Clustering based Priority Queue Algorithm for Spatial Task Assignment in Crowdsourcing

Yue Ma, Xiaofeng Gao, *Senior Member, IEEE,* Shahzad Sarwar Bhatti, and Guihai Chen, *Fellow, IEEE*

Abstract—Spatial crowdsourcing is an increasingly popular category in the era of mobile Internet and sharing economy, where tasks have spatio-temporal constraints and must be completed at specific locations. In this paper, we focus on *the **M**ulti-**O**bjective **S**patio-**T**emporal task assignment (MOST) problem* considering the worker heterogeneity in spatial crowdsourcing and model it as a combinatorial multi-objective optimization (MOO) problem with the goals of maximizing the overall task completion rate and minimizing the average task time cost. Finding the optimal global assignment turns out to be intractable since it does not simply imply optimality for an individual worker, as a typical nearest-neighbor heuristic generally does not render a satisfactory result. We prove that the problem is NP-hard. Subsequently, we formulate an efficient algorithm for the MOST problem — ***Ta**sk Clustering based **M**ixed **P**riority Queue Scheduling (TAMP)*. First, we improve the spectral clustering algorithm to evenly divide the task network into different subdomains according to tasks' geographical locations, considering the task clustering phenomena in real scenarios. We then design a mixed priority queue strategy considering the geographical influence and temporal urgency, to schedule workers finishing tasks in sequence. Experiments on synthetic and real datasets demonstrate the efficiency of our solution over other methods.

Index Terms—Spatial Crowdsourcing, Task Assignment, Worker Heterogeneity, Spectral Clustering, Queue Scheduling

1 Introduction

Crowdsourcing is simply the outsourcing of different tasks or work to a diverse group of individuals in an open call for the purpose of utilizing human intelligence [1]. A well-designed crowdsourcing system leverages the collective intelligence of the massive crowd workers to provide services and accomplish tasks cost-effectively, and thereby it has attracted extensive attention from both academia and industry [2] in recent years.

With the increasing pervasiveness of GPS-equipped smart mobile devices and decreased cost of wireless mobile network (e.g., 5G network), a new class of crowdsourcing has emerged, called *spatial crowdsourcing* (SC) [3]. Spatial crowdsourcing advances the potential of a crowd to perform tasks related to real-world scenarios involving physical locations, which were not feasible with conventional crowdsourcing methods. The main feature of spatial crowdsourcing is the presence of spatial tasks that require workers (with smartphones) to be physically present at a particular location for task fulfillment. Its natural connection with the physical world makes spatial crowdsourcing a computing paradigm for a broad spectrum of daily applications, such as real-time ride-hailing services (e.g., Uber) [4], product placement checking supermarkets [5], road condition monitoring [6], crowdsourcing-aided positioning [7], etc.

A representative spatial crowdsourcing model consists of three types of participants: *requesters* (clients), *workers* (the crowd), and *crowdsourcing platform* (server). The general framework of spatial crowdsourcing model for task assignment is shown in Fig. 1. The task publishers release tasks also known as human intelligence tasks (HITs), and then the worker requests these tasks. The crowdsourcing platform acts as a broker between the task publishers and the workers. The crowdsourcing platform aggregates the information of publishers and workers, then assigns tasks to suitable workers by the algorithm. In practice, a spatial crowdsourcing platform is the core of the system and often needs to manage massive tasks and workers every day.

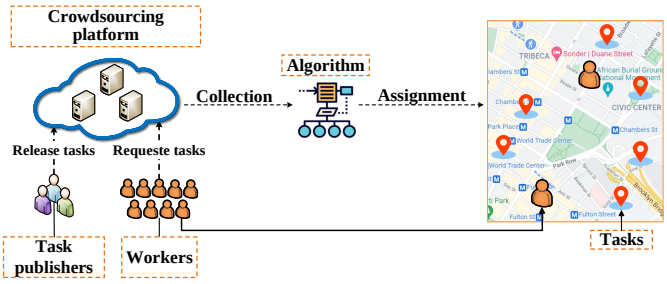


Fig. 1. A general framework of a spatial crowdsourcing model.

Thus, the major challenge of the spatial crowdsourcing platforms is how to assign the large-scale tasks to their workers, i.e., *task assignment*. In addition, most of the existing studies focus on task assignment based on the whole study area [8], [9], [10], [11], [12], [13].

The platforms usually aim to arrange the tasks to suitable workers with different optimization objectives, such as maximizing the total number of assigned tasks or the full payoff of the tasks to their assigned workers, minimizing the total traveling costs of the allocated workers. The objective is generally determined based on real needs and constraints. For example, one common challenge in spatial crowdsourc-

- *Y. Ma, X. Gao, and G. Chen are with MoE Key Lab of Artificial Intelligence, Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai 200240, China. E-mail: ma_yue@sjtu.edu.cn, {gao-xf, gchen}@cs.sjtu.edu.cn*
- *S.S. Bhatti is with Department of Information Sciences, Division of Science and Technology, University of Education, Lahore, Pakistan. E-mail: shahzad.sarwar@ue.edu.pk*
- *X. Gao is the corresponding author.*

ing is that the tasks reachable by each worker highly depend on the distance between origin and destination as well as the tightness of deadline, which have to be treated carefully in constructing the task assignment algorithm.

Therefore, designing an efficient assignment mechanism is of paramount importance for the SC platform, which could improve the system efficiency by increasing the income of workers and saving the cost of the platform. Based on the basic problem characteristics, task assignment in SC can be classified into two different categories: *task matching* and *task scheduling*. Task matching provides guidance on which tasks to perform: the assignment mechanism tries to match a set of tasks to workers. Task scheduling provides a plan (or order) to perform tasks located at different places: the assignment mechanism schedules the order of tasks for the workers. The problem of task scheduling is unique to spatial crowdsourcing.

Geographic information is of vital importance in the field of SC and is a necessary condition to allocate tasks in the spatial dimension. As the task allocation problem is NP-hard in its general form [14], it is easier to obtain an accurate solution by dividing a complex spatial problem into multiple sub-problems based on geographical information. However, most of these studies ignored the temporal information of workers and tasks, and thus do not apply readily to an SC application. Niu et al. [15] propose a pricing model based on the distance information and the number of workers, but the expiration time of tasks is not considered.

In addition, in real crowdsourcing situations, it is often observed the phenomena of *task clustering*, i.e., most of the tasks are concentrated in a few regions in the space instead of being distributed uniformly. The reason for this phenomenon is that community structures are quite common in real networks [16]. For example, in the take-out scene, the locations of tasks are often concentrated in densely populated areas, such as schools or office spaces. However, in sparsely populated places or suburbs, the distribution of tasks is relatively sparse. If let workers select tasks by their preferences, workers will only choose to complete tasks that are closer, resulting in remote tasks that have not been responded to forever, thereby affecting the overall task completion rate.

Based on the observation, we introduce *spectral clustering* [17], a graph clustering algorithm in our integrated algorithm to partition the task network. Besides, we improve the spectral clustering algorithm by applying θ-sparseness to reconstruct the affinity matrix for the reasonable time complexity and enhancing the fairness of subdomain division. In view of the divide-and-conquer idea, we first divide the task network into different subdomains based on their locations in space and allocate workers to the corresponding subdomains to finish tasks.

Compared to the previous work, the hardness of our problem lies in that, once the traveling cost associated with moving to tasks' locations, the expiration time of tasks, and the heterogeneity of workers are taken into account, the locally optimal assignment does not guarantee global optimality. In other words, assigning the most jobs to each worker does not necessarily imply the maximum number of accomplished tasks by all workers.

To the best of our knowledge, a unified assignment mechanism considering spatio-temporal constraints of tasks, worker heterogeneity, and task clustering, with multiple objectives in the multi-user dynamic environment in SC systems has not been all probed together, so far. In summary, we make the following contributions:

- To promote the overall performance of task assignment, the task queue scheduling problem is modeled as a multi-objective joint optimization problem, *the **M**ulti-**O**bjective **S**patio-**T**emporal task assignment (MOST) problem*, which focuses on maximizing the overall task completion rate and minimizing the average task time cost, and considers the worker heterogeneity simultaneously. Besides, we prove the problem is NP-hard.
- Considering clustering phenomena of spatial crowdsourcing tasks, spectral clustering is introduced to divide tasks into subdomains. At the same time, in order to better learn the spatial relationships between tasks and reduce the complexity of the algorithm, we creatively reconstruct the affinity matrix by θ-*sparseness* method to improve spectral clustering.
- The proposed algorithm, ***Ta**sk Clustering based **M**ixed **P**riority Queue Scheduling (TAMP)* algorithm, integrates two critical objectives, the temporal constraints and the spatial information of spatial tasks, into one joint metric and uses it to make decisions on sequences of task execution. The choice of the combined weight of these two metrics is dynamic and investigated in the design.
- Extensive experiments on both synthetic and real data are performed to compare the proposed scheme with different comparative techniques, and the results show that the new scheme outperforms others.

This paper extends the initial study [18], via (i) surveying up to date literature and summarizing the comparison of task assignment models in Sec. 2; (ii) reformulating the combinatorial multi-objective optimization problem and proving it as a NP-hard problem in Sec. 3; (iii) designing a sparseness method to improve the spectral clustering efficiency; (iv) adding more explanations for model framework and the time complexity analysis for algorithm in Section 3.2.3; (v) updating existing figures and add more diagrams; (vi) improving the organization and presentation of the paper by a major revision and careful proofreading.

2 Related Work

Spatial Crowdsourcing (SC) can be deemed as one of the main enablers to employ smart device carriers as workers to move to some specified locations and perform location-based tasks physically [38]. A recent survey on spatial crowdsourcing is [39], which reviews the existing research on major algorithmic issues such as task assignment, quality control, incentive mechanism design, and privacy protection. Moreover, the first challenge of the spatial crowdsourcing platforms is task assignment, which is the basis for other research studies.

In view of the task publishing mode, SC can be classified into two categories, namely Worker Selected Tasks (WST) mode and Server Assigned Tasks (SAT) mode [3]. WST

TABLE 1
Taxonomy and analysis of research concerning task assignment in spatial crowdsourcing

Ref	Mode	Objective	Constraints				Assignments		Algorithm
			Spatial	Time	Worker Heter	Others	Matching	Scheduling	
[19]	WST	maximize the overall satisfaction	✓	✓	✓	✓	✓		Greedy algorithm
[20]	WST	maximize the number of assigned tasks	✓	✓	✓	✓	✓		HCTD and Greedy algorithm
[21]	SAT	maximize the sum of the rewards	✓	✓		✓		✓	Approximation algorithm
[22]	SAT	maximize the sum of the rewards	✓	✓			✓		Greedy algorithm
[23]	SAT	maximize overall user satisfaction	✓	✓	✓	✓	✓		Greedy / Game algorithm
[24]	WST	maximize the task quality and minimize the incentive budget	✓	✓		✓		✓	Heuristic algorithm
[25]	SAT	maximize the number of assigned tasks	✓	✓		✓	✓		Heuristic algorithm
[26]	SAT	maximize the total payoff	✓	✓	✓	✓	✓		Greedy / Game algorithm
[27]	SAT	maximize the expected total utility			✓	✓	✓		Approximation algorithm
[28]	SAT	maximize task reliability for dynamic task assignment	✓	✓		✓	✓		Approximation algorithm
[29]	SAT	maximize platform profit considering worker	✓	✓		✓		✓	Greedy algorithm
[30]	SAT	maximize the expected quality	✓			✓	✓		Heuristic algorithm
[31]	SAT	maximize the spatial/temporal coverage	✓	✓			✓		Approximation / Heuristic algorithm
[32]	SAT	maximize the diversity score of assignment	✓	✓		✓	✓		Approximation algorithm
[33]	SAT	maximize the total number of task assignments	✓	✓		✓	✓		Greedy algorithm
[34]	SAT	minimize the overall cost	✓			✓	✓		Approximation algorithm
[35]	SAT	maximize the expectation of the number of answered tasks	✓	✓		✓		✓	Greedy algorithm
[36]	SAT	maximize the number of assigned tasks	✓			✓	✓		Heuristic algorithm
[37]	SAT	maximize the social surplus	✓	✓	✓	✓		✓	Game-theoretic incentive mechanism
[10]	SAT	maximize the number of assigned tasks and minimize the average task number difference	✓	✓		✓		✓	Reinforcement learning

mode gives workers the right to directly select the tasks based on their own preference without coordination with the server [19], [20], while SAT mode requires the server to assign tasks to the interested workers based on the system optimization goals [21], [22], [23]. In WST mode, no specific task allocation algorithm is required, and it suffices for the platform to receive and process the orders of the workers. The users explore the optimal task assignment to befit their own instead of the platform. One drawback of this mode is that the SC server has no control over task allocation. This may result in some spatial tasks never being assigned, while others may be assigned redundantly.

Another drawback of WST is that workers choose tasks based on their own objectives (e.g., choosing the closest spatial tasks to minimize their travel cost), which is not necessarily the ultimate objective of the SC-server (i.e., maximizing the overall task assignment). Besides, incentive mechanisms have been widely used in WST mode. Wang et al. [24] study a worker incentive model combined with both a genetic algorithm and an ant colony optimization algorithm to maximize the task completion quality while minimizing the incentive budget in the whole area. Zhu et al. [40] propose Incentive-aware Task Location (ITL) for a location-unspecific task with a fixed budget, the aim of which is to maximize the number of workers who are willing to participate in the task. And the work proposes three heuristic methods to solve it, including even clustering, uneven clustering, and greedy location methods.

In SAT mode, the server of the crowdsourcing platform assigns tasks to nearby workers usually based on the system optimization goals such as maximizing the number of assigned tasks after collecting all the locations of workers [25], maximizing the total payoff from assigned tasks [26], maximizing the expected total utility achieved by all workers [27], maximizing task reliability for dynamic task assignment [28], maximizing platform profit considering worker utilities simultaneously [29], maximizing the expected quality of results from workers by a real-time budget-aware task package allocation [30], or maximizing the spatial/temporal coverage where/when workers perform tasks [31].

Most existing studies adopt the SAT mode, where an SC server takes charge of the task assignment process. For example, Cheng et al. [32] propose a reliable diversity-based spatial crowdsourcing (RDB-SC) problem in SC, where an SC server assigns tasks to suitable workers in order to maximize the diversity score of assignments. Zhao et al. [33] propose a preference-based task assignment problem and design a tensor-decomposition-based algorithm to learn worker preferences, after which the assignment problem is transformed into a Minimum Cost Maximum Flow (MCMF) problem. However, they all assume that each worker can only perform tasks in a specific spatial region, while we do not exert in our model a hard constraint on the working area. Therefore, these works have a much smaller search space in their problem settings compared to ours.

Moreover, within the SAT publishing mode, tasks assignment can be further classified into two different modes: Single Task Assignment (STA) mode and Redundant Task Assignment (RTA) mode [3]. STA mode assumes that all the workers are trusted and can perform the tasks correctly without any malicious intentions so that each task is only assigned to one worker in STA mode. However, there inevitably exist some malicious workers that might intentionally complete tasks incorrectly. Therefore, RTA mode is proposed to improve the validity of task completion by assigning each task to several nearby workers. In RTA mode, the task completion result with the majority vote is regarded as correct [34], [35].

Among the above studies in SC, traveling cost is critical, due to the fact that SC workers have to physically move to the locations of spatial tasks in order to perform them [36], [37]. For instance, considering task localness, which refers to workers' preferences based on their traveling cost (i.e., workers are more likely to accept nearby tasks), [36] proposes an effective task assignment framework by modeling task acceptance rate as a decreasing function of travel distance. Cheung et al. [37] formulate the interactions among

users as a non-cooperative Task Selection Game (TSG), and propose an Asynchronous and Distributed Task Selection (ADTS) algorithm that balances the rewards and traveling costs of the workers for completing tasks.

In task assignment, modeling the assignment score as the shortest path in visiting the locations of multiple tasks becomes similar to the traveling salesman problem (TSP) and vehicle routing problem (VRP) [41]. Since there is only one worker in TSP, here we discuss VRP. Different variants of VRP have been studied [42], [43], still, there are differences between our task assignment problem and these variants. Compared with VRP, our goal is to maximize the overall task completion rate and minimize the average task time cost simultaneously, whereas VRP aims to minimize the total traveling time of all workers. Besides, in VRP, all workers start from the same location, whereas, in our setting, workers have different initial locations.

The latest model is extended to multiple workers in [10], which is the closest related work to our study. In that paper, the authors propose a Task Allocation with Geographic Partition (TAGP) framework for the Multi-Center-based Task Allocation problem (MCTA), which aims to maximize the allocated task number and achieve the allocation fairness among workers. More specifically, the work first utilizes a Voronoi diagram mechanism to decompose a complex multi-center graph into multiple smaller single-center-based graphs and then adopts a Reinforcement Learning method to allocate tasks by transforming the task allocation problem into a multiple traveling salesman problem (MTSP). The idea is similar to our work, which is to divide the whole area first and then assign tasks for different subdomains. However, this work still transforms the task assignment problem into a MTSP inconsistent with facts in spatial crowdsourcing, because the center of a graph assumed by the work does not exist in practice. Besides, worker heterogeneity is not taken into account.

Our problem, which is discussed in this paper, is a version of the task assignment problem considering traveling time cost and worker heterogeneity in STA mode.

TABLE 2
Summary of Symbols and Notations

Category	Symbols	Definitions
Attributes	s_i	The i^{th} spatial task
	w_k	The k^{th} worker
	ls_i, lw_k	Location of s_i, w_k
	e_i	Expiration time of s_i
	e_{w_k}	Deadline of w_k
	v_k, p_k	Traveling / Processing speed of w_k
Sets	S	Set of all tasks
	S_A	Set of accomplished tasks
	W	Set of workers $\{w_k\}$
	A	A task assignment strategy
	S_A^k	The achievable task set (ATS) for w_k
	Ω_k	The k^{th} cluster
Parameters	α	Weight of time
	θ	Weight of matrix sparseness
Calculation	δ	The overall task completion rate
	τ	The average task time cost
	$a_{k,i}$	The arrival time of w_k at ls_i
	d_i^k, $d_{i,j}$	Distance between s_i and w_k/s_j
	t_{p_k}	The total processing time of w_k
	d_{w_k}	The traveling distance of w_k
	Ψ_k	Task queue of w_k
	$\xi_k^{(t)}(i)$	Temporal priority of s_i in $\Psi_k^{(t)}$
	$\xi_k^{(d)}(i)$	Spatial priority of s_i in $\Psi_k^{(d)}$
	$\xi_k^{(m)}(i)$	Mixed priority of s_i in $\Psi_k^{(m)}$

3 The Proposed Scheme

In this section, we firstly present our model architecture and give a formal statement of the **M**ulti-**O**bjective **S**patio-**T**emporal task assignment (MOST) problem. Then, we explain each part of the proposed scheme in detail.

3.1 Model Architecture and Problem Statement

Here we investigate a kind of task assignment mechanism under the above spatial crowdsourcing model with Single Task Assignment (STA) mode, referred to as *single spatio-temporal task assignment*. Specifically, given a user's current location, the platform aims to find an optimal assignment between tasks and workers such that the overall task completion rate is maximized and the average task time cost rate is minimized. In particular, we note that the task assignment is actually made up of two sub-problems: 1) for each task, we need to assign it to a suitable worker; and 2) for each worker, we need to schedule a sequence that each worker follows to perform the assigned tasks. (The list of involved notations is given in Table 2.)

Before presenting our problem, we first formally define the spatial tasks and the workers in spatial crowdsourcing.

Definition 1 (Spatial Task)**.** *A spatial task s_i is characterized by a 2-tuple $s_i = \langle ls_i, e_i \rangle$, which implies that the task s_i is located at ls_i, and will expire at time e_i. ls_i is a position in 2D space expressed as its coordinate (x_i, y_i).*

For simplicity and without loss of generality, most studies assume that the processing time of each task is 0 and the workers' speeds are the same, but we consider the worker heterogeneity in our model.

Definition 2 (Worker)**.** *A worker w_k, characterized by a 4-tuple $w_k = \langle lw_k, v_k, p_k, e_{w_k} \rangle$, is a carrier of a mobile device who volunteers to perform spatial tasks. A worker can be in either online or offline mode. A worker is offline when she is unable to perform tasks and is online when she is ready to accept tasks. An online worker is associated with her current location $lw_k = (x_k, y_k)$, the traveling speed and processing speed of which are v_k and p_k respectively. In addition, she has to return to her initial departure location lw_k^0 before her deadline e_{w_k}.*

In spatial crowdsourcing, the query of a spatial task s_i can be answered only if a worker w_k is physically located at that location ls_i. Therefore, considering the expiration time of task s_i and the worker's deadline, it can be completed only if a worker w_k arrives at ls_i finishing the task before its expiration time e_i and returning back to her initial departure lw_k^0 before her deadline e_{w_k}, which implies the constraint

$$\begin{aligned} a_{k,i} + \frac{1}{p_k} &\le e_i, \\ a_{k,0} = a_{k,i} + \frac{1}{p_k} + \frac{d(ls_i, lw_k^0)}{v_k} &\le e_{w_k}, \end{aligned} \tag{1}$$

where $a_{k,i}$ is the arrival time of w_k at the location ls_i for task s_i, $a_{k,0}$ is the time w_k back to departure lw_k^0, $\frac{1}{p_k}$ is the time of w_k processing task s_i, and $d(s_i, lw_k^0)$ is the distance between task s_i and the initial location of w_k.

Note that in the STA mode, the platform can assign every spatial task to one worker only. Once worker w_k is online, she sends a task inquiry to the server, which includes her current location lw_k. The server will take all the available tasks and workers at the particular time instance into account and return a task sequence to w_k.

Let t denote the current time. The distance $d_i^k(t)$ between worker w_k and task s_i at t is calculated as their Euclidean distance, i.e.,

$$d_i^k(t) = \|ls_i - lw_k\|_2 = \sqrt{(x_i - x_k)^2 + (y_i - y_k)^2}, \quad (2)$$

where ls_i and lw_k are respectively the location of task s_i and worker w_k at the moment t.

The maximum allowable remaining time for task s_i is determined based on the time left for the task before its expiration time, given by

$$t_i = e_i - t. \quad (3)$$

Due to the foregoing descriptions, we formally formulate the problem statement as follows.

Problem Statement (MOST problem): In a model of spatial crowdsourcing containing a crowdsourcing platform, plenty of task publishers, tasks, and workers, how does the crowdsourcing platform with STA mode simultaneously consider the spatio-temporal interactive information of tasks and the heterogeneity of workers to assign tasks to suitable workers, so as to maximize the number of accomplished tasks and reduce traveling cost of workers?

Optimal Objective: To better analyze and solve the problem, we formalize it as a multiple-objective joint optimization problem, *MOST problem*, which has two optimal goals: maximizing the overall task completion rate δ and minimizing the average task time cost τ simultaneously.

Because a task only can be assigned to a suitable worker, then we assume $x_{k,i} = 1$ if worker w_k complete task s_i, otherwise $x_{k,i} = 0$. Let $S = \{s_1, s_2, \dots\}$ be the set of all tasks, and S_A denote the set of tasks that are accomplished by the task assignment strategy A. Obviously, $S_A \subseteq S$. Thus, the maximization of the overall task completion rate δ can be expressed as:

$$\max \delta = \frac{|S_A|}{|S|} = \frac{\sum x_{k,i}}{|S|}, \quad (4)$$

subject to:

$$\begin{aligned} &\sum_{k=1}^{|W|} x_{k,i} = 0, \text{ or } 1, \quad \forall\, i = 1, 2, \cdots, |S|, \\ &x_{k,i} \cdot a_{k,i} + \frac{1}{p_k} \le e_i, \\ &x_{k,i} \cdot a_{k,i} + \frac{1}{p_k} + \frac{d(ls_i, lw_k^0)}{v_k} \le e_{w_k}, \\ &x_{k,i} = 0, \text{ or } 1. \end{aligned} \quad (5)$$

Similarly, the minimization of average task time cost τ for accomplished tasks is expressed as:

$$\begin{aligned} \min \tau &= \frac{\sum_k \frac{d_{w_k}}{v_k} + \sum_k t_{p_k}}{\sum x_{k,i}} \\ &= \frac{\sum_k \frac{d_{w_k}}{v_k} + \sum_k \frac{\sum_i x_{k,i}}{p_k}}{\sum x_{k,i}}, \end{aligned} \quad (6)$$

subject to:

$$\begin{aligned} &\sum_{k=1}^{|W|} x_{k,i} = 0, \text{ or } 1, \quad \forall\, i = 1, 2, \cdots, |S|, \\ &x_{k,i} \cdot a_{k,i} + \frac{1}{p_k} \le e_i, \\ &x_{k,i} \cdot a_{k,i} + \frac{1}{p_k} + \frac{d(ls_i, lw_k^0)}{v_k} \le e_{w_k}, \\ &\frac{d_{w_k}}{v_k} + t_{p_k} \le e_{w_k}, \\ &x_{k,i} = 0, \text{ or } 1. \end{aligned} \quad (7)$$

Here, d_{w_k} is the traveling distance of worker w_k, and t_{p_k} is the time spent on processing tasks of worker w_k.

The *MOST problem* can be proved to be NP-hard by reduction from the Maximum Coverage (MC) problem. In the following, we give the definition of the achievable task set for subsequent proof and then prove the *MOST problem* as NP-hard.

Definition 3 (Achievable Task Set (ATS))**.** *A task set S_A^k is called an achievable task set (ATS) for a worker w_k, if there exists a task assignment strategy A^k, such that,*

- *all the tasks of S_A^k can be completed before their respective expiration time, i.e., $a_{k,i} + \frac{1}{p_k} \le e_i$ for each $s_i \in S_A^k$, and*
- *worker w_k can return back to departure on time after completing all tasks S_A^k, i.e., $a_{k,0} \le e_{w_k}$ for each $s_i \in S_A^k$.*

Lemma 1. *MOST problem is NP-hard.*

Proof. We first introduce the Maximum Coverage (MC) problem, which is proven to be NP-hard [44]. Given a collection of sets $\mathbb{R} = \{R_1, R_2, \cdots, R_K\}$ over a set of objects Ω, where $R_i \subseteq \Omega$, and a positive integer l, the MC problem is to find a subset $\mathbb{R}' \subseteq \mathbb{R}$ such that $|\mathbb{R}'| \le l$ and the number of covered elements by $\mathbb{R}'$ is maximized.

Getting the ATSs from a given task set $S = \{s_1, s_2, \dots\}$ for worker set $W = \{w_1, w_2, \dots, w_K\}$, e.g., $\mathbb{S} = \{S_{A_1}^1, S_{A_2}^1, \dots, S_{A_1}^2, S_{A_2}^2, \dots, S_{A_1}^K, S_{A_2}^K, \dots\}$, our *spatio-temporal task assignment problem* is actually to find the subset $\mathbb{S}' \subseteq \mathbb{S}$ such that $|\mathbb{S}'| \le K$ and the number of covered tasks by $\mathbb{S}'$ (e.g., $|\bigcup_{S_{A_i}^k \in \mathbb{S}'} S_{A_i}^k|$) is maximized, where each subset in $\mathbb{S}'$ belongs to one worker in W.

Consider a special case of *MOST problem*, where each worker has only one ATS, e.g., $\mathbb{S} = \{S_A^1, S_A^2, \dots, S_A^K\}$. The goal of our problem is to select at most n number of ATSs from $\mathbb{S}'$, such that the total number of tasks is maximum. That is, solving the maximum coverage problem is equal to finding a subset of the ATSs, $\mathbb{S}' \subseteq \mathbb{S}$, where $|\mathbb{S}'| (\le K)$ maximizes $|\bigcup_{S_A^k \in \mathbb{S}'} S_A^k|$.

Again, S_A^k for each worker $w_k \in W$ corresponds to each set R_k of the MC problem, where each task $s_i \in S$ corresponds to the object $\omega_i \in \Omega$, and the number of

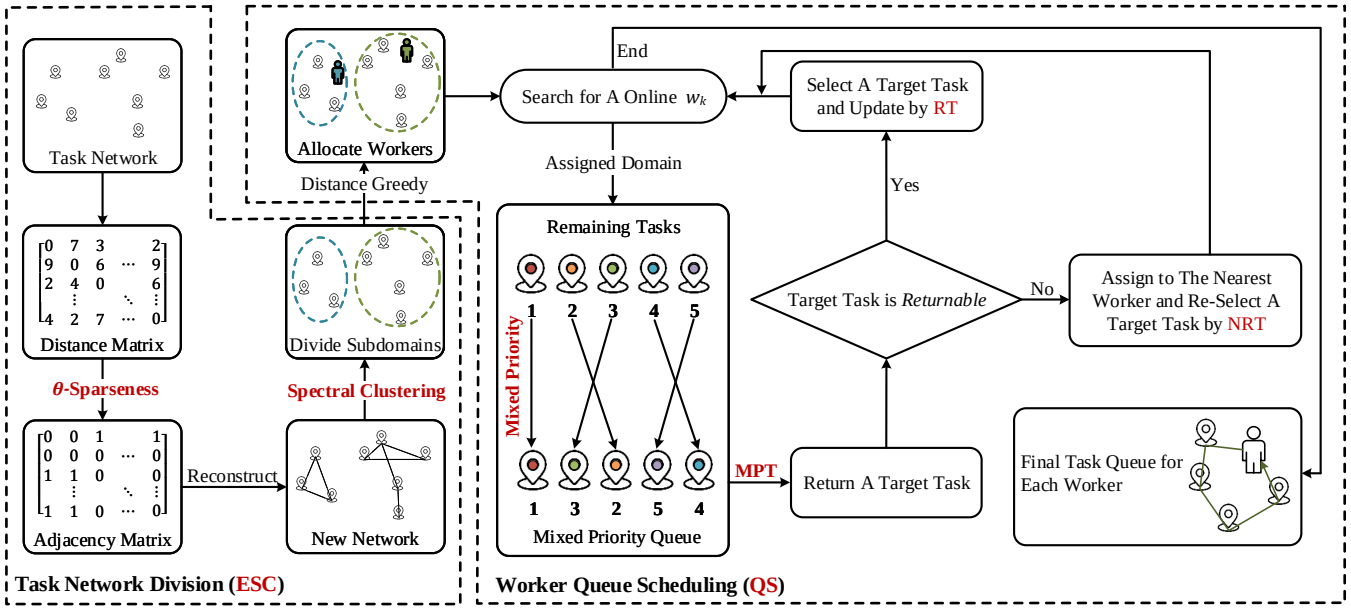


Fig. 2. The framework of TAMP algorithm with two parts: task network division and worker queue scheduling.

workers (i.e., K) corresponds to the positive integer l in the MC problem. Therefore, finding a subset $\mathbb{S}'$ form $\mathbb{S}$ (where $|\mathbb{S}'| \leq K$) with maximized $|\bigcup_{S_A^k \in \mathbb{S}'} S_A^k|$ is equivalent to solving the maximum coverage problem. Therefore, *MOST problem* with single ATS for each worker (i.e., the above special case) is NP-hard and *MOST problem* with multiple ATSs for each worker is also NP-hard. □

3.2 TAMP: Task Clustering based Mixed Priority Queue Scheduling

Since the *MOST problem* is NP-hard, a simple greedy algorithm is to use the maximum achievable task set for each worker as the assignment result. This can hardly be a satisfying result since multiple workers may be assigned the same set of tasks which may leave more tasks unassigned. In this paper, we propose the spectral clustering based scheme, ***Ta**sk Clustering-based **M**ixed **P**riority Queue Scheduling (TAMP)*, which works in the above problem setting for task assignment.

The whole framework of TAMP algorithm is shown in Fig. 2, with two parts: task network division and worker queue scheduling. First, TAMP initializes the network, and reconstructs the network by θ-sparseness. Then, enhanced spectral clustering divides the task network into subdomains according to tasks' geographical locations by Enhanced Spectral Clustering (ESC). Next, tasks of each subdomain are allocated to corresponding workers. Moreover, the task queue for a worker is rearranged by a mixed metric incorporating geographical location information as well as the task's temporal emergency. Finally, return the target task that the worker needs to accomplish in the next moment by Mixed Priority Task (MPT), which calls two sub-algorithms — Returnable Task (RT) and Not-Returnable Task (NRT). Finally, schedule workers to accomplish those tasks by the final task queue through Queue Scheduling (QS) algorithm.

3.2.1 Task Network Division

In order to group the network of tasks into subdomains, the spectral clustering algorithm is adopted to divide the network into subareas $\{\Omega_k\}, k \in \{1, \cdots, |W|\}$. Every Ω_k has a designated worker, who is mainly responsible for all tasks located inside.

To apply spectral clustering, the key step is to learn the affinity matrix to measure the similarity among data points. In the paper, we apply θ-sparseness to sparse the distance matrix for reconstructing the affinity matrix. Here we reconstruct the affinity matrix by matrix sparsification for two main reasons: 1) the spectral clustering algorithm needs to calculate the eigenvectors and eigenvalues of the affinity matrix, and the sparse processing could reduce the computational complexity in order to study the spatial relationships between tasks within a reasonable time; 2) the sparse processing saves the task information in a much closer neighborhood, and the subsequent subdomain division can divide the subarea as fairly as possible.

We first calculate the geographical distance matrix G, where $G_{i,j}$ is the Euclidean distance between each pair of tasks s_i and s_j in 2D space:

$$G_{i,j} = \|ls_i - ls_j\|_2 = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}. \quad (8)$$

Obviously, the distance matrix elements should be non-negative. Simply introducing the matrix into the spectral clustering algorithm does not impose any constraints on the graph sparsity, which will lead to expensive computing costs and might introduce noise (i.e., unimportant edges). Besides, it is not sparse enough that the spectral clustering algorithm cannot focus on the more proximity tasks.

Therefore, we extract the sparse non-negative adjacency matrix M from G by considering only the node pair with a much closer distance. To make the hyperparameter of the extraction threshold insensitive and not destroy the graph's sparsity distribution, we adopt a relative ranking strategy for the entire graph. Specifically, we mask off (i.e., set to

Algorithm 1: Enhanced Spectral Clustering (ESC)

Input: Tasks set $S = \{s_1, \cdots, s_n\}$, workers set W, θ.
Output: Task clusters $\{\Omega_l\}$

$k \leftarrow |W|$;
Affinity matrix $B \in \mathbb{R}^{n \times n}$ is calculated by Eq. (10);
$D \leftarrow diag(sum(B))$; //Degree matrix
$L \leftarrow D - B$; //Laplacian matrix
$L_s \leftarrow D^{-\frac{1}{2}} L D^{-\frac{1}{2}}$; //Normalized Laplacian matrix
Let $\mu_1, \mu_2, \cdots, \mu_k$ be the k smallest eigenvalues of L_s and $h_1, h_2, \cdots, h_k$ the corresponding eigenvectors; //Chosen to be orthogonal
for $i \leftarrow 1$ *to* k **do**
  $h_i \leftarrow \frac{h_i}{\|h_i\|_2}$; //Normalized
$H \leftarrow [h_1, h_2, \cdots, h_k]$;
for $i \leftarrow 1$ *to* n **do**
  for $j \leftarrow 1$ *to* k **do**
    $Y_{ij} = H_{ij} / (\sum_j H_{ij}^2)^{\frac{1}{2}}$; //Normalize H
Treat each row of Y as a point in $\mathbb{R}^k$, cluster them into clusters by K-means; //Minimize distortion
for *each* $s_i \in S$ **do**
  if Y_{i*} *is assigned to cluster* l **then**
    Assign s_i to task subdomian Ω_l;
return Clusters $\Omega_1, \Omega_2, \cdots, \Omega_k$ with $\Omega_l = \{s_i | y_i \in domain(\Omega_l)\}$

zero) those elements that are larger than a non-negative threshold, obtained by ranking the metric value in G. The adjacency matrix can be reconstructed by θ-*sparseness*

$$M_{ij} = \begin{cases} 1, & G_{ij} \leq \mathbf{Rank}_{\theta n}(G_{i*}), \\ 0, & otherwise, \end{cases} \tag{9}$$

where $\mathbf{Rank}_{\theta n}(G_{i*})$ returns the θn-th smallest value in ith row of distance matrix G, n is the number of nodes, and θ controls the overall sparsity of the generated graph.

It should be noticed that A is not necessarily symmetric based on the definition of the connectivity. In order to obtain a symmetric affinity matrix required in spectral clustering algorithm, we define the affinity matrix B as below:

$$B = \frac{1}{2}(M + M^{\mathrm{T}}). \tag{10}$$

It is different from the traditional sparse method that introduces KNN algorithm to calculate the affinity matrix [45]. The latter sets an absolute number threshold to select the neighbors by distance matrix. In our method, the hyperparameter θ could control the sparsity of the newly generated graph, and the number of removed elements could vary with the size of the graph.

As the value of parameter θ largely influences the obtained clusters, we need to carefully regulate the value of θ to make the number of tasks in each cluster more even, which considers the fairness for workers. The choice of θ will be discussed later in Section 4.

According to the above process, we can rebuild a new affinity matrix, and divide the task network into different subdomains by spectral clustering algorithm as shown in Algorithm 1 — Enhanced Spectral Clustering (ESC). Then we begin to consider how to assign a suitable worker to the corresponding subdomain and schedule the worker to accomplish assigned tasks.

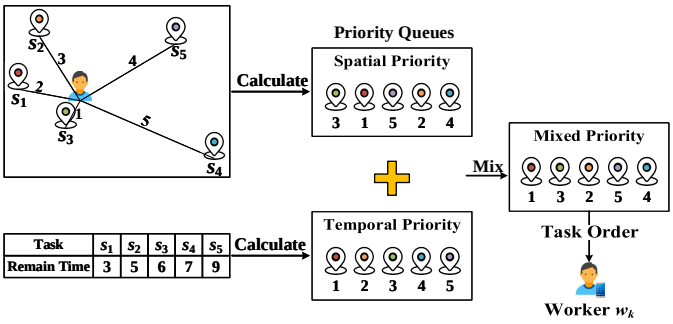


Fig. 3. Queue scheduling process for a single worker w_k.

3.2.2 *Worker Queue Scheduling*

At first, we note the center of the subdomain Ω_k as $(\overline{x}_k, \overline{y}_k)$, which is simply given by

$$(\overline{x}_k, \overline{y}_k) = \left(\frac{1}{|\Omega_k|} \sum_{s_i \in \Omega_k} x_i, \frac{1}{|\Omega_k|} \sum_{s_i \in \Omega_k} y_i \right). \tag{11}$$

Additionally, we set the number of clusters as $|W|$ when dividing the clusters. Thus, the tasks in every subdomain could be assigned to a specific worker, because the number of subdomains is equal to the number of workers, which is shown in the following part.

Here, we need to sort the subdomains $\{\Omega_k\}$ by their size $|\Omega_k|$, and the subdomain containing more tasks needs to be prioritized by the nearest worker for the reason that the more tasks in the subdomain, the less traveling cost need to be paid in the domain to ensure that more tasks are completed. Then the worker who is nearest to the subdomain center is allocated the tasks in the subdomain.

As shown in Fig. 3, the remaining assigned tasks for a single worker w_k could be formed into a task queue Ψ_k, which will be rearranged by the mixed priority strategy considering both her geographical distance to the task and the task's temporal emergency. This scheme will schedule the worker to accomplish the corresponding target task with higher mixed priority.

In the current task queue for some worker w_k, we denote the geographical distance of the nearest task by d_{min}^k and the furthest task by d_{max}^k. In order to make the distance comparable among workers, we first normalize the distances from w_k towards different tasks by defining her *spatial priority* of each task s_i as:

$$\xi_k^{(d)}(i) = \frac{d_i^k - d_{min}^k}{d_{max}^k - d_{min}^k + \epsilon}, \tag{12}$$

where ϵ is a very small number to prevent potential overflow due to division by zero, which is set to 10^{-6} by default.

Similarly, we define the temporal emergency of the task s_i for some worker w_k as the maximum allowable remaining traveling time t_i. Let t_{min} denote the remaining time of the

most urgent task and t_{max} denote the least. The *temporal priority* of task s_i can be calculated as:

$$\xi_k^{(t)}(i) = \frac{t_i - t_{min}}{t_{max} - t_{min} + \epsilon}. \tag{13}$$

To incorporate both metrics to evaluate the importance of a task s_i for w_k, a joint *mixed priority* $\xi_k^{(m)}(i)$ is obtained by combining spatial and temporal priorities:

$$\xi_k^{(m)}(i) = \alpha \xi_k^{(d)}(i) + (1-\alpha)\xi_k^{(t)}(i), \tag{14}$$

where parameter α balances weights of spatial and temporal priorities, which is between 0 and 1. The influence of α will be discussed and studied in Section 4 through experiments.

Generally, tasks with smaller values of mixed priorities, $\xi_k^{(m)}(i)$, are given higher priorities to be served first. The results of the algorithm proposed in this paper are quite different from those based solely on time constraints or spatial information. Here we give an example to explain the utility of the mixed priority strategy.

Example 1 (Utility of the mixed priority strategy)**.** *Fig. 4 shows the spatial and temporal information of worker w_1 and tasks s_1∼s_4. Besides, let w_1 could finish processing 4 tasks at one time unit, and the traveling speed is 2.*

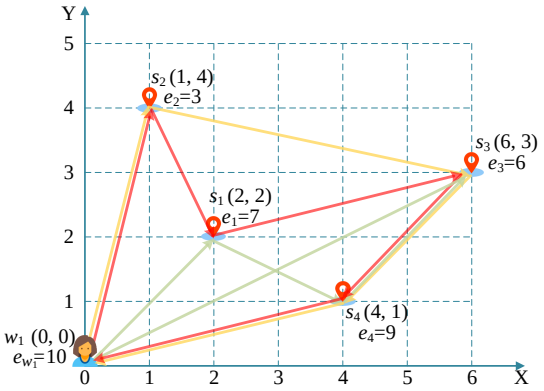


Fig. 4. Example 1 for a worker under different priority strategies.

The lines with arrowheads show the routes for w_1 under different priority strategies. w_1 will first go to process s_2, then s_3, s_4, and back to initial location considering temporal priority only (see the yellow route shown in Fig. 4), in which w_1 could not finish task s_1 before its expiration time (w_1 accomplish s_2, s_3 at time $t = \frac{\sqrt{17}+\sqrt{26}}{2} + \frac{1}{4} \times 2 = 4.61 + 0.5 = 5.11$, and she finishes processing s_1 at least at time $t = 5.11 + \frac{\sqrt{17}}{2} + \frac{1}{4} = 7.42$, which is exceeding s_1's expiration time $e_1 = 7$). Whereas if only the geographical location information is considered, w_1 will first go to process s_1, then s_4, s_3, and back to initial location (see the green route shown in Fig. 4), in which w_1 could not finish task s_2 before its expiration time (w_1 accomplish s_1 at time $t = \frac{2\sqrt{2}}{2} + \frac{1}{4} = 1.66$, and she finishes processing s_2 at least at time $t = 1.66 + \frac{\sqrt{5}}{2} + \frac{1}{4} = 3.03$, which is exceeding s_2's expiration time $e_2 = 3$).

However, for the mixed priority metric (set $\alpha = 0.5$) considering both temporal constraints and spatial information, the worker w_1 will finish the task s_2 first (the mixed priority of s_2 is minimum, which is $0.5 \times \frac{\sqrt{17}-2\sqrt{2}}{3\sqrt{5}-2\sqrt{2}+\epsilon} + 0.5 \times \frac{3-3}{9-3+\epsilon} = 0.167$), then s_1 and s_3, and then s_4 (see the red route shown in Fig. 4). Obviously, the number of accomplished tasks is most by mixed priority strategy, and the route of the mixed priority strategy is quite different from the other two priority strategies with pure temporal or spatial.

Algorithm 2: Queue Scheduling (QS)

Input: Task clusters $\{\Omega_l\}$, workers set W
Output: The final task queues for workers $\{L_{f_k}\}$

1 $State$ is the list recording the states of workers (Initial $State(k)$ as online);
2 L_t is the time list recording when the workers finished the last task (Initial $L_t(k)$ as 0);
3 L_d is the location list recording the location of the latest task;
4 Initial L_{f_k} as [ ];
5 Sort $\{\Omega_l\}$ by $|\Omega_l|$ in descending order;
6 Compute the center of Ω_l, note as $center_i = (\overline{x}_l, \overline{y}_l)$;
7 **for** $l \leftarrow 1$ *to* $|W|$ **do**
8 $\quad w \leftarrow \arg\min_{w \in W} d(w, center_i)$;
9 $\quad \Psi_w \leftarrow \Omega_l$; //Allocate w to Ω_l
10 $\quad W \leftarrow W \backslash \{w\}$; //Remove w from W
11 **while** *there are still workers online && the remaining task queue* $\neq \varnothing$ **do**
12 $\quad w_k \leftarrow \arg\min_w L_t$; // First finish previous tasks
13 $\quad$ Get the next task s_i, update $State(k)$ and the remaining task queues $\{\Psi_k\}$ by MPT;
14 $\quad$ **if** $s_i \neq \varnothing$ **then**
15 $\quad\quad L_{f_k} \leftarrow L_{f_k} \cup \{s_i\}$;
16 $\quad\quad$ Update $L_t(k)$ and $L_d(k)$ by s_i;
17 **return** $\{L_{f_k}\}$ for all workers

Although we could schedule the worker to process the assigned tasks in a subdomain by the mixed priority strategy, in some extreme cases, when a worker is unable to tackle currently assigned tasks, those tasks will be forwarded to a nearest worker for help following the specific forwarding rules.

In order to ensure the shortest collaboration paths and reduce the traveling time, we construct the shortest Hamilton path. We use H_i to represent the length of Hamiltonian path from worker w_k to the i^{th} task created based on nodes in Ψ_k including the location of w_k. We denote the distance between s_m and s_n as $d_{m,n}$, and H_i can be calculated as:

$$H_i = \begin{cases} d_i^k, & i = 1, \\ H_{i-1} + d_{u_i,e_i}, & 1 < i \leq |\Psi_k|, \end{cases} \tag{15}$$

where u_i is the first node of i^{th} path and e_i is the last node.

The worker queue scheduling scheme is mainly shown in QS algorithm, the part of which is split into MPT, RT and NRT algorithm. First, task subdomains are allocated to their designated workers (line 5-10). Moreover, the task queue for a worker is rearranged by a mixed metric incorporating geographical location information as well as the task's temporal emergency (line 11-16). Finally, return the target task that the worker needs to accomplish in the next moment by MPT algorithm. If current task could be

Algorithm 3: Mixed Priority Task (MPT)

Input: Remaining task queues $\{\Psi_k\}$, the time stamp $L_t(k)$, current location $L_d(k)$
Output: Target task s_i, $State(k)$, $\{\Psi_k\}$

Initial α;
Get current task queue Ψ_k for w_k;
for $i \leftarrow 1$ *to* $|\Psi_k|$ **do**
  Sort $\Psi_k^{(d)}$ by $d_i^k(t)$ according to Eq. (2);
  Sort $\Psi_k^{(t)}$ by t_i according to Eq. (3);
  Get $\xi_k^{(d)}(i)$ according to Eq. (12);
  Get $\xi_k^{(t)}(i)$ according to Eq. (13);
for $i \leftarrow 1$ *to* $|\Psi_k|$ **do**
  Calculate $\xi_k^{(m)}(i)$ according to Eq. (14);
Sort $\Psi_k^{(m)}$ by $\xi_k^{(m)}$ in ascending order;
Get top task of the sorted queue: s_i;
Test whether s_i is Returnable for w_k by Eq. (16);
if *Returnable* **then**
  call a sub-algorithm RT;
else
  call a sub-algorithm NRT;
return s_i, $State(k)$, $\{\Psi_k\}$

Returnable, the scheme select next task for the worker by RT algorithm. Otherwise, the scheme will re-select a new task for the worker by NRT algorithm. Besides, in case a task cannot be served by its initial designated worker, this work will send it to the nearest worker for help.

3.2.3 Scheme Analysis

In the task assignment architecture, when the crowdsourcing platform schedules workers to accomplish corresponding tasks, there are still remaining two important problems:

Algorithm 4: Returnable Task (RT)

Input: Task queues $\{\Psi_k\}$, the time stamp $L_t(k)$, current location $L_d(k)$
Output: Target task s_i, $State(k)$, $\{\Psi_k\}$

Test whether s_i is reachable by Eq. (19);
if *Reachable* **then**
  $\Psi_k \leftarrow \Psi_k \backslash \{s_i\}$; ///Remove s_i from Ψ_k
  if w_k *is Online-Idle* **then**
    $State(k) \leftarrow$ online-idle;
  return s_i, $State(k)$, $\{\Psi_k\}$;
else
  if w_q *is the nearest worker for* s_i **then**
    $\Psi_q \leftarrow \Psi_q \cup \{s_i\}$; //Append s_i to Ψ_q
    $\Psi_k \leftarrow \Psi_k \backslash \{s_i\}$; //Remove s_i from Ψ_k
  if w_k *is Online-Idle* **then**
    $State(k) \leftarrow$ online-idle;
    return $\varnothing$, $State(k)$, $\{\Psi_k\}$;
  else
    Re-select a new task by priority in $\Psi_k^{(m)}$;

Algorithm 5: Not-Returnable Task (NRT)

Input: Task queues $\{\Psi_k\}$, the time stamp $L_t(k)$, the location $L_d(k)$
Output: Target task s_i, $State(k)$, $\{\Psi_k\}$

if w_q *is the nearest worker for* s_i **then**
  $\Psi_q \leftarrow \Psi_q \cup \{s_i\}$; //Append s_i to Ψ_q
  $\Psi_k \leftarrow \Psi_k \backslash \{s_i\}$; //Remove s_i from Ψ_k
if w_k *is Offline* **then**
  $State(k) \leftarrow$ offline;
  return $\varnothing$, $State(k)$, $\{\Psi_k\}$;
else
  Re-select a new task by priority in $\Psi_k^{(m)}$;

whether the working time of the worker is exceeded and whether the allowable arriving time of the task is exceeded.

Returnable-In-Time Test: The worker w_k need to finish the assigned tasks and return back to initial departure location before her deadline e_{w_k}. After selecting a target task s_i, a worker will pre-calculate whether she could finish the task and return back to her departure location before her deadline. If so, she will move forward to the next task s_i; otherwise, she will re-select a new task in the mixed priority queue and forward current task s_i to a nearest worker:

$$\text{Returnable Test} = \begin{cases} s_i, & e_{w_k} \geq t + \frac{d_i^k(t)+d(s_i,lw_k^0)}{v_k} + \frac{1}{p_k}, \\ \text{re-select}, & \text{otherwise.} \end{cases} \tag{16}$$

Worker-Still-Online Test: When a worker is assigned a new task by a crowdsourcing platform, the worker should be tested whether she's still online.

When worker w_k is assigned a new task s_i, w_k couldn't finish and return back to the initial departure before her deadline. Then if the task is the last one in the current task queue of worker w_k, the worker would enter into the offline state; otherwise, the worker stays online and needs to re-select a new task in the remaining mixed priority queue.

$$\text{Offline Test} = \begin{cases} \text{offline}, & s_i \text{ is the last and not returnable,} \\ \text{online}, & \text{otherwise.} \end{cases} \tag{17}$$

When the worker enters into the offline state, the crowdsourcing platform does not consider assigning a new task to the worker anymore.

When a new task is assigned to some worker, and the worker could finish the task and return back to the initial departure before her deadline, if the set of remaining tasks for the worker is not empty after finishing the currently assigned task, then the state of the worker is still online. Otherwise, the worker enters into the online-idle state.

$$\text{Idle Test} = \begin{cases} \text{online-idle}, & \text{the remaining task set is empty,} \\ \text{online}, & \text{otherwise.} \end{cases} \tag{18}$$

If a worker is online-idle, the crowdsourcing platform will not actively assign a new task to the worker, unless the worker's neighboring workers can't complete the assigned

tasks (i.e., the task can't be finished by her neighboring worker and its nearest worker is her).

Reachable-in-Time Test: In the task assignment architecture, a subtle issue is that, no matter how hard the worker works, there might exist task requests that will never be served. Such requests should be either removed from the queue or delivered to a nearest region's worker for help.

When a task s_i request is received, w_k should compute the earliest time to reach the location of the task and compare it with the task's maximum allowable traveling time. If a worker cannot reach the task before the maximum remaining allowable traveling time, the task request should be simply dismissed or forwarded to the nearest subdomain for another online worker for help. The worker's finished time of the previous task s_{i-1} is t_{i-1} (i.e., $t_{i-1} = \frac{H_{i-1}}{v_k} + \sum_{j=1}^{i-1} \frac{1}{p_k}$).

This decision can be calculated by Equation (19) as well:

$$\text{Reachable Test} = \begin{cases} \text{serve}, & e_i \geq t_{i-1} + \frac{d_i^k(t_{i-1})}{v_k} + \frac{1}{p_k}, \\ \text{remove}, & \text{otherwise.} \end{cases} \tag{19}$$

The platform first allocates the worker to the corresponding subdomain and then assigns a new task to the worker who is the first to complete a previous task and still online. (When two or more workers finish a previous task at the same time, the platform would choose the worker whose id is smaller.) The Queue Scheduling (QS) algorithm is shown in Algorithm 2. Besides, the target task is assigned to the worker with the highest mixed priority by Algorithm 3 MPT algorithm. Moreover, If a worker could finish the current task before her deadline, Algorithm 4 RT algorithm will return the reachable task; If a worker could not finish the current task before her deadline, Algorithm 5 NRT algorithm will re-select a new task for the worker.

Conventional spectral clustering typically consists of two time-consuming phases, namely, affinity matrix construction and eigen-decomposition. It generally takes $O(N^2d)$ time to construct the affinity matrix, and takes $O(N^3)$ time to solve the eigen-decomposition problem [46], where N is the data size and d is the dimension. Thus, the time complexity of Enhanced Spectral Clustering in Algorithm 1 is $O(|S|^2 + |S|^3)$. The time complexity of Queue Scheduling in Algorithm 2 is $O(|W|^2 + |W|(|S| + |S|log|S|))$.

4 Experiments and Evaluation

4.1 General Setup

We report the results for two sets of experiments over the proposed scheme on both synthetic datasets (SYN) and real datasets (REAL). All the experiments are carried out on a machine with 6 cores of AMD R5-4600U and 16 GByte RAM.

In the first part of experiments, we evaluate the impact of the hyper-parameters, in particular, θ and α, on the performance of our approach on synthetic datasets. In these experiments, we evaluate the performance through 2 important metrics: 1) the overall task completion rate, and 2) the average task time cost.

In the second part of experiments, we fix the hyper-parameters determined in the first part and evaluate the scalability of our proposed approach by varying the number of tasks and workers on both synthetic and real datasets.

4.2 Experiments on Synthetic Data Sets

For the synthetic (SYN) datasets, we use random data following two different distributions: uniform (SYN-UNIFORM) and skewed (SYN-SKEWED). With regard to SYN datasets, 50% of the tasks are generated in twenty clusters (with standard deviation as 1 and randomly chosen centers) and the other 50% of the tasks are uniformly distributed, i.e., 50% of the tasks are SYN-SKEWED and others are SYN-UNIFORM. This is motivated by the clustering characteristic of tasks in practice.

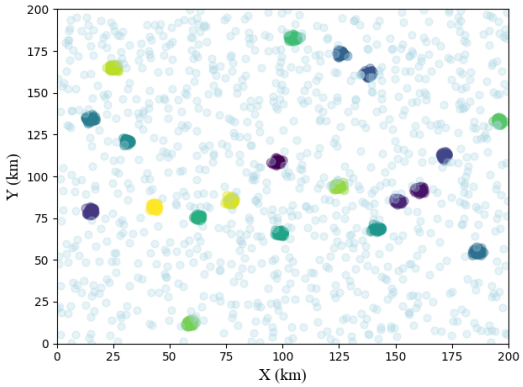


Fig. 5. A sample of synthetic data.

4.2.1 *Effect of Parameter θ*

Our experiments firstly decide the best value of parameter θ for applying θ-sparseness to reconstruct the affinity matrix in Algorithm 1. We conduct experiments on a 200×200 km^2 space where tasks with a cluster characteristic as illustrated in Fig. 5, where circles represent tasks in space. The default values of all the parameters used in our experiments are summarized in Table 3.

TABLE 3
Parameters of simulation

Parameters	Values
Space size (km^2)	200×200
Number of tasks	2000
Number of workers	150
Weight of time α	0.5
Expiration time of spatial tasks (h)	$[4, 10]$
Deadline of workers (h)	$[8, 10]$
Traveling speed of workers (km/h)	$[30, 60]$
Processing speed of workers (item/h)	$[1, 4]$

At the first time, we have no clear idea of the effects of parameter α in our algorithm, and we set $\alpha = 0.5$, which implies that the time and distance factors have the same level of impact on the priority of the tasks. Since the synthetic data are generated randomly, in order to reduce the impact of randomness on the experimental results, the experiments are repeated 1000 times for each value of θ, and the means of the metrics are reported.

In Fig. 6, we illustrate the overall task completion rate δ and average traveling time cost τ. We notice that in general, when we have the parameter $\theta = 0.007$ for θ-sparseness,

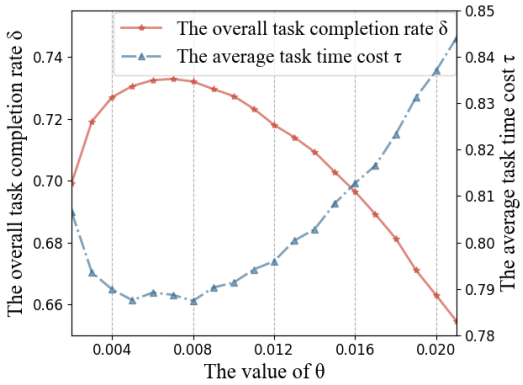


Fig. 6. Determining the value of parameter θ.

the overall task completion rate δ is largest, and the average traveling time cost τ is relatively low and close to the minimum obtained in the experiment. Besides, we can see from Fig. 6 when $\theta < 0.007$ or $\theta > 0.007$ both measures will get worse, which shows that the sparseness degree of the affinity matrix in Spectral clustering would affect the effect of the whole model. This suggests that $\theta = 0.007$ is a reasonable choice for Spectral clustering and we fix the value in the following experiments. At the same time, the value of θ is very small, which will have an impression on the complexity of Spectral clustering. A smaller value of θ can reduce the complexity of the whole algorithm to some extent, and improve the performance of algorithm.

4.2.2 Effect of Parameter α

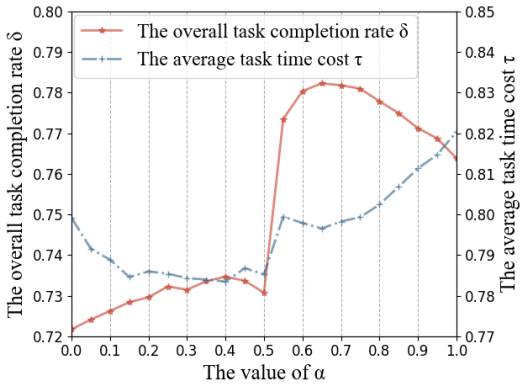


Fig. 7. Determining the weight of time α.

With the fixed value of parameter θ, in this part, we will decide the value of parameter α, which influences the weights of the temporal and spatial factors in the integrated priority function. The parameter θ is set as 0.007, and the other parameters are shown in Table 3 as well. Meanwhile, in order to reduce the impact of randomness on the experimental results, the experiments are repeated 1000 times for each value of α, and the means of the metrics are reported.

In Fig. 7, we present the value variation of δ and τ with respect to the value of α. We notice that when $\alpha = 0.65$, the accomplishment task rate δ is the highest, which is up to 0.782, and the traveling time cost rate τ is relatively low. In particular, when $\alpha > 0.5$ (i.e., $\frac{\alpha}{1-\alpha} > 1$), which indicates that the time priority is more important than the space priority, the task completion rate has a significant improvement compared to $\alpha < 0.5$. As shown in Fig. 7, the overall task completion rate δ is increased from 0.731 to 0.774, rising 5.9%, when parameter α is changed from 0.5 to 0.55. Thus, a relative proportion of time priority can improve the performance of our algorithm.

Such a result could be due to the proper combination of these two priorities. On one hand, if temporal priority is weighted too heavily, tasks that are too far away will be left alone to spend a lot of traveling costs. On the other hand, a heavy-weighted spatial priority may skip those tasks requiring immediate accomplishment with lower spatial priority.

Motivated by the results in Fig. 7, in the following experiments, we fix $\alpha = 0.65$. Because the task accomplishment rate at this time is higher than 70% in the SYN data, the TAMP algorithm is relatively stable and the optimal solution could be obtained.

4.3 Comparison with Other Algorithms

In this part, we use the values of two hyper-parameters determined in the first two experiments and compare TAMP with condign methods for task assignment on both synthetic and real data.

4.3.1 Baselines

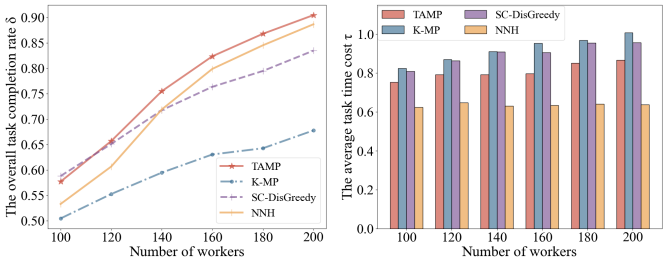
We first briefly present the baseline methods for comparative studies as follows.

- K-MP: The method clusters the tasks for different workers by K-means, then schedules the tasks for every worker by Mixed Priority Queue Scheduling.
- SC-DisGreedy: The method clusters the tasks for different workers by spectral clustering, and then every worker selects the nearest achievable task, which aims to reduce the traveling costs.
- NNH: In [47], Deng et al. propose an approximation algorithm named nearest neighbor heuristic (NNH). NNH exploits the spatial proximity between tasks by iteratively choosing the nearest available task to the last task added in the task sequence. At each iteration of NNH, the worker chooses one task which is available and the closest to his current position.

4.3.2 Results on Synthetic Data

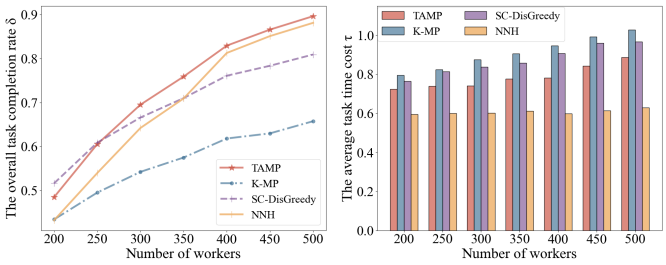
Fig. 8 and Fig. 9 show the evaluation metrics δ and τ achieved by different numbers of workers when there are 2000 and 5000 tasks in the SYN data network. The other parameters are the same as shown in Table 3.

Intuitively, the rates δ and τ both increase in the number of workers, as when more workers are available, it is more likely to find a worker for any specific task, and the minimum distance toward the task becomes lower. However, as the number of workers k increases, the growth of the overall task completion rate δ has begun to moderate. On the one hand, this could be due to the possibility that the remaining uncompleted tasks are located in remote locations. On the other hand, there are other factors that restrict the growth of the overall task completion rate δ, such as the traveling



(a) The rate δ (b) The value of τ

Fig. 8. Comparison of different methods in SYN data ($|S| = 2000$).



(a) The rate δ (b) The value of τ

Fig. 9. Comparison of different methods in SYN data ($|S| = 5000$).

speed of workers, the deadline of workers, and so on. Therefore, the growth of the overall task completion rate δ merely driven by the increase of workers could slow down and even saturate.

In terms of the overall task completion rate δ, the TAMP algorithm outperforms the three baseline methods in most of the cases. When $|W| = 200$ and $|W| = 250$ in Fig. 9 ($S = 5000$), the overall task completion rate δ is lower for TAMP than SC-DisGreedy, which means $\alpha = 0.65$ is not the best weight of time for our algorithm at this moment, i.e., the value of α needs to be adjusted with the model. Even if the parameter setting of the TAMP algorithm might not be optimal in all the scenarios (The parameters' values are not changed in subsequent experiments), the TAMP algorithm is still significantly better than other methods in most cases. When $|W| = 200$ in Fig. 8 and $|W| = 500$ in Fig. 9, the rate δ of TAMP algorithm is up to 0.9.

The NNH method has a lower running time since only considers the distance between workers and tasks, then the average task time cost τ is the lowest. The NNH method pays more attention to tasks in the neighborhood of workers, so as to minimize the traveling time cost of workers, but Our algorithm seeks the optimal solution of worker task assignment from a global perspective. Meanwhile, the TAMP algorithm is better than the two remaining methods in the average task time cost τ.

Generally speaking, the TAMP algorithm performed well on SYN data.

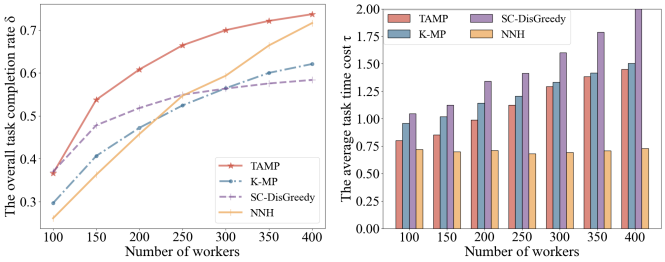
4.3.3 Results on Real Data

Considering the real social network scenarios, we use the open real-world dataset from Gowalla[1]. For simplicity, we only sample the data with longitude between -125 and -120, and latitude between 35 and 40 (approximately 440 km $\times$ 557 km rectangle region).

The dataset is a location-based social network, where users are able to check in to different spots in their vicinity. The check-ins include the location and the time that the users entered the spots. For our experiments, we use the check-in data over a period of one month (i.e., October 2010). Moreover, we assume that Gowalla nodes are the tasks of our spatial crowdsourcing system. Consequently, we assume all the chosen items happen in a single day.

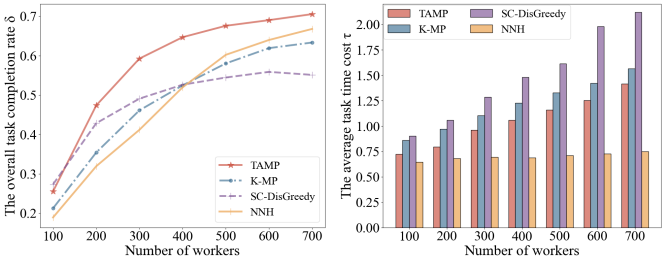
For each check-in, we use its location and time as the location and expiration time of the task. Intuitively, checking in a spot is equivalent to finishing a spatial task at that location. The worker heterogeneity is considered in the setting, then the processing time and traveling time are different when the same job is assigned to different workers. For the sake of simplicity, the traveling time cost is calculated by the Euclidean distance divided by the worker's traveling speed, and the processing time is calculated by the worker's processing speed.

The initial locations of workers are randomly generated in the restricted rectangle region, and the deadlines of workers are uniformly distributed from 6:00 pm to 8:00 pm. In this set of experiments, we evaluate the scalability of TAMP algorithm by different numbers $|S|$ of tasks, which is up to 8000. Fig. 10 and Fig. 11 show the rate δ and τ achieved by different numbers of workers when there are 5000 and 8000 tasks in the Gowalla data. Based on the experiments of REAL data, the advantage of TAMP algorithm in the overall task completion rate δ is more obvious. When $|W| = 400$ in Fig. 10 and $|W| = 700$ in Fig. 11, the rate δ of TAMP algorithm is more than 0.7.



(a) The rate δ (b) The value of τ

Fig. 10. Comparison of different methods in Gowalla data ($|S| = 5000$).



(a) The rate δ (b) The value of τ

Fig. 11. Comparison of different methods in Gowalla data ($|S| = 8000$).

1. http://snap.stanford.edu/data/loc-Gowalla.html

4.4 Discussion

The experiments present the efficiency and effectiveness of our proposed method. The real-world application scenarios, such as road condition monitoring [6] and crowdsourcing-aided positioning [7], are more applicable to the problem situations in our work. In these circumstances, the distance between the worker and the task will impact the number of completed tasks, with no special requirements on the worker's skills and strict time constraints on the task's completion. In many applications, the task assignment problem prefers to be dynamic rather than static. It is difficult to deal with the real-time task assignment problem in SC due to the unevenness of arriving tasks and workers, as well as the arrival time being random to the system. Our proposed method in this paper is applied in the static state of each batch, where the spatial-temporal information of tasks and workers is obtained in advance.

5 Conclusion

In this paper, we design an adequate task assignment mechanism in the context of spatial crowdsourcing, which assigns spatio-temporal tasks considering workers' heterogeneity. We formulate a combinatorial multi-objective optimization problem, i.e., MOST problem, and prove that it is NP-hard. To solve the above problem, we proposed the ***Ta**sk Clustering-based **M**ixed **P**riority Queue Scheduling (TAMP)* algorithm focusing on task network division and worker queue scheduling. At first, we apply θ-sparseness to the spectral clustering algorithm for optimizing the network partition to improve the scope of crowdsourcing services. Subsequently, the mixed priority queue scheduling scheme combines the temporal requirement as well as spatial features into a single priority metric, which schedules workers to complete the assigned tasks in turn. Extensive experiments on both synthetic and real data demonstrate the effectiveness and efficiency of our scheme.

The add-on of this work is to consider other properties of spatial tasks, such as the rewards of spatial tasks, the task workload, and others. Moreover, the work can be extended to spatial crowdsourcing in a real-time/online scenario.

Acknowledgments

This work was supported by the National Natural Science Foundation of China [U23A20309, 62272302, 62172276, 62372296]; Shanghai Municipal Science and Technology Major Project [2021SHZDZX0102]; CCF-DiDi GAIA [202307].

References

[1] Y. Zhen, A. Khan, S. Nazir, Z. Huiqi, A. Alharbi, and S. Khan, "Crowdsourcing usage, task assignment methods, and crowdsourcing platforms: A systematic literature review," *Journal of Software: Evolution and Process*, vol. 33, no. 8, p. e2368, 2021.

[2] J. Jarrett, M. B. Blake, and I. Saleh, "Crowdsourcing, mixed elastic systems and human-enhanced computing–a survey," *IEEE Transactions on Services Computing (TSC)*, vol. 11, no. 1, pp. 202–214, 2017.

[3] L. Kazemi and C. Shahabi, "Geocrowd: enabling query answering with spatial crowdsourcing," in *ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (ACM SIGSPATIAL)*, 2012, pp. 189–198.

[4] Q. Liu, L. Zheng, Y. Shen, and L. Chen, "Finish them on the fly: An incentive mechanism for real-time spatial crowdsourcing," in *Database Systems for Advanced Applications (DASFAA)*, 2020, pp. 694–710.

[5] B. Zheng, C. Huang, C. S. Jensen, L. Chen, N. Q. Viet Hung, G. Liu, G. Li, and K. Zheng, "Online trichromatic pickup and delivery scheduling in spatial crowdsourcing," in *IEEE International Conference on Data Engineering (ICDE)*, 2020, pp. 973–984.

[6] Z. Dai, C. H. Liu, Y. Ye, R. Han, Y. Yuan, G. Wang, and J. Tang, "Aoi-minimal uav crowdsensing by model-based graph convolutional reinforcement learning," in *IEEE International Conference on Computer Communications (INFOCOM)*, 2022, pp. 1029–1038.

[7] H. Lu, X. Gao, and G. Chen, "Efficient crowdsourcing-aided positioning and ground-truth-aided truth discovery for mobile wireless sensor networks in urban fields," *IEEE Transactions on Wireless Communications (TWC)*, vol. 21, no. 3, pp. 1652–1664, 2022.

[8] Y. Cheng, B. Li, X. Zhou, Y. Yuan, G. Wang, and L. Chen, "Real-time cross online matching in spatial crowdsourcing," in *IEEE International Conference on Data Engineering (ICDE)*, 2020, pp. 1–12.

[9] Y. Li, W. Xu, and M. L. Yiu, "Client-side service for recommending rewarding routes to mobile crowdsourcing workers," *IEEE Transactions on Services Computing (TSC)*, vol. 14, no. 6, pp. 1995–2010, 2021.

[10] G. Ye, Y. Zhao, X. Chen, and K. Zheng, "Task allocation with geographic partition in spatial crowdsourcing," in *ACM International Conference on Information & Knowledge Management (CIKM)*, 2021, pp. 2404–2413.

[11] X. Chen, Y. Zhao, K. Zheng, B. Yang, and C. S. Jensen, "Influence-aware task assignment in spatial crowdsourcing," in *IEEE International Conference on Data Engineering (ICDE)*, 2022, pp. 2141–2153.

[12] J. Yao, L. Yang, and X. Xu, "Online dependent task assignment in preference aware spatial crowdsourcing," *IEEE Transactions on Services Computing (TSC)*, vol. 16, no. 4, pp. 2827–2840, 2023.

[13] Y. Yang, Y. Cheng, Y. Yang, Y. Yuan, and G. Wang, "Batch-based cooperative task assignment in spatial crowdsourcing," in *IEEE International Conference on Data Engineering (ICDE)*, 2023, pp. 1180–1192.

[14] P. Cheng, L. Chen, and J. Ye, "Cooperation-aware task assignment in spatial crowdsourcing," in *IEEE International Conference on Data Engineering (ICDE)*, 2019, pp. 1442–1453.

[15] Y. Niu, Y. Zhang, and M. Song, "Pricing models for crowdsourcing tasks based on geographic information," in *IEEE/ACIS International Conference on Computer and Information Science (ICIS)*, 2018, pp. 794–797.

[16] M. E. Newman and M. Girvan, "Finding and evaluating community structure in networks," *Physical Review E (PHYS REV E)*, vol. 69, no. 2, p. 026113, 2004.

[17] A. Ng, M. Jordan, and Y. Weiss, "On spectral clustering: Analysis and an algorithm," *Advances in Neural Information Processing Systems (NIPS)*, vol. 14, pp. 849–856, 2001.

[18] Y. Ma, R. Ni, X. Gao, and G. Chen, "Mixed priority queue scheduling based on spectral clustering in spatial crowdsourcing," in *IEEE International Conference on Web Services (ICWS)*, 2021, pp. 293–302.

[19] X. Zhou, S. Liang, K. Li, Y. Gao, and K. Li, "Bilateral preference-aware task assignment in spatial crowdsourcing," in *IEEE International Conference on Data Engineering (ICDE)*, 2022, pp. 1687–1699.

[20] 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 Transactions on Knowledge and Data Engineering (TKDE)*, vol. 34, no. 7, pp. 3461–3477, 2022.

[21] S. S. Bhatti, J. Fan, K. Wang, X. Gao, F. Wu, and G. Chen, "An approximation algorithm for bounded task assignment problem in spatial crowdsourcing," *IEEE Transactions on Mobile Computing (TMC)*, vol. 20, no. 8, pp. 2536–2549, 2020.

[22] Z. Wang, Y. Zhao, X. Chen, and K. Zheng, "Task assignment with worker churn prediction in spatial crowdsourcing," in *ACM International Conference on Information & Knowledge Management (CIKM)*, 2021, pp. 2070–2079.

[23] Y. Xie, Y. Wang, K. Li, X. Zhou, Z. Liu, and K. Li, "Satisfaction-aware task assignment in spatial crowdsourcing," *Information Sciences*, vol. 622, pp. 512–535, 2023.

[24] Y. Wang, C. Zhao, and S. Xu, "Method for spatial crowdsourcing task assignment based on integrating of genetic algorithm and ant colony optimization," *IEEE Access*, vol. 8, pp. 68 311–68 319, 2020.

[25] P. Cheng, X. Lian, L. Chen, and C. Shahabi, "Prediction-based task assignment in spatial crowdsourcing," in *IEEE International Conference on Data Engineering (ICDE)*, 2017, pp. 997–1008.

[26] Z. Liu, K. Li, X. Zhou, N. Zhu, Y. Gao, and K. Li, "Multi-stage complex task assignment in spatial crowdsourcing," *Information Sciences*, vol. 586, pp. 119–139, 2022.
[27] Y. Tong, L. Chen, Z. Zhou, H. V. Jagadish, L. Shou, and W. Lv, "Slade: A smart large-scale task decomposer in crowdsourcing," *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, vol. 30, no. 8, pp. 1588–1601, 2018.
[28] U. ul Hassan and E. Curry, "Efficient task assignment for spatial crowdsourcing: A combinatorial fractional optimization approach with semi-bandit learning," *Expert Systems with Applications*, vol. 58, pp. 36–56, 2016.
[29] S. Sarker, M. A. Razzaque, M. M. Hassan, A. Almogren, G. Fortino, and M. Zhou, "Optimal selection of crowdsourcing workers balancing their utilities and platform profit," *IEEE Internet of Things Journal (IOT)*, vol. 6, no. 5, pp. 8602–8614, 2019.
[30] P. Wu, E. W. Ngai, and Y. Wu, "Toward a real-time and budget-aware task package allocation in spatial crowdsourcing," *Decision Support Systems*, vol. 110, pp. 107–117, 2018.
[31] Z. He, J. Cao, and X. Liu, "High quality participant recruitment in vehicle-based crowdsourcing using predictable mobility," in *IEEE Conference on Computer Communications (INFOCOM)*, 2015, pp. 2542–2550.
[32] P. Cheng, X. Lian, Z. Chen, R. Fu, L. Chen, J. Han, and J. Zhao, "Reliable diversity-based spatial crowdsourcing by moving workers," in *International Conference on Very Large Data Bases (VLDB)*, vol. 8, no. 10, 2015.
[33] Y. Zhao, J. Xia, G. Liu, H. Su, D. Lian, S. Shang, and K. Zheng, "Preference-aware task assignment in spatial crowdsourcing," in *AAAI Conference on Artificial Intelligence (AAAI)*, vol. 33, no. 01, 2019, pp. 2629–2636.
[34] X. Gao, H. Huang, C. Liu, F. Wu, and G. Chen, "Quality inference based task assignment in mobile crowdsensing," *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, vol. 33, no. 10, pp. 3410–3423, 2020.
[35] X. Miao, Y. Kang, Q. Ma, K. Liu, and L. Chen, "Quality-aware online task assignment in mobile crowdsourcing," *ACM Transactions on Sensor Networks (TOSN)*, vol. 16, no. 3, pp. 1–21, 2020.
[36] H. To, G. Ghinita, and C. Shahabi, "A framework for protecting worker location privacy in spatial crowdsourcing," in *International Conference on Very Large Data Bases (VLDB)*, vol. 7, no. 10, 2014, pp. 919–930.
[37] M. H. Cheung, F. Hou, J. Huang, and R. Southwell, "Distributed time-sensitive task selection in mobile crowdsensing," *IEEE Transactions on Mobile Computing (TMC), early access*, pp. 1–1, 2020.
[38] T. Song, Y. Tong, L. Wang, J. She, B. Yao, L. Chen, and K. Xu, "Trichromatic online matching in real-time spatial crowdsourcing," in *IEEE International Conference on Data Engineering (ICDE)*, 2017, pp. 1009–1020.
[39] Y. Tong, Z. Zhou, Y. Zeng, L. Chen, and C. Shahabi, "Spatial crowdsourcing: a survey," *The VLDB Journal (VLDBJ)*, vol. 29, no. 1, pp. 217–250, 2020.
[40] F. Zhu, S. Liu, J. Fang, and A. Liu, "Incentive-aware task location in spatial crowdsourcing," in *International Conference on Database Systems for Advanced Applications (DASFAA)*, 2021, pp. 650–657.
[41] Y. Li, D. Deng, U. Demiryurek, C. Shahabi, and S. Ravada, "Towards fast and accurate solutions to vehicle routing in a large-scale and dynamic environment," in *International Symposium on Spatial and Temporal Databases (SSTD)*, 2015, pp. 119–136.
[42] F. Bullo, E. Frazzoli, M. Pavone, K. Savla, and S. L. Smith, "Dynamic vehicle routing for robotic systems," *Proceedings of the IEEE (Proc. IEEE)*, vol. 99, no. 9, pp. 1482–1504, 2011.
[43] Y. Liu, B. Guo, C. Chen, H. Du, Z. Yu, D. Zhang, and H. Ma, "Foodnet: Toward an optimized food delivery network based on spatial crowdsourcing," *IEEE Transactions on Mobile Computing (TMC)*, vol. 18, no. 6, pp. 1288–1301, 2018.
[44] R. M. Karp, "Reducibility among combinatorial problems," in *Complexity of computer computations*. Springer, 1972, pp. 85–103.
[45] M. Lucińska and S. T. Wierzchoń, "Spectral clustering based on k-nearest neighbor graph," in *Computer Information Systems and Industrial Management (CISIM)*, 2012, pp. 254–265.
[46] U. Von Luxburg, "A tutorial on spectral clustering," *Statistics and computing*, vol. 17, no. 4, pp. 395–416, 2007.
[47] D. Deng, C. Shahabi, U. Demiryurek, and L. Zhu, "Task selection in spatial crowdsourcing from worker's perspective," *GeoInformatica*, vol. 20, no. 3, pp. 529–568, 2016.

Yue Ma received her B.S. degree in Mathematics from Shandong University, China, in 2016, and her M.S. degree in Operational Research and Cybernetics from Shandong University, China, in 2019. She is currently a Ph.D. candidate majoring in Computer Technology in Shanghai Jiao Tong University, China. Her research interests include crowdsourcing and combinatorial optimizations.

Xiaofeng Gao received the B.S. degree in information and computational science from Nankai University, China, in 2004; the M.S. degree in operations research and control theory from Tsinghua University, China, in 2006; and the Ph.D. degree in computer science from The University of Texas at Dallas, USA, in 2010. She is currently a Professor with the Department of Computer Science and Engineering, Shanghai Jiao Tong University, China. Her research interests include data engineering and combinatorial optimizations. She has published more than 200 peer reviewed papers in the related area, including well-archived international journals such as IEEE TKDE, IEEE TMC, ACM/IEEE TON, IEEE TC, IEEE TPDS, IEEE JSAC and also in well-known conference proceedings such as INFOCOM, SIGKDD, ICDE, SIGIR, WWW, NeurIPS, IJCAI, AAAI, etc. She is an IEEE/ACM Senior Member and CCF Distinguished Member. For more information, please visit http://www.cs.sjtu.edu.cn/gao-xf/.

Shahzad Sarwar Bhatti received the BS degree in electrical engineering (telecommunication) from the University of Engineering and Technology, Lahore, Pakistan, in 2000; and the MS degree in telecommunication engineering from the University of Engineering and Technology, Peshawar, Pakistan, in 2015; and the PhD degree in Computer Science and Technology from Shanghai Jiao Tong University, China, in 2020. He is currently an Assistant Professor with the Department of Information Sciences, University of Education, Lahore, Pakistan. His research interests include crowdsourcing, mobile crowdsourcing, task assignment, and combinatorial optimizations.

Guihai Chen is a distinguished professor of Shanghai Jiao Tong University and Nanjing University concurrently. He is an IEEE Fellow and CCF Fellow. He earned his Ph.D. degree in computer science from the University of Hong Kong in 1997. He had been invited as a visiting professor by Kyushu Institute of Technology in Japan, University of Queensland in Australia, and Wayne State University in USA. He has a wide range of research interests with focus on parallel computing, wireless networks, data centers, peer-to-peer computing, high-performance computer architecture, and data engineering. He has published more than 700 peer-reviewed papers, and most of them are in well-archived international journals such as more than 150 ACM/IEEE Transactions papers, and also in well-known conference proceedings such as HPCA, OSDI, SIGCOMM, MOBICOM, INFOCOM, ICNP, AAAI, and NIPS. He has won 15 best paper awards of conferences including ICNP 2015, DASFAA 2017, IWQoS 2020, and Ubicomp 2023. His papers are cited for more than 22000 times.