Task Allocation with Geographic Partition in Spatial Crowdsourcing

Guanyu Ye
University of Electronic Science and Technology of China
ygy@std.uestc.edu.cn

Yan Zhao*
Department of Computer Science, Aalborg University
yanz@cs.aau.dk

Xuanhao Chen
University of Electronic Science and Technology of China
xhc@std.uestc.edu.cn

Kai Zheng
University of Electronic Science and Technology of China
zhengkai@uestc.edu.cn

ABSTRACT

Recent years have witnessed a revolution in Spatial Crowdsourcing (SC), in which people with mobile connectivity can perform spatio-temporal tasks that involve travel to specified locations. In this paper, we identify and study in depth a new multi-center-based task allocation problem in the context of SC, where multiple allocation centers exist. In particular, we aim to maximize the total number of the allocated tasks while minimizing the average allocated task number difference. To solve the problem, we propose a two-phase framework, called Task Allocation with Geographic Partition, consisting of a geographic partition phase and a task allocation phase. The first phase is to divide the whole study area based on the allocation centers by using both a basic Voronoi diagram-based algorithm and an adaptive weighted Voronoi diagram-based algorithm. In the allocation phase, we utilize a Reinforcement Learning method to achieve the task allocation, where a graph neural network with the attention mechanism is used to learn the embeddings of allocation centers, delivery points, and workers. Extensive experiments give insight into the effectiveness and efficiency of the proposed solutions.

CCS CONCEPTS

• **Networks** → *Location based services*; • **Human-centered computing** → **Empirical studies in collaborative and social computing**.

KEYWORDS

geographic partition, task allocation, spatial crowdsourcing, reinforcement learning

ACM Reference Format:
Guanyu Ye, Yan Zhao, Xuanhao Chen, and Kai Zheng. 2021. Task Allocation with Geographic Partition in Spatial Crowdsourcing. In *Proceedings of the 30th ACM International Conference on Information and Knowledge Management (CIKM '21), November 1–5, 2021, Virtual Event, QLD, Australia.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3459637.3482300

*Corresponding author: Yan Zhao.

1 INTRODUCTION

The development of GPS-enabled smart devices and communication technologies flourishes the market of Spatial Crowdsourcing (SC) [4, 5, 8–10, 12, 22, 27, 32, 33, 40–42], where task requesters can issue spatial tasks to an SC server, and the server employs smart device carriers as workers to physically travel to the specified locations and accomplish these spatial tasks.

Most of the existing studies focus on task allocation based on the whole study area [1, 11, 21, 23, 24, 30, 31, 35–37, 39, 44–47]. For example, Abdullah et al. [1] propose an efficient framework to select optimal workers for each task with various specification by using a Bayesian Network in a spatial region. Wang et al. [35] 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. An implicit assumption shared by this work is that a worker can physically move to any place to perform tasks as long as the reachable constraint is not violated. While this is indeed realistic for many applications, we also observe some other scenarios where many allocation centers (aiming to allocate the tasks to their workers in their responsible areas) exist, and each worker has to work for a particular spatial region. For example, some chain supermarkets and fresh food markets (e.g., Carrefour and Hema Xiansheng) provide delivery services in a city, where each worker works for a particular allocation center (e.g., a particular supermarket or a particular fresh food market) and its corresponding responsible area. Under normal circumstances, there are many chain stores in a city, and each store only distributes deliveries within a certain geographic area.

In this paper, we investigate the task allocation of spatial crowdsourcing under such a problem setting, namely Multi-Center-based Task Allocation (MCTA). Specifically, given a set of allocation centers, a set of workers, a set of delivery points each with several tasks, it aims at finding an optimal allocation of tasks to workers to maximize the total number of allocated tasks while minimizing the allocated task number difference among workers, where the whole area is divided based on the distributions of allocation centers and delivery points. Few studies explore task allocation with multiple allocation centers. Recently, Zhao et al. [43] propose a fairness-aware task allocation framework, which also studies a multi-center-based

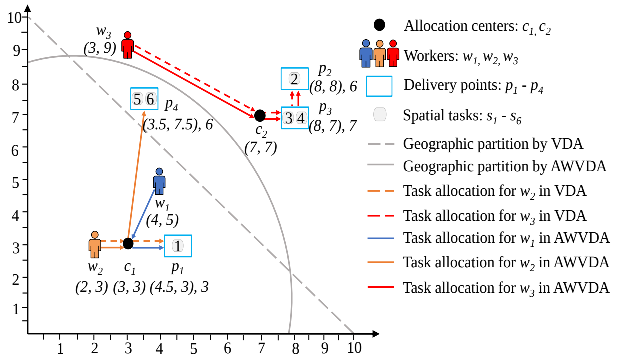


Figure 1: Running example

task allocation problem. However, it differs from our work in terms of problem setting and objectives. First, it assumes that each allocation center has fixed delivery points, while in our work, the delivery points are adaptively associated with allocated centers based on the dynamic tasks. Second, the goal of the study [43] is to minimize the payoff difference among workers while maximizing the average worker payoff, whereas our primary goal is to maximize the total number of allocated tasks, and the secondary goal is to minimize the average allocated task number difference.

We show the MCTA problem through a motivation example in Figure 1, that shows two allocation centers (e.g., c_1 and c_2, located at (3, 3) and (7, 7), respectively), three workers (e.g., $\{w_1, w_2, w_3\}$), and four delivery points (e.g., $\{p_1, p_2, p_3, p_4\}$). Each delivery point p is associated with a set of tasks $p.S$ (the deliveries sent to the location of p), e.g., p_3 has two tasks. For simplicity, we just use the index to denote tasks, e.g., $S = \{1, 2, ..., 6\}$. Further, tasks expire after a certain time, and the earliest expiration time e among the tasks in a delivery point is recorded (e.g., the earliest expiration time of $p_3.S$ is 7). The MCTA problem is to allocate tasks to workers so as to maximize the total number of allocated tasks while minimizing the average allocated task number difference. For simplicity, we assume that each worker has the same speed (1). Next, each worker has to move to the allocation center to receive a task (e.g., a package to be delivered) and must then travel to an allocated delivery point to complete the task (perform the delivery). When adopting a basic Voronoi Diagram-based Algorithm (VDA), we can obtain a geographic partition, $\{(c_1, \{p_1\}), (c_2, \{p_2, p_3, p_4\})\}$, based on which we can get a task allocation $\{(w_2, \{p_1\}), (w_3, \{p_2, p_3\})\}$ by applying our method (see Section 4.2), where the total number of allocated tasks is 4 and the average allocated task number difference is 2. However, this algorithm leaves tasks in delivery point p_4 unallocated.

To solve the MCTA problem, we propose a novel spatial crowdsourcing framework, namely Task Allocation with Geographic Partition (TAGP) which consists of a geographic partition phase and a task allocation phase. The first phase aims to partition the whole study area based on the allocation centers. More specifically, a Voronoi Diagram (VD) technique [18] is used to divide the delivery points based on the spatial distance between the delivery points and the allocation centers. We further design an Adaptive Weighted Voronoi Diagram-based Algorithm (AWVDA) to divide the delivery points adaptively by considering both the spatial distance and the dynamics of tasks. In the allocation part, we utilize a Reinforcement Learning (RL) method to achieve the task allocation, where a graph neural network with the attention mechanism is used to learn the graph embedding [34]. We utilize a samples batch approximation strategy to speed up the convergence, and integrate the total number of allocated tasks and the average allocated task number difference into the reward to achieve the learning process. In Figure 1, we can get the geographic partition, $\{(c_1, \{p_1, p_4\}), (c_2, \{p_2, p_3\})\}$, by applying the proposed AWVDA, and achieve a task allocation, $\{(w_1, \{p_1\}), (w_2, \{p_4\}), (w_3, \{p_2, p_3\})\}$ with the total number of 6 and a comparable average difference of 1.33.

The contributions of this paper can be summarized as follows:

1) We formulate a novel task allocation problem in SC, namely Multi-Center-based Task Allocation (MCTA), which aims to maximize the total number of allocated tasks while minimizing the average allocated task number difference among workers.

2) A basic Voronoi Diagram-based Algorithm (VDA) and an Adaptive Weighted Voronoi Diagram-based Algorithm (AWVDA) are developed to efficiently and effectively divide the whole area based on allocation centers.

3) We allocate tasks by a Reinforcement Learning (RL) method for each divided area, where the graph neural network and attention mechanism techniques are used.

4) Extensive experiments are conducted with real and synthetic data, where the empirical results confirm that our solutions are effective and efficient in allocating spatial tasks.

The remainder of this paper is organized as follows. Section 2 introduces the related work and Section 3 provides notations and the proposed problem. In Section 4, we design different strategies for geographic partition, and propose a RL-based task allocation algorithm, followed by the experimental results in Section 5. Finally, we conclude the paper in Section 6.

2 RELATED WORK

Research on Spatial Crowdsourcing (SC) has gained substantial attention in recent years; consequently, many task allocation techniques have been proposed for different application scenarios. A Multi-Objective task allocation method has also been considered in [2, 25]. For instance, Zhang et al. [2] main research multi-objective optimization in SC, using the Multi-Objective particle swarm optimization algorithm and the sorting strategy algorithm to find the optimal solution based on the conflict of goals.

Geo-information is extremely important in the field of SC and is a necessary condition for workers to allocate tasks in the spatial dimension. As the task allocation problem is NP-hard in its general form [7, 14], it is easier to obtain an accurate solution by dividing a complex spatial problem into multiple subproblems 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. [28] propose a pricing model based on the distance information and the number of workers, but the expiration time of tasks is not considered. Recently, Rohith et al. [15] applied Reinforcement Learning method to achieve task allocation. It differs from our work in terms of the task definition, the problem settings, and the objectives. First, they define the priority of tasks, so workers prefer to complete high-priority tasks and regardless of the cost. In contrast, we set the same priority for each task, which provides a more balance task allocation model. Second, while Rohith et al. [15] partition graph into grids and set 8 directions for each worker to sure her path. We utilize a graph neural network to learn the graph embedding, and use the attention mechanism to calculate the worker's importance to each delivery point. Third, Rohith et al. [15] aims to achieve the minimum average travel distance by completing all tasks, while we aim to maximize the completed task number and minimize the completed task difference among workers to achieve fairer and more task allocation. In particular, we design an adaptive geographic partition algorithm to find an optimal allocation method that can also bring a satisfying average task number for each worker.

3 PROBLEM DEFINITION

We proceed to present necessary preliminaries and then give the problem statement. Table 1 lists the major notations used throughout the paper.

Table 1: Summary of Notations

Notation	Definition
c	Allocation center
$c.l$	Location of allocation center c
$c.P$	The set of delivery points of allocation center c
$c.S$	The set of tasks of allocation center c
$c.W$	The set of workers of allocation center c
C	Allocation center set
p	Delivery point
$p.l$	Location of delivery point p
$p.S$	The set of tasks of delivery point p
P	Delivery point set
s	Spatial task
$s.p$	The delivery point of spatial task s
$s.e$	Expiration time of spatial task s
S	Spatial task set
w	Worker
$w.l$	Current location of worker w
$w.c$	Center of worker w
W	Worker set
P_w	A delivery point set of w
R	A delivery point sequence
$t(l)$	The arrival time of particular location l
t_{now}	The current time
$tc(a,b)$	Travel time from a to b
$VPS(w)$	A valid delivery point set for w
A	A spatial task allocation
$A.S$	Allocated task set of workers
$\|A.S\|$	Total number of allocated tasks
$\|A.S_{dif}\|$	Average allocated task number difference
$\mathbb{A}$	A spatial task allocation set

Definition 1 (Allocation Center). *An allocation center, denoted by $c = (l, P, S, W)$, has a location $c.l$, a set of delivery points $c.P$, a set of tasks $c.S$ to be allocated, and a set of workers $c.W$.*

Definition 2 (Delivery Point). *A delivery point, denoted by $p = (l, S)$, contains a location $p.l$, and a set of tasks $p.S$ that are deliveries from an allocation center c (that p belongs to) to delivery point p. It means that $c.S = \cup_{p \in c.P} p.S$.*

Definition 3 (Spatial Task). *A spatial task, denoted by $s = (p, e)$, consists of a delivery point $s.p$ (at which task s is to be delivered), and a task expiration time $s.e$.*

With spatial crowdsourcing, a spatial task s can be finished only if the worker is physically located at its location (i.e., the location of its delivery point $s.p$). Moreover, a task s can be completed only if the worker arrives at $s.p$ before its expiration time $s.e$. Note that with the single-task allocation mode [19], the server should allocate each spatial task to a worker at a time. For simplicity and without loss of generality, we assume that the processing time of each task is zero, which means that a worker will go to the location of the next task upon finishing the current task.

Definition 4 (Worker). *A worker, denoted as $w = (l, c)$, is able to perform spatial tasks. A worker can be in either online or offline mode. A worker is online when the worker is ready to accept tasks and offline when unavailable to perform tasks. An online worker w is associated with a current location $w.l$, and an allocation center $w.c$ the worker is going to work for.*

We assume that a worker can only work for a single allocation center, which is reasonable in practice. In this work, we assume that all the tasks are micro tasks (e.g., deliveries) and have the same reward. The server will consider all the available tasks and workers at a particular time instance, and it returns a sequence of delivery points (each with a set of spatial tasks) for each worker to visit in order to complete the spatial tasks while the spatio-temporal constraints are not violated. Once a delivery point sequence is allocated to a worker, the worker will go to the allocation center to take the deliveries (i.e., spatial tasks) and then go to the corresponding delivery points. The worker is offline until the allocated tasks are completed.

Figure 1 depicts two allocation centers that three workers work for, and four delivery points, each of which has several associated tasks. For example, workers w_1 and w_2 work for c_1, and delivery point p_3 has two tasks, i.e., s_3 and s_4.

Definition 5 (Delivery Point Sequence). *Given an online worker w and a set of allocated delivery points P_w, a delivery point sequence on P_w, denoted by $R(P_w)$, represents the order in which w visits the delivery points in P_w. The arrival time of w at delivery point $p_i \in P_w$ (the time of completing tasks related to p_i) can be computed as follows:*

$$t_{w,R}(p_i.l) = \begin{cases} t_{now} + tc(w.l, c.l) + tc(c.l, p_i.l) & if\ i = 1 \\ t_{w,R}(p_{i-1}.l) + tc(p_{i-1}.l, p_i.l) & if\ i > 1 \end{cases} \quad (1)$$

where t_{now} is the current time, $tc(a,b)$ is the travel time from location a to location b. When the context of w and R is clear, we use $t(p_i.l)$ to denote $t_{w,R}(p_i.l)$.

In Figure 1, worker w_3 can follow a delivery point sequence (p_3, p_2) and finish 3 tasks, since the times when w_3 arrives at these tasks are less than their expiration times.

For the sake of simplicity, we assume all workers share the same velocity, so the travel time between two locations can be estimated with their spatial distance. However, our proposed algorithms are not dependent on this assumption and can handle the case where the workers are moving at different speeds.

Definition 6 (Valid Delivery Point Set). *A delivery point set P_w is called a valid delivery point set (VPS) for a worker w, denoted as $VPS(w)$, if a delivery point sequence $R(P_w)$ exists, such that all the tasks located in $p \in P_w$ can be completed before their expiration times, i.e., $\forall s \in p.S, \forall p \in P_w\ (t(p.l) \leq s.e)$.*

It is worth noting that more than one delivery point sequences may exist for a given VPS. Among these, we consider only the one with the minimal travel time for each VPS. In Figure 1, $\{p_2, p_3\}$ is a VPS for worker w_3, and (p_2, p_3) and (p_3, p_2) are delivery point sequences. However, we only consider (p_3, p_2) since it has the lowest travel time.

Definition 7 (Spatial Task Allocation). *Given a set of allocation centers, a set of workers, and a set of delivery points, each with a set of tasks, a spatial task allocation, denoted by A, consists of a set of (worker, VPS) pairs in the form of* $(w_1, VPS(w_1))$, $(w_2, VPS(w_2))$,..., $(w_{|W|}, VPS(w_{|W|}))$, *where* $VPS(w_i) \cap VPS(w_j) = \emptyset, 1 \leq i \neq j \leq |W|$, *and W denotes the worker set.*

Let $A.S$ denote the set of tasks that are allocated to workers, $|A.S|$ denote the total number of allocated tasks, and $|A.S_{dif}|$ denote the average allocated task number difference between all workers, which reflects the allocation unfairness among workers to some extent and is calculated in Equation 2:

$$|A.S_{dif}| = \frac{\sum_{w_i \in W, w_j \in W, w_i \neq w_j} ||VPS(w_i)| - |VPS(w_j)||}{|W|(|W|-1)} \tag{2}$$

The problem investigated in our paper can be formally stated as follows.

Problem Statement. Given a set of allocation centers, a set of delivery points, a set of workers, and a set of tasks at the current time instance on an SC platform, our problem aims to find a task allocation A_{opt} that achieves the following goals:

1) primary optimization goal: maximize the total number of allocated tasks, i.e., $\forall A_i \in \mathbb{A}$ $(|A_{opt}.S| \geq |A_i.S|)$, where $\mathbb{A}$ denotes all possible allocations; and

2) secondary optimization goal: minimize the average allocated task number difference, i.e., $\forall A_i \in \mathbb{A}$ $(|A_{opt}.S_{dif}| \leq |A_i.S_{dif}|)$.

Our problem is NP-hard, which can be proved by reducing from the 0-1 knapsack problem in the similar way with [43].

4 ALGORITHM

In this paper, we propose a novel spatial crowdsourcing framework comprising of two components: geographic partition and task allocation.

The first component aims to divide the whole area into $|C|$ regions based on the allocation centers, where $|C|$ denotes the number of allocation centers. More specifically, we utilize a basic Voronoi Diagram-based Algorithm (VDA) to realize the partition of the delivery set according to the spatial distances between delivery points and allocation centers. Further, considering the dynamic tasks, we devise an Adaptive Weighted Voronoi Diagram-based Algorithm (AWVDA) that not only takes the distances between delivery points and allocation centers into account, but also utilizes the current number of tasks and workers to divide the delivery points adaptively.

The second component needs to allocate the tasks to the suitable workers by scheduling a task sequence for each worker in each allocation center to achieve the maximal task allocation while minimizing the average allocated task reward difference. We first utilize the compositional message-passing neural network (CMPNN) and the attention mechanism to obtain embeddings of allocation centers, delivery points and workers, and then use a Reinforcement Learning (RL) method to achieve task allocation by maximizing the expected reward of the allocation.

4.1 Geographic Partition

In this section, we mainly present two algorithms to achieve the geographic partition, i.e., Voronoi Diagram-based Algorithm (VDA) and Adaptive Weighted Voronoi Diagram-based Algorithm (AWVDA).

4.1.1 *Voronoi Diagram-based Algorithm (VDA).*

Voronoi diagram (VD), also known as Thiessen polygon or Dirichlet diagram, is composed of a set of polygons, where each polygon has a generator, and the boundary of two adjacent generators is their vertical bisector. VD has the following characteristics:

1) there is only one generator in each polygon;

2) for each point in a polygon, the distance between it and its corresponding generator is shorter than the distances between it and other generators;

3) for each point on a polygon's boundary, the distances between it and the generators sharing the boundary are the same.

In VDA, we regard the allocation centers as the generators and use spatial distance as the basis for achieving geographic partition, i.e., dividing the delivery points based on the spatial distances between them and the allocation centers. We elaborate the details of the VDA procedure in Algorithm 1. It takes an allocation center set C, and a delivery point set P as input, and outputs the delivery point sets for all the allocation centers (i.e., $C.P$). For each allocation center $c_i \in C$, we select the delivery points belonging to it by comparing the spatial distances between the delivery points and the allocation centers (lines 2-7). To be specific, a delivery point p is added into the delivery point set of the current allocation center c_i when the distance between p and c_i is not larger than the distances between p and other allocation centers, i.e., $\forall c_j \in C - \{c_i\}$ $(d(p, c_i) \leq d(p, c_j))$ (lines 6-7). After adding into the delivery point set of c_i, the delivery point p is removed from P (line 8), which guarantees that the delivery points located in the boundary of the VD can only be allocated once. Then $C.P$ and C are updated accordingly (lines 9-10). Finally, we find the delivery point set for each allocation center (line 11). According to the characteristics of VD, we can guarantee that each polygon (i.e., region) has only one allocation center, and each delivery point is located in a polygon.

Taking Figure 1 as an example, VDA can obtain a geographic partition, $\{(c_1, \{p_1\}), (c_2, \{p_2, p_3, p_4\})\}$, based on which we can get a task allocation $\{(w_2, \{p_1\}), (w_3, \{p_2, p_3\})\}$ by applying our method (see Section 4.2).

4.1.2 *Adaptive Weighted Voronoi Diagram-based Algorithm (AWVDA).*

Although VDA is effective in the process of geographic partition, it does not take the dynamics of tasks and workers into account, which may result in a worker-task imbalance, e.g., a large number of tasks are allocated to a small number of workers in a Voronoi polygon.

To solve this issue, we propose an Adaptive Weighted Voronoi Diagram-based Algorithm (AWVDA), where we utilize the adaptive distance (that considers the numbers of the current workers and tasks) instead of the spatial distance. For an allocation center c_i, the adaptive distance between it and a delivery point p can be

Algorithm 1: VDA

```
Input: $C, P$
Output: $C.P$
1  $C.P \leftarrow \emptyset$;
2  for each allocation center $c_i \in C$ do
3      $c_i.P \leftarrow \emptyset$;
4      for each delivery point $p \in P$ do
5          for each $c_j \in C - \{c_i\}$ do
6              if $d(p, c_i) \leq d(p, c_j)$ then
7                  $c_i.P \leftarrow c_i.P \cup p$;
8                  $P \leftarrow P - p$;
9          $C.P \leftarrow C.P \cup c_i.P$;
10         $C \leftarrow C - c_i$;
11 return $C.P$;
```

calculated in Equation 3.

$$ad(p, c_i) = aw_i \cdot d(p, c_i) \tag{3}$$

$$aw_i = \frac{|c_i.S|}{|c_i.W| + 1} \tag{4}$$

where $ad(p, c_i)$ denotes the adaptive distance between p and c_i, aw_i denotes an adaptive weight for c_i that can be computed in Equation 4, and $d(p, c_i)$ is the spatial distance between p and c_i. Next, $|c_i.S|$ is the number of tasks associated with c_i, and $|c_i.W|$ is the number of workers associated with c_i. Similar with VDA, in AWVDA, each delivery point in $c_i.P$ has a shorter adaptive distance from c_i than from other allocation centers, i.e., $c_i.P = \{p | ad(p, c_i) \leq ad(p, c_j), c_j \in C - \{c_i\}\}$.

Algorithm 2 illustrates the process of AWVDA. Given a set of allocation centers C, a set of delivery points P, a set of workers W, and a set of tasks S, AWVDA divides all the delivery points into $|C|$ subsets based on the allocation centers with a similar process of Algorithm 1. The main difference between the two algorithms is that Algorithm 2 sets an iteration h to adjust the partition continuously (line 2) and computes the adaptive distance that considers the numbers of the current workers and tasks (line 8), and adding the delivery point p into the delivery point set of c_i based on the adaptive distance (line 10).

Applying AWVDA in the running example in Figure 1, we can get a geographic partition, i.e., $\{(c_1, \{p_1, p_4\}), (c_2, \{p_2, p_3\})\}$. Applying our task allocation method in Section 4.2, a task allocation $\{(w_1, \{p_1\}), (w_2, \{p_4\}), (w_3, \{p_2, p_3\})\}$ with the total number of allocated tasks of 6 is obtained.

4.2 Task Allocation

After getting the geographic partition, we decompose the Multi-Center-based Task Allocation (MCTA) problem into $|C|$ single center-based sub-problems, where $|C|$ denotes the number of allocation centers. Each sub-problem can be transformed into a special multiple traveling salesman problem, where each salesman needs to arrive at cities before their expiration times. Then we give a Reinforcement Learning (RL) method by learning the embeddings of allocation centers, delivery points, and workers to solve each sub-problem to achieve a satisfying task allocation.

4.2.1 Allocation Center and Delivery Point Embeddings. According to the geographic partition, we can divide the whole area into $|C|$ parts each with an allocation center c. Each part can be regarded as a graph $g_c = (V_c, E_c)$, where V_c is a vertex set containing an

Algorithm 2: AWVDA

```
Input: $C, P, W, S$
Output: $C.P$
1  $C.P \leftarrow \emptyset$;
2  $h \leftarrow 1$;
3  repeat
4      for each allocation center $c_i \in C$ do
5          $c_i.P \leftarrow \emptyset$;
6          for each delivery point $p \in P$ do
7              for each $c_j \in C - \{c_i\}$ do
8                  Compute $ad(p, c_i)$ and $ad(p, c_j)$ based on
                     Equation 3;
9                  if $ad(p, c_i) \leq ad(p, c_j)$ then
10                     $c_i.P \leftarrow c_i.P \cup p$;
11                     $P \leftarrow P - p$;
12             $C.P \leftarrow C.P \cup c_i.P$;
13             $C \leftarrow C - c_i$;
14     $h = h + 1$
15 until $c.P^h = c.P^{h-1}$;
16 return $c.P$;
```

allocation center c and its delivery points $c.P$, i.e., $V_c = \{c, c.P\}$, and E_c is a set of edges connecting any two vertices in V_c. Besides, we define c as the first vertex in V_c.

Inspired by the success of the Compositional Message-passing Neural Network (CMPNN) [38] (a modified version of the Message Passing Neural Network (MPNN) [16]) for computing the feature embedding for each vertex efficiently, we employ CMPNN to learn the vertex embeddings (i.e., the allocation center and delivery point embeddings) of the graph. CMPNN enables the connected neighboring vertices to have more similar embeddings by message passing, which is the same as their spatial representation.

Specifically, in g_c, each vertex v_i in V_c has a two-dimensional coordinate as its input feature vector, denoted by f_i^0. Therefore, the input feature dimension is set to 2, i.e., $d_{in} = 2$. We set m types for each edge, the edge ed_{ij} of vertices v_i and v_j is associated with a one-hot feature vector, and the edge feature dimension d_{ed} is set to m. In order to pass message, CMPNN aggregates neighbor information of each vertex. In the following, we define the neighbor set for a vertex.

Definition 8 (Neighbor Set). *Given an undirected graph $G = (V, E)$ composed of a vertex set V and an edge set E. For each vertex v_i in V, we set the z closest neighbor vertices as its neighbor set $Ne(v_i)$, satisfying the following properties:*
1) $|Ne(v_i)| = z$, and
2) $\forall v_j \in Ne(v_i), v_p \in V - Ne(v_i)$ $(d(v_i, v_j) \leq d(v_i, v_p))$.

Like Convolutional Neural Network (CNN), CMPNN also designs a shift invariant global kernel $\mathbf{k} \in \mathbb{R}^{d_{ed} \times d_{in} \times d_{out}}$, where d_{out} is the output feature dimension. Based on the graph structure information and the neighbor set $Ne(v_i)$, the v_i's new feature in the $n+1$th iteration (n is a positive integer) f_i^{n+1}, can be formulated as:

$$f_i^{n+1} = \underset{v_j \in Ne(v_i)}{\mathbf{Agg}} \; ed_{ij} \mathbf{k} f_j^n \tag{5}$$

where **Agg** is an aggregation operator. However, due to the complexity of the graph, it is difficult to clarify the types of the edges. As a result, CMPNN uses a neural network h_{ed} to predict them. To pass message more effectively, CMPNN utilizes two different global invariant kernels, i.e., $\mathbf{k}_{ori}$ and $\mathbf{k}_{nei}$, to retain self-information and aggregate neighbors' information, respectively. The improved CMPNN is expressed as follows:

$$f_i^{n+1} = \underset{v_j \in Ne(v_i)}{\mathbf{Agg}} h_{ed}(ed_{ij})\,[\mathbf{k}_{ori} f_i^n + \mathbf{k}_{nei} f_j^n - \mathbf{k}_{nei} f_i^n] \tag{6}$$

After N iterations, each vertex embedding containing information from its N-hop neighborhoods, i.e., $f_i^N \in \mathbb{R}^{d_f}$, is obtained, where d_f is the dimension of vertex embedding. In the experiment part, we set N to 2. For simplicity, we use f_i to denote f_i^N.

4.2.2 Worker Embeddings.

4.2.2 Worker Embeddings. To obtain each worker embedding, we consider the graph contextual embedding and leverage an attention mechanism [29] to get attention coefficients, which reflect the importance of delivery point embeddings in constructing worker embeddings.

The embeddings of allocation centers and delivery points in the previous step have encoded the structure information of the graph. For graph g_c, its embedding f_{g_c} is computed by the max pooling from its vertex embeddings. It can be formally defined as $f_{g_c} = max\{f_1, f_2, ..., f_{|V_c|}\}$, where $f_{g_c} \in \mathbb{R}^{d_f}$. In our problem, each worker w must reach the allocation center $w.c$ (that the worker works for) first to get (delivery) tasks, and then go to the corresponding delivery points to complete tasks. So it is of vital importance for workers to know the information of their allocation centers. Since we set the allocation center as the first vertex in g_c, the graph contextual embedding $f_{cg_c} \in \mathbb{R}^{2d_f}$ ($2d_f$ is the dimension of the feature) is expressed as follows:

$$f_{cg_c} = [f_{g_c}; f_1] \tag{7}$$

where $[;]$ is a concatenation operation. Next, we utilize the notion of attention mechanism, which can compute the importance of the delivery point embeddings to the workers. We set query Q_w and key-value pair $\{K_{w,v_i}, V_{w,v_i}\}$ for each worker w and delivery point v_i. The Q_w is computed from the graph contextual embedding f_{cg_c} and $\{K_{w,v_i}, V_{w,v_i}\}$ comes from v_i's embedding f_i. The specific calculation method is as follows:

$$\begin{aligned} Q_w &= \theta_w f_{cg_c} \\ K_{w,v_i} &= \theta_{key} f_i \quad v_i \in V_c - c \\ V_{w,v_i} &= \theta_{value} f_i \quad v_i \in V_c - c \end{aligned} \tag{8}$$

where $\theta_w \in \mathbb{R}^{d_{key} \times 2d_f}$, $\theta_{key} \in \mathbb{R}^{d_{key} \times d_f}$, and $\theta_{value} \in \mathbb{R}^{d_{value} \times d_f}$ are embedding matrices, and d_{key} and d_{value} are the dimensions of key and value, respectively. For different workers, they own the same calculation process but different parameters. According to the query and key, we calculate the attention score $\mathcal{A}(w, v_i)$ between w and v_i:

$$\mathcal{A}(w, v_i) = \frac{Q_w^T K_{w,v_i}}{\sqrt{d_{key}}} \tag{9}$$

where Q_w^T denotes Q_w for transpose operation. We use a softmax function to normalize $\mathcal{A}(w, v_i)$, and obtain the attention weight $\mathcal{W}(w, v_i)$ between w and v_i:

$$\mathcal{W}(w, v_i) = \frac{e^{\mathcal{A}(w,v_i)}}{\sum_{v_j} e^{\mathcal{A}(w,v_j)}} \quad v_j \in V_c - c \tag{10}$$

Finally, according to the attention weights and values, we construct the worker embedding f_w in the following:

$$f_w = \sum_{v_i} \mathcal{W}(w, v_i) V_{w,v_i} \quad v_i \in V_c - c \tag{11}$$

4.2.3 Reinforcement Learning-based Task Allocation. After getting the embeddings of allocation centers, delivery points, and workers, we use a policy-based Reinforcement Learning (RL) method to achieve the task allocation. First, we utilize the attention mechanism to calculate the importance of workers to delivery points. Specifically, for worker w and delivery point v_i:

$$\begin{aligned} Q'_w &= \theta'_w f_w, \\ K'_{w,v_i} &= \theta'_{key} f_i \quad v_i \in V_c - c \\ \mathcal{A}'(w, v_i) &= \frac{Q'^T_w K'_{w,v_i}}{\sqrt{d'_{key}}} \quad v_i \in V_c - c \end{aligned} \tag{12}$$

where Q'_w is new query, K'_{w,v_i} is new key, $\mathcal{A}'(w, v_i)$ is new attention score, $\theta'_w \in \mathbb{R}^{d'_{key} \times d_{value}}$ and $\theta'_{key} \in \mathbb{R}^{d'_{key} \times d_f}$ are embedding matrices to project the embeddings back to d'_{key} dimensions, and d'_{key} is the dimension of the new key. For each delivery point v_i, the probability that worker w visits v_i is decided by w's importance to v_i. Therefore, we utilize a softmax function to calculate the probability value $\mathcal{P}_{w,v_i}$ in the following:

$$\mathcal{P}_{w,v_i} = \frac{e^{\mathcal{A}'(w,v_i)}}{\sum_w e^{\mathcal{A}'(w,v_i)}} \quad w \in c.W \tag{13}$$

In the learning phase, we randomly allocate delivery points to workers. ORTools solver[1] is an optimization tool developed by Google. It is used to find the best solution to a problem from many possible solutions by utilizing meta-heuristics [3]. For each worker w in g_c, we use ORTools to optimize w's point-performing path and get the $(w, VPS(w))$ pair. The allocation distribution of w is the product of the probability values that w visits delivery points in $VPS(w)$. Therefore, for this overall task allocation A_c, the distribution of A_c can be represented as follows:

$$\pi_\theta(A_c|g_c) = \prod_w \prod_{v_i} \mathcal{P}_{w,v_i} \quad w \in c.W \quad v_i \in VPS(w) \tag{14}$$

In the traditional MTSP problem, it always considers the average travel distance of the workers [26]. However, in our MCTA problem, we need to consider the primary optimization goal and the secondary optimization goal, i.e., the total number of allocated tasks and the average allocated task number difference. In order to achieve these goals, we calculate the reward of A_c, $Re(A_c)$, in Equation 15.

$$Re(A_c) = \frac{|A_c.S|}{A_c.dis \cdot |A_c.S_{dif}|} \tag{15}$$

where $|A_c.S|$ is the total number of allocated tasks associated with g_c, $A_c.dis$ is the average travel distance associated with g_c, and $|A_c.S_{dif}|$ is the average allocated task number difference associated with g_c.

To optimize the task allocation, we need to maximize the expected reward $\mathcal{L}_{Rc}(\theta)$:

$$\begin{aligned} \theta^* &= \arg\max \mathcal{L}_{Rc}(\theta) \\ \mathcal{L}_{Rc}(\theta) &= \mathbb{E}_{(g_c) \sim D} \sum_{A_c} \pi_\theta(A_c|g_c) Re(A_c) \end{aligned} \tag{16}$$

[1]https://developers.google.com/optimization/

where D is the training set. To speed up the convergence, we utilize a samples batch approximation [17], i.e., we randomly generate SA task allocations in g_c. To further decrease the variance while training, we utilize an advantage function. Specifically, for each allocation A_c^{sa}, its new reward can be represented as follows:

$$Re'(A_c^{sa}) \approx Re(A_c^{sa}) - \frac{1}{SA}\sum_{i=1}^{SA} Re(A_c^{i}) \tag{17}$$

Finally, we get the new expected reward and optimization function:

$$\mathcal{L}_{Rc}(\theta) \approx \mathbb{E}_{(g_c)\sim D} \sum_{sa=1}^{SA} \pi_\theta(A_c^{sa}|g_c) Re'(A_c^{sa}) \tag{18}$$

$$\nabla_\theta \mathcal{L}_{Rc}(\theta) \approx \mathbb{E}_{(g_c)\sim D} \sum_{sa=1}^{SA} \Big(\sum_{w}\sum_{v_i} \nabla_\theta \log \mathcal{P}_{w,v_i}\Big) Re'(A_c^{sa}) \quad w \in c.W \quad v_i \in VPS(w) \tag{19}$$

where $\nabla_\theta \mathcal{L}_{Rc}(\theta)$ is the partial derivative of the expected reward on θ.

5 EXPERIMENTAL EVALUATION

5.1 Experimental Setup

The experiments are carried out on both real and synthetic datasets, including a gMission dataset (marked as GM) and a synthetic dataset (marked as SYN). gMission is an open source SC dataset [6], in which each task is regarded as a delivery point that has a location, and each worker is also associated with a location. As gMission is not associated with allocation centers, we randomly generate $|C|$ allocation centers, where $|C| = 5, 10, 15, 20, 25$ with a default value of 5. Each worker is allocated to the closet allocation center and works for it. For each delivery point, we uniformly generate n_s tasks, where n_s is an integer and $n_s \in [0, \frac{2|S|}{|W|}]$ (to guarantee that the task number of each delivery point is non-negative). For the synthetic dataset, we generate the locations of allocation centers, workers and delivery points following a uniform distribution within a 2D space $[0, 1000]^2$ based on observations from real datasets (e.g., gMission). Other settings are set in the same way with the GM dataset.

We introduce a Genetic Algorithm (GA) [13] and a Simulated Annealing (SA) [20] as baselines to allocate tasks on the divided areas generated by the Voronoi Diagram-based Algorithm (VDA) or the Adaptive Weighted Voronoi Diagram-based Algorithm (AWVDA). GA is a computational model to search for the optimal solution by simulating the natural evolution process, and SA is a random optimization algorithm based on the annealing process of solids. Specifically, we compare the performance of the following algorithms:

1) VDA+GA: VDA is used to achieve the geographic partition, and GA is used to allocate tasks.

2) VDA+SA: VDA is used to achieve the geographic partition, and SA is used to allocate tasks.

3) VDA+RL: VDA is used to achieve the geographic partition, and our Reinforcement Learning (RL) model is used to allocate tasks.

4) AWVDA+GA: AWVDA is used to achieve the geographic partition, and GA is used to allocate tasks.

5) AWVDA+SA: AWVDA is used to achieve the geographic partition, and SA is used to allocate tasks.

6) AWVDA+RL: AWVDA is used to achieve the geographic partition, and our RL model is used to allocate tasks.

Three main metrics are compared among the above algorithms, i.e., CPU time, total number of allocated tasks, and average allocated task number difference (cf. Equation 2), for finding the final task allocation. Table 2 shows our experimental settings, where the default values of all parameters are underlined. All the experiments are implemented on an Intel(R) Xeon(R) CPU E5-2680 v4 @ 2.40GHz, and NVidia GeForce RTX 2080Ti GPU.

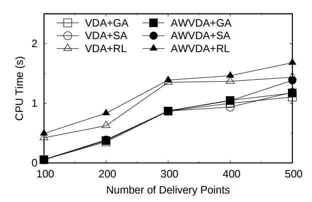
Table 2: Experiment Parameters

Parameter	Value
Number of delivery points (GM), $\|P\|$	100, 200, 300, 400, 500
Number of delivery points (SYN), $\|P\|$	1K, 2K, 3K, 4K, 5K
Number of workers (GM), $\|W\|$	100, 200, 300, 400, 500
Number of workers (SYN), $\|W\|$	1K, 2K, 3K, 4K, 5K
Number of allocation centers (GM), $\|C\|$	5, 10, 15, 20, 25
Number of allocation centers (SYN), $\|C\|$	50, 100, 150, 200, 250
Number of tasks (GM), $\|S\|$	200, 400, 600, 800, 1000
Number of tasks (SYN), $\|S\|$	2K, 4K, 6K, 8K, 10K
Expiration time of tasks (GM, SYN), e	0.5h, 1h, 1.5h, 2h, 2.5h

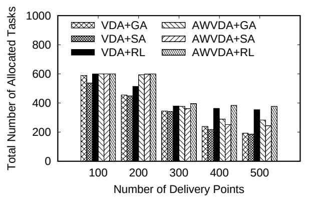
5.2 Experimental Results

Effect of $|P|$**.** To study the scalability of all the methods, we generate 5 datasets containing 100 to 500 (1000 to 5000) delivery points by random selection from the GM (SYN). In Figures 2(a) and 3(a), the CPU time of all methods is on the rise as $|P|$ increases. Moreover, independently of $|P|$, VDA-related methods (i.e., VDA+GA, VDA+SA, and VDA+RL) always run slower than their counterparts (i.e., AWVDA-related methods including AWVDA+GA, AWVDA+SA, and AWVDA+RL), while they also allocate fewer tasks and have imbalanced allocations of tasks when compared to their counterparts. As shown in Figure 2(b) and 3(b), the total number of allocated tasks of all the methods exhibits a decreasing trend when $|P|$ grows. All our proposed methods have similar performance when the number of delivery points is low, which means that there are not many benefits gained from the optimizations. However, the benefits of AWVDA become more obvious when $|P| \geq 300$ (3000). Apparently, AWVDA+RL achieves the highest total number of allocated tasks in both GM and SYN. Even when $|P| = 5000$ in SYN, AWVDA+RL can still complete up to 99.9% of all tasks. Another observation is that, AWVDA-related methods are consistently higher than VDA-related methods in terms of the total number of allocated tasks, which demonstrates the superiority of the adaptive weight strategy. In Figures 2(c) and 3(c), with more delivery points, each delivery point is more likely to have a more similar number of tasks, and the result is that workers are allocated a more balanced number of tasks. RL method can allocate tasks more reasonably by considering the importance of workers to delivery points, so with more delivery points, RL-related methods (i.e., VDA+RL and AWVDA+RL) show the fastest decrease on the average allocated task number difference and achieve the lowest ones, followed by GA-related methods (i.e., VDA+GA and AWVDA+GA) and SA-related methods (i.e., VDA+SA and AWVDA+SA) in both the GM and SYN. When $|P| \geq 200(2000)$, the average allocated task number differences obtained by the RL-related methods are less than 2.

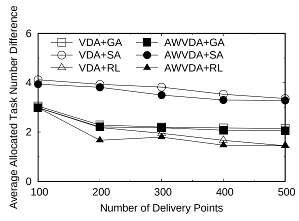
Effect of $|W|$**.** Next, we study the effect of $|W|$, the number of workers to be allocated. As illustrated in Figures 4(a) and 5(a), with the varying $|W|$, we can see that the CPU time shows an



(a) CPU Time

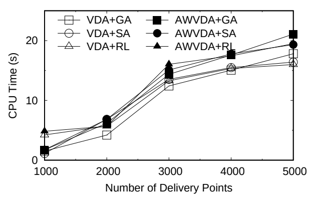


(b) Total Number of Allocated Tasks

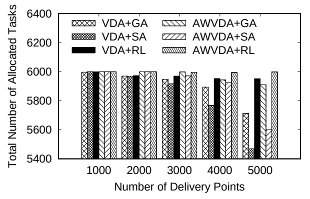


(c) Average Allocated Task Number Difference

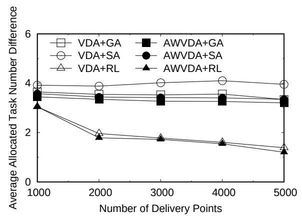
Figure 2: Performance of Task Allocation: Effect of $|P|$ on GM



(a) CPU Time

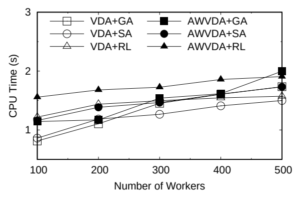


(b) Total Number of Allocated Tasks

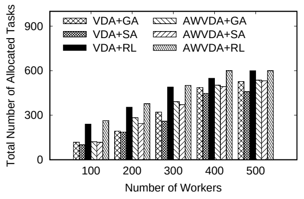


(c) Average Allocated Task Number Difference

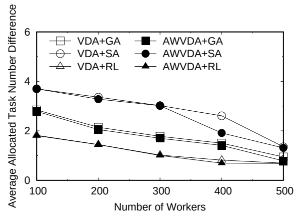
Figure 3: Performance of Task Allocation: Effect of $|P|$ on SYN



(a) CPU Time

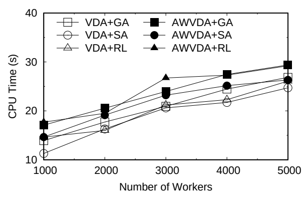


(b) Total Number of Allocated Tasks

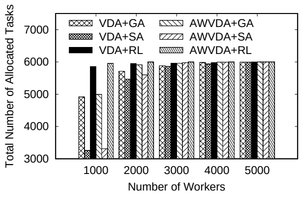


(c) Average Allocated Task Number Difference

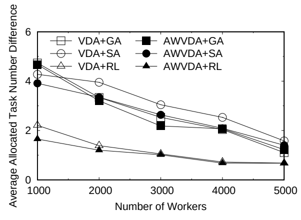
Figure 4: Performance of Task Allocation: Effect of $|W|$ on GM



(a) CPU Time



(b) Total Number of Allocated Tasks



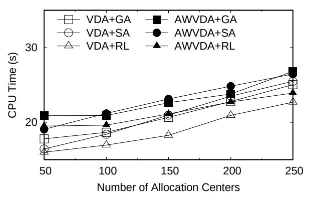
(c) Average Allocated Task Number Difference

Figure 5: Performance of Task Allocation: Effect of $|W|$ on SYN

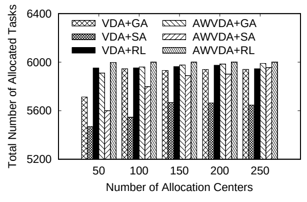
upward trend, since more workers means more allocations. Although AWVDA-related methods are more time-consuming than VDA-related methods, the computing efficiency of AWVDA-related methods is acceptable. As shown in Figures 4(b) and 5(b), RL-related methods generate the highest number of completed tasks followed by GA-related methods and SA-related methods. More specifically, in SYN, when $|W| = 1000$, the RL-related methods allocate almost all tasks. They even allocate 20% more tasks than the GA-related methods. In Figures 4(c) and 5(c), with the increase of $|W|$, the average allocated task number differences of all methods gradually decrease. RL-related methods have the smallest ones in two datasets, which proves the effectiveness of our RL-based task allocation algorithm. In addition, AWVDA has a certain role in reducing the average allocated task number difference and our AWVDA+RL method obviously has the best performance. This again confirms the effectiveness of our methods. To save space, in the following experiments, we will not report the results of the gMission dataset, which show similarity to those of the synthetic dataset.

Effect of $|C|$. We next study the effect of $|C|$. In Figure 6(a), with the number of allocation centers $|C|$ increasing, the CPU time gradually rises for all methods. When $|C| \geq 100$, the RL-related methods have the lowest CPU time. As illustrated in Figure 6(b), we can see that the AWVDA-related methods allocate more tasks than their counterparts. RL-related methods perform the best, followed by the GA-related methods and the SA-related methods. In Figure 6(c), there is no doubt that AWVDA+RL has the best performance and strongest stability. Another observation is that, the gap between AWVDA-related methods and their counterparts is increasing with more allocation centers, since AWVDA is easier to partition tasks with more reasonable number for each allocation center.

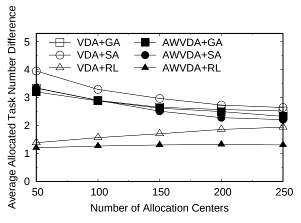
Effect of $|S|$. Figure 7 shows the effect of the number of tasks $|S|$. With the increase of $|S|$, the CPU time of all the methods grows



(a) CPU Time

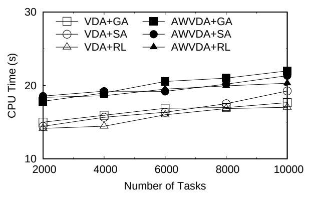


(b) Total Number of Allocated Tasks

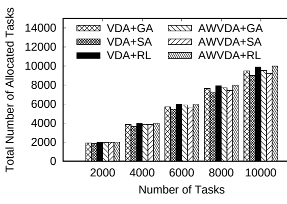


(c) Average Allocated Task Number Difference

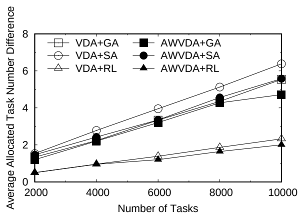
Figure 6: Performance of Task Allocation: Effect of $|C|$ on SYN



(a) CPU Time

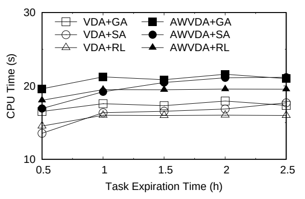


(b) Total Number of Allocated Tasks

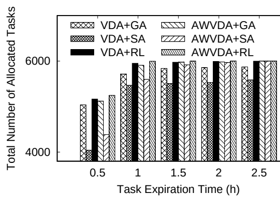


(c) Average Allocated Task Number Difference

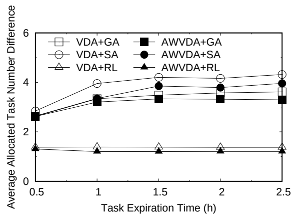
Figure 7: Performance of Task Allocation: Effect of $|S|$ on SYN



(a) CPU Time



(b) Total Number of Allocated Tasks



(c) Average Allocated Task Number Difference

Figure 8: Performance of Task Allocation: Effect of e on SYN

slowly in Figure 7(a), since the numbers of workers and delivery points have not changed. There is no doubt that the number of tasks of all methods is increasing in Figure 7(b). The AWVDA-related methods allocate more tasks with the different number of tasks. In Figure 7(c), with the increase of $|S|$, each delivery point is allocated more tasks. As a result, the difference in the number of tasks of the delivery points becomes larger, so the average allocated task number differences of all methods show a clear upward trend. The gap between the RL-related methods and others is increasing. This shows that for a more imbalanced distribution of tasks, our task allocation algorithm has a better performance.

Effect of *e*. In the final set of experiments, we study the effect of e. Not surprisingly, as can be seen in Figure 8(a), the tendency of all the methods are ascending at first and stable afterwards. This is because initially, with larger e, each worker tends to have more reachable tasks and allocating these tasks needs more CPU time. Then, as the expiration time of tasks continue to be extended, the number of reachable tasks for each worker will keep stable due to other spatio-temporal constraints such that the CPU time of finding the task allocation maintains stability. In Figure 8(b), AWVDA+RL still generates the maximal number of allocated tasks, and the AWVDA-related methods outperform the VDA-related methods. When $e =$ 0.5h, all the methods cannot allocate tasks well, but the AWVDA-related methods still perform better than their counterparts. In such circumstances, the benefits of AWVDA become more obvious. In Figure 8(c), the RL-related methods achieve the lowest average difference, followed by the GA-related and SA-related methods.

6 CONCLUSION AND FUTURE WORK

The development of the 5G communication technology and the sharp reduction of hardware cost offer a capable foundation for the deployment of Spatial Crowdsourcing (SC), which consists of location-specific tasks and requires workers to physically be at specific locations to complete them. In this paper, we propose and offer solutions to a problem termed Multi-Center-based Task Allocation (MCTA), which aims to maximize the allocated task number and achieve the allocation fairness among workers. To settle the intractable complexity of this problem, we propose a Task Allocation with Geographic Partition (TAGP) framework. More specifically, we first utilize a Voronoi diagram mechanism (including a basic Voronoi diagram-based algorithm and an adaptive weighted Voronoi diagram-based algorithm) to decompose a complex multi-center graph (i.e., the whole study are) into multiple smaller single-center-based graphs. Then we adopt a Reinforcement Learning method to allocate tasks by transforming the task allocation problem into a multiple traveling salesman problem. To the best of our knowledge, this is the first study in SC that allocates tasks based on adaptive geographic partition. An empirical study with real and synthetic data offers evidence that our proposed solutions can improve the effectiveness and efficiency of task allocation.

ACKNOWLEDGMENTS

This work is partially supported by Natural Science Foundation of China (No. 61972069, 61836007 and 61832017).

REFERENCES

[1] N. Abdullah, M. M. Rahman, M. Rahman, and K. I. Ghauth. A framework for optimal worker selection in spatial crowdsourcing using bayesian network. *Access*, 8:120218–120233, 2020.

[2] Afra A. Alabbadi and M. Abulkhair. Multi-objective task scheduling optimization in spatial crowdsourcing. *Algorithms*, 14:77, 2021.

[3] C. Blum and A. Roli. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *Comput. Surv.*, 35:268–308, 2003.

[4] Z. Chen, P. Cheng, Y. Zeng, and L. Chen. Minimizing maximum delay of task assignment in spatial crowdsourcing. *ICDE*, pages 1454–1465, 2019.

[5] Z. Chen, Peng Cheng, Liquan Chen, Xuemin Lin, and C. Shahabi. Fair task assignment in spatial crowdsourcing. *PVLDB*, 13:2479 – 2492, 2020.

[6] Z. Chen, Rui Fu, Ziyuan Zhao, Z. Liu, Leihao Xia, Lei Chen, P. Cheng, Caleb Chen Cao, Yongxin Tong, and C. Zhang. gmission: A general spatial crowdsourcing platform. *PVLDB*, 7:1629–1632, 2014.

[7] P. Cheng, L. Chen, and J. Ye. Cooperation-aware task assignment in spatial crowdsourcing. *ICDE*, pages 1442–1453, 2019.

[8] P. Cheng, Xun Jian, and Lei Chen. An experimental evaluation of task assignment in spatial crowdsourcing. *PVLDB*, 11:1428–1440, 2018.

[9] P. Cheng, Xiang Lian, Lei Chen, and C. Shahabi. Prediction-based task assignment in spatial crowdsourcing. *ICDE*, pages 997–1008, 2017.

[10] P. Cheng, Xiang Lian, Xun Jian, and Lei Chen. Frog: A fast and reliable crowdsourcing framework. *TKDE*, 31:894–908, 2019.

[11] Y. Cheng, Boyang Li, Xiangmin Zhou, Y. Yuan, G. Wang, and L. Chen. Real-time cross online matching in spatial crowdsourcing. *ICDE*, pages 1–12, 2020.

[12] Yue Cui, Liwei Deng, Yan Zhao, Bin Yao, Vincent W Zheng, and Kai Zheng. Hidden poi ranking with spatial crowdsourcing. In *KDD*, pages 814–824, 2019.

[13] K. Deb, S. Agrawal, Amrit Pratap, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *Trans. Evol. Comput.*, 6:182–197, 2002.

[14] Dingxiong Deng, C. Shahabi, and L. Zhu. Task matching and scheduling for multiple workers in spatial crowdsourcing. *SIGSPATIAL*, pages 21:1–21:10, 2015.

[15] Rohith Gandhi Ganesan, Samantha Kappagoda, Giuseppe Loianno, and David K. A. Mordecai. Comparative analysis of agent-oriented task assignment and path planning algorithms applied to drone swarms. *ArXiv*, abs/2101.05161, 2021.

[16] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. In *ICML*, volume 70, pages 1263–1272, 2017.

[17] Yujiao Hu, Yuan Yao, and Wee Sun Lee. A reinforcement learning approach for optimizing multiple traveling salesman problems over graphs. *KNOWL-BASED SYST*, 204:106244, 2020.

[18] Ziyun Huang, Danny Ziyi Chen, and Jinhui Xu. Influence-based voronoi diagrams of clusters. *Comput. Geom.*, 96:101746, 2021.

[19] Leyla Kazemi and Cyrus Shahabi. Geocrowd:enabling query answering with spatial crowdsourcing. In *GIS*, pages 189–198, 2012.

[20] S. Kirkpatrick, C. D. Gelatt, and M. Vecchi. Optimization by simulated annealing. *Science*, 220:671 – 680, 1983.

[21] Xiang Li, Yan Zhao, Jiannan Guo, and Kai Zheng. Group task assignment with social impact-based preference in spatial crowdsourcing. In *DASFAA*, pages 677–693, 2020.

[22] Xiang Li, Yan Zhao, Xiaofang Zhou, and Kai Zheng. Consensus-based group task assignment with social impact in spatial crowdsourcing. *Data Science and Engineering*, 5(4):375–390, 2020.

[23] Feng Lin, Jianhao Wei, Junyi Li, J. Zhang, and Bo Yin. Local privacy-preserving dynamic worker locations in spatial crowdsourcing. *Access*, 9:27359–27373, 2021.

[24] Qiyu Liu, Libin Zheng, Y. Shen, and L. Chen. Finish them on the fly: An incentive mechanism for real-time spatial crowdsourcing. In *DASFAA*, 2020.

[25] Ellen Mitsopoulou, Juliana Litou, and V. Kalogeraki. Multi-objective online task allocation in spatial crowdsourcing systems. *ICDCS*, pages 1123–1133, 2020.

[26] R. Necula, Mihaela Breaban, and Madalina Raschip. Tackling the bi-criteria facet of multiple traveling salesman problem with ant colony systems. *ICTAI*, pages 873–880, 2015.

[27] Wangze Ni, P. Cheng, L. Chen, and Xuemin Lin. Task allocation in dependency-aware spatial crowdsourcing. *ICDE*, pages 985–996, 2020.

[28] Yuxin Niu, Y. Zhang, and M. Song. Pricing models for crowdsourcing tasks based on geographic information. *ICIS*, pages 794–797, 2018.

[29] Neil Robertson and Paul Seymour. Graph minors. ii. algorithmic aspects of tree-width. *Journal of Algorithms*, 7(3):309–322, 1986.

[30] Qian Tao, Yongxin Tong, Zimu Zhou, Yexuan Shi, L. Chen, and K. Xu. Differentially private online task assignment in spatial crowdsourcing: A tree-based approach. *ICDE*, pages 517–528, 2020.

[31] Yongxin Tong, Yu xiang Zeng, Bolin Ding, L. Wang, and L. Chen. Two-sided online micro-task assignment in spatial crowdsourcing. *TKDE*, 33:2295–2309, 2021.

[32] Luan Tran, Hien To, Liyue Fan, and C. Shahabi. A real-time framework for task assignment in hyperlocal spatial crowdsourcing. *TIST*, 9:1 – 26, 2018.

[33] Jiayang Tu, P. Cheng, and L. Chen. Quality-assured synchronized task assignment in crowdsourcing. *TKDE*, 33:1156–1168, 2021.

[34] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NIPS*, pages 5998–6008, 2017.

[35] Yang Wang, Chenxi Zhao, and Shanshan Xu. Method for spatial crowdsourcing task assignment based on integrating of genetic algorithm and ant colony optimization. *Access*, 8:68311–68319, 2020.

[36] Z. Wang, Yubing Li, Kun Zhao, W. Shi, Liangliang Lin, and J. Zhao. Worker collaborative group estimation in spatial crowdsourcing. *Neurocomputing*, 428:385–391, 2021.

[37] Jinfu Xia, Yan Zhao, Guanfeng Liu, Jiajie Xu, Min Zhang, and Kai Zheng. Profit-driven task assignment in spatial crowdsourcing. In *IJCAI*, pages 1914–1920, 2019.

[38] Zhenyu Zhang and W. S. Lee. Deep graphical feature learning for the feature matching problem. *ICCV*, pages 5086–5095, 2019.

[39] Yan Zhao, Jiannan Guo, Xuanhao Chen, Jianye Hao, Xiaofang Zhou, and Kai Zheng. Coalition-based task assignment in spatial crowdsourcing. In *ICDE*, pages 241–252, 2021.

[40] Yan Zhao, Yang Li, Yu Wang, Han Su, and Kai Zheng. Destination-aware task assignment in spatial crowdsourcing. In *CIKM*, pages 297–306, 2017.

[41] Yan Zhao, Jinfu Xia, Guanfeng Liu, Han Su, Defu Lian, Shuo Shang, and Kai Zheng. Preference-aware task assignment in spatial crowdsourcing. In *AAAI*, pages 2629–2636, 2019.

[42] Yan Zhao, Kai Zheng, Yue Cui, Han Su, Feida Zhu, and Xiaofang Zhou. Predictive task assignment in spatial crowdsourcing: a data-driven approach. In *ICDE*, pages 13–24, 2020.

[43] Yan Zhao, Kai Zheng, Jiannan Guo, Bin Yang, Torben Bach Pedersen, and Christian S Jensen. Fairness-aware task assignment in spatial crowdsourcing: Game-theoretic approaches. In *ICDE*, pages 265–276, 2021.

[44] Yan Zhao, Kai Zheng, Yang Li, H. Su, Jiajun Liu, and Xiaofang Zhou. Destination-aware task assignment in spatial crowdsourcing: A worker decomposition approach. *TKDE*, 32:2336–2350, 2020.

[45] Yan Zhao, Kai Zheng, Hongzhi Yin, Guanfeng Liu, Junhua Fang, and Xiaofang Zhou. Preference-aware task assignment in spatial crowdsourcing: from individuals to groups. *TKDE*, 2020.

[46] Libin Zheng and Lei Chen. Multi-campaign oriented spatial crowdsourcing. *TKDE*, 32:700–713, 2020.

[47] Feibai Zhu, Shushu Liu, Junhua Fang, and An Liu. Incentive-aware task location in spatial crowdsourcing. In *DASFAA*, volume 12681, pages 650–657, 2021.