)$ denotes the set of tasks assigned to worker w_j, and t^* is the most preferred task of w_j in its preference list.

In Definition 10, the parameter μ denotes the trade-off between worker and task satisfaction, and is set to $\mu = 0.5$ in this paper to indicate equal priority for both.

Definition 11 (BDTA problem). Given a dynamically arriving worker set $W = \{w_1, w_2, \ldots, w_{|W|}\}$ and a task set $T = \{t_1, t_2, \ldots, t_{|T|}\}$, the goal of BDTA is to identify a worker-task matching set M maximizing the overall satisfaction $\check{S}_M$. Formally, it is described as:

$$\max \check{S}_M \ \textbf{s.t.} \begin{cases} AD_{w_j} > 0,\ AD_{t_i} > 0 \\ \hat{d}(w_j.p, t_i){\le}w_j.r \\ w_j.s \geq t_i.minR \\ P_{t_i}(w_j) > 0,\ P_{w_j}(t_i) > 0 \\ \sum_{i=1}^{|T|} x_{ji}{\le}w_j.c,\ \sum_{j=1}^{|W|} x_{ji}{\le}1. \end{cases} \tag{2}$$

Algorithm 1: Hybrid Batch Processing Framework.

```
Input: A time interval threshold $b.t$ and a task count threshold $b.n$
Output: A worker-task matching set $M_b$ in each batch
1 Initialize $b.W \leftarrow \varnothing; b.T \leftarrow \varnothing; M_b \leftarrow \varnothing$;
2 while current timestamp is within $b.t$ and the new task count is less than $b.n$ do
3     Compute the unassigned task set $b.T$;
4     Compute the available worker set $b.W$;
5     Compute the task-worker matching set $M_b$ for tasks $t \in b.T$ and workers $w \in b.W$ using the methods outlined in Sections IV, V or VI;
6     foreach worker-task pair $(w_i, t_j) \in M_b$ do
7         Remove $t_j$ from $b.T$;
8         Update $w_i$;
```

Here, $w_j \in W, t_i \in T, x_{ji} \in \{0,1\}, j \in [1,|W|]$, and $i \in [1,|T|]$. AD is the available detour distance for the matched task/worker. $\hat{d}(w_j.p, t_i)$ is the minimum distance between w_j and t_i, $P_{t_i}(w_j)$ and $P_{w_j}(t_i)$ is the preferences of worker w_j and task t_i for each other, respectively. The binary variable x_{ji} equals 1 if task t_i is allocated to worker w_j, and 0 otherwise.

Theorem 1. BDTA is an NP-hard problem.

Remark: The reputation scores of workers can be obtained based on their history of task completion [26], with their routine trajectories derived from past routing data. From the view of task requester, a stronger reputation score signifies better service quality. Following [27], we treat workers' reputation scores as task preferences. Additionally, as discussed in [25], both the worker and task requester supply other relevant information, including voluntarily submitted historical records or other data.

III. Hybrid Batch Processing Framework

In BDTA, tasks and workers arrive in real time. This makes it challenging to find a globally optimal solution. Thus, we employ a batch processing technique to enhance computational efficiency and gain a local optimal solution.

In this subsection, we develop a hybrid batch strategy that considers the uneven temporal and spatial distribution of workers and tasks, focusing on the task count and the time window. Note that the batch processing window is generally determined by the tasks count, rather than the worker count, as SC platforms aim to maximize revenue by completing as many tasks as possible.

As shown in Algorithm 1, Line 1 initializes the available worker set $b.W$, the available task set $b.T$, and the worker-task matching set M_b. The hybrid batch processing framework partitions tasks and workers into batches, each with at most $b.n$ tasks and arriving within the time interval $b.t$ (Line 2). In Lines 3-4, the task set $b.T$ and worker set $b.W$ are computed from new tasks and workers, along with unmatched ones from the previous round that have remaining capacity and detour distance, with end times later than the current timestamp. After that, the algorithms described in Sections IV, V, and VI are applied to compute the worker-task matching set M_b for tasks in the current batch (Line 5). Finally, the tasks in M_b are removed from $b.T$, and the information of each worker in $b.W$ is updated accordingly (Lines 6-8).

IV. Task-Initiated Bidirectional Algorithm

In this section, *task-initiated bidirectional* (TIB) algorithm is presented to solve BDTA, by the TPPG strategy from [1]. The algorithm prioritizes task preferences and mitigates the impact of task order on the matching results through the following strategies.

Bidirectional Selection Strategy: Most existing research on online task allocation processes tasks in the order of their arrival time. As a result, early arriving tasks are easier to assign to their preferred workers, while later arriving tasks often struggle to be assigned to workers who meet their preferences. This creates an unfair problem for tasks arriving later. To address this issue, TIB integrates a bidirectional selection strategy by incorporating the Boston Mechanism within the hybrid batch processing framework.

Initially, tasks send requests to their most preferred workers. Workers then rank the received tasks and sequentially accept the ones they prefer most, according to their priority. Tasks that remain unmatched in a round are re-offered to workers based on their next preferences in the hierarchy.

In the Boston mechanism, matched workers face reduced availability for new tasks due to time and capacity constraints, leaving some tasks unassigned to preferred workers in later rounds [28]. To mitigate this, we dynamically update the preferences of tasks based on workers' current matches.

Algorithm: As shown in Algorithm 2, it first initializes an available worker set AW (workers with remaining capacity and positive detour distance), an available task set AT (Unassigned tasks with unvisited workers in their PT), a worker-task matching set M_b, and a candidate matching set M_C (Line 1). In lines 2-3, it computes the initial preference lists $PT(t)$, $PW(w)$ for all tasks $t \in b.T$ and workers $w \in b.W$, and sets the dynamic preference list $PT'(t)$ to $PT(t)$. In lines 5-10, each $t \in AT$ requests its most preferred available and unrequested worker $w \in AW$ in its $PT'(t)$. If the (w,t) formed under M_b satisfies the constraints of (2), (w,t) is added to M_C (lines 7-8); Otherwise, t is removed from AT (line 9). In lines 11-18, for each worker $w \in M_C$, tasks $t' \in M_C(w)$ are added to M_b in descending order of $PW'(w)$, provided they satisfy the constraint in (2). Once a worker can no.longer take new tasks, they are removed from AW. In lines 19-20, Update $PT'(t)$ of the unassigned task $t \in AT$ using the *PreferenceUpdate* procedure. Finally, when there are no.more workers or tasks in AT or AW, the algorithm stops, and M_b is produced as the final matching set (Line 22).

The *PreferenceUpdate function* adjusts preferences based on the urgency of each task and worker (lines 23-31). The task's new preference is determined (line 29) based on the worker's remaining capacity (line 26), available detour distance (line 27), and the proportion of rejected tasks t (line 28).

Example 5. Assume a batch window b starts at time 0 with a 5.2-unit interval, as shown in Fig. 1. Under this setting, entities $\{t_6, t_7, t_1, t_2, t_4, w_1, w_2, w_3, t_3\}$ arrive in order within b. By TIB, we first compute the initial PT and PW, with the results shown in Example 3. Then, tasks t_6, t_7, t_1, t_2, t_4, and t_3 simultaneously send requests to their most preferred

Algorithm 2: TIB Algorithm.

```
Input: Task set $b.T$, worker set $b.W$
Output: Worker-task matching set $M_b$
Initialize $AT \leftarrow b.T$; $AW \leftarrow b.W$; $M_b, M_C \leftarrow \varnothing$; $j = 1$;
Compute initial preference lists $PT(t)$ and $PW(w)$ for all $t \in b.T$ and $w \in b.W$;
$PT'(t) \leftarrow PT(t)$;
while $AT \neq \varnothing$ and $AW \neq \varnothing$ do
    foreach $t \in AT$ do
        $w \leftarrow$the best worker for $t$ from $PT^{'}(t) \cap AW$;
        if $(w, t) \cup M_b$ satisfying the constraints of Eq. (2) then
            Add $(w, t)$ to $M_C$;
        else
            Remove $t$ from $AT$;
    foreach $w \in M_C$ do
        while $w.\bar{c} \leq w.c$ do
            Add $(w, t') \in M_C$ with the maximum $P_{t'}(w)$ satisfying Eq. (2) to $M_b$ ;
            $w.\bar{c} = w.\bar{c} + 1$;
            Remove $t$ from $AT$ ;
            Update $AD$ of each $t \in M_b(w)$ and $w$ ;
        if $w.\bar{c}$ is equal to $w.c$ then
            Remove $w$ from $AW$;
    foreach $t \in AT$ do
        $PT^{'}(t) \leftarrow$ PreferenceUpdate();
return $M_b$.
Function PreferenceUpdate()
    Initialize $PT_t^* \leftarrow \varnothing$;
    foreach $w \in AW \cap PT'(t)$ do
        $f_1 \leftarrow 1 - \frac{|M_w|}{c}$;
        $f_2 \leftarrow 1 - \frac{AD_w}{(w.d - w.t) \cdot speed}$ ;
        $f_3 \leftarrow 1 - \frac{j}{|PT(t)|+1}$;
        $P_w(t) \leftarrow PT_w(t) \cdot f_1 \cdot f_2 \cdot f_3$;
        Add $P_w(t)$ to a descending sorted list $PT_t^*$;
    return $PT_t^*$
```

workers w_3, w_3, w_1, w_2, w_1, and w_1, respectively. Worker w_3 first accepts t_6, then rejects t_7 due to capacity, preference, and detour constraints: $w_3.c = 2 \geq |\{t_3, t_7\}|$, $P_{t_6}(w_3) > P_{t_7}(w_3)$, and $AD_{w_3} - \hat{d}(w_3.p, t_6) > 0$, while the detour constraint for t_7 is violated, i.e., $AD_{w_3} - \hat{d}(w_3.p, t_6) - \hat{d}(w_3.p, t_7) < 0$. Similarly, w_1 rejects t_3 due to insufficient capacity after accepting tasks t_1 and t_4. The pair (w_2, t_2) is directly matched. Next phase, unassigned tasks $\{t_7, t_3\}$ update their preferences via the *preferenceUpda()* function (e.g., new $PT_{w_2}(t_7) = 1.4454$, $PT_{w_2}(t_3) = 0.9636$) and request their most preferred worker w_2. Worker w_2 accepts the more preferred task t_7, as it cannot simultaneously accept t_3 due to capacity constraints: $w_2.c - |\{t_2, t_7, t_3\}| = -1 < 0$. Now, t_3 has no.other available workers to request. Finally, we obtain a matching result $M_3 = \{(w_1, t_1), (w_2, t_2), (w_1, t_4), (w_3, t_6), (w_2, t_7)\}$ with an overall satisfaction of 0.699.

Time Complexity: For each task t, generating its preference list requires $O(|W| \cdot |P'|) + O(|W| \cdot \log |W|)$. During matching, identifying the best available worker per takes $O(|W|)$. Additionally, each worker requires $O(|T| \cdot \log |T|) + O(|T|)$ to sort and select a preferred task. Thus, the overall time complexity of TIB is $O(|W| \cdot |T| \cdot (\log |T| + \log |W| + |P'|))$, where $|P'|$ denotes the maximum trajectory points count per task.

Theorem 2. TIB cannot generate a stable matching.

The proof of Theorem 2 is similar to the proof of Theorem 2 in our previous work [1], and we omit the detailed proof here to avoid repetition.

Remark: Similar worker-initiated bidirectional strategies can also be applied to task allocation. Note that tasks are no. allocated to workers simultaneously to match their capacity; instead, each worker accepts only one task at a time.

V. Approximate Deferred-Acceptance Algorithms

As shown in Theorem 2, TIB does no. guarantee a stable matching. Although TSDA (ICDE version) ensures strict stability in static settings by eliminating all blocking pairs, such strict stability is impractical in BDTA scenarios where tasks, worker availability, and temporal constraints dynamically evolve. To address this, we propose an α-approximate solution via a generalized deferred-acceptance (GDA) approach, relaxing strict stability to accommodate real-world constraints like budget limits and supply-demand imbalance, thus balancing stability with practical feasibility.

A. α-Approximate Model

To enhance the stability of matching results, we aim to introduce a stable matching strategy. However, due to the complex constraints in dynamic task distribution problems, strict stability conditions cannot always be satisfied. Thus, this section introduces the approximate stability to improve the quality of task assignment.

Definition 12 (α-Blocking Coalition [29]). Given a current matching set M, and another matching set M', a blocking coalition $M'(w)$ for a worker w exists if:

- M' satisfies the conditions in (2).
- $P_{w'}(t) > P_w(t)$ for any worker-task pair $(w', t) \in M' \setminus M$ and $(w, t) \in M$.
- $u(M'(w)) > \alpha \cdot u(M(w))$.

Here, $u(M(w))$ (or $u(M'(w))$) denotes the utility of worker w in M (or M') and is given by $u(M(w)) = \sum_{t_i \in M(w)} P_{t_i}(w)$.

Definition 13 (α-Stable Matching). A matching M is an α-stable matching if no α-blocking coalitions exist in M.

Example 6. Going back to Example 1, it holds $M_1(t_4)=\{w_2\}$ and $M_1(w_1)=\{t_1, t_7\}$ for the feasible assignment M_1. However, worker w_1 prefers t_4 to t_1, and t_4 also prefers worker w_1 to worker w_2. If task t_4 is directly added to $M_1(w_1)$, it cannot satisfy the conditions in Definition 12, i.e., capacity condition. If task t_4 replaces task t_1 for worker w_1, we have $AD_{w_1}=(25-5)\times5-80-1.7\times2+2.8\times2=22.2>0$, $AD_{t_7}=(25-5)\times5-30-2.35\times2-1.7=63.6>0$, and $AD_{t_4}=122.65>0$. Therefore, (w_1, t_4) forms a worker-task pair in a 1-blocking coalition, meaning that t_4 can replace t_1 in $M_1(w_1)$.

Definition 14 (α-Approximation). Given $\alpha \geq 1$ and a worker $w \in W$, a sequential choice function Ch_w on $PW(w)$ is called α-approximate if

$$\alpha \cdot u(M(w)) = \alpha \cdot u(Ch_w(T')) \geq \max\{u(\mathcal{M}(w)) : \text{Satisfies Eq. (2) constraints}\},$$

where $\mathcal{M}(w)$ is the matching maximizing the utility of w.

B. Task-Initiated Generalized Deferred-Acceptance Algorithm

In this subsection, we design a *task-initiated generalized deferred-acceptance* (TIDA) algorithm and show that TIDA guarantees the α-approximation ratio.

To design a mechanism that ensures approximately stable matchings and facilitates efficient allocations under budget constraints, we modify the *Generalized Deferred Acceptance* (GDA) algorithm proposed by Hatfield and Milgrom [30]. By ensuring the properties of substitutability, irrelevance of rejected requests, and the law of aggregate demand, our mechanism guarantees the stability of matchings within the hybrid batch processing framework of the BDTA problem.

In the GDA mechanism, it first computes preferred workers for each task sequentially, then workers temporarily accept tasks according to their preferences. For a worker w, if he/she has received a set of tasks, and a new task t' arrives that exceeds the constraints of the worker but is more preferred, t' can replace an existing task t, one of w's accepted tasks that is less preferred than t'. In this case, worker w will defer receiving t' until evaluating whether replacing t is beneficial.

To effectively improve worker utility in GDA, we integrate a *Local Search* (LG) algorithm inspired by [31], as the per-worker assignment aligns with the setting considered therein.

Algorithm: As depicted in Algorithm 3, TIDA first initializes an available task set AT and a worker-task matching set M_b (Line 1). The preference list $PT(t)$ and $PW(w)$ for each task t and each worker is computed according to Definitions 7 and 6 (Line 2). The rest of TIDA is executed iteratively until the available task set AT is empty (Lines 3-25). Before each iteration, nAT is initialized to AT for the next round of matching (Line 4), and tasks $t \in AT$ are processed in order (Lines 5-24). Each task $t \in AT$ requests its most preferred unrequested worker w according to its $PT(t)$ (Line 6). we first check if t has been rejected by all workers in $PT(t)$ (Line 7). If so, task t will wait for the next window to match (Lines 22-24). Otherwise, if w directly accepts t, satisfying the constraint in (2) (line 8), a matching (w, t) is formed. The detour distances of worker w, task t', the tasks in $M_b(w)$, and $w.c$ are updated based on Definitions 4 and 5. t is removed from nAT (lines 9–13). If no., we use *LocalSearch* function to find a replacement task $t' \in M_b(w)$ satisfying Definition 14 (Line 15). If such a task t' exists (Line 16), we replace the pair (w, t) with (w, t') (Lines 17-18) and update the information of affected tasks and w (Lines 19-20). Now, task t does no. request a match in the next round, while task t' will submit a fresh assignment request (Line 21). Once all tasks in AT are matched or no.available worker is left, the final matching set M_b is produced (Line 25).

The LocalSearch Function determines whether replacing task t' in $M_b(w)$ with a new task t increases the utility. Specifically, the task t' that satisfies the constraints in (2) and Definition 14, and yields the highest marginal reward Δ, is replaced by t and returned (Lines 27–37).

Example 7. On the assumption in Example 5, using TIDA, We first compute the preference lists for tasks and workers as shown in Example 3. In the first round of assignment, following the task arrival order, task t_6 requests its most preferred worker w_3 from $PT(t_6)$. Since $PW(w_3) = \{t_6, t_7\}$ and w_3, t_6 satisfy the constraint in (2), a temporary matching $M_t = \{(w_3, t_6)\}$ is formed, updating the information of $M_t(w_3)$ and w_3. Next, t_7 requests its most preferred $w_3 \in PT(t_7) = \{w_3, w_1, w_2\}$, but is rejected because (i) w_3 cannot reach the destination on time if both t_6 and t_7 are assigned, i.e., $AD_{w_3} - d(w_3, t_6) - d(w_3, t_7) < 0$. (ii) There is no.replaceable task for t_7, as marginal reward $\Delta(t_6, t_7, M_t(w_3)) = P_{t_7}(w_3) - P_{t_6}(w_3) = -0.2 < 0$. Similarly, tasks t_1, t_2, and t_4 successively request w_1, w_2, and w_1, respectively, updating $M_t = \{(w_1, t_1), (w_2, t_2), (w_1, t_4), (w_3, t_6)\}$. when t_3 requests w_1, it is rejected because (i) $w_1.c = 2 = |M_t(w_1)| = |\{t_4, t_1\}|$ is full. (ii) $\Delta(t_3, t_4, M_t(w_1)) = -0.4, \Delta(t_3, t_1, M_t(w_1)) = -2.3$. Second round, tasks t_7 and t_3 in $AT = \{t_7, t_3\}$ successively request w_1 and w_2 and

Algorithm 3: TIDA Algorithm

```
Input: Task set $b.T$, worker set $b.W$, approximation factor $\alpha$
Output: Worker-task matching set $M_b$
1  Initialize $AT \leftarrow b.T$; $M_b \leftarrow \varnothing$; $nAT \leftarrow \varnothing$;
2  Compute the preference list $PT(t)$ and $PW(w)$ for each
     $t \in b.T$ and $w \in b.W$;
3  while $AT$ is not empty do
4      $nAT \leftarrow AT$;
5      foreach task $t \in AT$ do
6          $w \leftarrow$ highest-ranked unrequested worker of $t$ in
             $PT(t)$;
7          if $w$ exists then
8              if $w$ accepts $t$ satisfying the constraints of
                 Eq. (2) then
9                  $M_b(t) \leftarrow w$;
10                 $M_b(w) \leftarrow \{t\} \cup M_b(w)$;
11                 Update $AD_w$ and $w.c$ (Def. 4);
12                 Update $AD_{t^*}$ for each task
                     $t^* \in M_b(w)$(Def. 5);
13                 $nAT \leftarrow nAT/\{t\}$;
14             else
15                 $t' \leftarrow$ LocalSearch$(w, t, M_b(w), b.T, \alpha)$;
16                 if $t'$ exists then
17                     $M_b(t) \leftarrow w$, $M_b(t') \leftarrow \varnothing$;
18                     $M_b(w) \leftarrow M_b(w) \cup \{t\}/\{t'\}$;
19                     Update $AD_{t'}$ and $AD_w$ (Defs. 4 and 5);
20                     Update $AD_{t^*}$ for each $t^* \in M_b(w)$
                         (Def. 5);
21                     $nAT \leftarrow nAT/\{t\} \cup \{t'\}$;
22         else
23             $nAT \leftarrow nAT/\{t\}$;
24             $t$ enqueue $(b+1).T$;
25     $AT \leftarrow nAT$;
26 return $M_b$;
27 Function LocalSearch$(w, t, S_w, b.T, \alpha)$:
28     Initialize $bestTask \leftarrow \varnothing$, $maxMarginal \leftarrow -\infty$;
29     foreach $t' \in S_w$ do
30         if Replacing $t'$ with $t$ satisfies the constraints for $w$
             then
31             $newUtility \leftarrow u(S_w \setminus \{t'\} \cup \{t\})$;
32             if $newUtility > u(S_w)$ then
33                 $\Delta(t, t', S_w) = newUtility - u(S_w)$;
34                 if $\Delta(t, t', S_w) > maxMarginal$ then
35                     $maxMarginal \leftarrow \Delta(t, t', S_w)$;
36                     $bestTask \leftarrow t'$;
37     return $bestTask$;
```

are matched. Here, w_1 replaces t_1 with t_7 because $w_1.c = 2$, $\Delta(t_7, t_1, M_t(w_1)) = 4.2 - 2.8 = 1.4 > \Delta(t_7, t_4, M_t(w_1)) = 4.2 - 4.3 = -1.4$, and the constraints in (2) are satisfied. $M_t = \{(w_2, t_2), (w_2, t_3), (w_1, t_4), (w_3, t_6), (w_1, t_7)\}$.
In the final round, $AT = \{t_1\}$ and $PT(t_1) = \{w_1\}$. However, t_1 has no.available workers, as w_1 has rejected it. Thus, the matching ends with $M = M_t$ and overall satisfaction 0.712.

Theorem 3. If the local selection function LS is α-approximate for each worker $w \in W$, then TIDA produces an α-stable matching M.

Theorem 4. Let M be a β-approximate local optimal solution obtained by the LS algorithm, Ω denotes the optimal assignment, and OPT is the optimal solution value. Then, $u(M) \geq \frac{\beta}{\beta+1}$OPT.

For BDTA, the feasible set $M(w)$ of each worker satisfies a k-set constraint, which allows the use of a PTAS for k-set packing and yields a β-approximation [32]. Consequently, the stability guarantee of TIDA is given by $\alpha = \frac{1+\beta}{\beta}$.

Time Complexity: TIDA has two phases: initialization and matching. In the initialization phase, computing and sorting each task's preference list takes $O(|W| \cdot |P'|) + O(|W| \log |W|)$. In the matching phase, each task requires $O(|W|)$ to find an available worker and $O(C)$ for workers to accept or replace tasks. The overall time complexity of TIDA is $O(|W| \cdot |T| \cdot (|P'| + \log |W| + C))$, where C represents the worker capacity, and $|P'|$ represents the maximum trajectory point count per trajectory.

Remark: The *Worker-initiated Generalized Deferred-Acceptance* (WIDA) Algorithm is initiated by the worker first, and the task performs matching operations such as acceptance, replacement, or rejection.

VI. Reverse Generalized Deferred-Acceptance Algorithm

The RGDA and WIDA algorithms are both unilateral-initiated approaches. However, there is an inherent trade-off between task and worker satisfaction, making it challenging to achieve optimal satisfaction for both simultaneously. Additionally, in these two algorithms, the side with the larger number of participants has more opportunities to choose, thereby enhancing matching flexibility [33]. Therefore, the *reverse generalized deferred-acceptance* (RGDA) algorithm takes into account the available task count and the remaining available worker capacity in each batch. It dynamically selects either TIDA or WIDA, aiming to balance worker and task satisfaction while enhancing overall matching performance.

Algorithm: As depicted in Algorithm 4, RGDA first initializes an available task set AT, an available worker set AW, an unassigned task set UT, and two sets of worker-task matching set M_b and $\hat{M}_b$ (Line 1). The preference lists $PT(t)$ and $PW(w)$ for each task t and each worker w are computed from Definitions 7 and 6 (Line 2). The rest of RGDA involves selecting between the WIDA and TIDA algorithms. If the available task count $|AT|$ exceeds the total available workers' capacity, $|AW| \cdot w.c$ (Line 3), TIDA is applied to compute the worker-task matching set M_b (Line 4). Next, the available worker set AW and the unassigned task set UT are updated (Lines 5-6). If AW and UT are no. empty, WIDA is used to compute another worker-task matching set $\hat{M}_b$ for tasks $t \in UT$ and workers $w \in AW$ (Lines 7-8). Otherwise, WIDA is first applied to obtain M_b, and then TIDA is executed on the unassigned task set UT and the available worker set AW to obtain the worker-task matching set $\hat{M}_b$ (Lines 10–14). Finally, the worker-task matching set M_b is updated to $M_b \cup \hat{M}_b$ and produced as the final matching set (Lines 15-16).

Algorithm 4: RGDA Algorithm.

```
Input: Task set $b.T$, worker set $b.W$
Output: Worker-task matching set $M_b$
1  Initialize $AT \leftarrow b.T$; $AW \leftarrow b.W$; $UT, M_b, \hat{M}_b \leftarrow \varnothing$;
2  Compute a list of preferences $PW$ for each $w \in b.W$ and
     $PT$ for each $t \in b.T$;
3  if $|AT| \geq |AW| \cdot w.c$ then
4      Compute $M_b$ by TIDA($b.W, b.T$);
5      Calculate available worker set $AW$ with unfilled
         capacity from $b.W$ ;
6      Calculate unassigned task set $UT$ for $w$ from $b.T$ ;
7      if $AW \neq \varnothing$ and $UT \neq \varnothing$ then
8          Compute $\hat{M}_b$ by WIDA($AW, UT$);
9  else
10     Compute $M_b$ by WIDA($b.W, b.T$);
11     Calculate available worker set $AW$ with unfilled
         capacity from $b.W$;
12     Calculate unassigned task set $UT$ for $w$ from $b.T$ ;
13     if $AW \neq \varnothing$ and $UT \neq \varnothing$ then
14         Compute $\hat{M}_b$ by TIDA($AW, UT$);
15 $M_b \leftarrow M_b \cup \hat{M}_b$;
16 return $M_b$.
```

Example 8. Following Example 5, with the batch window set to [0-5.2] and each worker $w \in AW$ having capacity $w.c$ =2, we have the available task set $AT=\{t_6, t_7, t_1, t_2, t_4, t_3\}$ and available worker set $AW=\{w_1, w_2, w_3\}$ within the batch window. RGDA first computes $|AT|$=6 and $|AW| \cdot w.c$=3×2=6. Since $|AT| \geq |AW| \cdot w.c$, RGDA is applied and returns the assignment $M=\{(w_2, t_2), (w_2, t_3), (w_1, t_4), (w_3, t_6), (w_1, t_7)\}$. Then, WIDA is executed for the unassigned task set $\{t_1\}$ and available worker set $\{w_3\}$, whose capacity is no. yet full. However, w_3 cannot be assigned to t_1 because t_1 is out of his/her service radius, i.e., $d(w_3, t_1)$=9.99>3. At last, M is produced as the final matching set.

Time complexity: RGDA invokes both the TIDA and WIDA algorithms to compute the worker-task matching set M_b. Accordingly, the total time complexity of RGDA is $O(|W| \cdot |T| \cdot (|P'| + \log |T| + \log |W| + C))$, where C represents the worker capacity, and $|P'|$ represents the maximum trajectory point count per trajectory.

VII. Experiments

This section outlines the experimental setup from Section VII-A, evaluates the performance of our algorithms in Section VII-B, and concludes with a summary of the results in Section VII-C.

A. Experiment Setup

Datasets: We validate our algorithms on two datasets, as shown in Table V, namely Berlin [34] and T-Drive [35], [36]. Berlin is a real dataset comprising 236 bus routes representing

TABLE VI
PARAMETER SETTING

Parameter	Setting
$\|T\|$	Berlin: 1000, 1500, **2000**, 2500, 3000 T-Drive: 1000, 5000, **10000**, 15000, 20000
$\|W\|$	Berlin: 100, 250, **500**, 750 T-Drive: 1000, 2000, **3000**, 4000, 5000
$w.r$	Berlin, T-Drive: 500, 1000, **1500**, 2000, 2500
$b.n$	Berlin, T-Drive: 50, **200** , 800, 1600
$b.t$	Berlin, T-Drive: 2, 10, **50**, 100
$w.c$	Berlin, T-Drive: 1, 3, **5**, 7

worker trajectories and 3,083 Points-of-Interest (POIs) representing task locations. When the worker count exceeds 236, trajectories are randomly sampled and repeated to ensure broad coverage. Task locations are directly extracted from the POIs. T-Drive is a synthetic dataset that includes 10,357 taxi trajectories in Beijing, collected from February 2 to February 8, 2008. Given the high density of points, we selected 40 points per trajectory, spaced at every fourth point, to simulate worker paths and generated synthetic task data around these points. Task locations in T-Drive are uniformly distributed around the worker trajectories.

In line with common SC platform experimental setups, all generated attributes adhere to a uniform manner standard, ensuring dataset consistency [37]. To align with our BDTA problem requirements, we apply a Gaussian distribution model on both T-Drive and Berlin to generate attributes such as worker reputation, minimum task reputation, task rewards, and other relevant parameters. Additionally, each worker's deadline $w.d$ is calculated as $w.d = w.t + \lambda \cdot (1 + D_w)$, where $\lambda \in [0.2, 0.8]$ is a random deadline coefficient, and D_w represents the worker's daily trajectory time. We assume uniform values for the service radius $w.r$, deadline coefficient λ, and capacity $w.c$ across all workers.

Algorithms: This study conducts the first investigation of BDTA, and current approaches are no. directly applicable to it. Instead, we propose a new baseline algorithm, called Greedy, which is an extended of our closely related work [19]. Thus, the following four algorithms are evaluated in the experiments.

- Task-initiated bidirectional (TIB) algorithm in Section IV.
- Task-initiated generalized deferred-acceptance (TIDA) algorithm in Section V-B.
- Reverse generalized deferred-acceptance (RGDA) algorithm in Section VI.
- The Greedy algorithm: For all tasks, the nearest worker satisfying the constraints in (2) is selected.

Parameters and Metrics: In the experiments, we perform 20 repetitions for each parameter test and report the average results. All algorithms are assessed by varying parameters like task count $|T|$, worker count $|W|$, worker service radius $w.r$, worker capacity $w.c$, time interval threshold $b.t$, and task count threshold $b.n$ within the hybrid batch processing framework. Note that the reported runtime refers to the total time for processing all batches, no. a single batch.

Setting: Table VI lists the parameter settings, with defaults in bold. All algorithms are evaluated by runtime (s) and satisfaction (%), using GNU C++ on a server with an Intel(R) Xeon(R) Gold 5218R CPU @ 2.10 GHz and 256 GB RAM.

B. Experiment Results

Effect of task count $|T|$: As depicted in Fig. 2(a), the satisfaction of the four algorithms increases with the growth of $|T|$ on the Berlin dataset, whereas a decline is observed on the T-Drive dataset. This divergence stems from the differing characteristics of the two datasets. In T-Drive, when $|T|$ grows to 20,000—far exceeding the 3,000 available workers—the limited matching capacity, combined with spatial and temporal constraints, results in a large number of unmatched tasks and a sharp drop in task satisfaction. Although more tasks offer workers increased choice, the overall matching success rate still decreases. Consequently, overall satisfaction declines due to the dominant influence of task-side dissatisfaction. In contrast, the Berlin dataset features repetitive and spatially dispersed bus routes. As $|T|$ increases, more tasks align with these fixed trajectories, increasing spatial overlap. The lower competition and consistent coverage lead to more stable matchings, improving both task and worker satisfaction.

Algorithm Satisfaction Analysis: Among the four algorithms, RGDA consistently achieves the highest satisfaction in most scenarios, followed by TIDA and TIB, whereas Greedy yields the lowest satisfaction, with the maximum difference reaching up to 7%. This performance disparity stems from the design principles of the algorithms. Greedy follows a one-shot matching strategy that prioritizes local proximity without considering global allocation balance, especially in situations of intense competition or an unbalanced task-to-worker ratio, leading to low satisfaction.In contrast, TIDA and RGDA incorporate iterative and bidirectional mechanisms that allow both tasks and workers to express preferences, enabling mutual selections and more balanced matching. Specifically, RGDA dynamically selects task- or worker-prioritized deferred acceptance based on the current numbers of tasks and workers, ensuring that more preferred and compatible matches are achieved across the system. TIB updates task preferences in each round based on urgency, helping improve task match success and satisfaction. However, since the adjustment only happens on the task side, it is less effective than TIDA or RGDA in finding globally balanced matchings.

Algorithm Runtime Analysis: The execution times of all four algorithms increase with $|T|$, reflecting the same pattern observed in satisfaction trends. Greedy is the fastest due to its single-pass design, while TIB is the slowest because of repeated updates to task-side preferences. TIDA and RGDA involve additional evaluations but maintain reasonable scalability. These differences stem from the underlying preference construction and update mechanisms, indicating a trade-off between computational efficiency and solution quality.

Effect of worker count $|W|$: The experimental results of the four algorithms with varying $|W|$ are shown in Fig. 2(b). As the number of workers increases, overall satisfaction tends to decline, while runtime increases. This is primarily due to

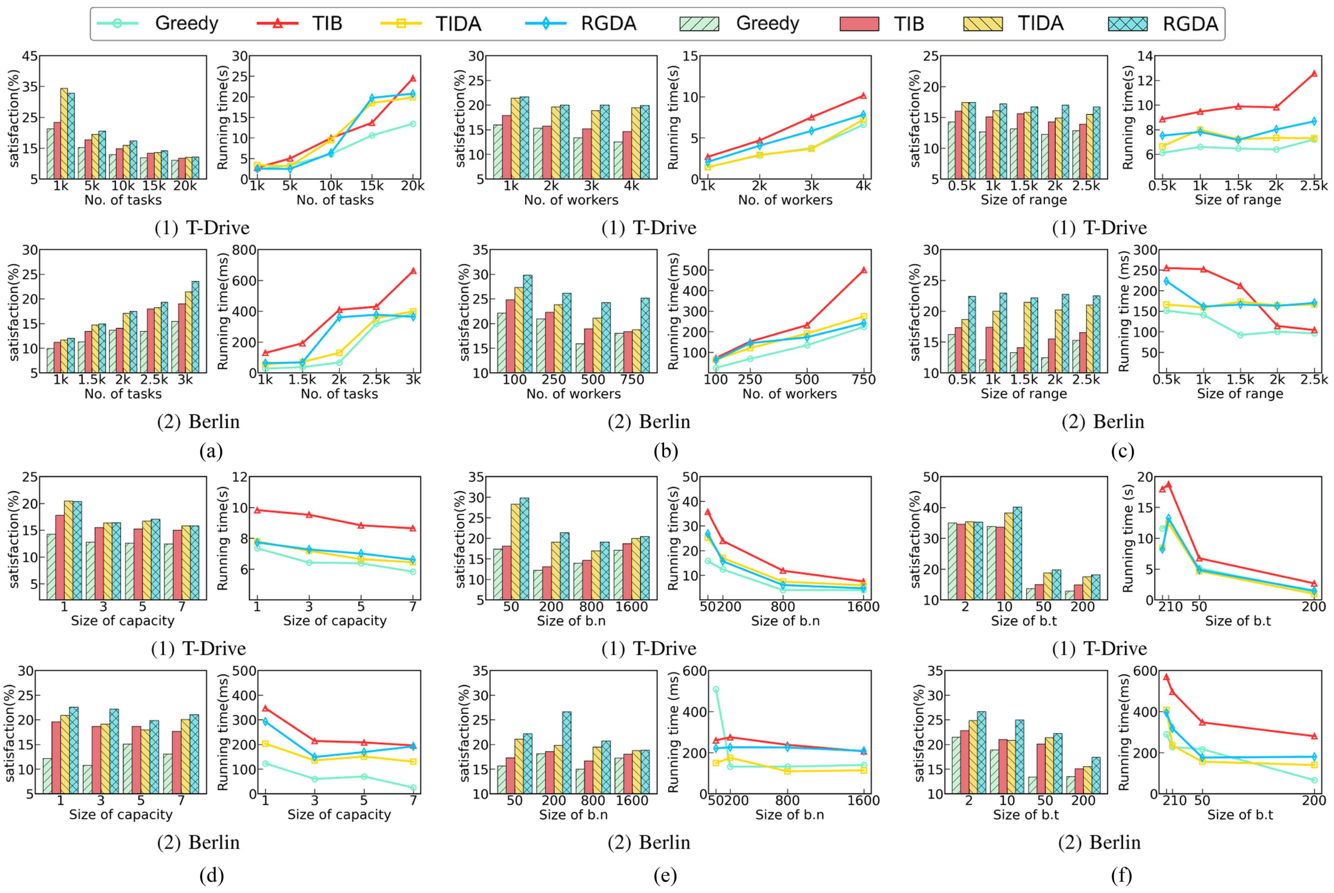
Fig. 2. (a) Effect of task count $|T|$. (b) Effect of worker count $|W|$. (c) Effect of service radius $w.r$. (d) Effect of capacity $w.c$. (e) Effect of task count threshold $b.n$. (f) Effect of time interval threshold $b.t$.

reduced worker utilization and greater computational overhead. Specifically, as $|W|$ grows, although the task has more candidate workers, theoretically increasing matching flexibility, the relatively fixed task count leads to a large number of idle workers, thereby reducing overall satisfaction. As $|W|$ increases, computational overhead grows, particularly for TIB due to frequent preference updates. Greedy is fastest but suffers from lower satisfaction. In contrast, our algorithms achieve a satisfaction improvement of 1% to 7%, reflecting a favorable trade-off between efficiency and solution quality, consistent with trends observed under increasing $|T|$. Notably, even the most time-consuming algorithm, TIB, is only 4 seconds slower than Greedy on large datasets and 0.1 seconds slower on small datasets. In return, our algorithms improve satisfaction by 1% to 7%, demonstrating a balanced trade-off between efficiency and solution quality, consistent with trends observed when increasing $|T|$.

Effect of service radius $w.r$: The experimental results of the four algorithms with varying $w.r$ are shown in Fig. 2(c). As $w.r$ increases, satisfaction and runtime exhibit only minor changes on both T-Drive and Berlin. A larger detour radius expands each worker's candidate task set. However, due to spatiotemporal constraints among tasks and between workers and tasks, the number of feasible matches does no. grow proportionally. Although better compatibility can slightly improve pairwise satisfaction, the limited number of successful matches often flattens or reduces overall satisfaction. Runtime on T-Drive increases with larger $w.r$, due to the density and diversity of worker trajectories. The expansion of candidate task sets intensifies local selection conflicts, raising computational costs. In contrast, Berlin's repetitive and spatially dispersed routes lead to less overlap among workers, so runtime decreases as reachability improves. These results, consistent with the trend under increasing $|T|$, demonstrate that our algorithms sustain high satisfaction and efficiency across varying service radius.

Effect of capacity $w.c$: The experimental results of the four algorithms with varying $w.c$ are depicted in Fig. 2(d). As $w.c$ increases, the overall satisfaction slightly declines on T-Drive, while it remains relatively stable on Berlin. In both datasets, runtime drops significantly. Since the average satisfaction of workers is linked to the ratio of satisfaction with their best task, increasing task-side satisfaction but diluting individual worker satisfaction can lower the overall score. Additionally, higher capacity improves task allocation flexibility, preventing bottlenecks and speeding up task matching. Again, RGDA outperforms TIDA, which in turn outperforms TIDA, with Greedy being the least effective. Even the least efficient TIB algorithm achieves 3% to 8% higher satisfaction compared to the greedy algorithm, while its runtime increases only slightly by 0.2 to 3 seconds. This further demonstrates that our algorithms enhance

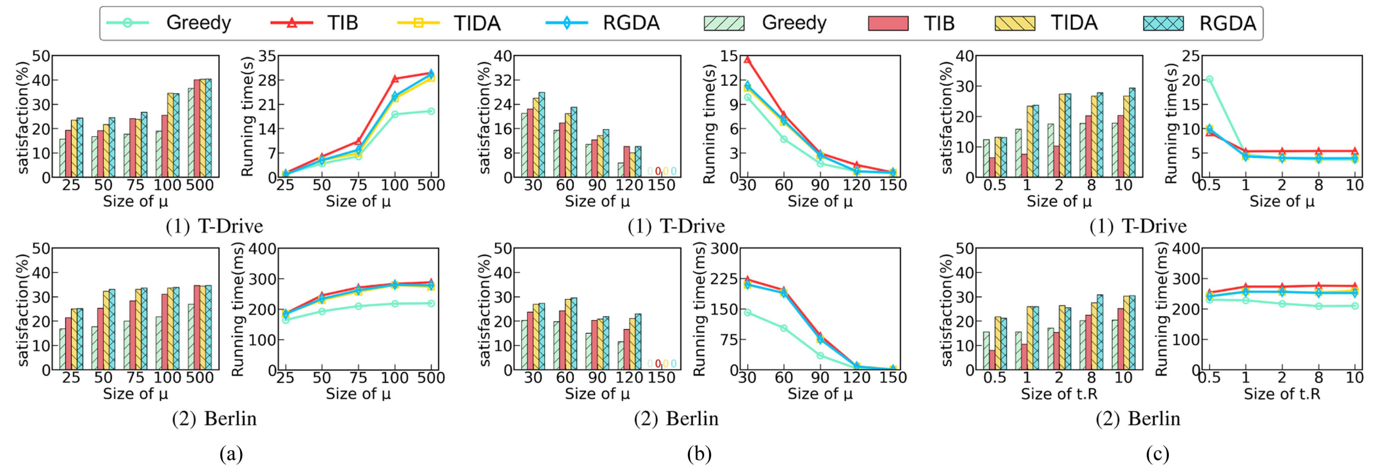

Fig. 3. (a) Effect of task limit $t.d - t.a$. (b) Effect of minimum required reputation. (c) Effect of task reward.

matching satisfaction while maintaining both effectiveness and efficiency.

Effect of task count threshold $b.n$: Fig. 2(e) shows that as $b.n$ changes, the performance of the four algorithms in terms of satisfaction and runtime exhibits slight variations across datasets. On T-Drive, where tasks and workers are sparse and dynamic, smaller $b.n$ (e.g., 50) triggers more frequent matching windows. Under such high-frequency, low-load conditions, the game-theoretic mechanisms in TIDA and RGDA become more effective, yielding the highest satisfaction. However, the increased scheduling frequency leads to higher runtime. As $b.n$ increases, the number of tasks per window theoretically grows, but the fixed time interval constraint $b.t$ limits the actual increase in task arrivals, resulting in only marginal improvements in satisfaction and gradually reduced runtime due to fewer matching rounds. In contrast, the Berlin dataset features denser and more stable spatiotemporal distributions along worker trajectories. Consequently, due to the active time interval constraint, the effectiveness of matching becomes less sensitive to changes in $b.n$, and both satisfaction and runtime remain relatively stable as $b.n$ increases. In summary, the performance trends under $b.n$ are consistent with those observed under $|T|$, while the overall impact still depends on the batch window time threshold $b.t$ and the spatiotemporal characteristics of the dataset.

Effect of time interval threshold $b.t$: Fig. 2(f) shows the experimental results with varying $b.w$. As $b.t$ increases beyond 10, both satisfaction and running time decrease. The runtime reduction slows as the task count threshold $b.n$ begins to dominate batching behavior. Larger time intervals lead to longer waiting times, causing task and worker invalidation, which in turn reduces satisfaction. These invalidations no. only reduce satisfaction but also decrease the computational workload. On T-Drive, TIDA and RGDA achieve similar runtimes to Greedy but with higher satisfaction. TIB shows lower satisfaction and running time than Greedy at smaller time intervals, but outperforms Greedy in larger intervals. On Berlin, runtime differences among all algorithms are within 0.2 seconds, with each achieving higher satisfaction than Greedy. Overall, the impact of $b.t$ on system performance is similar to that of $b.n$. Therefore, balancing these parameters is critical to optimizing the trade-off between running time and satisfaction in our framework and algorithms.

Effect of task limit $(t.d - t.a)$: To demonstrate the impact of different task time limit on the assignment results, we simulate varying task time limits by fixing the appearance time $t.a$ for each task and sample its deadline $t.d$ from a normal distribution, i.e., $t.d \sim \mathcal{N}(\mu, 20^2)$. As shown in Fig. 3(a), both the average satisfaction and the runtime of each algorithm increase with the task deadline $t.d$ on the Berlin and T-Drive datasets. As tasks remain in the system longer, more workers have overlapping availability, increasing the chances of forming mutually satisfactory matches. Extended deadlines also relax detour constraints, allowing workers to complete more tasks and expanding the search space. When $\mu = 500$, the enlarged matching space enables algorithms to better leverage workers' multi-tasking ability, reducing conflicts and priority competition. Consequently, runtimes and satisfaction converge across algorithms. Note that task-worker matching is determined by the overlap of their time windows. Therefore, modifying task time limits has an equivalent effect to adjusting worker availability windows. As a result, we omit separate experiments on worker time limits.

Effect of task minimum required reputation $t.minR \sim \mathcal{N}(\mu, 5^2)$: The number of tasks and workers in each preference list depends on whether a worker's reputation $w.r$ satisfies the task's minimum reputation $t.minR$. We fix $w.r \sim \mathcal{N}(60, 20^2)$ and sample $t.minR \sim \mathcal{N}(\mu, 5^2)$ with varying means μ. As shown in Fig. 3(b), both the satisfaction and runtime of Greedy, TIB, TIDA, and RGDA decrease as $t.minR$ increases. Higher $t.minR$ impose stricter constraints on tasks, thereby pruning more low-reputation workers from the candidate pool and reducing the number of feasible task-worker pairs. In extreme cases (e.g., $t.minR = 150$), no.workers meet the requirement, causing all algorithms to fail in matching tasks, demonstrating the severe impact of stringent preferences on matching feasibility and platform efficiency.

The experiment once again shows that the three algorithms we proposed are always more satisfied than the location-based greedy. Varying $t.minR$ indirectly modifies worker preferences by limiting accessible tasks, reducing preference diversity and matching flexibility. Therefore, further analysis on $w.r$ is omitted, as varying $t.minR$ captures its equivalent effect on compatibility.

Effect of task reward $t.r$: Fig. 3(c) shows how task rewards affect assignment outcomes. We model task completion cost in $(0, 3]$ and generate rewards from a Gaussian distribution between 0.5 and 10 units. As rewards increase, more worker-task pairs become feasible, improving overall satisfaction. Under low-reward settings, Greedy achieves higher satisfaction than TIB. Since workers are less reward-sensitive, Greedy can focus solely on spatial proximity to form fast, conflict-free matches. In contrast, TIB adopts the Boston mechanism. When reward differences are small, task priorities become nearly indistinguishable, rendering the sorting mechanism essentially ineffective. After workers complete early matches, the availability of workers for later unmatched tasks decreases, leading to increased matching delays and failure rates.By comparison, TIDA and RGDA update matching decisions dynamically over rounds, gradually forming stable outcomes and maintaining high satisfaction across all reward levels. Notably, Greedy incurs the highest runtime under low rewards, as many nearby tasks are unprofitable, requiring repeated candidate trials before successful matches.

C. Summary of Experiments

Across both datasets, TIB, TIDA, and RGDA achieve consistently higher satisfaction than Greedy, with comparable runtime performance.

Our RGDA algorithm consistently achieves the highest satisfaction, while Greedy generally yields the lowest. Greedy makes assignments based on proximity, neglecting future utility. TIB employs the Boston mechanism with bilateral preferences but prioritizes immediate decisions. TIDA allows relaxation of the bilateral stability matching condition, improving adaptability. RGDA further enhances satisfaction by dynamically adjusting the proposer. Although TIB has the highest runtime due to frequent preference recalculations in dense environments, its execution time exceeds that of Greedy by only 0.3 seconds on Berlin and 3 seconds on T-Drive, demonstrating good scalability.

Our hybrid batch framework, parameterized by $b.n$ and $b.t$, enables effective control over responsiveness and satisfaction. As shown in Figs. 2(e) and 2(f), appropriate tuning of thresholds yields consistent gains. Dataset properties shape outcomes: on Berlin (sparser, moderate concurrency), TIDA and RGDA perform steadily; on T-Drive (dense, high volume), RGDA achieves up to 15% higher satisfaction over Greedy. Overall, our methods deliver robust performance across varying spatiotemporal scenarios with minimal overhead.

D. Discussion Section

This discussion section presents a comprehensive analysis of BDTA, covering its practical applications, representative use cases, scalability challenges, and directions for future research.

The proposed BDTA solution demonstrates strong potential for real-world deployment and can be integrated into existing platforms such as Gigwalk[5] and Field Agent,[6] where workers are assigned to geo-tagged tasks such as storefront photography, retail display audits, and signage verification.

Furthermore, by explicitly incorporating destination constraints, BDTA is also applicable to ride-hailing services such as Uber[1] and DiDi Chuxing[3], where both pickup and drop-off points are critical. These cases underscore BDTA's flexibility in managing real-time, location-sensitive matching scenarios.

To address scalability in large-scale dynamic settings, we propose a batch-processing framework that groups tasks and workers by time intervals and workload. This design significantly reduces both response time and computational overhead. However, the model assumes fixed travel speeds and does no. yet incorporate environmental dynamics such as real-time traffic or weather disruptions, which may affect task feasibility and routing decisions [38]. These limitations constrain the robustness of BDTA in complex real-world deployments.

Future work may integrate BDTA with real-time traffic data, spatiotemporal prediction models, and deep learning and machine learning algorithms to improve task-worker matching and adapt to dynamic environments [39]. These enhancements could broaden the model's application to smart cities, urban logistics, and drone deployment. In Industry 4.0 settings, BDTA may support real-time robot dispatch based on equipment status and dynamic task priorities. In healthcare, BDTA may assist hospitals in dynamically allocating medical staff to patients based on urgency, proximity, and specialty matching.

VIII. Related Work

This section reviews research on task distribution and other related studies within Spatial Crowdsourcing.

A. Task Assignment in Spatial Crowdsourcing

Crowdsourcing is a collaborative model where tasks are distributed among participants [26]. SC is an extension of crowdsourcing, which requires participants to appear in a specific location [11], [13]. Typical applications include ridesharing like Uber and DiDi, as well as urban sensing projects.

Task allocation is central to SC, ensuring that tasks are efficiently and effectively allocated to workers. Most previous studies have focused on *unilateral preferences* in worker-based/task-based predominant studies. Xie et al. [7] prepackage tasks and assign them by worker preferences and trajectories to boost matches and cut communication overhead. However, task coalitions and budget limits exclude many worker groups, preventing reassignment to higher-preference workers. Sakai et al. [14] prioritized urgent tasks by classifying tasks as urgent or routine, but ignored skill matching and lacked adaptability in complex settings. Additionally, maximizing task allocation count or improving individual utility has also been explored. Zeng et al. [40] proposed offline algorithms with approximation and competitive guarantees to balance matching quantity and

[5]https://www.gigwalk.com/
[6]https://www.fieldagent.net/

delay under limited real-time worker availability. Chen et al. [5] studied online approximation algorithms to minimize maximum latency. Li et al. [41] proposed a gain model for cross-platform task distribution to ensure fairness via multi-party coordination.

B. Task Assignment Based on Bilateral Preference

Early studies focused on unilateral preferences, neglecting mutual satisfaction [42], [43], [44]. Later work introduced bilateral models [19], [45], but progress on crowdsourcing platforms remains preliminary.

Chen et al. [46] designed a convolutional neural network model that captures the mutual expectations of job seekers and job recruitment scenarios, considering bilateral preferences. Hu et al. [47] introduced the Bilateral Occupational-Suitability-aware Recommender System for online recruitment. However, its model focuses more on action sequences in the hiring process than real-time dynamic scenarios. Zhou et al. [1] developed a stable match method based on bilateral preferences to implement task assignment by combining task rewards and worker trajectories, but the problem is an offline scenario. Yang et al. [18] established a dual-perspective selection preference from job seekers and employers in an online recruitment platform to optimize person-job fit.

The stable matching framework proposed by Gale and Shapley has been extended from classical domains to modern crowdsourcing systems [45], especially in applications involving spatiotemporal constraints [4], [16], [42]. In particular, Yucel et al. [48] designed several stable matching algorithms for static scenarios to achieve stable bilateral worker-task matching across different MCS systems, and proved that no.single algorithm can guarantee stable matching in all system settings. Yin et al. [49] assign workers with varying skill levels to mutually preferred tasks in static scenarios, aiming to maximize worker satisfaction. Huang et al. [22] use bipartite graphs and historical data to learn preferences, applying linear programming for stable matching, but the method suits static settings with offline workers only. Yucel et al. [16] focus on a bilateral preference-based approach under uncertain trajectories. Although the algorithm aims for stable matching under uncertain conditions, there are cases where workers do no. visit the scheduled region, resulting in ineffective matching. Dai et al. [50] proposed a many-to-many stable matching method that allows sellers to adaptively adjust their asking prices based on matching results, thereby better matching buyers and sellers. In addition, in three-dimensional stable matching, Li et al. [51] designed an approximate algorithm to ensure the stability of the matching results for tasks, workers, and workplaces.

In summary, although bilateral preference research in SC is enriching, several key gaps remain. First, most methods assume static scenarios with known tasks and workers, making it difficult to adapt to dynamic, time-sensitive scenarios. Second, most approaches focus on worker locations and ignore their work trajectories. Third, probabilistic models for uncertain mobility cannot ensure that assigned workers reach the designated regions, resulting in ineffective or inefficient matches. These gaps motivate the need for dynamic, bilateral preference-aware, and mobility-robust assignment mechanisms.

IX. Conclusion

In this paper, we explore the problem of Bilateral Preference-aware Dynamic Task Assignment (BDTA). We first introduce a hybrid batch processing framework to handle tasks in batches. After that, we present a task-initiated bidirectional algorithm and two generalized deferred acceptance algorithms to assign tasks online, improving overall satisfaction and efficiency. Our extensive experimental results on two datasets validate the effectiveness and efficiency of the BDTA method. Future research may explore more innovative big-data-driven scenarios to enhance the generalizability of BDTA, such as modeling complex worker–task interactions via heterogeneous graphs built from user behavior. Graph neural networks, predictive models, and transfer learning can be employed to support adaptive matching under diverse conditions. Moreover, by leveraging both historical and predicted attributes of workers and tasks, the system may enable pre-allocation, thereby improving scheduling efficiency and personalization. Based on the above optimization techniques, BDTA can be applied to smart cities, industrial automation, and healthcare to improve coordination among heterogeneous entities.

Acknowledgment

This paper is a substantial extension of the conference version “Bilateral Preference-aware Task Assignment in Spatial Crowdsourcing”, which has been published in ICDE 2022.

References

[1] X. Zhou, S. Liang, K. Li, Y. Gao, and K. Li, “Bilateral preference-aware task assignment in spatial crowdsourcing,” in *Proc. IEEE Int. Conf. Data Eng.*, 2022, pp. 1687–1699.
[2] T. Ren, X. Zhou, K. Li, Y. Gao, J. Zhang, and K. Li, “Efficient cross dynamic task assignment in spatial crowdsourcing,” in *Proc. 39th IEEE Int. Conf. Data Eng.*, 2023, pp. 1420–1432.
[3] K. Liu, S. Peng, W. Gong, B. Zhang, and C. Li, “Hybrid user-based task assignment for mobile crowdsensing: Problem and algorithm,” *IEEE Internet of Things J.*, vol. 11, no. 11, pp. 19589–19601, Jun. 2024.
[4] Y. Tong, Y. Zeng, B. Ding, L. Wang, and L. Chen, “Two-sided online micro-task assignment in spatial crowdsourcing,” *IEEE Trans. Knowl. Data Eng.*, vol. 33, no. 5, pp. 2295–2309, May 2021.
[5] 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.
[6] J. Yao, L. Yang, and X. Xu, “Online dependent task assignment in preference aware spatial crowdsourcing,” *IEEE Trans. Services Comput.*, vol. 16, no. 4, pp. 2827–2840, Jul./Aug. 2023.
[7] Y. Xie et al., “Trajectory-aware task coalition assignment in spatial crowdsourcing,” *IEEE Trans. Knowl. Data Eng.*, vol. 36, no. 11, pp. 7201–7216, Nov. 2024.
[8] X. Bei and S. Zhang, “Algorithms for trip-vehicle assignment in ride-sharing,” in *Proc. AAAI Conf. Artif. Intell.*, 2018, Art. no. 1.
[9] Y. Tong, J. She, B. Ding, L. Chen, T. Wo, and K. Xu, “Online minimum matching in real-time spatial data: Experiments and analysis,” in *Proc. VLDB Endowment*, vol. 9, no. 12, pp. 1053–1064, 2016.
[10] Y. Li, Y. Li, Y. Peng, X. Fu, J. Xu, and M. Xu, “Auction-based crowdsourced first and last mile logistics,” *IEEE. Trans. Mobile Comput.*, vol. 23, no. 1, pp. 180–193, Jan. 2024.
[11] J. Xia, Y. Zhao, G. Liu, J. Xu, M. Zhang, and K. Zheng, “Profit-driven task assignment in spatial crowdsourcing,” in *Proc. Int. Joint Conf. Artif. Intell.*, 2019, pp. 1914–1920.

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

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

[14] K. Sakai, K. Takenaka, M.-T. Sun, and W.-S. Ku, "Priority-aware task assignment in opportunistic network-based mobile crowdsourcing," *IEEE Trans. Netw. Sci. Eng.*, vol. 11, no. 2, pp. 2124–2137, Mar./Apr. 2024.

[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] F. Yucel and E. Bulut, "Online stable task assignment in opportunistic mobile crowdsensing with uncertain trajectories," *IEEE Internet of Things J.*, vol. 9, no. 11, pp. 9086–9101, Jun. 2022.

[17] 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.

[18] Z. Chen, H. T. Shen, X. Zhou, and J. X. Yu, "Monitoring path nearest neighbor in road networks," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2009, pp. 591–602.

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

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

[21] D. Hettiachchi, V. Kostakos, and J. Goncalves, "A survey on task assignment in crowdsourcing," *ACM Comput. Surv.*, vol. 55, no. 3, Feb. 2022, Art. no. 49.

[22] W. Huang, P. Li, B. Li, L. Nie, and H. Bao, "Towards stable task assignment with preference lists and ties in spatial crowdsourcing," *Inf. Sci.*, vol. 620, pp. 16–30, 2023.

[23] T. Lai, Y. Zhao, W. Qian, and K. Zheng, "Loyalty-based task assignment in spatial crowdsourcing," in *Proc. ACM Int. Conf. Inf. Knowl. Manage.*, 2022, pp. 1014–1023.

[24] D. Deng, C. Shahabi, and L. Zhu, "Task matching and scheduling for multiple workers in spatial crowdsourcing," in *Proc. 23rd ACM SIGSPATIAL Int. Conf. Adv. Geographic Inf. Syst.*, 2015, pp. 1–10.

[25] Y. Zhao, Y. Li, Y. Wang, H. Su, and K. Zheng, "Destination-aware task assignment in spatial crowdsourcing," in *Proc. ACM Int. Conf. Inf. Knowl. Manage.*, 2017, pp. 297–306.

[26] S. Han, Z. Xu, Y. Zeng, and L. Chen, "Fluid: A blockchain based framework for crowdsourcing," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2019, pp. 1921–1924.

[27] L. Kazemi and C. Shahabi, "GeoCrowd: Enabling query answering with spatial crowdsourcing," in *Proc. 20th ACM SIGSPATIAL Int. Conf. Adv. Geographic Inf. Syst.*, 2012, pp. 189–198.

[28] F. Kojima and M. U. Ünver, "The "Boston" school-choice mechanism: An axiomatic approach," *Econ. Theory*, vol. 55, pp. 515–544, 2014.

[29] Y. Kawase and A. Iwasaki, "Approximately stable matchings with budget constraints," in *Proc. AAAI Conf. Artif. Intell.*, 2018, pp. 1113–1120.

[30] J. W. Hatfield and P. R. Milgrom, "Matching with contracts," *Amer. Econ. Rev.*, vol. 95, no. 4, pp. 913–935, 2005.

[31] M. L. Fisher, G. L. Nemhauser, and L. A. Wolsey, *An Analysis of Approximations for Maximizing Submodular Set Functions-II*. Berlin, Germany: Springer, 1978.

[32] M. Cygan, "Improved approximation for 3-dimensional matching via bounded pathwidth local search," in *Proc. IEEE 54th Annu. Symp. Found. Comput. Sci.*, 2013, pp. 509–518.

[33] D. Gale and L. S. Shapley, "College admissions and the stability of marriage," *Amer. Math. Monthly*, vol. 69, no. 1, pp. 9–15, 1962.

[34] E. Ahmadi and M. Nascimento, "Datasets of roads, public transportation and points-of-interest in Amsterdam, Berlin and Oslo," 2017. [Online]. Available: https://sites.google.com/ualberta.ca/nascimentodatasets/

[35] J. Yuan, Y. Zheng, X. Xie, and G. Sun, "Driving with knowledge from the physical world," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2011, pp. 316–324.

[36] J. Yuan et al., "T-drive: Driving directions based on taxi trajectories," in *Proc. 18th ACM SIGSPATIAL Int. Conf. Adv. Geographic Inf. Syst.*, 2010, pp. 99–108.

[37] Y. Tong, J. She, B. Ding, L. Wang, and L. Chen, "Online mobile micro-task allocation in spatial crowdsourcing," in *Proc. IEEE Int. Conf. Data Eng.*, 2016, pp. 49–60.

[38] Z. Liu, Z. Gong, J. Li, and K. Wu, "Mobility-aware dynamic taxi ridesharing," in *Proc. 36th IEEE Int. Conf. Data Eng.*, 2020, pp. 961–972.

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

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

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

[42] 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.

[43] Y. Zhao et al., "Preference-aware task assignment in spatial crowdsourcing," in *Proc. AAAI Conf. Artif. Intell.*, 2019, pp. 2629–2636.

[44] L. Chen, Q. Zhong, X. Xiao, Y. Gao, P. Jin, and C. S. Jensen, "Price-and-time-aware dynamic ridesharing," in *Proc. IEEE Int. Conf. Data Eng.*, 2018, pp. 1061–1072.

[45] R. C. Wong, Y. Tao, A. W. Fu, and X. Xiao, "On efficient spatial matching," in *Proc. Int. Conf. Very Large Data Bases*, 2007, pp. 579–590.

[46] C. Yang, Y. Hou, Y. Song, T. Zhang, J. Wen, and W. X. Zhao, "Modeling two-way selection preference for person-job fit," in *Proc. 16th ACM Conf. Recommender Syst.*, 2022, pp. 102–112.

[47] X. Hu, Y. Cheng, Z. Zheng, Y. Wang, X. Chi, and H. Zhu, "BOSS: A bilateral occupational-suitability-aware recommender system for online recruitment," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2023, pp. 4146–4155.

[48] F. Yucel, M. Yuksel, and E. Bulut, "QoS-based budget constrained stable task assignment in mobile crowdsensing," *IEEE. Trans. Mobile Comput.*, vol. 20, no. 11, pp. 3194–3210, Nov. 2021.

[49] X. Yin, Y. Chen, C. Xu, S. Yu, and B. Li, "MatchMaker: Stable task assignment with bounded constraints for crowdsourcing platforms," *IEEE Internet of Things J.*, vol. 8, no. 3, pp. 1599–1610, Feb. 2021.

[50] C. Dai, X. Wang, K. Liu, D. Qi, W. Lin, and P. Zhou, "Stable task assignment for mobile crowdsensing with budget constraint," *IEEE Trans. Mobile Comput.*, vol. 20, no. 12, pp. 3439–3452, Dec. 2021.

[51] B. Li, Y. Cheng, Y. Yuan, G. Wang, and L. Chen, "Three-dimensional stable matching problem for spatial crowdsourcing platforms," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2019, pp. 1643–1653.

Yang Huang is currently working toward the doctoral degree with the College of Computer Science and Electronic Engineering, Hunan University, Changsha, China. Her research interests include spatial crowdsourcing and graph computing.

Yumeng Liu received the doctoral degree from the University of Chinese Academy of Sciences, in 2023. He is a senior engineer with the Institute of Software, Chinese Academy of Sciences. His research interest covers data mining and database technology.

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

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

Zhibang Yang received the PhD degree in computer science from Hunan University, Changsha, China, in 2012. He is currently an associate professor with Changsha University. His major research contains cyber security, industrial control system, and e.g.-cloud computing.

Keqin Li (Fellow, IEEE) is a SUNY distinguished professor of computer science with the State University of New York and a national distinguished professor with Hunan University, China. His research interests include cloud computing, fog and e.g. computing, energy-efficient systems, embedded systems, and computer architectures. He has authored more than 860 publications and received several best paper awards. He is a fellow of the AAIA and a member of Academia Europaea.

Kenli Li (Senior Member, IEEE) received the PhD degree in computer science from the Huazhong University of Science and Technology, China, in 2003. He is currently a Cheung Kong professor of computer science and technology with Hunan University, Hunan University. His major research interests include parallel and distributed processing.