Trajectory-Aware Task Coalition Assignment in Spatial Crowdsourcing

Yuan Xie, Fan Wu, Xu Zhou, Wensheng Luo, Yifang Yin, Roger Zimmermann, *Senior Member, IEEE*, Keqin Li, *Fellow, IEEE*, and Kenli Li, *Senior Member, IEEE*

***Abstract*—With the popularity of GPS-equipped smart devices, spatial crowdsourcing (SC) techniques have attracted growing attention in both academia and industry. A fundamental problem in SC is assigning location-based tasks to workers under spatial-temporal constraints. In many real-life applications, workers choose tasks on the basis of their preferred trajectories. However, by existing trajectory-aware task assignment approaches, tasks assigned to a worker may be far apart from each other, resulting in a higher detour cost as the worker needs to deviate from the original trajectory more often than necessary. Motivated by the above observations, we investigate a *trajectory-aware task coalition assignment* (TCA) problem and prove it to be NP-hard. The goal is to maximize the number of assigned tasks by assigning task coalitions to workers based on their preferred trajectories. For tackling the TCA problem, we develop a batch-based three-stage framework consisting of task grouping, planning, and assignment. First, we design greedy and spanning grouping approaches to generate task coalitions. Second, to gain candidate task coalitions for each worker efficiently, we design task-based and trajectory-based pruning strategies to reduce the search space. Furthermore, a 2-approximate algorithm, termed MST-Euler, is proposed to obtain a route among each worker and task coalition with a minimal detour cost. Third, the MST-Euler Greedy (MEG) algorithm is presented to compute an assignment that results in the maximal number of tasks assigned and a parallel strategy is introduced to boost its efficiency. Extensive experiments on real and synthetic datasets demonstrate the effectiveness and efficiency of the proposed algorithms.**

***Index Terms*—Spatial crowdsourcing, task assignment, task coalition, trajectory.**

Manuscript received 4 December 2022; revised 7 October 2023; accepted 10 November 2023. Date of publication 29 November 2023; date of current version 27 September 2024. This work was supported in part by the National Key R&D Program of China under Grant 2021ZD40303, in part by the Creative Research Groups Program of the National Natural Science Foundation of China under Grant 62321003, in part by the Programs of NSFC under Grant 62172146, in part by the Natural Science Foundation of Hunan Province under Grants 2022JJ30009, 2023JJ10016, 2023JJ30083. in part by the Singapore Ministry of Education Academic Research Fund Tier 2 under MOE's official under Grant T2EP20221-0023. Recommended for acceptance by Y. Tong. *(Corresponding author: Fan Wu.)*

Yuan Xie is with the College of Computer Science, Electronic Engineering, Hunan University, Changsha 410012, China, and also with the Institute of Data Science, National University of Singapore, Singapore 119077 (e-mail: yxie@hnu.edu.cn).

Fan Wu, Xu Zhou, and Kenli Li are with the College of Computer Science, Electronic Engineering, Hunan University, Changsha 410012, China (e-mail: wufan@hnu.edu.cn; zhxu@hnu.edu.cn; lkl@hnu.edu.cn).

Wensheng Luo is with the School of data science, Chinese University of Hong Kong Shenzhen, Hong Kong (e-mail: luowensheng@cuhk.edu.cn).

Yifang Yin is with the Institute for Infocomm Research (I^{2}R), A*STAR, Singapore 138632 (e-mail: yin_yifang@i2r.a-star.edu.sg).

Roger Zimmermann is with the School of Computing, National University of Singapore, Singapore 119077 (e-mail: rogerz@comp.nus.edu.sg).

Keqin Li is with the Department of Computer Science, State University of New York, New Paltz, New York 12561 USA (e-mail: lik@newpaltz.edu).

Digital Object Identifier 10.1109/TKDE.2023.3336642

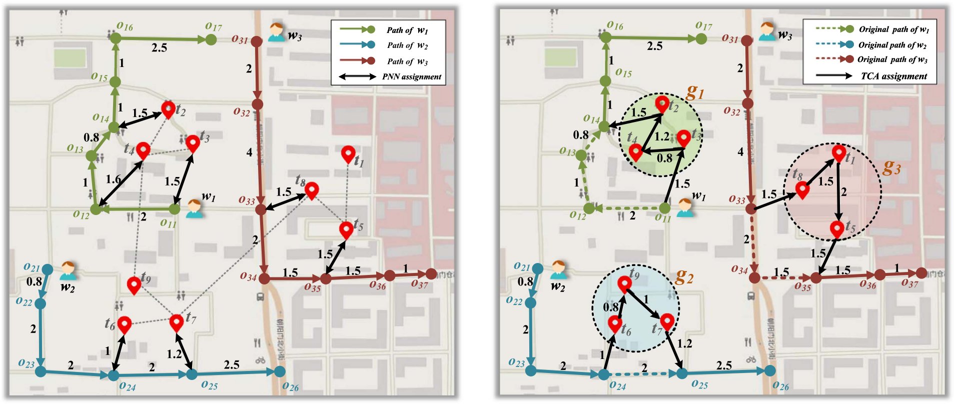
I. Introduction

Spatial Crowdsourcing (SC) is a novel computing paradigm, which employs people with sensor-equipped devices as workers to perform tasks at designated locations. SC has attracted increasing attention from both industry and academia for its wide application in delivery (i.e., Meituan[1]), map navigation (i.e., Google Maps,[2] Amap[3]), online car-hailing (i.e., Didi Chuxing,[4] Grab[5]), and other real-world applications. It also plays an important role in solving tedious and manual tasks like collecting vital spatial-temporal data (i.e., taking a store photo, reporting traffic information, and monitoring air quality).

Task assignment is a fundamental problem in SC by matching workers with appropriate tasks based on their locations under spatial-temporal constraints [1], [2], [3], [4], [5], [6], [7], [8]. Most existing approaches assume that workers will depart from a specific starting point and return to their destination after completing multiple tasks. The detour cost for workers is the distance they actually traveled minus the origin-destination distance. However, the above assumption ignores workers' preferences for paths.

In real scenarios, workers expect to perform as many tasks near their preferred path as possible to reduce detour costs. Therefore, part-time workers are often reluctant to take additional detours to perform further tasks, instead prefer to complete them on their daily path or commute. The ridesharing applications, i.e., Didi, Grab, and Uber, engage part-time drivers to deliver passengers that align with the driver's daily path or commute to minimize detour costs, consequently maximizing profits. Specifically, worker w_1 drives his car from home to the company regularly. Passengers r_1 and r_2 share the same destination as w_1, and the start points are located near the w_1's daily path. By utilizing the advantageous proximity, w_1 can accommodate both r_1 and r_2 as passengers, ensuring that the incurred detour costs are under the constraints. Furthermore, the applicability also can extend to some spatial tasks, i.e., taking

[1]https://www.meituan.com/
[2]https://www.google.com/maps/
[3]https://www.amap.com/
[4]https://www.didiglobal.com/
[5]https://www.grab.com/



(a) PNN assignment (b) TCA assignment

Fig. 1. Illustrations of the PNN assignment and the TCA assignment.

TABLE I
TASKS AND WORKERS

Worker	$w.p$	α	$w.\tau$	$w.c$
w_1	p_1	0.6	5.28	3
w_2	p_2	0.7	6.51	3
w_3	p_3	0.6	7.20	3

Task	$t.s$	$t.d$
t_1	17:30	18:10
t_2	16:00	17:40
t_3	16:00	17:10
t_4	16:00	17:20
t_5	17:30	18:30
t_6	18:00	19:00
t_7	18:00	19:30
t_8	17:30	18:00
t_9	18:00	19:10

photos for stores or check-in POIs, wherein employing part-time workers to execute tasks that are aligned with predefined routes. After part-time workers complete the assigned tasks, they return to their routine routes, bringing the advantages of the distance proximity and reducing the inconvenience of workers significantly.

To alleviate this issue, some recent methods focus on trajectory-aware task assignment problems in SC [9], [10], [11], [12]. They allow workers to deviate from their original trajectory to perform a task and come back to the former egress. Nevertheless, the assigned tasks for a worker may be far apart since the distribution of tasks is ignored. If nearby tasks are gathered and assigned together to a suitable worker, the detour cost can be reduced further.

Example 1: Paidian[6] is published by Meituan for collecting stores' information such as a store's photo. As shown in Fig. 1, there are three workers w_1, w_2, and w_3, and nine task requests (e.g., taking a photo of a specific store) $t_1 \sim t_9$ released. Table I shows the spatial-temporal information of workers and tasks. For each task t, $t.s$, and $t.d$ are the release time and deadline, respectively. All tasks must not exceed deadlines. For each worker w, $w.p$ and $w.c$ are the preferred trajectory and the capacity of w, respectively.

Take the state-of-the-art algorithm Path Nearest Neighbor (PNN) [9] as an example. In the PNN assignment, workers search for tasks near to their preferred trajectories and greedily choose the nearest one. After completing one of the assigned tasks, each worker will return to the same egress as their ingress. In Fig. 1(a), the total distance of the preferred trajectory p_1 of w_1 is 8.8 units. The maximum detour distance of w_1 is $w_1.\tau = 0.6 \times 8.8 = 5.28$ units and the capacity is 3. In general, the distance of deviating from the preferred trajectory cannot exceed the maximum detour distance. Task t_2 is the nearest to w_1 and is first qualified to be assigned to w_1 because the detour cost is 2 (i.e., $1\times 2 < 5.28$). Next, t_3 is qualified to be assigned to w_1 since the total detour cost of assigning t_3 and t_4 to w_1 is 5, which is also less than the maximum detour distance. Task t_4 cannot be assigned since the total detour cost exceeds the maximum detour distance if t_2 and t_3 are already assigned. Thus, a feasible assignment result of the PNN algorithm is $A_p = \{(w_1, t_2), (w_1, t_3), (w_2, t_6), (w_2, t_7), (w_3, t_5), (w_3, t_8)\}$ with the goal of maximizing the number of assigned tasks.

The above PNN assignment is limited in the following two aspects. First, the workers are restricted to come back to the same egress after tasks are performed, which inevitably increases the detour cost. Second, tasks are assigned and completed one by one in sequence. The spatial distribution of tasks is overlooked, resulting in that the tasks assigned to the same worker may be far apart from each other. These two limitations significantly hinder the workers from doing their jobs efficiently as they need to deviate from their trajectories more frequently than necessary.

To address the above concerns, we investigate a novel problem in SC, namely the trajectory-aware task coalition assignment (TCA), which aims to maximize the number of assigned tasks. Different from existing methods [13], [14], [15], [16], our approach focuses on assigning a task coalition to each worker according to her preferred trajectory. In particular, a path with minimal detour cost is planned for each worker to deviate from egress and return to another ingress on their trajectories. This is more in line with real-life scenarios. Besides, a task coalition

[6]https://paidian.meituan.com/

is computed for each worker based on their preferred trajectory each time by aggregating nearby tasks.

To illustrate the TCA problem clearly, we present a task assignment scenario in spatial crowdsourcing in the following example.

Example 2: In the TCA problem, shown in Fig. 1(b), task coalitions are first generated and we obtain three coalitions $g_1 = \{t_2, t_3, t_4\}, g_2 = \{t_6, t_7, t_9\}$, and $g_3 = \{t_1, t_5, t_8\}$. The task coalitions g_1, g_2, and g_3 are assigned to workers w_1, w_2, and w_3, respectively. Based on these task coalitions, we have another assignment $A_t = \{(w_1, g_1), (w_2, g_2), (w_3, g_3)\}$. Take worker w_1 as an example. She first leaves her preferred trajectory from o_{11} to complete tasks t_3, t_4, and t_2 in sequence, and comes back to the trajectory from the point o_{14}. In this way, worker w_1 can finish three tasks t_2, t_3, and t_4 by deviating from the preferred trajectory only one time. The detour cost of w_1 is 1.2 (i.e., $1.5 + 0.8 + 1.2 + 1.5 - 2 - 1 - 0.8 = 1.2$). Similarly, the detour cost of w_2 is 2 and w_3's is 3, respectively.

In the above examples, the TCA assignment results in the assignment of more tasks, where workers complete 9 tasks deviating from their preferred trajectories, with a total detour cost of 6.2. Conversely, by the PNN assignment, workers can only complete 6 tasks and the overall detour cost is 16. This suggests that the TCA assignment increases the number of completed tasks that deviate from the preferred trajectory, and also greatly reduces detour costs for workers.

Challenges: To the best of our knowledge, our study is the first to investigate the TCA problem and we prove its NP-hardness. The TCA problem consists of three subproblems, namely task grouping, planning, and assignment, which faces three main challenges: (1) existing methods, such as DBSCAN [17] and k-means [18], are inefficient in generating task coalitions based on the spatial distribution of tasks; (2) it is time-consuming to plan a route for each worker to choose the best exit pair (i.e., egress and ingress) of the preferred trajectory such that the worker can deviate from egress and return to the ingress; and (3) it is difficult to effectively assign task coalitions to proper workers such that all workers and tasks satisfy the given spatio-temporal constraints.

For effective processing, we first explore a batch-based three-stage framework, consisting of task grouping, task planning, and task assignment. For **Challenge (1)**, two new task grouping approaches, Greedy and Spanning, are designed for generating task coalitions, which achieve these coalitions by more efficiently considering the spatial distribution of tasks than existing task clustering algorithms [17], [18]. For **Challenge (2)**, an approximate algorithm with a 2-approximate ratio is proposed based on two new pruning strategies to solve the task planning problem. It computes routes with a minimal detour cost for each worker-and-coalition pair. For **Challenge (3)**, the MST-Euler Greedy (MEG) algorithm is designed to gain an assignment with the maximal number of assigned tasks. Additionally, a parallel strategy is introduced to improve the performance of MEG. Extensive experiments show that the proposed algorithms are up to 3 orders of magnitude faster than the exact method while obtaining very close results. In addition, they can also accomplish a higher number of assigned tasks than the existing PNN [9] method.

Briefly, our contributions are illustrated as follows.

- We identify a new task assignment problem, named TCA, to maximize the number of assigned tasks by assigning task coalitions to workers based on their trajectories, and prove its NP-hardness (Section III).
- To solve the trajectory-based task planning problem, we propose pruning strategies to improve the efficiency, and a 2-approximate ratio approximate algorithm named MST-Euler (Section VI).
- We develop the MEG algorithm to gain the assignment result with the goal of maximizing the number of assigned tasks and a parallel strategy to boost the performance (Section VII).
- We conduct extensive experiments on real and synthetic datasets to show the effectiveness and efficiency of our proposed algorithms (Section VIII).

Related work is reviewed in Section II. Section IV outlines the framework and Section V introduces the task grouping methods. Finally, Section IX concludes the article and Section X is the discussion.

II. Related Work

A. Task Assignment Problem

Location-Aware Task Assignment Problem: Spatial crowdsourcing (SC) is comprised of four main research areas, namely task assignment [19], [20], [21], [22], quality control [23], [24], incentives [25], and privacy protection [26], [27], [28]. Among these, task assignment is the foundational problem of SC. As introduced by Tong et al. [29], task assignment problems can be classified into assignment and planning problems. An assignment problem relates to managing vast numbers of tasks and workers through an SC platform; for example, Didi Chuxing needs to deal with millions of order requests every day. Thus, how to assign large-scale tasks to a massive number of workers is the foundational challenge in spatial crowdsourcing. Zhao et al. [30] studied the destination-aware task assignment problem for achieving the maximal total number of completed tasks under the constraints of the deadline. Tong et al. [31] were concerned with assigning suitable workers to tasks while they appear on the platform in real time [32]. Zhao et al. [7], [8] developed a preference-aware task assignment problem to predict the workers' preference to the tasks for achieving the maximal expected preference value. These location-aware task assignment studies concentrate on assigning proper workers to tasks based on their exact locations. However, they all ignore workers' preferred trajectories.

Trajectory-Aware Task Assignment Problem: Assigning suitable tasks to proper workers based on the workers' trajectories is more inline with real applications. There exist some trajectory-aware task assignment studies for SC problems. The problem setting of Costa et al. [10], [12], named In-Route Task Selection (IRTS) in spatial crowdsourcing, is similar to our work, which assigns the tasks to the workers based on their preferred routes. A worker is allowed to deviate from an exit point on the route to the spatial task and returns to the same egress under a limited budget for maximizing the total profit [33]. The studies in [34], [35] are similar to our problem of assigning suitable tasks to workers based on each worker's preferred trajectory. However, the TCA problem is allowing the worker to return to other exits for a lower detour cost. Besides, the nearby tasks are clustered to make the most use of a spatial advantage when a worker proceeds for the assigned tasks. Morever, deep learning technologies are also applied in the trajectory-aware task assignment to predict the future location of tasks and paths of workers [36], [37], the goal of which is maximizing the expected number of assigned tasks.

B. Task Planning Problem

The planning problem [38], [39], [40] is different from the assignment problem. A planning problem is related to applications such as food delivery and ride-sharing, where the SC platform should plan a route for the workers to guide them to complete as many tasks as possible. Deng et al. [41], [42] first studied the planning problem to maximize the number of assigned tasks and also prove its NP-hardness in the case of one worker to many tasks. [41] schedule a feasible route for the worker by inserting the task into the best position, which has a similar research problem to our TCA problem. However, [41] lacks the task grouping stage but selects tasks from the global set at the cost of more extensive search spaces. Besides, the start and destination are fixed for each worker [41]. But in the TCA problem, all points on the worker's trajectory are regarded as equally important, where each trajectory point on the worker's route can be the start and endpoint, bringing more flexible routes. Moreover, the TCA problem schedules a route considering the global distance of the worker's trajectory and tasks' locations while [41] inserts tasks one by one into the worker's schedule route. Thus, the method of [41] cannot directly solve the proposed TCA problem.

Additionally, the dial-a-ride (DARP) problem and courier delivery problems proposed in [43] are associated with the schedule from origin to destination of a vehicle's route for maximizing the number of transported customers. They are both similar to the subproblem, trajectory-based task coalition planning, of TCA in this paper. Compared to DARP, our TCA problem is much different since the DARP problem needs to pick up their passengers and deliver them to their destinations. However, the worker in TCA is required to perform their tasks when arriving at its location, only ”check-in” one location. But the workers need to follow a predefined route to decide the best degrees and ingress, which adds another level of complexity to the problem. Tong et al. [44], [45] focus on developing the insertion-based framework to solve the flexible multi-objective route planning for shared mobility. Zeng et al. [46] aim to solve shared-route planning queries in ridesharing problems. Zeng et al. [47] propose a guarantee algorithm for optimizing and minimizing the makespan of couriers and the total latency of requesters simultaneously for solving the Last-Mile Delivery problem. However, the optimizing goals of these planning problems are different from our TCA problem, which is not designing the feasible route by considering the worker's preferred route.

TABLE II
SYMBOLS AND DESCRIPTIONS

Symbols	Descriptions
P	The trajectory of worker
G	The task coalition set
g	One coalition in task coalition set G
$g.s$	The task coalition size
$g.r$	The task coalition range
$t.l$	The location of task
$o_i.l$	The location of each point on trajectory P
$\pi(g)$	The task completion sequence of coalition g
$c(t_i.l)$	Total travelling cost for completing all tasks in g
o_u	The egress on worker's trajectory
o_b	The ingress on worker's trajectory
$\mathcal{D}(o_u, g, o_b)$	The detour distance from o_u to o_b
$\mathcal{C}(o_u, g, o_b)$	The detour cost of $Ddis(o_u, g, o_b)$
$dist(a, b)$	The distance from location a to location b
$p(a, b)$	The distance from a to b alongside trajectory P

III. Preliminaries

In this subsection, we present the important definitions and formulate the TCA problem. Table II lists the important notations frequently utilized in this paper.

A. Notations and Definitions

Definition 1 (Spatial Task): Given a task set $T = \{t_1, t_2, \ldots, t_n\}$, each spatial task $t = \langle t.\gamma, t.l, t.s, t.d\rangle$ is available at the timestamp γ, released at start time $t.s$ in its location $t.l$, and needs to be finished before the deadline $t.d$.

Definition 2 (Task Coalition): A task coalition is comprised of a set of tasks close to each other. Given a range constraint r and a size constraint k, it satisfies

$$g = \{g_i \in G | \underset{\{t_i, t_j \in g\}}{dist} (t_i, t_j) < r, |g| \leq k\}, \tag{1}$$

where G is the set of task coalitions, the distance between any two tasks in g is no more than r, and the size of each task coalition cannot exceed k.

Definition 3 (Task Sequence): For a given task coalition g and a worker w_j, a task schedule $\pi(g) = \{t_1, t_2, \ldots, t_{|\pi|}\}$ is defined as a specific completion sequence of tasks in g by the worker w_j. Here, w_j will complete all tasks in g following their order in $\pi(g)$. After finishing tasks in g, the travel distance of the worker w_j is

$$c_{w,g}(t_i.l) = \begin{cases} c(t_{i-1}.l) + dist(t_{i-1}.l, t_i.l) & i \neq 1, \\ dist(o_u.l, t_i.l) & i = 1, \end{cases}$$

where $dist(t_{i-1}.l, t_i.l)$ is the traveling cost between tasks t_{i-1} and t_i, $dist(o_u.l, t_1.l)$ is the distance from the egress to the coalition when there is only one task in the coalition. If given a specific worker w and a task coalition g, we use $c(t_i.l)$ to denote $c_{w,g}(t_i.l)$.

For the sake of simplicity, we have an assumption that all workers share the same velocity during the whole moving period, and the travel cost between two locations can be estimated by their distance. Note that the proposed algorithms are not limited to this assumption. Besides, our TCA problem is a static task assignment in Spatial Crowdsourcing. All spatial-temporal information is prior known for the platform to achieve an optimal assignment outcome.

Definition 4 (Spatial Worker): A worker is represented as $w = \langle w.p, \alpha, w.\tau, w.c\rangle$, where her preferred trajectory $w.p = \{o_1, o_2, \ldots, o_e\}$ is comprised of intersection points o_i on the road network for $1 \leq i \leq m$. $w.c$ is the capacity of the worker w. Besides, given the detour rate α of the worker w, her maximum detour cost $w.\tau$ is computed as the total distance multiplied by the rate α (i.e., $w.\tau = \alpha \cdot \sum_{o_i \in P} dist(o_{i-1}.l, o_i.l), 2 \leq i < e$).

It is noted that the worker only provides the detour rate limitation, which results in the maximum detour distance for each worker. $w.\tau$ is also rewritten as τ in the following.

Definition 5 (Worker-and-Coalition Pair): For a set of workers W and a set of task coalitions G, a worker-and-coalition pair (w, g) is valid if all tasks within coalition g are created and worker w is available, worker w can arrive at each task in g before the deadline, and worker w will not deviate from his trajectory more than $w.\tau$.

For each valid worker-and-coalition pair (w, g), the solution needs to guarantee that worker w can arrive at all required locations of the tasks in task coalition g. The travel cost from the start point on the trajectory to the location of the last task in g is $p(o_s.l, o_u.l) + c(t_i.l)$, where $c(t_i.l)$ is defined in Definition 3 for completing all tasks in g from the egress, and $p(o_s.l, o_u.l)$ is the distance from the starting point of workers to the egress alongside trajectory P. Every worker should arrive at each task assigned before the respective deadline, which requires $(p(o_s.l, o_u.l) + c(t_i.l))/speed_w \leq t_i.d$. Here, $speed_w$ is the average speed of the worker.

Definition 6 (Detour Distance): For a preferred trajectory P and a task coalition g, the detour distance is denoted as $\mathcal{D}(o_u, g, o_b)$, which is the distance from egress, to complete all tasks in g and come back to the ingress for $o_u, o_b \in P$. Thus, we have

$$\mathcal{D}(o_u, g, o_b) = c(t_{|\pi|}.l) + dist(t_{|\pi|}.l, o_b.l). \quad (2)$$

Here, $t_{|\pi|}$ is the last task in the complete sequence, $c(t_{|\pi|}.l)$ is the distance from egress o_u to complete all tasks in g as defined in Definition 3, $dist(t_{|\pi|}.l, o_b.l)$ is the distance from the last task in g and return to the ingress o_b on the trajectory.

Definition 7 (Detour Cost): The detour cost is denoted as $\mathcal{C}(o_u, g, o_b)$, which is the extra traveling cost for completing tasks in g. Thus, we have

$$\mathcal{C}(o_u, g, o_b) = \mathcal{D}(o_u, g, o_b) - p(o_u.l, o_b.l), \quad (3)$$

where $p(o_u.l, o_b.l)$ is the distance from egress and ingress alongside trajectory P.

The detour distance is the traveling cost of worker w for completing tasks on the detour from his trajectory. However, the real detour cost is the extra cost for accomplishing the task coalition, which requires the subtraction of the trajectory distance from o_u to o_b alongside trajectory P on the basis of detour distance.

B. Problem Formalization

Trajectory-Aware Task Coalition Assignment (TCA) Problem: Given a worker set W and a task set T, the objective is to obtain the assignment consisting of worker-and-coalition pairs that minimize total detour costs while ensuring the maximal number of assigned tasks in priority. Besides, the assignment must meet the following constraints:

1) *Detour distance constraint:* Worker w can only be assigned to task coalitions with detour distances no more than the maximal distance budget $w.\tau$.
2) *Deadline constraint:* Worker w can only be assigned to a task coalition such that she can arrive at all tasks in the coalition before their deadlines.
3) *Capacity constraint:* Each worker can only complete $w.c$ assigned tasks at most.

Remark: The TCA problem contains three subproblems, namely task grouping, planning, and assignment. In the task grouping phase, tasks are divided into different coalitions based on their locations such that tasks in a coalition are close to each other. Next, the task planning phase aims to compute qualified worker-and-coalition pairs. To achieve this goal, a route is generated to minimize the detour cost of each worker-and-coalition pair. Notably, the detour cost for each worker-and-coalition pair remains constant beyond the planning stage, serving as a fixed input for the subsequent task assignment phase. Thereafter, the task assignment phase is conducted to assign each task coalition to an available worker with the goal of maximizing the number of assigned tasks in priority and minimizing the total detour costs as the secondary goal.

Theorem 1: The TCA problem is NP-hard.

Proof: The NP-hardness proof can be achieved by transforming a Hamiltonion Path Problem, which has been proven to be NP-hard [48], to an instance of the TCA problem.

Hamiltionion Path Problem (HPP): Given a graph $G = (V, E)$ with $|V| = n$ nodes, a start node (v_{start}) and a stop node (v_{stop}), the problem asks to compute a simple path, beginning with node v_{start} and ending with node v_{start}, to traverse all nodes exactly once.

The goal of the TCA problem is to assign a task coalition g to each worker w to maximize the number of assigned tasks and minimize the detour cost for each worker-and-coalition pair simultaneously. This means that for an egress o_u and an ingress o_b of the worker trajectory $w.p$, the TCA is required to generate all the routes from o_u to o_b for completing tasks in the coalition g and select the optimal route with the minimal detour costs, where the worker's capacity is equal to the size of the coalition, $w.c = |g|$. Besides, we set the detour rate to ensure each worker completes all tasks within the coalition under a maximal detour distance.

Consider the following special instance of the TCA problem with only one worker. Let a node v_i represent the location of a task t_i in the task coalition g for $1 \leq i \leq k$. We could set the start node v_{start} as the egress o_u and the last node v_{stop} as the ingress o_b. The cost $cost(v_i, v_j)$ between nodes v_i and v_j is computed as the distance $dist(t_i, t_j)$ between corresponding tasks t_i and t_j.

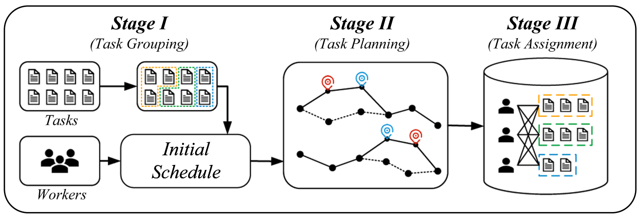


Fig. 2. Batch-based three-stage framework.

Algorithm 1: Batch-Based Three-Stage Framework.

```
Input: Time interval $\Gamma$
Output: Best assignment $A_p$ for all workers within the time
        interval $\gamma$
1 while current time $\gamma$ is in $\Gamma$ do
2     Compute all the available spatial tasks in $T$;
3     Compute all the available workers in $W$;
4     Divide tasks into coalitions and gain a task coalition $g$;
5     Apply the MST-Euler algorithm for the optimal routes of
        candidate worker-and-coalition pairs;
6     Compute an assignment result $A_p$ by the MEG algorithm;
7     foreach worker-and-coalition pair $(w_i, g_i) \in A$ do
8         Assign worker $w_i$ to conduct task coalition $g_i$ based
            on the proposed planed route;
```

The above instance of our TCA problem aims to compute a route with the minimal cost to complete all the tasks, which requires determining all the paths from the start node to the end node to traverse all tasks in g. This goal is equal to computing all routes from v_{start} such that they cover each node v_i for one time in the Hamiltonian Path Problem.

From the above derivation, we can reduce the Hamiltonian Path Problem to an instance of the TCA problem. Since the Hamiltonian Path Problem is NP-hard, the TCA problem is also NP-hard. □

It is worth noting that our TCA problem is much more complex than the Hamiltonian Path Problem, because it needs to plan tasks for multiple workers, the egress and ingress may be different points, and it needs to compute the best egress and ingress for each worker-and-coalition pair.

IV. Framework for the TCA Problem

In this section, we introduce a batch-based three-stage framework as shown in Algorithm 1, which iteratively assigns tasks to workers in multiple batches. In each batch, the optimal assignment is achieved by a three-stage process consisting of task grouping, planning and assignment phases. Fig. 2 illustrates the roadmap of our batch-based three-stage framework.

As depicted in Algorithm 1, for timestamp γ, we retrieve all available tasks T and workers W. Here, available tasks refer to those that have not been assigned to any worker or those that newly appeared after the last timestamp. Available workers refer to those who are not assigned to any task coalitions in the former batch or those who newly appeared in the current timestamp (Lines 2–3).

In the first phase, tasks are divided into coalitions by task grouping strategies (Line 4). For a given range $g.r$ and size $g.s$, the distance of any two tasks in the coalition cannot be more than $g.r$ and the cardinality of each coalition is no more than $g.s$. The task grouping methods are discussed in Section V. In the second phase, for the worker-and-coalition pairs, we plan a feasible route to minimize the detour cost from the worker trajectory to the task coalition with the MST-Euler algorithm, which integrates the pruning strategies detailed in Section VI (Line 5). In the third phase, the MST-Euler Greedy (MEG) algorithm is applied to obtain the optimal task assignment result for maximizing the number of assigned tasks, and a parallel strategy is introduced in Section VII (Line 6). Finally, according to the optimal assignment, the platform informs workers to conduct task coalitions based on the specific planning routes (Lines 7–8).

We will introduce the methods of each stage in detail in the following sections. Note that this paper assumes the preferred route is optimal for each worker in the real application. That is, the worker can follow this route forward to the destination at the lowest cost. The point of the TCA problem is to focus on the detour costs. Despite this, we set the negative detour cost to 0 since the negative costs will not bring the extra detour costs.

V. Task Grouping

Task grouping is an important component in our TCA problem, and it has a significant impact on the subsequent assignment result.

DBSCAN [17] is a general task grouping method that can exploit the spatial distribution of tasks. It generates distance-sensitive task clusters as the task coalitions. Although nearby tasks are clustered into task coalitions, it faces an imbalance problem, i.e., some coalitions have many tasks while others have few. Besides, tasks far away from clusters form separate task coalitions, leading to higher detour costs. To solve this problem, we present a new grouping method, namely the spanning grouping method, which constructs a minimum spanning tree to balance the average distance among all task coalitions. Since the high running time cost of the spanning grouping method to construct a spanning tree of all tasks, we also develop the greedy grouping method utilizing a greedy strategy to separate tasks.

Spanning Grouping Method: The spanning method first constructs a minimum spanning tree to connect all tasks based on the distances between the tasks. After that, we decompose the spanning tree iteratively into subtrees whose sizes are no more than the required group size. Here, the tasks in each subtree form a task coalition. The Edge-Delete algorithm [49] is introduced to decompose the minimum spanning tree evenly. The variance of each edge is $\sigma(e) = \frac{1}{|E|}\sum_{e\in E}(w(e)^2) - (\frac{1}{|E|}\sum_{e\in E}(w(e))^2$, where $w(e)$ is the distance, $|E|$ is the number of edges. The weight of each edge is $\sigma(E) - \sigma(E/e)$, meaning the decrease in the variance. Tasks will be grouped evenly if removing the edge that decreases the variance the most in the minimum spanning tree. The decomposition is stopped if the task size in a subtree is smaller than the required coalition size. As a result, each subtree represents a task coalition. Next, we present an example to illustrate the process of the spanning grouping method.

Example: We illustrate the example in Fig. 1(a). First, we build the MST for all tasks, $\{t_1, t_2, \ldots, t_9\}$, based on the distance, denoted by the grey dotted line. Assuming the edge $e(t_7, t_8)$ has the highest weight of variance difference, we delete $e(t_7, t_8)$ to separate the tree into two parts. As a result, the MST is divided as two subtrees, $s_1 = \{t_1, t_5, t_8\}$, and $s_2 = \{t_2, t_3, t_4, t_6, t_7, t_9\}$. Since the coalition size of s_2 is 3, smaller than the worker's capacity, which is set as 5, we stop splitting the s_1 since it already can form a task coalition. Suppose the edge $e(t_4, t_9)$ has the highest weight in the s_2; we repeat the same processes above to split the s_2 into $\{t_2, t_3, t_4\}$ and $\{t_6, t_7, t_9\}$. The size of each subtree is smaller than the worker's capacity, so stop splitting the subtrees and forming the task coalitions eventually.

Greedy Grouping Method: Since the spanning grouping method of constructing the minimum spanning tree for all tasks requires a high running time, we introduce a more time-efficient alternative method, named the greedy grouping method. The Greedy method greedily assembles nearby tasks in a given coalition range to form task coalitions of a specified size. In particular, it starts from a random task t_s and searches for the nearest task t_n. If the task t_n is not included in any task coalition, t_n can be included in the task coalition of t_i. When the number of the assembled tasks is equal to the task coalition size $g.s$, we choose another new task as a starting task and repeat the above steps.

The experiments in Section VIII verify that the greedy and the spanning grouping methods are superior to DBSCAN in terms of result quality and efficiency.

VI. Task Coalition Planning

Next we introduce the trajectory-based task coalition planning problem. To tackle the problem effectively, we present efficient pruning strategies and propose an approximation algorithm with a 2-approximation ratio.

A. Trajectory-Based Task Coalition Planning Problem

In the second phase of the TCA problem, for each qualified worker-and-coalition pair, a feasible route with minimal detour cost is generated for each task coalition.

It is time-consuming to plan a route for a worker from her trajectory to complete assigned tasks. The time complexity depends on the number of trajectory points and the task completion sequence. That is, the planning problem faces two main challenges: 1) how to find a task completion sequence that minimizes the travel cost for completing all tasks in the coalition, and 2) how to choose the best egress and ingress points among all possible exit-entrance pairs on the original trajectory to divert to the newly assigned tasks while minimizing the total detour cost.

To address the two challenges, we design task-based and trajectory-based pruning strategies to reduce the search space. Additionally, the MST-Euler Algorithm with an approximate ratio of 2 is developed to compute a feasible route for each worker.

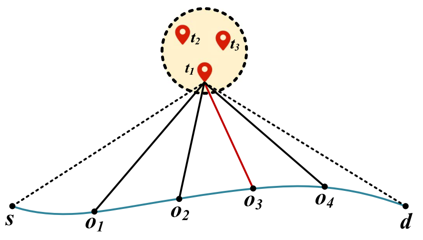


Fig. 3. Illustration of the trajectory $P = \{s, o_1, o_2, o_3, o_4, d\}$ and task coalition $g_i = \{t_1, t_2, t_3\}$. We assume the $dist(t_1, o_3)$ is the nearest distance from the worker's trajectory to the task coalition.

B. Pruning Strategies

In this subsection, we present the design of two pruning strategies, namely a task-based and a trajectory-based pruning strategy, to reduce the search space as detailed below.

1) Task-Based Pruning Strategy: Under a limited detour distance, some coalitions which are far from workers' trajectories may be excluded in advance. Inspired by this observation, we first propose the task-based pruning strategy.

Before we describe the task pruning strategy, we observe that $dist^{\downarrow}(g, P)$ has to be less than $\tau/2$, where $dist^{\downarrow}(g, P)$ is the minimal distance from a coalition to the trajectory of w (i.e., $min_{t \in g}\{min_{o_i \in P}\{dist(t.l, o_i.l)\}\}$). That is, if w cannot reach the nearest task and return back to the same offset within the distance constraint of τ, other tasks cannot be completed in the same task coalition as well.

Lemma 1: Given the set of coalitions, and a specific worker w, the candidate coalitions satisfy the following condition:

$$dist(o_i.l, t_n.l) + dist(t_n.l, o_j.l) - p(o_i.l, o_j.l) \leq \tau \qquad (4)$$

where t_n is the nearest task to trajectory P, and $o_i, o_j \in P$.

Proof: We will prove the above lemma by contradiction and by utilizing Fig. 3. Without loss of generality, there is a coalition containing the tasks t_1, t_2, and t_3. Assuming the shortest distance from trajectory P to the coalition is $dist(t_1, o_3)$, the worker deviates from s and comes back to d. We have $dist(s, t_1) + dist(t_1, d) - p(s, d) < 2 \cdot dist(t_1, o_3)$ owing to the definition of triangle inequality, where $dist(s, t_1) - p(s, o_3) < dist(t_1, o_3)$ and $dist(t_1, d) - p(o_3, d) < dist(t_1, o_3)$. If $dist(s, t_1) + dist(t_1, d) - p(s, d) > \tau$, it holds that $dist(t_1, o_3) > \tau/2$ (i.e., $dist^{\downarrow}(g, P) > \tau/2$). If the detour cost of any task in the coalition exceeds τ, the coalition can be pruned safely since it cannot satisfy the detour distance constraint. □

Based on Lemma 1, we can prune unreachable coalitions if the detour cost exceeds τ. This not only avoids planning for a task completion sequence but also averts checking all possible exit-entrance pairs. On the other hand, we propose the following pruning strategies to remove some invalid trajectory pairs as early as possible.

2) Trajectory-Based Pruning Strategy: Because of the detour limitation and deadline constraints, not all exit pairs on a trajectory are available. For example, in Fig. 3, if choosing s as egress, the candidate ingress set is $\{o_1, o_2, o_3, o_4, d\}$. However, due to the distance and time limitations, not all points may be included in the candidate ingress set, and some of which could be deleted in advance. Therefore, we develop trajectory-based pruning strategies to delete some unavailable points on the trajectories of workers.

Lemma 2: Given a preferred trajectory P, the valid set of exit points, denoted as $g.V$, is

$$g.V = \{o|dist(t,o) \leq \tau - dist^{\downarrow}(g,P), t \in g, o \in P\}, \quad (5)$$

where $dist^{\downarrow}(g,P)$ is the lower bound distance from trajectory to task coalition and $dist(t,o)$ is the distance from the task in coalition to a point on the trajectory.

Proof: Take Fig. 3 as the example again. Assume $dist(o_3, t_1)$ is the nearest distance from the task coalition to the trajectory (i.e., $dist^{\downarrow}(g,P)$); the task coalition completion sequence is t_2, t_3, and t_1; o_3 is the ingress. For any detour point except o_3, we can choose another one as the exit point. We take o_1 as the egress and t_2 as the first task. If $dist(o_1,t_2) > \tau - dist(o_3,t_1)$, the detour distance exceeds τ because $\pi(t_2,t_3,t_1).d$ is no less than zero. Thus, we have $dist(o_1,t_2) + \pi(t_2,t_3,t_1).d + dist(o_3,t_1) > \tau$, where $\pi(t_2,t_3,t_1).d$ is the travel cost of the task coalition and also is denoted as $\pi(g).d$. Thus, in this case, o_1 is not a valid detour point for the task coalition g, which cannot be added as a valid exit point. Accordingly, the point on P whose distance to any task in g exceeds $\tau - dist^{\downarrow}(g,P)$ can be pruned safely. The lemma holds. □

Lemma 3: If the detour points on a preferred trajectory P satisfy $\mathcal{D}(o_2,g,o_3) < \tau$ and $\mathcal{D}(o_1,g,o_4) < \tau$, then $\mathcal{C}(o_2,g,o_3)$ is greater than $\mathcal{C}(o_1,g,o_4)$.

Proof: As shown in Fig. 3, the detour cost of worker w to complete task coalition g from (o_1,o_4) is $\mathcal{C}(o_1,t,o_4) = dist(o_1, g) + \pi(g).d + dist(o_4,g) - p(o_1,o_4)$. The detour cost of choosing
deviating from o_2 and returning to o_3 is $\mathcal{C}(o_2,t,o_3) = dist(o_2,g) + \pi(g).d + dist(o_3,g) - p(o_2,o_3)$. The detour cost difference holds that $\mathcal{C}(o_1,g,o_4) - \mathcal{C}(o_2,g,o_3) = dist(o_1,g) - p(o_1,o_2) - dist(o_2,g) + dist(o_4,g) - p(o_3,o_4) - dist(o_3,g)$. According to the triangle inequality theorem, we have $dist(o_1,g) - p(o_1,o_2) - dist(o_2,g) < 0$ and $dist(o_4,g) - p(o_3,o_4) - dist(o_3,g) < 0$. Therefore, $\mathcal{C}(o_1,t,o_4)$ is less than $\mathcal{C}(o_2,t,o_3)$, and this lemma holds. □

For all exit pairs among the candidate set, the detour cost of exit pairs that have a longer detour distance is smaller than that of other pairs. Therefore, the exit pairs which have a smaller detour distance can be deleted safely according to Lemma 3.

Lemma 4: Given the preferred trajectory P of a worker w, a task coalition g, and o_n is the nearest offset, available detour points on P are separated by the point o_n into two sets, $n.LV$ and $n.RV$, where $n.LV$ includes o_n and detour points earlier than o_n when the user is moving from s to d along P, and $n.RV$ includes points visited later than o_n. The egress and ingress of the best exit pair o_l and o_r are from $n.LV$ and $n.RV$, respectively.

Proof: If the two points o_l and o_r are both in $n.LV$ for $l, r \leq n$, the detour cost is $\mathcal{C}(o_l,g,o_r) = dist(o_l,\pi^1) + \pi(g).d + dist(\pi^k,o_r) - p(o_l,o_r)$. Here, π^1 is the first task in $\pi(g)$ and π^k is the last task. For the exit pair egress o_l and ingress o_n, the detour cost is $\mathcal{C}(o_l,g,o_n) = dist(o_l,\pi^1) + \pi(g).d + dist(\pi^k,o_n) - p(o_l,o_n)$. According to the triangle inequality principle, the difference between two edges is lower than the third edge. Therefore, we have $\mathcal{C}(o_l,g,o_r) < \mathcal{C}(o_l,g,o_n)$. Accordingly, (o_l,o_n) is not the best exit pair of g. Therefore, o_l and o_r must be from $n.LV$ and $n.RV$, respectively. □

Lemma 5: Given a fixed task sequence of g, for any candidate detour point on the trajectory P, the upper bound distance from the point on P to the coalition is $\tau - \pi(g).d - dist^{\downarrow}(g,P)$.

Proof: For any trajectory point o_i on the trajectory, if $dist(g,o_i)$ is greater than $\tau - \pi(g).d - dist^{\downarrow}(g,P)$ for $o_i \in P$, we have $dist(g,o_e) + \pi(g).d + dist^{\downarrow}(g,P) > \tau$. Here, $\pi(g).d$ is the travel cost for completing all tasks in the coalition from the start task to the end task, and $dist^{\downarrow}(g,P)$ is the minimal distance from the trajectory P to the given coalition. □

Based on the above pruning strategies, the task coalitions far from a given worker's trajectory and invalid trajectory points can be pruned safely in advance.

C. The MST-Euler Algorithm

By integrating task-based and trajectory-based pruning strategies, we propose the MST-Euler algorithm, inspired by [50], to plan a feasible route for each worker-and-coalition pair efficiently. The approximation ratio of MST-Euler is also presented. Note that the Christofides algorithm [51] is a seminal solution for the TSP problem, which has similar steps as the following MST-Euler algorithm. However, the application scenarios of the Christofides algorithm and our TCA problem are inherently distinct, in which Christofides schedules a route that visits all nodes, encompassing both tasks and trajectory points. In contrast, the planning stage revolves around ensuring visiting all nodes and a subset of the worker's trajectory points. This critical variation in route requirements renders the direct application of the Christofides algorithm unsuitable for addressing the challenges presented by the TCA problem. Inspired by this, we propose the MST-Euler algorithm.

The main idea of the MST-Euler algorithm is first to combine tasks and trajectory points as the node set V. Then, to form the minimum spanning tree (MST) of V and construct an Euler tour among all connected node pairs. Next, we obtain a Hamiltonian cycle connecting all tasks based on the short-cutting strategy [50]. By integrating the two pruning strategies in Section VI-B, the MST-Euler algorithm returns the planning route with minimal travel cost for each qualified worker-and-coalition pair.

Short-Cutting Strategy: To obtain a task coalition completed sequence, we introduce the short-cutting [50], [51] strategy in the MST-Euler algorithm. First, we establish the minimum spanning tree of a worker's trajectory and all tasks in the coalition, such as Fig. 4(a). Next, the Euler tour for each connected node pair, denoted as the dotted line, is established based on the MST, shown in Fig. 4(b). Each connected node pair has two arrows directed at each other. To obtain the task completion sequence, directly connecting one task to another by skipping the hollow points can form a Hamiltonian cycle among all tasks since it follows the triangle inequality principle, as shown in Fig. 4(c).

Example: A set of tasks $T = \{t_1, t_2, \ldots, t_7\}$ are denoted as hollow dots and a set of trajectory points $O = \{o_{11}, o_{12}, o_{13}\}$

Algorithm 2: MST-Euler Algorithm.

```
   Input: The worker-and-coalition pair $(w, g)$.
   Output: The best planning route $s^*$.
1  $s^* \leftarrow \emptyset$;
2  if $dist^{\downarrow}(g, P) > \tau/2$ then
3      $s^* \leftarrow \emptyset$;                                   // Lemma 1
4  else
5      $V \leftarrow \{t\}_{t \in g} \cup \{o_i\}_{o_i \in P}$;
6      MST ← minimum spanning tree of $V$;
7      E-MST ← Euler tour of MST;
8      H ← Hamiltonian cycle of E-MST;
9      foreach $o \in P$ do
10         if $min\{dist(t, o) | t \in g\} \leq \tau - dist^{\downarrow}(g, P)$ then
11             $tj_w \leftarrow \{o\}$;                               // Lemma 2
12     foreach $\pi \in H$ do
13         foreach $o \in tj_w$ do
14             if $dist(\pi, o) > \tau - \pi.d - dist^{\downarrow}(g, P)$ then
15                 $tj_w \leftarrow tj_w / \{o\}$;                      // Lemma 5
16             Let $o_n \in P$ as the nearest point from $g$ to $P$;
17             if $o < o_n$ then
18                 $n.LV \leftarrow \{o\}$;
19             else
20                 $n.RV \leftarrow \{o\}$;                            // Lemma 4
21         foreach $o_l \in n.LV$ and $o_r \in n.RV$ do
22             if $\mathcal{D}(o_l, \pi(g), o_r) < \tau$ then
23                 break;                                          // Lemma 3
24     $s^* \leftarrow \arg min_{(o_l, o_r) \in tj_w, \pi \in H} \mathcal{C}(o_l, \pi(g), o_r)$;
25 return $s^*$
```

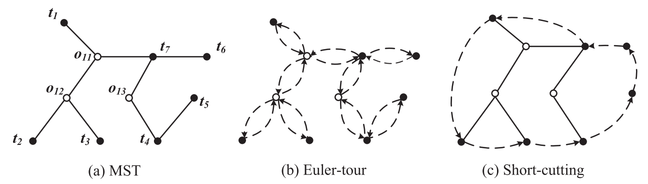


Fig. 4. (a) shows the minimum spanning tree of tasks and trajectories, (b) presents the Euler-tour, and (c) obtains the Hamiltonian circle by applying the short-cutting strategy.

are represent to solid dots in Fig. 4. The minimum spanning tree (MST) is constructed from the union node set $T \cup O$ as shown in Fig. 4(a). Alongside the MST, bilateral connect each intersect node pair to construct a bidirectional Euler graph as shown in Fig. 4(b). Then, directly guide a route from one task to another by skipping all connected hollow dots. As presented as Fig. 4(c), task t_1 is directly connected to t_2, but skip o_{11} and o_{12}. Finally, the task completed circle sequence is given by $g^{\pi} = \{t_1, t_2, t_3, t_4, t_5, t_6, t_7\}$.

Algorithm: In Algorithm 2, the MST-Euler algorithm first initializes s^* as the best planning route result of worker-and-coalition pair (w, g). From Lemma 1, the distance from the available coalition to w's trajectory has to be less than $\tau/2$. If the distance exceeds $\tau/2$, the return is an empty set as the final planning result since a feasible route cannot be generated for the worker-and-coalition pair (Lines 2–3). For available task coalitions and the trajectory point set, it constructs a minimum spanning tree (MST) of V starting from a random task, presented in Fig. 4(a) (Lines 5–6). After that, the Euler-tour is established on the MST by building bidirectionally connected routes for each pair of connected nodes, such as shown in Fig. 4(b) (Line 7). Next, each task of a coalition in the MST is reachable by the short-cutting strategy and we obtain a Hamiltonian cycle to connect all tasks as shown in Fig. 4(c) (Line 8). Then, candidate trajectory points are added into tj_w based on Lemma 2 (Lines 9–11). If the distance from the exit pair on P to a task coalition exceeds $\tau - \pi(g).d - dist^{\downarrow}(g, P)$, which means the worker cannot complete the coalition under the limitation of τ, the trajectory point can be pruned safely according to Lemma 5 (Lines 14–15). Based on Lemmas 3 and 4, the optimal exit pair is selected from $n.LV$ and $n.RV$, respectively (Lines 16–23). Finally, MST-Euler returns the optimal planning route of the worker-and-coalition pair with minimal detour cost (Line 24).

Example: Consider the example in Fig. 1 again. For worker w_2, by the MST-Euler algorithm, task coalition g_2 is generated which contains tasks t_6, t_7, and t_9. The detour distance of w_2 is 6.51 and the shortest distance from p_2 to task coalition g_2 is $dist(t_6, o_{24}) = 0.6$, which is smaller than half of the detour distance, i.e., $0.6 < 6.51/2$. Therefore, g_2 is an available task coalition for w_2. Next, we establish a graph among the tasks in g_2 and all trajectory points of w_2, where the point set is $V = \{o_{21}, \ldots, o_{26}, t_6, t_7, t_9\}$ and the weight of each edge is the distance between corresponding nodes. After that, we search the minimum spanning tree of the established graph and obtain the Hamiltonian cycle. Next, we obtain the final task completing sequence as $\pi(g_2) = \{t_6, t_9, t_7\}$, and o_{24} and o_{25} can be selected as the optimal egress and ingress, respectively.

Below, we provide the approximate ratio proof of the proposed MST-Euler algorithm.

Theorem 2: The MST-Euler algorithm is a 2-approximation algorithm.

Proof: Let a set V be comprised of tasks in the coalition g and trajectory points of worker w. The optimal travel cost from trajectory to task coalition of pair (w, g) is denoted as OPT, and the cost of MST-Euler is represented as $cost(C)$. In Algorithm 2, the cost to form the minimum spanning tree (MST) is denoted as $cost(T)$. Thus, we have $cost(T) \leq OPT$ since the travel cost of the MST (i.e., Fig. 4(a)). is no more than the optimal travel cost for traversing all points in V. For creating the Euler tour of the MST, we need $2 \cdot cost(T)$ since the Euler tour contains twice edges of the MST (i.e., Fig. 4(b)), denoted as $cost(\widehat{T})$. Thus, any solid point pair is reachable according to the constructed two-way Eulerian graph. Because of the triangle inequality, we have $cost(C) \leq cost(\widehat{T})$ after the "short-cutting" step (i.e., Fig. 4(c)). Combining the above inequalities, we have $cost(C) \leq 2 \cdot OPT$, and Theorem 2 holds. In conclusion, the MST-Euler approach is a 2-approximation algorithm. □

Time Complexity: Assume there are k tasks in the coalition g and $|P|$ points on the worker's trajectory. If the distance from the coalition to the trajectory exceeds half of $w.\tau$, the coalition will be pruned directly (i.e., Lemma 1), which costs $O(k \cdot |P|)$ (Lines 2–3). Prim's algorithm is applied for forming a minimum spanning tree, and it costs $O((k + |P|)^2)$ (Line 6). Creating an Euler tree and Hamiltonian cycle also needs $O((k + |P|)^2)$ (Lines 7–8). According to Lemma 2, we require $O(k \cdot |P|)$ to

Algorithm 3: MEG Algorithm.

```
Input: A set of workers in $W$ and a set of coalitions in $G$.
Output: Assignment results $A$.
1  Initialize $M$ as the candidate worker-and-coalition set;
2  foreach $w$ in $W$ do
3      foreach $g$ in $G$ do
4          Apply the MST-Euler algorithm to plan each
             worker-and-coalition pair, $(w, g)$;
5          if $P(w, g)$ is valid then
6              $M \leftarrow (w, g)$;
7          else
8              $M \leftarrow \emptyset$;
9  Sort the candidate pairs in $M$ based on $g.s$ and $\mathcal{C}$;
10 foreach $(w, g) \in M$ do
11     if $w$ not in $A$ and $g$ not in $A$ then
12         $A \leftarrow (w, g)$;
13 return Assignment $A$;
```

choose candidate points of the trajectory P (Lines 10–12). The time cost of finding the best pair is $O(k{\cdot}|P|)$ according to Lemmas 3, 4 and 5 (Lines 13–23). Thus, the total time complexity is $O(3{\cdot}k{\cdot}|P| + 2{\cdot}(k + |P|)^2)$.

VII. Task Coalition Assignment

In the third phase of solving the TCA problem, we propose the MST-Euler Greedy (MEG) algorithm to assign task coalitions to proper workers. To compute qualified worker-and-task coalition pairs, a route is generated for each pair to minimize the detour cost in the task planning phase detailed in Section VI. To attain a higher number of assigned tasks among all valid worker-and-coalition pairs, we introduce the MST-Euler Greedy algorithm. Furthermore, a parallel strategy is applied to boost performance.

To achieve the optimal assignment result, we need to search all available worker-and-coalition pairs in the planning phase. Therefore, the main idea of the MEG algorithm is to generate a feasible route for each available worker-and-coalition pair. This is followed by the greedy strategy to obtain the final assignment that achieves the maximal number of assigned tasks with minimal total detour costs.

Algorithm: In Algorithm 3, we show the pseudocode of the MEG algorithm. Specifically, it initializes a candidate worker-and-coalition set M (Line 1). For each worker-and-coalition pair, we apply the MST-Euler algorithm to plan an optimal route in order to guide workers in completing all tasks in the coalition (Lines 2–4). Under the deadline constraint and detour distance limitation, if the worker-and-coalition pair has a valid planning route $P(w, g)$, the worker-and-coalition pair is to be updated and included in M (Lines 5–6). Otherwise, if worker w cannot complete this task coalition, the pair cannot be included in M (Lines 7–8). Next, we sort the candidate set according to the group size and the detour cost of $P(w, g)$ (Line 9). Finally, we select worker-and-coalition pairs iteratively and include them in the final assignment A if they have not being assigned so far (Lines 10–12). Note that, in this assignment phase, we not only achieve the goal of maximizing the number of assigned tasks but also generate a route with the minimum detour cost of each assigned worker-and-coalition pair.

Time Complexity: Assume there are m workers, n tasks, and g coalitions after the division into task coalitions. The time cost of the MST-Euler algorithm is $O(3{\cdot}k{\cdot}|P| + 2{\cdot}(k + |P|)^2)$ as analyzed in Algorithm 2 (Line 4). Besides, it costs $O(1)$ to identify if the pair (w, g) is available or not (Lines 5–8). The planning phase costs $O(mg{\cdot}(3{\cdot}k{\cdot}|P| + 2{\cdot}(k + |P|)^2))$ to generate feasible routes for all worker-and-coalition pairs, and the time complexity of the selection is $O(mn)$ (Lines 10–12). Therefore, the total time complexity of MEG is $O(mg{\cdot}(3{\cdot}k{\cdot}|P| + 2{\cdot}(k + |P|)^2))$.

Parallelization: To boost the performance of task assignments, a parallel strategy is developed to further reduce the execution time of the MEG approach. The proposed batch-based framework is divided into three phases, task grouping, planning, and assignment. Among the three phases, task planning is the most time consuming. It requires invoking the MST-Euler algorithm $O(|W|{\cdot}|G|)$ times for planning routes of all worker-and-coalition pairs, which accounts for a large proportion of the running time. To further improve the performance of the proposed algorithms, we introduce parallel techniques to compute routes for worker-and-coalition pairs in parallel. In this way, given $\hat{t}$ CPU threads, it costs $O(|W|{\cdot}|G|/\hat{t})$ to compute the planning route for all worker-and-coalition pairs.

VIII. Experiments

We evaluate the efficiency and effectiveness of the proposed algorithms through extensive experiments. Moreover, we also evaluate different task grouping methods, such as Greedy, Spanning, and DBSCAN, to generate task coalitions. In addition, we conduct experiments to show the effectiveness of the parallel strategy.

A. Experiment Setup

Datasets: We evaluate our proposed methods on two real datasets, Berlin and Ams [52]. The Berlin dataset contains 5,548 POIs on a road network involving 428,769 vertices and 504,229 edges. The Ams dataset consists of 1,446 POIs on a road network involving 106,600 vertices and 130,091 edges. In the two datasets, the bus routes and bus stops are taken as the trajectories and their points. Besides, the locations of coffee shops, bars, banks, and restaurants are considered realistic task locations. We choose POIs in Berlin and Ams as tasks and bus routes as the workers' trajectories.

Settings: We choose the parameters settings as existing works [1], [53]. The capacity of workers is set to 5 tasks by default, according to the number of workers and tasks in our datasets. Since the task coalition size is equal to the capacity of workers in our paper, we define the default task coalition size to be 5 as well. After testing multiple parameters and considering the workers' maximal detour distance, we finalize the default value of the task coalition range to be 0.5 km for the best performance. For each worker, the detour distance is equal to the total trajectory distance multiplied by the detour rate. Similar to [54], we split the bus route every 10 points as the worker's

TABLE III
EXPERIMENTAL SETTINGS

Parameters	Value
Number of workers W	Berlin: 400, 600, **800**, 1k, 1.2k Ams: 60, 80, **100**, 120, 140
Number of tasks T	Berlin: 1k, 2k, **2.5k**, 3k, 5k Ams: 100, 200, **300**, 400, 500
Detour rate α	0.5, 0.6, **0.7**, 0.8, 0.9
Task coalition size $g.s$	3, 4, **5**, 6, 7
Task coalition range (km) $g.r$	0.5, 0.6, **0.7**, 0.8, 0.9

single trajectory, i.e., $|P| = 10$. The task's expiration time is $dt \times dist(o.l, t.l)/speed_w$ [1], where dt is the hyperparameter which is set as 4, $dist(o.l, t.l)$ is the maximum distance from the worker's original location the task's location, and the worker's average speed $speed_w$ is 60. Table III summarizes the experimental settings.

Algorithms: We evaluate the performance of the following algorithms. Em-greedy aims to enumerate all possible egress and ingress comprised of the worker's trajectory and list all feasible routes to complete the task coalition separately for exploring the optimal route with minimal detour costs in the planning stage; then, utilize the greedy-inspired algorithm in the task assignment stage. Emt-greedy adds tricks, i.e., the detour distance constraint, to filter out invalid routes in advance. To determine the optimal number of assigned tasks, we utilize the min-cost max-flow (MM) algorithm in the task assignment stage compared to Em-greedy, denoted as Em-MM. PNN [9] limits workers to return to the same egress after completing the task under the detour distance constraint. Additionally, we modified three methods from [41], namely GALS, Insert, and InsertPrun, which research a similar work to this work. Specifically, GALS assigns the proper workers to suitable tasks from the global task set by the max-flow min-cost algorithm, then insert tasks into the worker's trajectory to schedule the optimal route for each worker to filter some failed tasks and assign them again in the next assignment round until all valid tasks are assigned. We add a detour distance constraint in the GALS. Insert has the same framework as our method, including task grouping, planning, and assignment stages. In the task planning stage, enumerates all egress and ingress as the origin and destination and insert the tasks in the coalition after finding the best position. The task grouping and assignment stages are the same as our approaches. InsertPrun applies the pruning strategies designed in our work on the basis of Insert.

- *Em-greedy:* Enumerate the egress and ingress pairs and task completed sequence separately to explore an optimal route in the planning stage and utilize the greedy algorithm in the assignment stage.
- *Emt-greedy:* Filter out the invalid routes in advance with the detour distance constraint on the basis of Em-greedy.
- *Em-MM:* The algorithm aims to enumerate all possible egress and ingress comprised of the worker's trajectory and apply the min-cost max-flow algorithm in the task assignment stage.

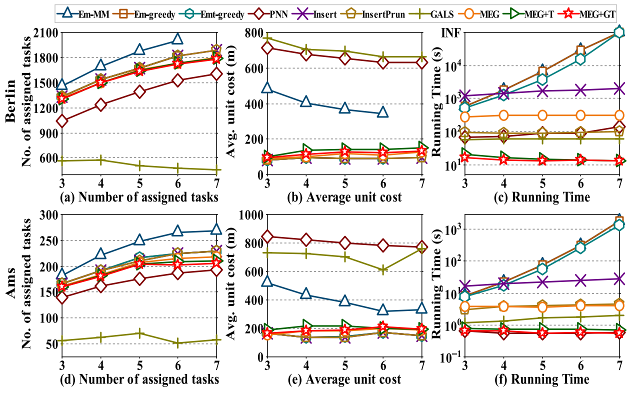


Fig. 5. Effect of the task coalition size $g.s$.

- *PNN:* The Path Nearest Neighbor method allows workers to return to the same offset point after completing the assigned tasks.
- Insert: Enumerates all egress and ingress as the origin and destination and find the best position to insert the tasks in the coalition in the task planning stage; then, apply the greedy assignment algorithm to obtain the final assignment result.
- *InsertPrun:* Applying the designed pruning strategies based on Insert.
- *GALS:* Repeat the steps of assigning tasks from the global tasks by the max-flow min-cost algorithm and insert tasks into workers' trajectory under the detour distance constraint until all valid tasks are assigned.
- *MEG:* The MEG algorithm applies the MST-Euler algorithm without any pruning strategies.
- *MEG+G:* The MEG+G method adds the task-based pruning strategy based to the MEG algorithm.
- *MEG+GT:* The MEG+GT method adds both the task-based and the trajectory-based pruning strategies based to the MEG algorithm.

Metrics: We use three key metrics to evaluate the algorithms:

- *Assigned Tasks:* This metric counts the number of tasks successfully assigned to task coalitions.
- *Average Detour Cost:* It measures the mean detour cost required to complete a single task.
- *Running Time:* This metric reflects the total time taken to process all workers and tasks.

These metrics serve to assess the effectiveness and efficiency of the algorithms.

All experiments are conducted on an Intel(R) Xeon(R) Gold 6140 CPU @ 2.30 GHz in Python 3.6.

B. Experimental Results

We list the experimental results of the task coalition $g.s$, task coalition range $g.s$, worker's detour rate α, number of tasks $|T|$, and number of workers $|W|$ as follows.

Effect of the Task Coalition Size $g.s$: Fig. 5 depicts the experimental results when varying the coalition size from 3 to 7. Fig. 5(a), (b), and (c) show the results of Berlin and Fig. 5(d),

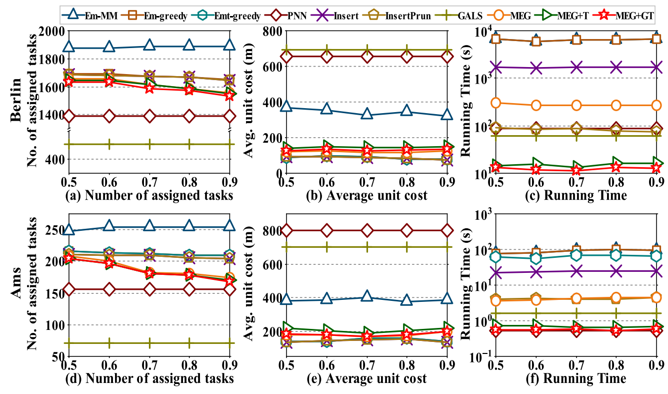


Fig. 6. Effect of the task coalition range $g.r$.

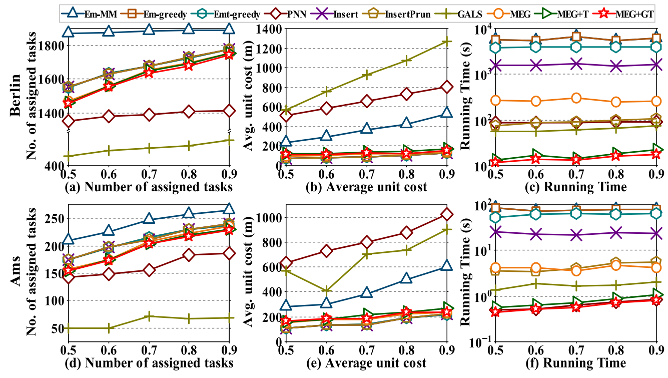


Fig. 7. Effect of the detour rate α.

(e), and (f) represent the results of Ams. The number of assigned tasks of most methods increases when more tasks are in a single coalition. That is because if the size of the task coalition is expanded, each worker can complete more tasks. The proposed methods, MEG, MEG+T, and MEG+GT, attain similar numbers of assigned tasks to Em-greedy and Emt-greedy. This fact demonstrates that packaging the nearby tasks is effective. Insert and InsertPrun achieve a slightly higher number of assigned tasks and lower average distance than MEG-related methods. The primary reason is that the insert-related algorithms identify good positions and sequentially insert tasks into workers' trajectories, allowing for the exploration of more viable routes for each worker-coalition pair. However, MEG-related methods first obtain an execution order for the tasks in the task coalition, then find a position to insert the whole coalition into worker's trajectory, leading to overlooking some good solutions. The running time of MEG-related methods requires at most two lower orders of magnitude than the Insert and InsertPrun, which is more acceptable for all parties in crowdsourcing platforms. Besides, the MEG-related methods perform better than other baselines, PNN and GALS. Moreover, when the number of tasks is larger than 7, the running time to attain the assignment result of Em-MM, Em-greedy and Emt-greedy exceeds 2.78 hours (e.g., 10^5 seconds), which is not an acceptable waiting time in our problem and we set it as INF. The reason is that the Em-related methods need to enumerate all feasible routes to complete tasks in task coalition. The time complexity depends on the number of assigned tasks.

Effect of Task Coalition Range $g.r$: Fig. 6 shows the experimental results by varying the task coalition range from 0.5 km to 0.9 km. As shown in Fig. 6(a) and (d), with the expansion of the coalition range, the number of assigned tasks of all the algorithms decreases. That is because the larger the coalition range, the sparser the coalitions become, which requires a higher detour cost to finish each task coalition. Therefore, given the same limited maximal detour cost for each worker, fewer task coalitions can be completed. PNN and GALS have the same number of assigned tasks no matter how we change the coalition range since these two methods don't have the task grouping stage, which is not affected by the task coalition range. Fig. 6(b) and (e) illustrate the average detour cost. MEG+GT results in less detour cost than MEG+G, while they require a similar time cost. In Fig. 6(c) and (f), PNN and GALS require nearly three times the average detour cost of our proposed methods in the Berlin dataset and nearly two times in the Ams dataset. This fact shows that our methods can reduce the detour cost significantly with better efficiency.

Effect of the Detour Rate α: The experimental results of varying the detour rate are shown in Fig. 7. Illustrated in Fig. 7(a) and (d), the number of assigned tasks increases when the detour rate increase. This is because workers are allowed to go further to complete more tasks since the detour ratio is linear with the maximal detour distance for each worker. After relaxing the detour rate, our proposed methods significantly increase the number of assigned tasks. That is, more tasks are completed, resulting in higher detour costs. Em-MM has the highest number of assigned tasks since applying the max-flow min-cost algorithm in the assignment stage. Em-greedy and Emt-greedy obtain better performance than MEG-related methods since enumerating all ingress and egress pairs. The Insert and InsertPrun achieve a slightly higher number of assigned tasks since they can search for more feasible routes for each worker but require almost two orders of magnitude higher time than the MEG-related methods. Fig. 7(b) and (e) indicate that the average unit cost of each worker increases when expanding the detour rate. The MEG-related methods achieve a higher number of assigned tasks and lower average travel costs than PNN and GALS, and the running time of MEG-related methods is lower than that of PNN and GALS. The reason is that the PNN is forced to return to the original predefined route resulting in a higher detour cost, and GALS selects tasks first and schedules later, also bringing worse results when applying to the TCA problem. MEG+GT has a lower running time than most methods in both two datasets, shown in Fig. 7(c) and (f), respectively. This fact reveals the effectiveness and efficiency of our proposed methods.

Effect of the Number of Tasks $|T|$: Fig. 8 depicts the experimental results when increasing the number of tasks. Fig. 8(a) and (d) show that, with more nearby tasks being valid for workers, the number of assigned tasks increases. MEG, MEG+G, and MEG+GT achieve a similar number of tasks assigned but slightly lower than that of Em-MM and Em-greedy. This fact

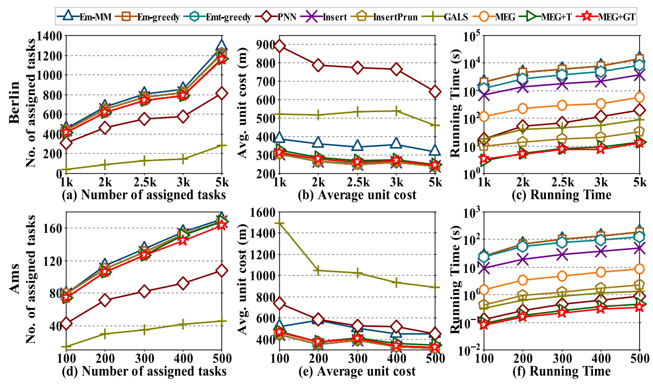


Fig. 8. Effect of the number of tasks $|T|$.

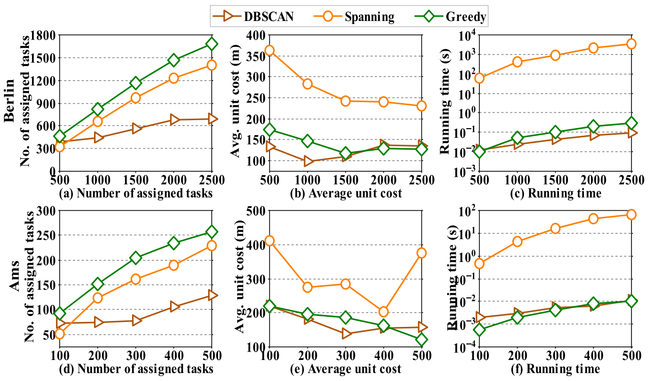


Fig. 10. Effect of different task grouping methods.

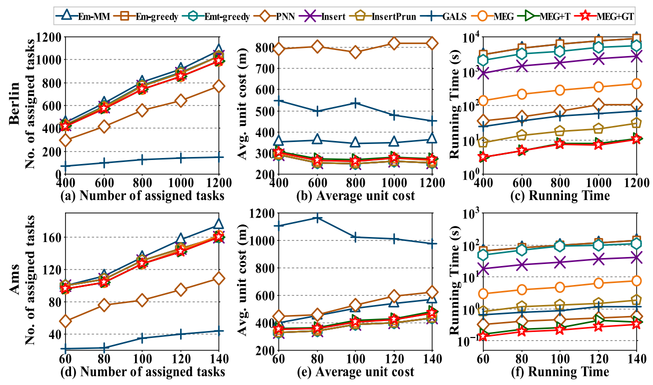


Fig. 9. Effect of the number of workers $|W|$.

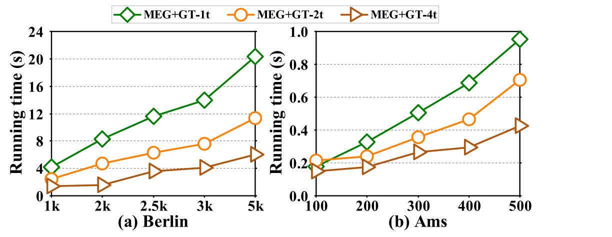


Fig. 11. Effect of parallel strategy.

illustrates the effectiveness of our pruning strategies. Fig. 8(b) and (e) show that the average detour cost decreases since more nearby tasks can be selected for suitable workers. Illustrated in Fig. 8(c) and (f), to schedule more task coalitions, more time is required. Therefore, the running time increases when expanding the number of tasks. Specifically, MEG's costs are two orders of magnitude higher than for MEG+GT while they achieve a similar number of tasks assigned and average detour cost. Furthermore, the running time of Em-MM and Em-greedy is nearly three orders of magnitude higher than MEG+GT, which indicates the efficiency of the proposed methods.

Effect of the Number of Workers $|W|$: Fig. 9 shows the experimental results by increasing the number of workers. In Fig. 9(a) and (d), with the addition of workers, the number of assigned tasks of all algorithms increases. That is because more workers can complete more task coalitions. In Fig. 9(b), the average cost of all methods is stable even with the number of assigned tasks rising in the Berlin dataset. Fig. 9(e) shows that the average detour cost increases with an increasing number of workers in the Ams dataset. That is, more tasks result in higher total detour costs. As shown in Fig. 9(c) and (f), compared with Em-MM and Em-greedy, MEG+GT needs three orders of magnitude less computation. On the other hand, PNN and GALS require a higher time cost but complete fewer assigned tasks at a higher detour cost. Insert and InsertPrun also cost a higher time but achieve a slightly higher number of assigned tasks. All these facts support the effectiveness and efficiency of the proposed MEG-related method can achieve good performance to solve the TCA problem.

Effect of Different Task Grouping Methods: Fig. 10 depicts the experimental results of the proposed algorithms by applying different task grouping methods. The number of assigned tasks increases when the number of tasks grows from 500 to 2500 in the Berline dataset and from 100 to 500 in the Ams dataset since more task coalitions are available. The running time is the time cost of dividing tasks into different coalitions. Among the three grouping methods, DBSCAN achieves the lowest number of assigned tasks. The number of assigned tasks of Spanning is higher than DBSCAN, but the running time is significantly higher than both DBSCAN and Greedy. This is because forming a minimum spanning tree among all tasks requires more time than the other methods. In terms of the number of assigned tasks, running time, and average detour cost, Greedy is a practical choice since it can achieve the largest number of assigned tasks with less running time.

Effect of Parallel Strategy: In this experiment, we evaluate the efficiency of the parallel strategy. Fig. 11 shows the parallel strategy of MEG+GT by varying the number of threads. The speedup of the algorithm is approximately linear by varying the number of threads from 1 to 4. The running time of MEG+GT is reduced from 20.31 s to 5.99 s in the Berlin datasets with 5,000 tasks and from 0.95 s to 0.42 s in the Ams datasets with 500 tasks on 4 threads. The performance of other algorithms follows a consistent trend. As shown, the running time of MEG+GT is reduced roughly linearly by introducing the parallel strategy, which demonstrates the efficiency of the proposed parallel strategy.

Summary: The proposed MEG, MEG+G, and MEG+GT algorithms perform better than GALS and PNN in terms of the number of assigned tasks, the average detour cost, and the running time. The number of assigned tasks and the average detour cost of the insert-related algorithm is slightly higher than our proposed algorithms. Although the insert-related algorithm has been applied in real-world applications, like Didichuxing, to schedule drivers' routes, the efficiency of the solutions still needs improvement, especially in time-sensitive scenarios. Our MEG-related algorithms take approximately 10 seconds on the Berlin dataset, while InsertPrun needs approximately 100 seconds. This significant difference in time expenditure could potentially lead to user attrition. Additionally, by utilizing the task-based and trajectory-based pruning strategies, our approaches reduce the running time by at most three orders of magnitude on the two real test datasets, which indicates the efficiency of our designed pruning strategies.

IX. Conclusion

In this paper, we study the TCA problem to assign workers to proper tasks according to their preferred trajectories. We propose a three-stage framework comprised of task grouping, planning, and assignment. To plan a feasible route for each qualified worker-and-coalition pair, we propose the MST-Euler approximate algorithm. In addition, task-based and trajectory-based pruning strategies are designed to improve efficiency. For the assignment stage, we explore the greedy-inspired MEG algorithm. The algorithms can be parallelized to accelerate the running time. Extensive experiments demonstrate the effectiveness and efficiency of our proposed algorithms. In our TCA problem, we assume that each worker deviates from their trajectory for only one task coalition. We can extend our solutions to consider workers that deviate from multiple task coalitions in our future work. Besides, predicting the worker's future trajectory and task location in the next timestamps in the TCA problem is also promising research work in the future.

X. Discussion

The greedy-inspired MEG algorithm is heuristic when the task coalitions are independent in our TCA problem. However, if task coalitions with dependency, the MEG algorithm is approximate and has a guarantee of the approximate rate. We present the theorem as follows.

Theorem 3: The optimal assignment A_{opt} of the MEG algorithm for another case of TCA where task coalitions being dependent is at least $(1-\frac{1}{e})\cdot A_{opt}$.

Proof: Let $|(w_i, g_i)|$ as the size of the task coalition g_i. Then, the objective function of MEG can be rewritten as the $Sum(A) = \sum_{w_i \in W, g_j \in G} |(w_i, g_j)|$. The value of the $|(w_i, g_j)|$ is always positive when assigning g_j to coalition w_i, resulting in $\sum_{w_i \in W, g_j \in G} |(w_i, g_j)|$ being monotone. If we have $Sum(\widehat{A}) - Sum(\widetilde{A}) \leq Sum(\overline{A}) - Sum(A)$, the submodular is proven, where $Sum(\widehat{A}) = \{A \cup (w_i, g_i), (w_j, g_j)\}$, $Sum(\widetilde{A}) = \{A \cup (w_i, g_i)\}$, and $Sum(\overline{A}) = \{A \cup (w_j, g_j)\}$. $\widehat{A}$ is the assignment result where task coalition g_j be assigned to the activate worker w_j after g_i is already be assigned. On the other hand, for $\overline{A}$, g_i is not assigned yet. Assume task coalition g_i and g_j have a dependency relationship, g_j can be completed in $\overline{A}$ but cannot be finished by $\widehat{A}$. Therefore, we have $Sum(\widehat{A}) - Sum(\widetilde{A}) = 0$, and $Sum(\overline{A}) - Sum(A) = |(w_j, g_j)|$. In summary, $Sum(A)$ is monotone and submodular. Since $Sum(A)$ is monotone and submodular, the proven holds according to [55], [56]. □

References

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

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

[3] 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.

[4] D. Shi, Y. Tong, Z. Zhou, B. Song, W. Lv, and Q. Yang, "Learning to assign: Towards fair task assignment in large-scale ride hailing," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2021, pp. 3549–3557.

[5] X. Tang et al., "Value function is all you need: A unified learning framework for ride hailing platforms," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2021, pp. 3605–3615.

[6] 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 *Proc. AAAI Conf. Artif. Intell.*, 2019, pp. 2245–2252.

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

[8] 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.

[9] 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.

[10] C. F. Costa and M. A. Nascimento, "In-route task selection in crowdsourcing," in *Proc. 20th Int. Conf. Adv. Geographic Inf. Syst.*, 2018, pp. 524–527.

[11] C. F. Costa and M. A. Nascimento, "In-route task selection in spatial crowdsourcing," *ACM Trans. Spatial Algorithms Syst.*, vol. 6, pp. 7:1–7:45, 2020.

[12] C. F. Costa and M. A. Nascimento, "Online in-route task selection in spatial crowdsourcing," in *Proc. 20th Int. Conf. Adv. Geographic Inf. Syst.*, 2020, pp. 239–250.

[13] Z. Chen, P. Cheng, L. Chen, X. Lin, and C. Shahabi, "Fair task assignment in spatial crowdsourcing," *VLDB Endow.*, vol. 13, pp. 2479–2492, 2020.

[14] P. Cheng, J. Jin, L. Chen, X. Lin, and L. Zheng, "A queueing-theoretic framework for vehicle dispatching in dynamic car-hailing," *VLDB Endow.*, vol. 14, pp. 2177–2189, 2021.

[15] P. Cheng, C. Feng, L. Chen, and Z. Wang, "A queueing-theoretic framework for vehicle dispatching in dynamic car-hailing," in *Proc. Int. Conf. Data Eng.*, 2019, pp. 1622–1625.

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

[17] M. Ester, H. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 1996, pp. 226–231.

[18] D. J. MacKay et al., *Information Theory, Inference and Learning Algorithms*, Cambridge, U.K.: Cambridge Univ. Press, 2003, pp. 284–292.

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

[20] P. Cheng, X. Lian, L. Chen, J. Han, and J. Zhao, "Task assignment on multi-skill oriented spatial crowdsourcing," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 8, pp. 2201–2215, Aug. 2016.

[21] Z. Liu, K. Li, X. Zhou, N. Zhu, Y. Gao, and K. Li, “Multi-stage complex task assignment in spatial crowdsourcing,” *Inf. Sci.*, vol. 586, pp. 119–139, 2022.

[22] Y. Xie, Y. Wang, K. Li, X. Zhou, Z. Liu, and K. Li, “Satisfaction-aware task assignment in spatial crowdsourcing,” *Inf. Sci.*, vol. 622, pp. 512–535, 2023.

[23] L. Kazemi, C. Shahabi, and L. Chen, “Geotrucrowd: Trustworthy query answering with spatial crowdsourcing,” in *Proc. 20th Int. Conf. Adv. Geographic Inf. Syst.*, 2013, pp. 304–313.

[24] J. Tu, P. Cheng, and L. Chen, “Quality-assured synchronized task assignment in crowdsourcing,” *IEEE Trans. Knowl. Data Eng.*, vol. 33, no. 3, pp. 1156–1168, Mar. 2021.

[25] Z. Liu, K. Li, X. Zhou, N. Zhu, and K. Li, “Incentive mechanisms for crowdsensing: Motivating users to preprocess data for the crowdsourcer,” *ACM Trans. Sensor Netw.*, vol. 16, pp. 39:1–39:24, 2020.

[26] H. To, C. Shahabi, and L. Xiong, “Privacy-preserving online task assignment in spatial crowdsourcing with untrusted server,” in *Proc. Int. Conf. Data Eng.*, 2018, pp. 833–844.

[27] Q. Tao, Y. Tong, Z. Zhou, Y. Shi, L. Chen, and K. Xu, “Differentially private online task assignment in spatial crowdsourcing: A tree-based approach,” in *Proc. Int. Conf. Data Eng.*, 2020, pp. 517–528.

[28] M. Li et al., “Privacy-preserving batch-based task assignment in spatial crowdsourcing with untrusted server,” in *Proc. Conf. Inf. Knowl. Manage.*, 2021, pp. 947–956.

[29] Y. Tong, Z. Zhou, Y. Zeng, L. Chen, and C. Shahabi, “Spatial crowdsourcing: A survey,” *VLDB J.*, vol. 29, pp. 217–250, 2020.

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

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

[32] L. Tran, H. To, L. Fan, and C. Shahabi, “A real-time framework for task assignment in hyperlocal spatial crowdsourcing,” *ACM Trans. Intell. Syst. Technol.*, vol. 37, pp. 37:1–37:26, 2018.

[33] C. Liu, K. Li, K. Li, and R. Buyya, “A new service mechanism for profit optimizations of a cloud provider and its users,” *IEEE Trans. Cloud Comput.*, vol. 9, no. 1, pp. 14–26, First Quarter, 2021.

[34] S. Shekhar and J. S. Yoo, “Processing in-route nearest neighbor queries: A comparison of alternative approaches,” in *Proc. 11th ACM Int. Symp. Adv. Geographic Inf. Syst.*, 2003, pp. 9–16.

[35] J. S. Yoo and S. Shekhar, “In-route nearest neighbor queries,” *GeoInformatica*, vol. 9, pp. 117–137, 2005.

[36] 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. Int. Conf. Data Eng.*, 2020, pp. 13–24.

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

[38] D. Sun et al., “Online delivery route recommendation in spatial crowdsourcing,” *World Wide Web*, vol. 22, pp. 2083–2104, 2019.

[39] S. Ma, Y. Zheng, and O. Wolfson, “T-share: A large-scale dynamic taxi ridesharing service,” in *Proc. Int. Conf. Data Eng.*, 2013, pp. 410–421.

[40] K. Li, X. Tang, and K. Li, “Energy-efficient stochastic task scheduling on heterogeneous computing systems,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 11, pp. 2867–2876, Nov. 2013.

[41] D. Deng, C. Shahabi, and L. Zhu, “Task matching and scheduling for multiple workers in spatial crowdsourcing,” in *Proc. 20th Int. Conf. Adv. Geographic Inf. Syst.*, 2015, Art. no. 21.

[42] D. Deng, C. Shahabi, and U. Demiryurek, “Maximizing the number of worker’s self-selected tasks in spatial crowdsourcing,” in *Proc. 20th Int. Conf. Adv. Geographic Inf. Syst.*, 2013, pp. 314–323.

[43] J. Cordeau and G. Laporte, “The dial-a-ride problem (DARP): Variants, modeling issues and algorithms,” *Quart. J. Belg. Fr. Italian Operations Res. Societies*, vol. 1, pp. 89–101, 2003.

[44] Y. Tong, Y. Zeng, Z. Zhou, L. Chen, and K. Xu, “Unified route planning for shared mobility: An insertion-based framework,” *ACM Trans. Database Syst.*, vol. 47, pp. 1–48, 2022.

[45] Y. Tong, Y. Zeng, Z. Zhou, L. Chen, J. Ye, and K. Xu, “A unified approach to route planning for shared mobility,” in *Proc. VLDB Endow.*, vol. 11, no. 11, pp. 1633–1646, 2018.

[46] Y. Zeng, Y. Tong, Y. Song, and L. Chen, “The simpler the better: An indexing approach for shared-route planning queries,” in *Proc. VLDB Endow.*, vol. 13, no. 13, pp. 3517–3530, 2020.

[47] Y. Zeng, Y. Tong, and L. Chen, “Last-mile delivery made practical: An efficient route planning framework with theoretical guarantees,” in *Proc. VLDB Endow.*, vol. 13, no. 3, pp. 320–333, 2019.

[48] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York, NY, USA: W. H. Freeman & Co., 1979.

[49] C. Liu et al., “Spatio-temporal hierarchical adaptive dispatching for ridesharing systems,” in *Proc. 20th Int. Conf. Adv. Geographic Inf. Syst.*, 2020, pp. 227–238.

[50] V.V. Vazirani,, “Approximation algorithms,” in *Science Business Media*, Berlin, Germany: Springer, 2014.

[51] T. M. Goodrich and R. Tamassia, “The christofides approximation algorithm,” *Algorithm Des. Appl.*, vol. 363, pp. 513–514, 2015.

[52] 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/

[53] S. Shang, K. Deng, and K. Xie, “Best point detour query in road networks,” in *Proc. 11th ACM Int. Symp. Adv. Geographic Inf. Syst.*, 2010, pp. 71–80.

[54] X. Zhou, S. Liang, K. Li, Y. Gao, and K. Li, “Bilateral preference-aware task assignment in spatial crowdsourcing,” in *Proc. Int. Conf. Data Eng.*, 2022, pp. 1687–1699.

[55] S. Khuller, A. Moss, and J. S. Naor, “The budgeted maximum coverage problem,” *Inf. Process. Lett.*, vol. 70, pp. 39–45, 1999.

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

Yuan Xie is currently working toward the doctoral degree with the College of Computer Science and Electronic Engineering, Hunan University, Changsha, China. She is a visiting student with the National University of Singapore, Singapore. Her research interests include spatial crowdsourcing and urban computing.

Fan Wu received the MS degree in computer science from the University of Wuhan University, Wuhan, China, in 2004, and the PhD degree in computer science from the University of Hunan University, Changsha, China, in 2017. He is currently a vice professor with the College of Computer Science and Electronic Engineering, Hunan University. His research interests include parallel and distributed computing, machine learning, and DNA computing.

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

Wensheng Luo received the PhD degree from the College of Computer Science and Electronic Engineering, Hunan University, in 2022. He is a postdoctoral fellow with the School of Data Science, Chinese University of Hong Kong, Shenzhen. His research interests include parallel computing and data management/mining, and especially for the graph data.

Yifang Yin received the BE degree from the Department of Computer Science and Technology, Northeastern University, Shenyang, China, in 2011, and the PhD degree in computer science from National University of Singapore, Singapore, in 2016. She is currently a senior scientist with the Machine Intellection Department, Institute for Infocomm Research, A*STAR. Her research interests include machine learning, multimodal multimedia analysis, and spatiotemporal data mining.

Roger Zimmermann (Senior Member, IEEE) received the PhD degree in computer science from the University of Southern California, in 1998. He is a professor in computer science from the School of Computing, National University of Singapore (NUS), Singapore. His current research interests include streaming media and AR/VR architectures, dynamic adaptive streaming over HTTP (DASH), software defined networking (SDN), applications of machine/deep learning, mobile location-based services, spatial data management.

Keqin Li (Fellow, IEEE) is a SUNY distinguished professor of computer science with the State University of New York. He is also a national distinguished professor with Hunan University, China. His current research interests include cloud computing, fog computing and mobile edge computing, energy-efficient computing and communication, embedded systems and computer architectures and systems. He has authored or coauthored more than 860 journal articles, book chapters, and refereed conference papers, and has received several best paper awards. He is an AAIA fellow. He is also 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, the dean of the College of Computer Science and Electronic Engineering, Hunan University. His major research interests include parallel and distributed processing. He serves on the editorial board of the IEEE-TC.