Cost-efficient Heterogeneous Worker Recruitment under Coverage Requirement in Spatial Crowdsourcing

Ning Wang, *Student Member, IEEE,* and Jie Wu, *Fellow, IEEE*

Abstract—With the progress of mobile devices and the successful forms of using the wisdom of crowds, spatial crowdsourcing has attracted much attention from the research community. The idea of spatial crowdsourcing is recruiting a set of available crowds to finish the spatial tasks located in crowdsourcing locations, e.g., landmarks, by using their handheld devices. This paper addresses the worker recruitment problem in spatial crowdsourcing under the coverage and workload-balancing requirements. The coverage constraint means that any crowdsourcing location should be visited by at least one of the recruited workers to satisfy the Quality-of-Service requirement, e.g., traffic monitoring or climate forecast. In addition, we argue that each crowdsourcing operation has a cost in reality, e.g., data traffic or energy consumption and the resource may be limited at each crowdsourcing location. The objective of this paper is to solve a Coverage and Balanced Crowdsourcing Recruiting (CBCR) problem, which ensures the coverage requirement and minimizes the maximum crowdsourcing cost for any crowdsourcing location. We prove that the CBCR problem is NP-hard in the general case. Then, we discuss the CBCR problem in the 1-D scenario. In the 1-D scenario, we first propose a directionally coverage scheme and further extend it to a Polynomial-Time Approximation Scheme (PTAS) to trade-off the computation complexity and the performance. The performance can be bounded to 2 + ε, where ε can be an arbitrary small value. Then, we found that there exists a sub-optimal structure, and thus the dynamic programming approach is proposed to find the optimal solution in the 1-D scenario. In the general 2-D scenario, we first prove that it has a sub-modular property and thus the naive greedy algorithm has an approximation ratio of $\ln n + 1$. In addition, we propose a randomized rounding algorithm with an expectation bound of $O(\frac{\log n}{\log \log n})$. Extensive experiments on realistic traces demonstrate the effectiveness of the proposed algorithms.

Index Terms—Mobile edge computing, spatial crowdsourcing, task allocation, mobile networks.

1 Introduction

With the ubiquity of mobile devices and vehicles equipped with high-fidelity sensors and the development of wireless networks (e.g., WiFi and LTE) in past years, all kinds of data have become widely available and large in amount. Traditional infrastructure-based computing approaches or systems [1, 2] began to show their limitations, i.e, high system implementation costs, difficult-to-handle dynamic environments and hard-to-utilize kinds of big data that accelerate ubiquitous intelligence in the real-world. To address the aforementioned two challenges, spatial crowdsourcing [3] also called participatory crowdsourcing, has emerged in the past few years. The idea of spatial crowdsourcing is to recruit a set of people/vehicles, called workers, to actively collect and report data using their mobile devices for a given campaign. Therefore, there is no need to build a specific system using the ubiquitous sensors in smart devices/vehicles and enjoy a pay-as-you-go character. The difference between online crowdsourcing [4] and spatial crowdsourcing is that for online crowdsourcing, everyone can register as a worker to conduct a crowdsourcing task, however, spatial crowdsourcing consists of location-specific tasks, the people who agree to participate in the spatial crowdsourcing have to physically be at specific locations to complete the tasks. There are many applications, such as sharing economy (e.g., Uber and Waze [5, 6]), geographical data generation (e.g., OpenStreetMap [7]), and road traffic monitoring (e.g., Waze [6]).

Existing spatial crowdsourcing mechanisms [8–11] do not consider the coverage requirement, i.e., collected data from all crowdsourcing locations in an urban area. However, the coverage problem is the requirement of many practical applications, such as weather forecasts, e.g., DroneSense [13], and traffic route optimization, e.g., CalTel [14], Nericell [15], and GreenGPS [16]. For example, for the navigation service provider, it can provide a good route only after the server gets all traffic information in the monitoring area. Otherwise, if the server does not have the traffic information for certain road segment in the recommended route, the result might be far from optimal, e.g., an accident in the unknown road segment. That is, missing data from any crowdsourcing location might have a huge impact on the final result, therefore, the desired performance (e.g., QoS) cannot be guaranteed without the coverage constraint.

The second requirement, which is important but often ignored in spatial crowdsourcing, is the workload balancing requirement for each crowdsourcing location. In reality, there is a cost (e.g., battery consumption, time, cellular traffic, and pay-off) for each crowdsourcing operation, and the budget (e.g., battery, money, etc.) in each crowdsourcing location is limited. For example, workers collect environment/traffic information from a Wireless Sensor

• *N. Wang and J. Wu are with Department of Computer and Information Sciences, Temple University, USA, PA, 19122.*
E-mail: ning.wang@temple.edu

Manuscript received April 19, 2005; revised August 26, 2015.

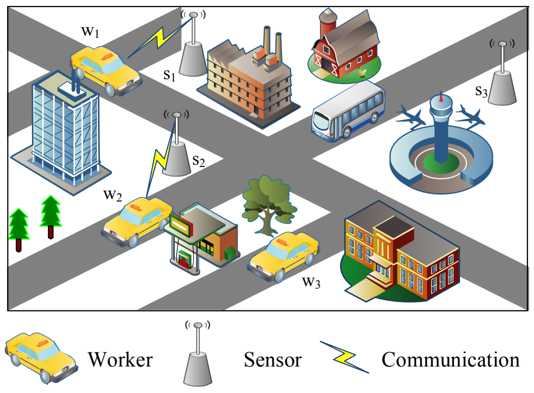


Fig. 1. An illustration of the network model in real applications.

Network (WSN) in a target area, and the data collection operation is a wireless communication operation and it will consume a certain energy. The energy consumption amount is related to the communication distance, current network Signal-to-Interference-plus-Noise Ratio (SINR), etc. The solar-charging sensor in each location has limited battery per day and if the communication energy consumption cannot exceed the any sensor's battery in the network, the whole WSN dies. Therefore, to maximize the WSN's lifetime, we need to balance the energy-consumption of each sensor and such a requirement has widely explored in WSNs [17]. Another application scenario is Waze carpool [6]. The different between Waze carpool with taxi and Uber is that Waze drivers already have their own driving routing, and drivers only pick passengers whose destination is on their commute routes. In such a scenario, there are limited crowdsourcing tasks, waiting for passengers, in each crowdsourcing location and if many workers arrive at a crowdsourcing location and find that there is no remaining task, it will discourage many people from taking part in the crowdsourcing platform. To avoid such a situation, the maximum cost for any crowdsourcing location should be minimized.

Motivated by the aforementioned two challenges, this paper addresses the worker recruitment problem in spatial crowdsourcing. In the proposed model, there are a set of spatial crowdsourcing locations and workers. The trajectory of each worker is known, which is reasonable considering that fact navigation softwares are widely used today and the trajectory information is generated and collected. In addition, there is a crowdsourcing cost when it visits a crowdsourcing location. Our goal is to find a recruitment solution which ensures all the crowdsourcing locations in an area can be visited by at least one worker, while minimizing the maximal cost for all crowdsourcing locations. We refer to this problem as the Coverage and Balanced Crowdsourcing Recruiting (CBCR) problem. The potential applications of the CBCR problem are intelligent applications in smart cities, e.g., the intelligent transportation system.

An illustration of the network model and CBCR problem is shown in Fig. 1, where there are four vehicles and 3 of them can be recruited as workers. There are 3 traffic sensors, which collect traffic-related information in this area and can communicate with vehicles through short-range communication interface, e.g., WiFi and Bluetooth. Among them, worker w_1 can collect data from sensors s_1 and s_2, worker w_2 can collect data from sensors s_2 and s_3, and worker w_3 does not collect data from sensors in this toy example. For each data collection, there is a communication energy consumption, which is related to the distance between them. Therefore, the worker w_1 has a larger cost for collecting data from sensor 2 than sensor 1. To satisfy the coverage constraint, workers w_1 and w_2 should be recruited together.

The proposed CBCR problem is proven to be NP-hard in the general 2-D scenario. Then, we discuss the solution of the CBCR problem in the 1-D scenario. In the 1-D scenario, we first observe that the naive greedy algorithm might lead to cost accumulation, and thus the maximal workload of a crowdsourcing location can be huge. Then, we propose a worker recruitment scheme by covering crowdsourcing locations directionally from one side to another side. Therefore, at most two workers will cover one crowdsourcing location, and thus the cost accumulation is controlled. After that, a PTAS extension is applied in the aforementioned coverage method to trade-off the computation complexity and the performance. The idea is that we divide workers into two sets, i.e., a cheap worker set and a costly worker set, and only select workers in the cheap worker set. Therefore, the performance can be bounded to $2 + \varepsilon$, where ε can be an arbitrary small value. Lastly, we found that a sub-optimal structure exists and thus the dynamic programming approach is proposed to find the optimal solution in the 1-D scenario. In the general 2-D scenario, the sub-optimal solution does not exist anymore. To address the CBCR problem, we first prove that it has a sub-modular property, and thus an approximation ratio of $\ln n + 1$. In addition, we propose a randomized rounding algorithm with an expectation bound.

The contributions of this paper are as follows:

- To our best knowledge, we are the first to consider the coverage requirement and the workload balancing of the crowdsourcing locations in spatial crowdsourcing.
- In the 1-D scenario, we propose a PTAS solution. Later, we propose a dynamic programming approach to find the optimal solution.
- In the general 2-D scenario, we prove that the proposed CBCR problem has a sub-modular property, and thus there is an approximation ratio of $1 + \ln n$ for the greedy algorithm.
- In the general 2-D scenario, we propose a randomized rounding algorithm which can solve the program effectively and with a high probability.
- The effectiveness of the proposed algorithms are demonstrated through three real datasets.

The remainder of the paper is organized as follows. The related works about spatial crowdsourcing and its main challenges are in Section II. The problem statement and its challenges are introduced in Section III. The greedy scheme and optimal solution in the 1-D scenario are provided and analyzed in Section IV. The general solution and its analysis for the 2-D scenario are presented in Section V. The experimental results are shown in Section VI. We conclude the paper in Section VII.

2 Related Works

With the wide adaptation of crowdsourcing applications, task coverage and participant selection in the crowdsourcing system have drawn much attention from researchers in recent years [8–11, 18–20]. We start from initial online crowdsourcing then move to the spatial crowdsourcing. The existing works in spatial crowdsourcing can be mainly categorized into two types based on whether the worker's trajectory is predetermined or not.

2.1 Worker Trajectory Planning

In this category, the worker's trajectory is controlled and planned by the server [10, 11, 21]. Therefore, the major problem is a trajectory planning problem, i.e., the planned trajectory of a worker can maximize the benefit. The most famous application example is Uber, some other applications are TaskRabbit and WeGoLook [22], which are crowdsourcing agent platforms. Previous researchers have produced many works where there is only one worker in their spatial crowdsourcing model. In [10], each task has a deadline and a feasible route of a worker should make sure that all the tasks in the route can be finished before their deadline. Facing this constraint, they proposed an approach to maximize the number of tasks that a worker can finish by using dynamic programming. In [8], the authors noticed that there are time conflicts for task assignments, which further complicates the planning problem. A mapping solution with pruning will reduce the complexity. The authors argues that there are multiple workers in the network and workers may have competing relationships. Therefore, the optimal trajectory for a particular worker might not be the optimal trajectory in terms of benefiting of the network. To address this situation, a simple greedy collaborative trajectory planning scheme is that if there are unassigned tasks and workers, the system selects a worker whose feasible trajectory achieves the maximum benefit. However, this approach does not really address the multiple worker collaborative crowdsourcing. That is also the reason why the method in [8] does not have a performance bound. Similar problems and workers are in [21, 23]. In [24], they considered the issue where tasks are not pre-known but arrived online. In [25], authors tried use semi-bandit learning method to maximize task reliability and minimize travel costs.

2.2 Trajectory Coverage

In this category, each worker's trajectory is predetermined [18, 26]. Therefore, the major problem is recruiting the most cost-efficient worker to maximize the crowdsourcing task. The application background is the sharing economy, it would be great if people could earn some benefits and not be bothered, i.e., detouring, by the crowdsourcing platform at the same time. An application example is Waze Carpool [6], people can earn some money by picking-up people on their commute trajectories. There are many theoretical studies on task assignments and participant selection problems, playing trade-offs among the crowdsourcing budgets and the coverage range [18, 26, 27]. The difference between their models and proposed model in this paper is that they consider maximizing the coverage range. However, in many scenarios, ensuring the coverage in a certain area, such as traffic monitoring or surveillance, with the minimal cost is very important. In [19], authors considered the coverage requirement in the crowdsourcing and minimized the overall recruiting cost. A greedy algorithm with a performance analysis is proposed. The difference between their works and the proposed work in this paper is that they consider the overall crowdsourcing cost. We argue that the workload balance is very important, thus, we consider the crowdsourcing cost for each crowdsourcing location. In [28], the authors considered the workload constraint but the coverage constraint was not considered. In [29], the authors considered the trade-off between load balance of each worker and utility maximization by modeling the recruitment as a Nash bargaining game. In [30], they considered the situation where the result of a worker may not reach a certain quality and argued that for each crowdsourcing location, the total quality of workers pass it should be maximized. In [31], authors jointly consider the amount of assigned tasks and coverage area of workers.

2.3 Time-involving Recruitment

Considering the fact that workers arrives in different time in spatial crowdsourcing systems, it is a fundamental problem to optimize the system performance in a dynamic way. Given the future trajectories of participants, authors in [29] consider how to maximize coverage in a series of time window with limited budget. In [24], authors consider an online user-task matching problem. The worker arrive in an online manner and every worker has to be assigned to a task upon its arrival. The object is to minimize the movement length of all workers. They provide a simple and greedy algorithm which turns out to be have a constant approximation ratio. In [32], they further considered tasks and workers all arrive in an online manner. However, in this case, they consider each task has a deadline and thus how to design the worker's trajectory so that the amount of processed tasks is maximized. In [33], they consider a 3-dimensional online matching, where the worker and requester have to meet a specific location and a bounded greedy algorithm was proposed which has a tighter competitive ratio. In [34], the authors considered the budget distribution problem overtime. Since the workers arrives dynamically, there is a trade-off to distribute the fixed budget in each time window or dynamically change the budget in each time window.

3 Model and Problem

In this section, we introduce the network model used in this paper, followed by the problem formulation. The hardness of the proposed problem is shown at the end.

3.1 Model

In this paper, we discuss a spatial crowdsourcing scenario under a centralized matter, which is the case in many spatial crowdsourcing applications (e.g., [5, 6, 22]). We assume there are n workers (e.g., people or vehicles) who agree to accept the crowdsourcing task at a time slot, $W = \{w_1, w_2, \cdots, w_n\}$. Each worker has a known crowdsourcing trajectory, t_i, which is reasonable since navigation

TABLE 1
Summary of symbols

Symbol	Interpretation
n	The total number of available workers
m	The total number of total crowdsourcing location
w_i	A available worker
t_i	The trajectory of worker w_i
x_i	A recruiting decision boolean variable for w_i
X	A recruiting decision vector for n available workers
l_i	A crowdsourcing location
c_{ij}	The crowdsourcing cost for w_i to location l_j
U	A set of selected workers
$f(U)$	The maximum crowdsourcing location cost using U

applications/devices are widely used today. For example, in Waze [6], the drivers share the traffic information on their way home. Therefore, we have a trajectory vector, T, $T = \{t_1, t_2, \cdots, t_n\}$. To simplify the illustration of the trajectory, the network is assumed to be discretized into grids. The trajectory of each worker is approximated based on grids, and the calculation error is bounded by the discretization level, i.e., the grid length. In addition, we assume that the spatial crowdsourcing area can be mapped into a 2-D grid topology and each crowdsourcing location belongs to a grid. Note that if there are multiple crowdsourcing locations in one grid, we will increase the discretization level until there is at most only one crowdsourcing location in one grid. There are two types of locations: a crowdsourcing location, denoted as a black block in Fig. 2, and a regular location. In a crowdsourcing location, there is a crowdsourcing task. In a regular location, there is not a crowdsourcing task. Suppose there are m crowdsourcing locations, $L = \{l_1, l_2, \cdots, l_m\}$. The length of t_i is denoted as $|t_i|$, which is the number of crowdsourcing locations that w_i passes. Workers can only move in four directions, up, down, left, and right, to mimic the way that people move along roads in reality. An illustration of the network model is shown in Fig. 2.

In this paper, we assume if there is a crowdsourcing task available for a worker, i.e., the trajectory of that worker covers the crowdsourcing location, the worker will conduct that crowdsourcing task. Therefore, there is no crowdsourcing task selection freedom for the worker. However, the proposed model can be extended to a general scenario where each worker has the freedom to select which crowdsourcing task it performs by a simple transformation. The transformation is that if a worker prefers not to conduct a crowdsourcing task for a crowdsourcing location, it is equivalent to the case that the worker takes a detour and does not pass that crowdsourcing location. In this paper, we have a 2-D cost vector, $\mathbf{C}$,

$$\mathbf{C} = \begin{pmatrix} c_{11} & c_{12} & \cdots & c_{1m} \\ c_{21} & c_{22} & \cdots & c_{2m} \\ \vdots & \vdots & \vdots & \vdots \\ c_{n1} & c_{n2} & \cdots & c_{nm} \end{pmatrix}.$$

There is a heterogeneous crowdsourcing cost, e.g., pay-off, energy consumption for each crowdsourcing operation, and c_{ij} denotes the cost for the crowdsourcing location l_j by the worker w_i. It is a more general model than the previous homogeneous model in [35]. In [35], the cost vector is a 1-D vector, where $C = \{c_1, \cdots, c_n\}$.

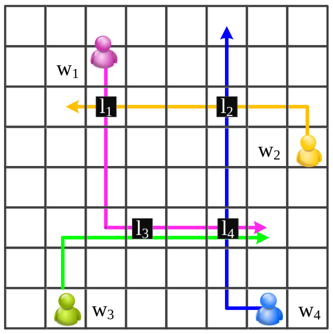


Fig. 2. An illustration of crowdsourcing model in this paper, where the grids with black blocks are spatial crowdsourcing locations.

A recruitment policy is a vector $X = \{x_1, x_2, \cdots, x_n\}$, which determines whether the worker w_i is selected ($x_i = 1$) or not ($x_i = 0$) to conduct the crowdsourcing task. The workload of a crowdsourcing location l_j is denoted as $\sum_{l_j \in t_i} c_{ij} x_i$. Note that there is a reachability issue for each worker. When calculating the crowdsourcing location workload of l_j, it counts the workload only when the recruited a worker trajectory t_i includes this crowdsourcing location, i.e., $l_j \in t_i$. If l_j does not in w_i's trajectory, i.e., $l_j \notin t_i$, the cost of recruiting w_i will not be added to the location l_j.

In addition, we use U to denote the set of workers which are selected in a recruitment policy. That is, $U = \{W | x_i = 1\}$ and $f(U)$ to denote the maximum location cost with set U.

3.2 Problem Formulation

To address the coverage requirement of spatial crowdsourcing, e.g., traffic monitoring, route recommendation, climate forecast, and surveillance systems, we argue that the data should be collected from all $|L|$ crowdsourcing locations before calculation to ensure a certain quality level. In addition, a practical issue in crowdsourcing is that there is a crowdsourcing cost for paying the visited workers and there should be a budget for each crowdsourcing location. Therefore, we propose the Coverage and Balanced Crowdsourcing Recruiting (CBCR) problem in this paper, which is formulated as follows.

As for the problem formulation, the proposed CBCR problem is changed into

$$\begin{aligned} \min \quad & \max_j \sum_{l_j \in t_i} c_{ij} x_i \\ \text{s.t.} \quad & \sum_{l_j \in t_i} x_i \geq 1, \quad \forall l_j \quad x_i \in \{0, 1\}, \end{aligned} \tag{1}$$

where the objective is to find a worker recruitment solution so that the maximum crowdsourcing location cost for all crowdsourcing locations is minimized and the constraint ensures that every crowdsourcing location is covered.

To illustrate the CBCR problem, there is a motivational example in Fig. 2, where there are 4 workers, w_1, w_2, w_3, and w_4, with 4 crowdsourcing locations, namely, l_1, l_2, l_3, and l_4. Assume workers have an identical cost for visiting any crowdsourcing location in this toy example. In Fig. 1, the worker w_1 visits l_1, l_3, and l_4. We cannot only select w_1, since this recruiting solution does not satisfy the coverage constraint. To satisfy the coverage constraint, there are three feasible recruitment strategies, i.e., $\{w_1, w_2\}$, $\{w_1, w_4\}$, and $\{w_2, w_3\}$. For the first two solutions, the crowdsourcing location l_1 or l_4 is covered twice. However, in the last solution, all the crowdsourcing locations are only covered once, which is better in terms of balancing the workload.

3.3 Hardness of CBCR problem

Theorem 1. *The proposed CBCR problem is NP-hard.*

Proof. To show the decision version of the CBCR (CBCR-D) $\in$ NP, suppose a recruiting policy, X, is given. Clearly, we can verify the correctness of X in polynomial time. Specifically, the complexity of the verification algorithm is $O(m)$, where m is the number of crowdsourcing locations.

To show that CBCR-D $\in$ NP-hard, we reduce the tripartite matching problem to it in polynomial time, which is NP-complete [36]. The tripartite matching problem can be formulated as: given sets B, G, and H, each containing n elements and a ternary relation $T \in B \times G \times H$, that is, T consists of triples (b, g, h) such that $b \in B, g \in G$, and $h \in H$. find a set of n triples in T, no two of which have a component in common. We present a polynomial time reduction and construct an instance of the CBCR-D as follows:

($\leftarrow$) Consider an instance of the CBCR-D in which $c_{ij} = 1$. m is a $multiple$ of 3 and m crowdsourcing locations can be partitioned into three equal sets. Each worker visits exactly 3 crowdsourcing locations from these 3 location sets respectively. It is equivalent to that in which we build a graph $G = (V, E)$, where V has m nodes and each node represents a crowdsourcing location. All nodes are partitioned into three sets and each of the sets has $m/3$ nodes. The nodes formulate sets B and G, and H. If a worker visits two locations successively, there is an edge between these two nodes. And for the locations that each worker visited formulate a 3-tuple set T, which contains one element from each of B, G, and H. We now argue that the instance of tripartite matching is a "yes" instance, i.e. there is an instance of the CBCR-D.

($\rightarrow$) If the constructed instance of the CBCR-D satisfies all the criterion, the selected trajectories form an instance of tripartite matching according to the problem definition. Therefore, the CBCR-D problem is NP-hard. □

4 1-D Scenario

In this scenario, we assume that all the crowdsourcing locations are in a line-topology. Application for this type of line-topology is road segment monitoring, e.g., a highway situation. On the highway, different vehicles enter in different entrances and leave at different exits. Without loss of generality, let us denote the crowdsourcing locations from one side to the another side as l_1 to l_m for the remainder of this section. The line topology simplifies the CBCR problem, since the overlapping relationship between workers is relatively simple.

Algorithm 1 MG algorithm

```
Input: The vectors of $T$ and $C$.
Output: The recruiting vector $X$.
1: while $\exists l \notin \cup t_i$ do
2:   Set $I_{min} = \infty$ and $idx = -1$
3:   for $i$ from 1 to $W \backslash U$ do
4:     for $j$ from 1 to $m$ do
5:       if $l_j \in t_i$ and $f(U \cup \{w_i\}) - f(U) < I_{min}$ then
6:         $I_{min} = f(U \cup \{w_i\}) - f(U)$, $idx = i$.
7:   Add $w_{idx}$ to $U$ and set $x_{idx} = 1$.
```

Algorithm 2 CO algorithm

```
Input: The vectors of $T$ and $C$.
Output: The recruiting vector $X$.
1: while $\exists l \notin \cup t_i$ do
2:   Set $I_{max} = -\infty$ and $idx = -1$
3:   for $\forall t$, where $l_i \in t$ do
4:     if $|\cup_{w_j \in U \cup \{w_i\}} t_j| - |\cup_{w_j \in U} t_j| > I_{max}$ then
5:       $I_{max} = |\cup_{w_j \in U \cup \{w_i\}} t_j| - |\cup_{w_j \in U} t_j|$, $idx = i$.
6:   Add $w_{idx}$ to $U$ and set $x_{idx} = 1$.
```

4.1 Greedy Approaches

In this subsection, we propose two greedy solutions first, followed by the performance analysis and limitations with the proposed greedy algorithms. The first naive greedy algorithm is widely used in many applications. We observe its drawbacks in the 1-D scenario and propose the second greedy algorithm to improve the naive greedy algorithm.

4.1.1 Naive greedy algorithm

A natural idea is to select the worker who increases the maximal workload cost the least to cover the network in each round. Therefore, we propose the Min-max Greedy (MG) algorithm as a baseline algorithm in this paper. The detailed algorithm is shown in Algorithm 1, which starts with an empty set. In each iteration, we check all the unassigned players and add the matchmaking which minimizes the marginal gain of the objective function, i.e.,

$$w \leftarrow \arg \min_{w_i \in W \backslash U} \{f(U \cup w_i) - f(U)\} \tag{2}$$

The drawback of the MG algorithm is that a single crowdsourcing location might be unnecessarily covered multiple times in the MG algorithm and such unnecessary coverage can accumulate and lead to a bad result.

Theorem 2. *In the optimal solution, a crowdsourcing location will not be covered by 3 workers in the 1-D scenario.*

Proof. This theorem can be proven by contradiction. If we project all the trajectories for a special crowdsourcing location into the road, we can always find a trajectory whose starting position is the left-most, and we can also find a trajectory whose end position is the right-most. Then, we can use these two trajectories, possibly as one trajectory, to cover this crowdsourcing location. □

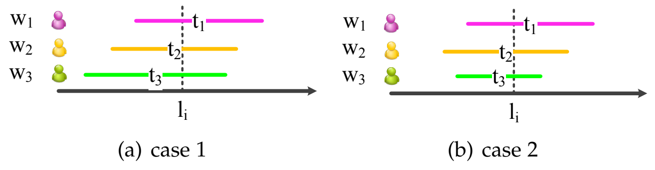


Fig. 3. Two cases for a crowdsourcing location.

4.1.2 Directional coverage algorithm

Facing this drawback of the MG algorithm and theorem 2, we propose the Coverage-Only (CO) algorithm, which covers all the crowdsourcing locations from one side to the other side, i.e., from l_1 to l_m. Each time, we select the worker in an uncovered area, which increases the coverage most from left to right. That is,

$$w \leftarrow \arg\max\{|\cup_{w_j \in U \cup \{w_i\}} t_j| - |\cup_{w_j \in U} t_j|\} \quad (3)$$

where $\cup_{w_j \in U} t_j$ denotes all crowdsourcing locations covered so far by set U, and $|\cup_{w_j \in U} t_j|$ denotes the number of covered locations by $\cup_{w_j \in U} t_j$. The detailed algorithm is shown in Algorithm 2, which selects a worker which can extend the coverage at most in a round. It is easy to prove that the result of the CO algorithm is always feasible and it can be proven by contradiction. The CO algorithm takes advantage of the contiguous trajectory overlapping property in the 1-D scenario, as shown in theorem 2, and achieves the performance bound, shown in theorem 3.

Theorem 3. *The CO algorithm has a $2c_{\max}$ approximation ratio in the 1-D scenario, where $c_{\max}$ equals to $\max \frac{c_{ij}}{c_{i'j'}}, \forall i, i', j, j'$.*

Proof. The proof insight is that each time, we guarantee that a new crowdsourcing location will be covered, so that it avoids unnecessary coverage accumulation. The problem can be proven through contradiction, assuming that there is a crowdsourcing location, l_i, that 3 workers can visit. Based on the end position of their trajectories from left to right, we denote these three trajectories as t_1, t_2, and t_3 and their starting positions as s_1, s_2, and s_3. Then, for the starting points of t_1 and t_2, two cases exist: $s_1 < s_2$ or $s_1 > s_2$, as shown in Fig. 3. In the first case, the CO algorithm will select t_1 rather than t_2. In the second case, the CO algorithm will select t_2 instead of t_1. Therefore, neither of two cases exist in the CO algorithm. Otherwise, there would be a contradiction. As a result, there are at most two workers visiting a crowdsourcing location and the cost is bounded by $2c_{\max}, c_{\max} = \max \frac{c_{ij}}{c_{i'j'}}, \forall i, i', j, j'$. □

4.2 CO-PTAS algorithm

We observe that the bad performance of the CO algorithm is due to the improper recruitment of workers with a large recruiting cost. Therefore, if these jobs are scheduled with a low priority, they cannot have a big influence on the final result. Based on this observation, we propose a Polynomial-Time Approximation Scheme (PTAS) algorithm. Given all the workers, we partition the workers into two sets: costly workers and cheap workers. Let us use r_i to denote the ratio between the maximum crowdsourcing cost of a worker and the maximum crowdsourcing cost in the network, i.e., $r_i = \max \frac{c_{ij}, \forall j}{c_{ij}, \forall i, j}$. We call a worker, w_i, a costly worker, if $r_i > \varepsilon/2$. Let B and S denote the set of costly workers and cheap workers respectively, i.e., $B = \{w_i : r_i > \varepsilon/2\}$ and $S = \{w_i : r_i \leq \varepsilon/2\}$.

Algorithm 3 CO-PTAS algorithm

```
Input: The vector of $T$, $C$, and $\varepsilon$.
1: for from $t_0$ to $t_n$ do
2:     if $c_{ij} \leq \frac{\varepsilon}{2} \max_{\forall i,j} c_{ij}$ $\forall i, j$ then
3:         Add $w_i$ to $S$.
4:     else
5:         Add $w_i$ to $B$.
6: Call CO algorithm for workers/trajectories in $S$.
7: if the result is a coverage then
8:     Modify the guess, $\varepsilon$, and repeat.
9: else
10:    Return the best result so far.
```

Algorithm 4 Dynamic programming algorithm

```
Input: The vector of $T$ and $C$.
1: Initialize the state record $d[\cdot\cdot]$
2: for check crowdsourcing location $i$ from $l_1$ to $l_m$ do
3:     for for all crowdsourcing locations $l_{i'}$, $l_{i'} \leq l_i$ do
4:         if $t_j$ can reach location $l_{i'}$ then
5:             $d[i,j] = min_{i'<i,j'\leq j} \max\{d[i',j'],$
6:                 $\max_{\forall l_k \in t_j} \{c_{j'k} + c_{jk}\}\}$
7:     Update the $d[i,j]$ for location $l_i$
8: Find min $d[m\cdot]$
9: Return $X$
```

The optimal ε can be found through the binary search. Initially, we find the maximum worker cost in the network. Then, we set $\varepsilon = 1$ and try to check if we can find a feasible schedule through the CO algorithm. If so, we decrease the value of ε to a half, otherwise we increase the value of ε by 2. After the ε is changed, we calculate the corresponding sets B and S and apply the CO algorithm in the new set S. Then, the problem becomes finding a feasible solution in set S. Then, according to theorem 2, the optimal value is at most twice this value.

Theorem 4. *The CO-PTAS algorithm can achieve a $2 + \varepsilon$ approximation ratio in the proposed problem.*

The proof is similar to the approximation ratio proof of the CO algorithm. The insight is that if all crowdsourcing operation costs are the same, the CO algorithm achieves an approximation of 2. The PTAS scheme gradually removes costly workers so that only the most cost-efficient workers are selected. For the complexity of the CO-PTAS algorithm, the CO-PATS algorithm at most calls the CO algorithm $\log(n)$ times according to the binary search. Therefore, the overall complexity of the CO-PTAS algorithm is $O(mnlog(n))$.

An illustration of these three algorithms is in Fig. 4, where there are four workers, and their crowdsourcing operation cost at each crowdsourcing location is shown in the figure. The MG algorithm will select w_1 first, since it will only increase by 1.4 at most, followed by w_4, which further increases by 0.6 for the maximum workload cost.

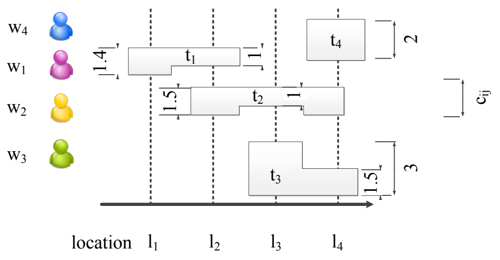


Fig. 4. An example of 1-D network in the heterogeneous model.

Then, to finish the coverage requirement, w_2 should also be selected. As a result, the maximum workload cost is 3.5 at crowdsourcing location l_4. As for the CO algorithm, it directionally covers from l_1 to l_4. Therefore, w_1 is selected first, followed by w_3 in the worst case. Then, the maximum workload cost is 3 at crowdsourcing location l_3. However, in the CO-PTAS algorithm, when $\varepsilon = 1$, w_3 will have a lower priority for being selected. Then, the CO-PTAS algorithm will use w_1 and w_2 to finish the coverage, and the maximum workload cost is 2.5 at crowdsourcing location l_2.

4.3 Dynamic Programming Approach

We notice that the trajectory of a worker can only have influence on contiguous locations in the 1-D scenario. Therefore, we can partition the problem into a series of sub-problems to further overcome the improper selection of cost trajectories.

Assume the crowdsourcing locations from one side to another side (e.g., from left to right) is $l_1, l_2, \cdots l_m$. Without loss of generality, we assume trajectories are ordered based on their ending positions. In dynamic programming, we maintain a 2-D vector $d[\cdot,\cdot]$ to store the best result so far. That is, $d[i,j]$ is the optimal solution from crowdsourcing locations l_1 to l_i, and j denotes that the last trajectory is used to cover location l_i. The objective of dynamic programming is to find $\min d[m,j], \forall j$. Note that if a trajectory, t_j, does not cover l_i, $d[i,j]$ equals ∞, which ensures that we will not select t_j to cover location l_i. Initially $d[0,j] = 0, \forall j$, which means that before we select any trajectories, the minimal maximum cost is 0. Then we can get the following relationship:

$$d[i,j] = \begin{cases} 0 & i = 0 \\ \underset{i'<i, j'\le j}{min} \max\{d[i',j'], \max_{\forall l_k \in t_j}\{c_{j'k} + c_{jk}\}\} & i > 0 \end{cases} \quad (4)$$

where location $l_{i'}$ is any crowdsourcing location that is no smaller than the first crowdsourcing location before the starting point of t_j. j' is any trajectory whose index is smaller than j. For example, if $j = 3$, l'_i can be crowdsourcing locations l_1 and l_2, $t_{j'}$ can be t_1 or t_2. The idea behind Eq. 4 is that if we want to find the optimal solutions up to crowdsourcing location l_i with trajectory t_j as the last trajectory, we only need to check all sub-optimal solutions from the previous location i' with trajectory $t_{j'}$ to ensure the coverage constraint. If adding t_j to the crowdsourcing location from $l_{i'}$ to l_i does not increase the maximum cost (i.e., $d[i',j'] > \max_{\forall l_k \in t_j}\{c_{j'k} + c_{jk}\}$), we keep the maximum cost, (i.e., $d[i,j] = d[i',j']$). Otherwise, we update the maximum cost of $d[i,j]$, (i.e., $d[i,j] = \max_{\forall l_k \in t_j}\{c_{j'k} + c_{jk}\}$).

Algorithm 5 Rounding algorithm

Input: The vector of T and C.
Output: The recruiting vector X.

```
1: Relax the problem into a linear programming.
2: Solve the relaxed linear programming problem and get
   fractional assignment vector $X^\star$.
3: for for $l_1$ to $l_m$ do
4:   for $\forall l_i \in t_i$ do
5:     Assign each $t_i$ to an interval between 0 and $\sum x_i^\star$.
6:     Randomly generate a value in the whole range.
7:     Pick the $x_i$, if the random value is in its interval.
```

Note that $c_{j'k}$ can be zero, in which case $t_{j'}$ ends at the previous crowdsourcing location before t_j starts. Note that the reason that we can maintain the number of trajectories covering one crowdsourcing location rather than the set of combinations of these trajectories is that the latter always leads to a worse result.

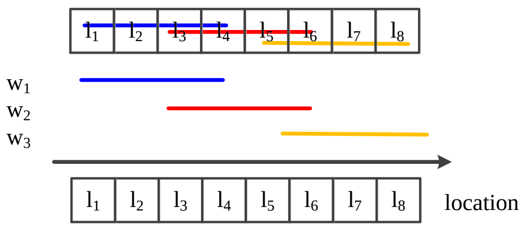
An example to illustrate the dynamic programming approach is shown in Fig. 4. For crowdsourcing location l_1, since it is only covered by trajectory t_i, $d[1,1]$ equals 1.4. Similarly, there are two trajectories covering crowdsourcing location l_2, thus, we have $d[2,1] = 1.4$ and $d[2,2] = 2.5$. The optimal solution that covers the crowdsourcing location from l_1 to l_2 is $\min\{d[2,1], d[2,2]\} = 1.4$. For crowdsourcing location l_3, there are two trajectories, t_2 and t_3, covering it. For t_2, it is already calculated in the previous crowdsourcing location, therefore, $d[3,2] = d[2,2] = 2.5$. For t_3, it checks all the previous solutions up until crowdsourcing location l_2. Therefore, $d[3,3] = \min\{\max\{d[2,1], c_{33}\}, \max\{d[2,2], c_{23} + c_{33}\}\} = 3$. In the former case, t_1 ends before t_3, hence $c_{13} = 0$. Therefore, the optimal solution up to l_3 is $\min\{d[3,2], d[3,3]\} = 2.5$. For crowdsourcing location t_4, the optimal solution up to l_4 is $\min\{d[4,2], d[4,3], d[4,4]\} = 2.5$.

The complexity analysis of dynamic programming is shown as follows. Dynamic programming requires storing $O(mn)$ states. For each state update, we need to check $O(mn \log n)$ times at most. This is because the algorithm needs to trace back to find the optimal solution in each previous location, $O(m)$, and check the optimal solution at that crowdsourcing location, which is $O(n)$. The dynamic programming approach still needs a sorting algorithm to find the smallest c_j, which is $O(\log n)$. Thus, the overall time complexity is $O(m^2n^2 \log n)$.

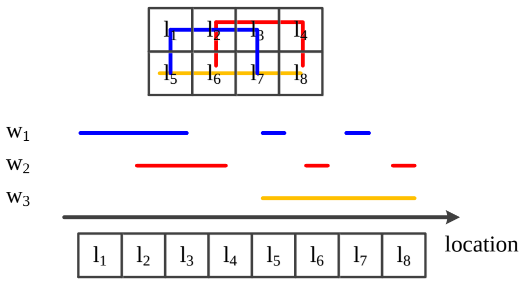
5 General 2-D Scenario

In this section, we further discuss the general solution in the 2-D scenario by exploring the inherent property of the CBCR problem. The application scenario for the 2-D scenario is traffic monitoring in an urban area.

In the 1-D scenario, the overlapping relationship between different trajectories is simple. There are only two cases in total for two trajectories. (1) They do not have overlapping relationships with each other. (2) They overlap with each other, and all the overlapping locations are contiguous. However, there is no such property in the general 2-D case. An illustration of this property in the 1-D scenario is shown



(a) 1-D scenario



(b) 2-D scenario

Fig. 5. The trajectory of workers in the 1-D and 2-D scenarios.

in Fig. 5, where there are 8 crowdsourcing locations. In Fig. 5(a), if we map workers' trajectories into a 1-D dimension, their trajectories are contiguous. In Fig. 5(a), w_1 and w_3 do not overlap with each other. Workers w_2 and w_3 overlap at crowdsourcing locations l_3 and l_4, which are adjacent to each other. However, in Fig. 5(b), if we map workers' trajectories into a 1-D dimension, their trajectories might be discontinuous, e.g., w_1's trajectory is $\{l_1, l_2, l_3, l_5, l_7\}$ in Fig. 5(b). Its trajectory overlaps with worker w_3's trajectory at crowdsourcing locations l_5 and l_7, which are not adjacent with each other. Therefore, there is no sub-optimal structure, that is, the optimal solution in a sub-problem may have an impact on the later recruitment assignment.

5.1 Sub-modular property

Due to the fact that the CBCR problem is NP-hard in the 2-D scenario, it is impossible to try every combination and backtracking if a matchmaking combination leads to a bad result. Instead, we try to gradually expand the matchmaking assignment. However, during the expanding procedure, the error might increase. We prove that the error can be bounded. That is the insight of the sub-modular property.

Theorem 5. *The objective function, $f(U)$, of the CBCR problem is nonnegative, monotone, and sub-modular.*

Proof. According to the definition of cost, it has a minimal value of 0 and it cannot be negative. Therefore, $f(U)$ is nonnegative. If there exists two sets U' and U'', and $U' \subseteq U'' \subseteq U$. Let us denote the worker set whose trajectories cover the crowdsourcing location l_k as U'_k, and U''_k, respectively. Clearly, $U'_k \subseteq U''_k$, otherwise, there is a contradiction that $U' \subseteq U''$. Then, based on the inclusion relation, $f_k(U'_k) = \sum_{w_i \in U'} c_{ik} \leq f_k(U''_k) = \sum_{w_i \in U''} c_{ik}$ and $f_k(U) = \max_{k \in [1,m]} f_k(U_k)$, where $f_k(\cdot)$ is the objective value in l_k. Therefore, $f(U') \leq f(U'')$ and $f(U)$ is monotone.

Based on the calculation of $f_k(U_k)$, for any crowdsourcing location l_k,

$$f_k(U_k \cup w'_i) = \max\{f(U_k), \sum_{u_i \in \{U_k \cup w'_i\}} c_{ik}\}. \tag{5}$$

Let us denote $w_{i'}$ and $w_{i''}$ as two newly recruited workers and $w_{i'}$ and $w_{i''} \in W \backslash U$. If the following inequation is true,

$$f(U \cup \{w_{i'}\}) + f(U \cup w_{i''}\}) \geq f(U \cup \{w_{i'}, w_{i''}\}) + f(U), \tag{6}$$

the CBCR problem is submodular. When $w_{i'}$ and $w_{i''}$ do not have overlaps in their trajectories, the two sides of above inequation are equal since $w_{i'}$ and $w_{i''}$ have no influence on each other. Therefore, we focus on the following condition, where $w_{i'}$ and $w_{i''}$ have overlaps in their trajectories at crowdsourcing location g_k, that is,

$$\begin{aligned} &f_k(U_k \cup \{w_{i'}\}) + f_k(U_k \cup w_{i''}\}) \\ &\geq f_k(U_k \cup \{w_{i'}, w_{i''}\}) + f(U_k), \end{aligned} \tag{7}$$

Based on Eq. 5, we prove that the inequation 7 is true in all the following cases.

(1) if $\sum_{w_i \in \{U_k \cup w'_i\}} c_{ik} \leq f_k(U_k)$ and $\sum_{w_i \in \{U_k \cup w''_i\}} c_{ik} \leq f_k(U_k)$. In this case, the two sides of InEq. 7 are the same and equal to $2f_k(U_k)$ according to Eq. 5. Therefore, InEq. 6 is true. (2) if $\sum_{w_i \in \{U_k \cup w'_i\}} c_{ik} \geq f_k(U_k)$ and $\sum_{w_i \in \{U_k \cup w''_i\}} c_{ik} \geq f_k(U_k)$, the left side of Eq. 6 is $\sum_{w_i \in \{U_k \cup w'_i\}} c_{ik} + \sum_{w_i \in \{U_k \cup w''_i\}} c_{ik} \geq \sum_{u_i \in \{U_k \cup \{w'_i, w''_i\}\}} w_i + f_k(U_k)$ and InEq. 6 is true. (3) if $\sum_{w_i \in \{U_k \cup w'_i\}} c_{ik} \geq f_k(U_k)$ or $\sum_{u_i \in \{U_k \cup w''_i\}} w_i \geq f_k(U_k)$, two sides of InEq. 7 equal to $\sum_{w_i \in \{U_k \cup \{w'_i, w''_i\}\}} c_{ik} + f_k(U_k)$ and InEq. 7 is true. Therefore, $f(U)$ is sub-modular. □

According to the results in [37], the MG algorithm in the general case has an approximation ratio of $1 + \ln n$.

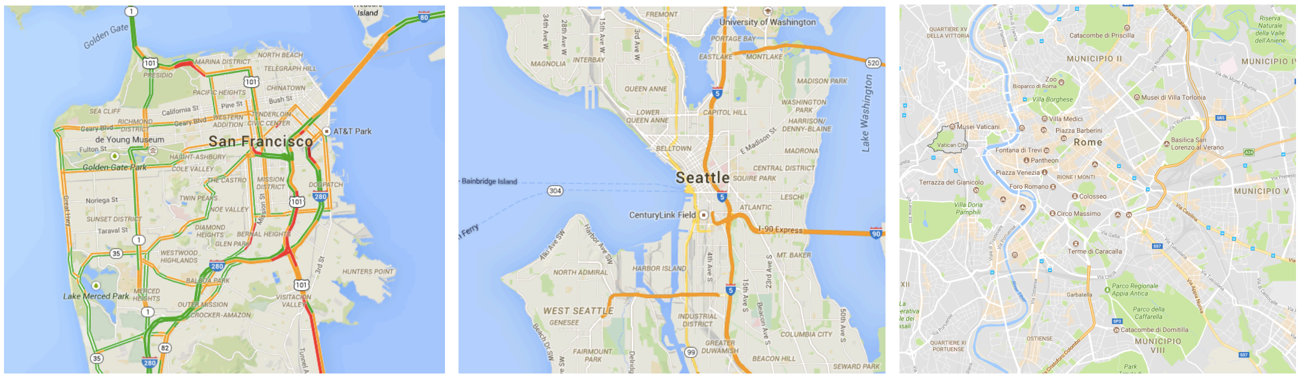
5.2 Randomized Rounding Approach

In this subsection, we propose to using a rounding technique [38] to propose a randomized rounding algorithm. For the original problem in Eq. 1, it is equivalent to finding a smallest value θ, which ensures that the workload of any crowdsourcing location is no larger than θ. Then, if we relax the formulation from $x_i \in \{0, 1\}$ to $x_i \in [0, 1]$, Eq. 1 becomes a linear programming problem as follows

$$\begin{aligned} \min \quad & \theta \\ \text{s.t.} \quad & \sum_{l_j \in t_i} c_{ij} x_i \leq \theta, \\ & \sum_{l_j \in t_i} x_i \geq 1, \qquad x_i \in [0, 1] \quad \forall i, j. \end{aligned} \tag{8}$$

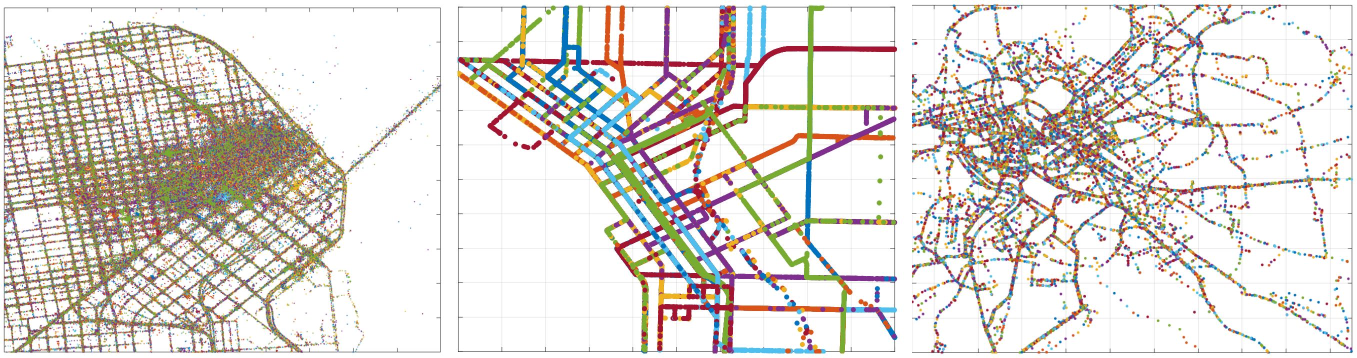
Eq. 8 can be optimally solved using the linear programming solver and therefore, we can get the optimal $\theta^\star$ and the corresponding assignment vector $\{x_1^\star, x_2^\star, \cdots, x_n^\star\}$.

For the original problem, we can use $\{x_1^\star, x_2^\star, \cdots, x_n^\star\}$ to get a randomized rounding result. In detail, the randomized rounding algorithm gives each worker a probability of being recruited. For every crowdsourcing location l_i from l_1 to l_m,



(a) San Francisco (b) Seattle (c) Rome

Fig. 6. The city map



(a) San Francisco (b) Seattle (c) Rome

Fig. 7. The vehicles' movement history

we check all trajectories that include l_i. These trajectories are selected based on the their corresponding $x^\star$ in the linear programming solution. For example, we have three workers with $x_1^\star = 0.5$, $x_2^\star = 0.3$, and $x_3^\star = 0.2$ who visit a crowdsourcing location. The randomized rounding algorithm will randomly generate a number between 0 and 1, the sum of 0.5, 0.3, and 0.2. If the number is between 0 and 0.5, the randomized rounding algorithm will pick the first worker. If the number is between 0.5 and 0.8, it will pick the second worker, and if the number is between 0.8 and 1, it will pick the third worker. The result generated by the randomized rounding algorithm is always a feasible solution. This is because the random assignment ensures that one trajectory for a crowdsourcing location will be selected.

The performance of the randomized rounding algorithm is as follows. Since all workers are selected with probability x_i independently, we get

$$\begin{aligned} Pr[\sum_{T:l_j \in t_i} c_{ij}E[x_i]] &= \sum_{T:l_j \in t_i} c_{ij}E[x_i] \\ &= \sum c_{ij} \cdot x_i^\star \leq \theta^\star, \end{aligned} \tag{9}$$

the expected cost on any location is at most $\theta^\star$. However, since we have many crowdsourcing locations in the network, some crowdsourcing locations may end up with a larger cost than the expectation. We would like to show that there is some number λ such that for every crowdsourcing location, $Pr[c_{ij}x_i \geq \lambda\theta^\star] \leq \frac{1}{nm}$. Then, by the union bound, $Pr[\exists l_i, \sum c_{ij}x_i \geq \lambda\theta^\star] \leq \frac{|L|}{mn} \leq \frac{1}{n}$. That is, we would get a λ approximation with a $\frac{1}{n}$ probability to exceed the $\lambda\theta^\star$. In the following, we will prove that $\lambda = O(\frac{\log n}{\log\log n})$.

Theorem 6. *The proposed randomized rounding algorithm has an $O(\frac{\log n}{\log\log n})$ approximation ratio.*

Proof. Without loss of generality, let us assume that all rounds of the randomized rounding algorithm are all disjoint events. Workers are selected with probabilities $\{x_1^\star, x_2^\star, ..., x_n^\star\}$ for any crowdsourcing location l_i, where $x_i^\star$ follows the independent and identically distributed random distribution, the probability that $\sum x_i^\star$ is normalized to 1. Let random variables $x_i \in [0,1]$. Therefore, $E[\sum c_{ij}x_i] = \sum_1^n c_{ij}E[x_i] =: \theta^\star$. Then, for any $\lambda > 0$ by using the Chernoff bounds, we get the following result due to the random selection:

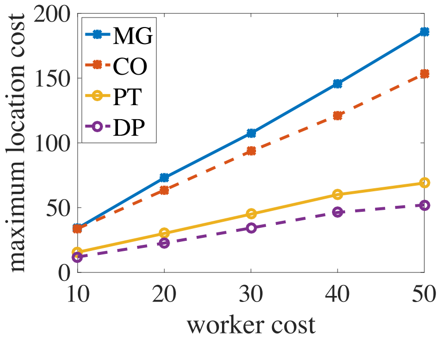
$$Pr[\sum c_{ij}x_i \geq (1+\lambda)\theta^\star] \leq \left(\frac{e^\lambda}{(1+\lambda)^{1+\lambda}}\right)^{\theta^\star} \tag{10}$$

It is equivalent to

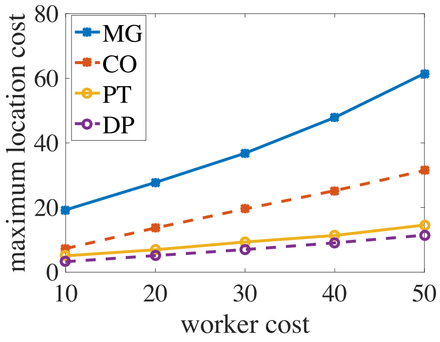
$$\begin{aligned} &Pr[\sum c_{ij}x_i > \lambda\theta^\star] \\ &\leq (\frac{e^{\lambda-1}}{\lambda^\lambda})^{-\lambda\theta^\star} \leq (\lambda/e)^{-\lambda} \end{aligned} \tag{11}$$

If we set $\lambda^\lambda \approx n$, we have $\lambda = O(\frac{\log n}{\log\log n})$ and Eq. 11 $\leq \frac{1}{n}$. □

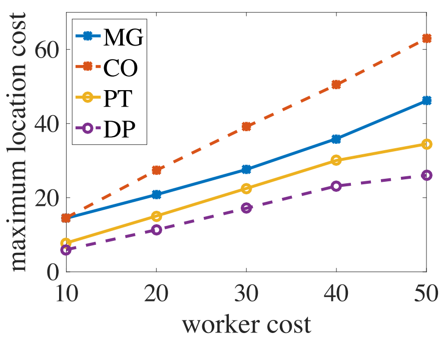
The insight of the randomized rounding algorithm is that the lower bound of the optimal recruitment assignment can be calculated and the random walk can reach the optimal assignment with a designed probability.



(a) San Franciso

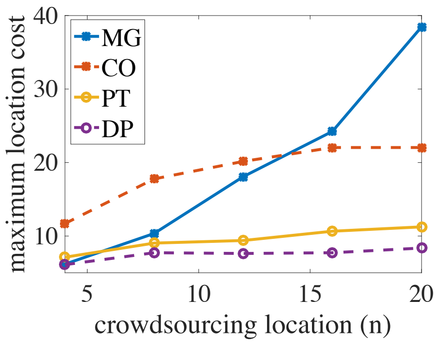


(b) Seattle

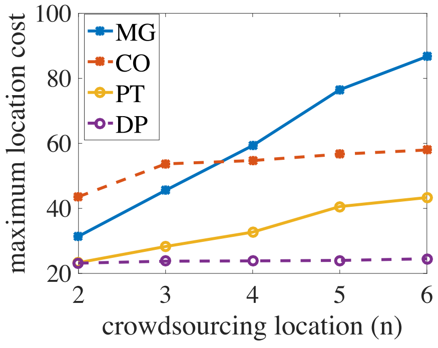


(c) Rome

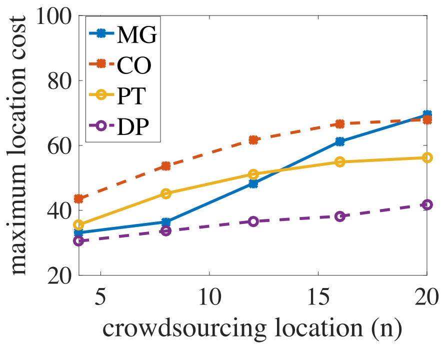
Fig. 8. Performance evaluation of different cost ranges in the 1-D scenario.



(a) San Francisco



(b) Seattle



(c) Rome

Fig. 9. Performance evaluation of different amounts of crowdsourcing locations in the 1-D scenario.

6 Performance Evaluation

In this section, we first discuss the experimental setting, followed by the algorithm introduction. The last part of this section is the experimental results under different settings.

6.1 Traces

The EPFL [39] trace is the taxi trace collected from San Francisco, USA. It contains GPS coordinates of 537 taxies over 30 days. Another trace that we use is the Seattle bus [40] trace. The traces were collected from 750 buses while on different routes in Seattle, USA for several weeks. The Rome taxi trace [41] contains GPS coordinates of 320 taxies in Rome, Italy. The experimental areas are shown in Figs. 7, where the trajectories of taxies in the EPFL trace and buses in the Seattle trace can be fitted into grids well. However, the trajectories of taxies in the Rome trace represent a more general case. Also, buses and taxies have different movement patterns, i.e., where a taxi visits most area of the city, however each bus has a limited coverage area. Therefore, these there datasets represent three different scenarios.

Some detailed experiment parameters are as follows: we choose the centers of these three cities, 10,000 (ft) $\times$ 10,000 (ft), as the experiment area. Then, we divide the experiment area into grids. The grid size is 200 (ft) $\times$ 200 (ft), which is the typical WiFi range under 2.4 GHz in 802.11 protocol for outdoor environment [42]. The experiment area and the vehicles' movement history are shown in Fig. 6 and 7. We consider that once a vehicle reaches a grid, it can successfully finish the crowdsourcing task. Since the data record amount is huge in these three traces, in the experiment, we choose the first 40 taxis in the EPFL trace, $2,183,479$ records, and we choose 236 buses in the Seattle bus trace, $401,577$ records. In the Rome trace, we choose the first 109 taxies, $1,000,000$ records. Since we do not have the cost information in these three traces, we generate five different costs, which refers to 5 different costs in Uber cars [5], i.e., UberPool, UberX, UberXL, UberSelect, and UberBlack in reality.

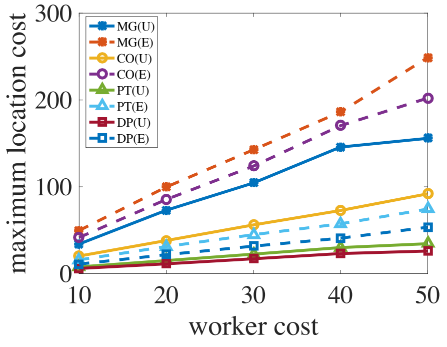
6.2 Experiment Setting

The trajectory of a taxi/bus is all grids that it visited in the experimental area. In the 1-D scenario, we randomly select a row in the area to conduct the experiments. The crowdsourcing locations are randomly selected among the grids [4, 20] and [2, 6], respectively in three datasets. For each vehicle, we use the uniform distribution and the exponential distribution with parameter 1 to assign a crowdsourcing operation cost. The enter position and the exit position are considered as the left-most grid and the right-most grid, respectively. In the 2-D scenario, we randomly select [5, 25] and [4, 20] crowdsourcing locations in the 2-D scenario. In addition, we use the uniform distribution to assign the vehicle cost. All the experiments are repeated 500 to 2000 times to ensure convergence.

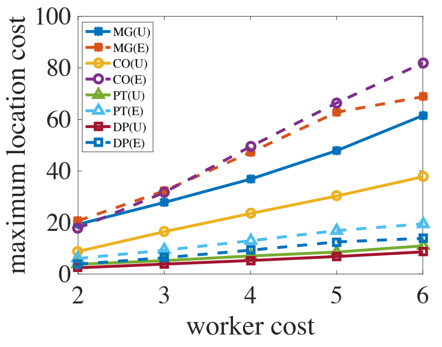
6.3 Algorithm Comparison

We propose four algorithms in the 1-D scenario.

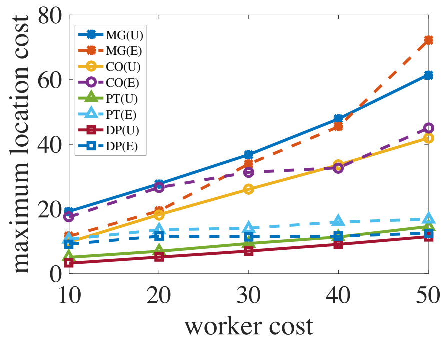
- *Min-max greedy* (MG) algorithm selects the worker candidate sets that increase the max-cost of any crowdsourcing location at least, then among them, the worker who can increase the coverage most is selected until the network is covered, which is widely used in combination optimization [37].



(a) San Franciso

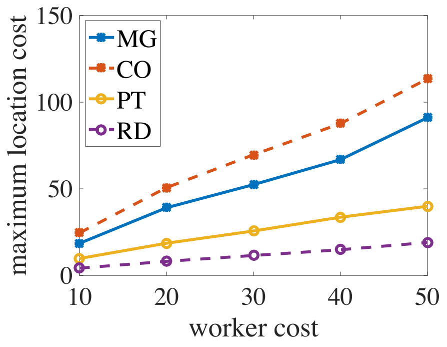


(b) Seattle

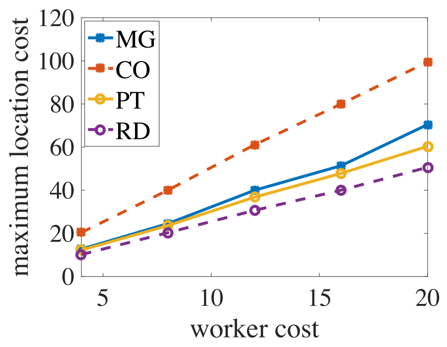


(c) Rome

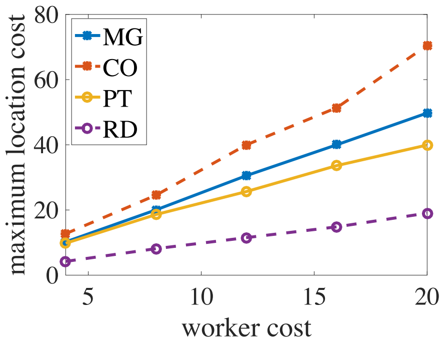
Fig. 10. Performance evaluation of different cost distributions in the 1-D scenario.



(a) San Franciso



(b) Seattle



(c) Rome

Fig. 11. Performance comparison of different cost ranges in the 2-D scenario.

- *Coverage-only* (CO) algorithm selects a set of workers from one side to another side without considering their cost.
- *PTAS* (PT) algorithm divides the workers into to two sets according to their corresponding costs, then, uses the worker with the lowest cost to cover the network. Then, we try to find the optimal set partition. The ε is 0.1 in the experiments.
- *Dynamic programming* (DP) algorithm uses the proposed dynamic programming technique to find the optimal solution.

In the generate 2-D scenario, the MG algorithm can be applied directly. For the CO and PT algorithms, they can be extended into the 2-D scenario by covering the area row-by-row/column-by-column. We compare their performance with the *randomized rounding* (RD) algorithm, which uses the randomized rounding technique to select the workers.

6.4 Experimental results

In this subsection, we will discuss the average maximum crowdsourcing location cost of the proposed algorithms in the 1-D scenario and the 2-D scenario in terms of different average cost values, different crowdsourcing locations, and different cost distributions.

6.4.1 Different cost ranges in the 1-D scenario

Fig. 8 shows the performance results of the proposed four algorithms in the 1-D scenario. The results show that along with the worker cost increase, the performance difference between the proposed four algorithms increases. The DP algorithm always achieves the best performance, i.e., the lowest maximum workload cost, followed by the PT algorithm in all three datasets. As for the MG and CO algorithms, CO algorithm has a better performance in the EPFL and Seattle traces but a worse performance in the Rome trace, which is related to the experimental setting. The reason is that the roads in the first two traces can be approximated as grids well. Therefore, it is easy to find a worker who can cover multiple crowdsourcing locations. However, for the Rome trace, the majority of workers' trajectories can only cover a few crowdsourcing locations in 1-D scenario. Therefore, in first two traces, the CO algorithm can significantly reduce the total amount of selected workers, and thus leads to a good performance. For the Rome trace, the CO algorithm cannot reduce the total amount of selected worker well and its drawback appears, i.e., worker with high cost might be selected. On average, the PT algorithm's performance is very close to the optimal algorithm, 10% of the performance loss at the benefit of a lower computation complexity. The CO and MG algorithms only achieve 25% to 30% performance compared with the optimal solution. However, their computation complexity is much lower.

6.4.2 Different crowdsourcing location amounts in the 1-D scenario

The performance results of different crowdsourcing location amounts in three datasets are shown in Fig. 9. It is clear that along with the increase of crowdsourcing locations, the performance of the MG algorithm decreases sharply. When there are only a few crowdsourcing locations, the MG algorithm achieves the second-best algorithm. However, it achieves the worst performance in the case where there are many crowdsourcing locations. For the remaining three algorithms, along with the increase of crowdsourcing locations, the cost does not increase much, which shows the effectiveness of directional coverage, i.e., the error cannot accumulate. The PT algorithm reduces the cost about 35% more on average in these three datasets than CO algorithm at the cost of extra computation complexity. Therefore, if the crowdsourcing locations are few, the MG algorithm achieves a good performance-cost trade-off. If there are many crowdsourcing locations, the PT algorithm achieves a good performance-cost trade-off. The insight behind it is that when the crowdsourcing location is sparse, the improper selection of the MG algorithm increases. However, then the crowdsourcing location becomes dense, and the improper of the MG algorithm decreases, since it becomes easy to cover some uncovered areas.

6.4.3 Different cost distributions in the 1-D scenario

In the experiments, we discuss the influence of different cost distributions, i.e., uniform distribution and exponential cost distribution, in three datasets. Let us use F(U) to denote the uniform cost distribution and F(E) to denote the exponential distribution for each algorithm, respectively. For example, MG(U) is the performance result of the MG algorithm under uniform distribution. The results are shown in Fig. 10. From Figs. 10(a), 10(b) and 10(c), we can conclude that the exponential crowdsourcing cost distribution leads to a worse performance than the uniform cost distribution. The reason is that we focus on the worst situation rather than the average situation in the CBCR problem. There is a higher probability in the exponential distribution that all the available workers for a crowdsourcing location has a high crowdsourcing operation cost than the uniform distribution and there might be high error accumulation in some crowdsourcing locations. In detail, Figs. 10(a) and 10(b) show similar results where MG(E) and CO(E) achieve the worst and the second-worst performances, followed by MG(U), CO(U), PT(E), DP(E), PT(U), and DP(U) algorithms. Though the average crowdsourcing costs are the same in two distributions, the maximum crowdsourcing location is doubled. The result in the Rome trace is a little different as shown in Fig. 10(c). The MG(E) algorithm is better than the MG(U) algorithm when the worker cost is small.

6.4.4 Different cost ranges in the 2-D scenario

Fig. 11 shows the performance results of the proposed four algorithms in the 2-D scenario. Similarly, along with the worker cost increase, the performance difference between the proposed four algorithms increases. The RD algorithm always achieves the best performance, i.e., the lowest maximum workload cost, followed by the PT algorithm in all three datasets. As for the MG and CO algorithms, the MG algorithm always has better performance than the CO algorithm, which is different from the 1-D scenario. The reason is that in the 2-D scenario, the CO algorithm tries to avoid redundant coverage, and thus covers row-by-row/column-by-column. However, the performance is not that good considering the fact that there are many redundancies from previously covered rows/columns. Therefore, the CO algorithm does not have a good performance in the 2-D scenario. In addition, crowdsourcing cost is not considered in the CO algorithm. The PT algorithm achieves a similar performance to the MG algorithm at the cost of an extra computation complexity as shown in Figs. 11(b) and 11(c). The RD algorithm achieves a 15% performance gain compared with the optimal solution. However, their computation complexity is the highest.

6.4.5 Different crowdsourcing location amounts in the 2-D scenario

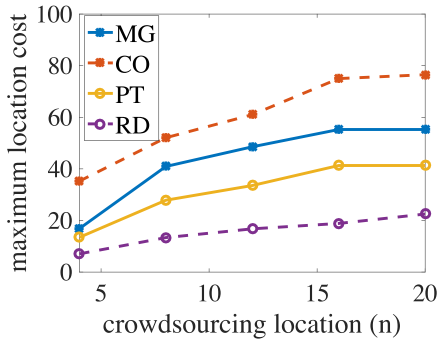
The performance results of different crowdsourcing location amounts in the 2-D scenario are shown in Fig. 12. It is clear that along with the increase of the amount of crowdsourcing locations, the performance of the proposed four algorithm becomes worse. Among them, the performance difference between the RD algorithm and the other three algorithm becomes larger. For the remaining three algorithms, the MG algorithm's performance is better than the PT algorithm in the Seattle and Rome traces, which shows that directional coverage will still lead to huge error accumulation. The worst algorithm is the CO algorithm in all three datasets. Note that the MG algorithm achieves a 40% performance gain over the CO algorithm with a similar time complexity. The RD algorithm has a 30% performance gain at the cost of extra time complexity. In Figs. 12(a) and 12(c), the CO algorithm achieves a larger maximum workload cost than the MG algorithm. However, in the Seattle trace, the CO algorithm achieves 30% of the maximum workload cost of that of the MG algorithm. The reason might be that the taxies visit the whole grid area more uniformly than the buses. Therefore, we can more easily find a taxi with a smaller cost to cover the same area. However, different buses' routes are more different.

6.4.6 Different cost distributions in the 2-D scenario

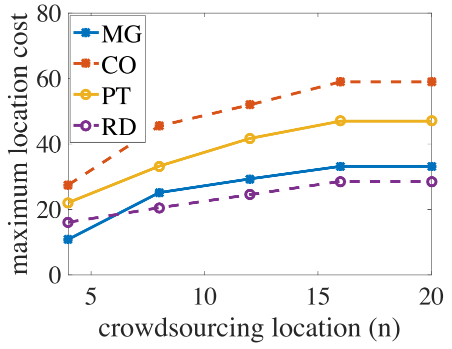
Fig. 13 shows the result of different cost distributions in the 2-D scenario. Figs. 13(a), 13(b), and 13(c) show a similar performance order of the four algorithms in terms of different cost distribution. That is, the maximum workload cost decreases following the order of the MG, CO, PT, and RD algorithms. The difference from the 1-D scenario is that the performance difference in two cost distributions becomes even larger. The insight behind it is that when the crowdsourcing cost distribution is exponential, the improper selection of the MG algorithm leads to a worse result. From Figs. 13(a), 13(b) and 13(c), we notice that the experimental distribution further increases the cost of the CO algorithm by $150\%, 200\%$, and 250%, respectively. However, for the MG and RD algorithms, the cost increase is less than 50%, which demonstrates the effectiveness of these two algorithms in the general case.

7 Conclusion and Future Work

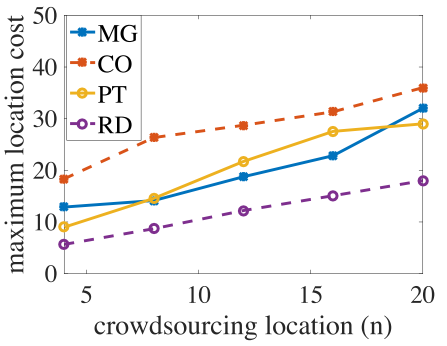
In this paper, we address coverage and workload-balancing in spatial crowdsourcing, which are ignored in existing approaches. The coverage constraint means that any crowdsourcing location should be visited by at least one of the recruited workers to satisfy the Quality-of-Service requirement for some applications. In addition, there is a heterogeneous cost for each crowdsourcing operation in reality, e.g., data traffic or energy/recourse consumption and the resource/energy may be limited at each crowdsourcing location. The objective of this paper is to solve the Coverage



(a) San Francisco

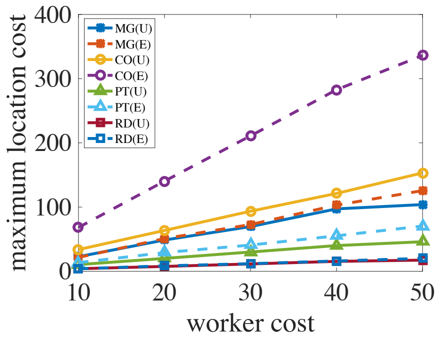


(b) Seattle

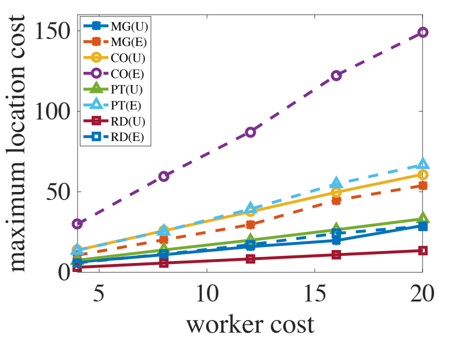


(c) Rome

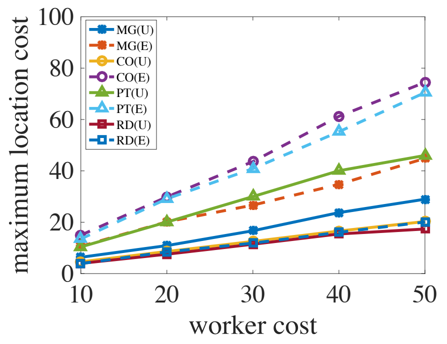
Fig. 12. Performance comparison of different amounts of crowdsourcing locations in the 2-D scenario.



(a) San Francisco



(b) Seattle



(c) Rome

Fig. 13. Performance comparison of different cost distributions in the 2-D scenario.

and Balanced Crowdsourcing Recruiting (CBCR) problem, which ensures the coverage requirement and minimizes the maximum crowdsourcing cost for any crowdsourcing location at the same time.

We prove that the CBCR problem is NP-hard in the general case. Then, we discuss the solution of CBCR problem in 1-D scenario. In the 1-D scenario, we first propose a naive greedy algorithm and a PTAS scheme to trade-off the computation complexity and the performance. The performance can be bounded to $2 + \varepsilon$, where ε can be an arbitrary small value. Then, we found that a sub-optimal structure exists, and thus the optimal dynamic programming approach is proposed to find the optimal solution in the 1-D scenario. In the general 2-D scenario, we first prove that the CBCR problem has a sub-modular property and thus the naive greedy algorithm has an approximation ratio of $\ln n + 1$. In addition, we propose a randomized rounding algorithm with an expectation bound of $O(\frac{\log n}{\log \log n})$. Extensive experiments from three realistic vehicle traces demonstrate the effectiveness of the proposed algorithms.

In this paper, we assume that each worker's trajectory is totally deterministic and this is no detour flexibility at all. One of the future directions is to conduct research on a limited flexibility model, where each worker has their original trajectories and a detour distance threshold. That is, it is acceptable to detour for a short distance to earn larger benefit. The real application is like UberPool. In this case, how to optimally recruit workers in spatial crowdsourcing under budget constraint is our interest. In addition, we are seeking opportunity to implement the proposed spatial crowdsourcing approaches into real system, therefore, we can adjust the proposed algorithm into the real setting.

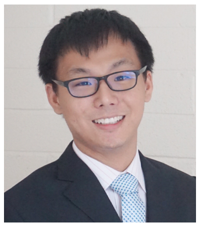
Acknowledgments

This research was supported in part by NSF grants CNS 1757533, CNS1629746, CNS 1564128, CNS 1449860, CNS 1461932, CNS 1460971, and IIP 1439672.

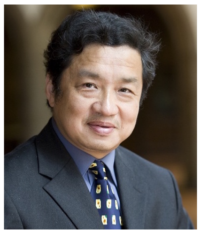
References

[1] N. Wang and J. Wu, "Opportunistic wifi offloading in a vehicular environment: Waiting or downloading now?" in *Proceedings of the IEEE INFOCOM*, 2016.
[2] J. Yick, B. Mukherjee, and D. Ghosal, "Wireless sensor network survey," *Computer networks*, vol. 52, no. 12, pp. 2292–2330, 2008.
[3] Y. Zhao and Q. Han, "Spatial crowdsourcing: current state and future directions," *IEEE Communications Magazine*, vol. 54, no. 7, pp. 102–107, 2016.
[4] D. C. Brabham, "Crowdsourcing as a model for problem solving: An introduction and cases," *Convergence*, vol. 14, no. 1, pp. 75–90, 2008.
[5] https://en.wikipedia.org/wiki/Uber_(company).
[6] https://www.waze.com.
[7] https://www.openstreetmap.org.
[8] J. She, Y. Tong, and L. Chen, "Utility-aware social event-participant planning," in *Proceedings of the ACM SIGMOD ICMD*, 2015.
[9] Z. He and D. Zhang, "Cost-efficient traffic-aware data collection protocol in vanet," *Ad Hoc Networks*, 2016.
[10] D. Deng, C. Shahabi, and U. Demiryurek, "Maximizing the number of worker's self-selected tasks in spatial crowdsourcing," in *Proceedings of the ACM SIGSPATIAL*, 2013.
[11] J. She, Y. Tong, L. Chen, and C. C. Cao, "Conflict-aware event-participant arrangement," in *Proceedings of the IEEE ICDE*, 2015.
[12] N. Wang and J. Wu, "Optimal cellular traffic offloading through opportunistic mobile networks by data partitioning," in *Proceedings of the IEEE ICC*, 2018.

[13] W. Sun, Q. Li, and C.-K. Tham, “Wireless deployed and participatory sensing system for environmental monitoring,” in *Proceedings of the IEEE SECON*, 2014.
[14] B. Hull, V. Bychkovsky, Y. Zhang, K. Chen, M. Goraczko, A. Miu, E. Shih, H. Balakrishnan, and S. Madden, “Cartel: a distributed mobile sensor computing system,” in *Proceedings of the ACM SenSys*, 2006.
[15] P. Mohan, V. N. Padmanabhan, and R. Ramjee, “Nericell: rich monitoring of road and traffic conditions using mobile smartphones,” in *Proceedings of the ACM SenSys*, 2008.
[16] F. Saremi, O. Fatemieh, H. Ahmadi, H. Wang, T. Abdelzaher, R. Ganti, H. Liu, S. Hu, S. Li, and L. Su, “Experiences with greengpsfuel-efficient navigation using participatory sensing,” *IEEE Transactions on Mobile Computing*, vol. 15, no. 3, pp. 672–689, 2016.
[17] M. Liaqat, A. Gani, M. H. Anisi, S. H. Ab Hamid, A. Akhunzada, M. K. Khan, and R. L. Ali, “Distance-based and low energy adaptive clustering protocol for wireless sensor networks,” *PloS one*, vol. 11, no. 9, p. e0161340, 2016.
[18] Y. Chon, N. D. Lane, Y. Kim, F. Zhao, and H. Cha, “Understanding the coverage and scalability of place-centric crowdsensing,” in *Proceedings of the ACM Ubicomp*, 2013.
[19] X. Zhang, Z. Yang, Y. Liu, and S. Tang, “On reliable task assignment for spatial crowdsourcing,” *IEEE Transactions on Emerging Topics in Computing*.
[20] B. Guo, Y. Liu, W. Wu, Z. Yu, and Q. Han, “Activecrowd: A framework for optimized multitask allocation in mobile crowdsensing systems,” *IEEE Transactions on Human-Machine Systems*, 2016.
[21] H. Zheng, N. Wang, and J. Wu, “Minimizing deep sea data collection delay with autonomous underwater vehicles,” *Journal of Parallel and Distributed Computing*, vol. 104, pp. 99–113, 2017.
[22] J. Prassl and M. Risak, “Uber, taskrabbit, and co.: Platforms as employers-rethinking the legal analysis of crowdwork,” *Comp. Lab. L. & Pol'y J.*, vol. 37, p. 619, 2015.
[23] N. Wang and J. Wu, “Trajectory scheduling for timely data report in underwater wireless sensor networks,” in *Proceedings of the IEEE GLOBECOM*, 2015.
[24] Y. Tong, J. She, B. Ding, L. Wang, and L. Chen, “Online mobile micro-task allocation in spatial crowdsourcing,” in *Proceedings of the IEEE ICDE*, 2016.
[25] 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.
[26] D. Zhao, H. Ma, and L. Liu, “Energy-efficient opportunistic coverage for people-centric urban sensing,” *Wireless networks*, vol. 20, no. 6, pp. 1461–1476, 2014.
[27] H. Xiong, D. Zhang, G. Chen, L. Wang, and V. Gauthier, “Crowdtasker: Maximizing coverage quality in piggyback crowdsensing under budget constraint,” in *Proceedings of the IEEE PerCom*, 2015.
[28] D.-H. Shin, S. He, and J. Zhang, “Joint sensing task and subband allocation for large-scale spectrum profiling,” in *Proceedings of the IEEE INFOCOM*, 2015.
[29] Z. He, J. Cao, and X. Liu, “High quality participant recruitment in vehicle-based crowdsourcing using predictable mobility,” in *Proceedings of the IEEE INFOCOM*, 2015.
[30] P. Cheng, X. Lian, Z. Chen, R. Fu, L. Chen, J. Han, and J. Zhao, “Reliable diversity-based spatial crowdsourcing by moving workers,” *Proceedings of the VLDB Endowment*, vol. 8, no. 10, pp. 1022–1033, 2015.
[31] X. Zhang, Z. Yang, Y.-J. Gong, Y. Liu, and S. Tang, “Spatialrecruiter: maximizing sensing coverage in selecting workers for spatial crowdsourcing,” *IEEE Transactions on Vehicular Technology*, vol. 66, no. 6, pp. 5229–5240, 2017.
[32] Y. Tong, L. Wang, Z. Zhou, B. Ding, L. Chen, J. Ye, and K. Xu, “Flexible online task assignment in real-time spatial data,” *Proceedings of the VLDB Endowment*, 2017.
[33] T. Song, Y. Tong, L. Wang, J. She, B. Yao, L. Chen, and K. Xu, “Trichromatic online matching in real-time spatial crowdsourcing,” in *Proceedings of the IEEE ICDE*, 2017.
[34] H. To, L. Fan, L. Tran, and C. Shahabi, “Real-time task assignment in hyperlocal spatial crowdsourcing under budget constraints,” in *Proceedings of the IEEE PerCom*, 2016.
[35] N. Wang, J. Wu, and P. Ostovari, “Coverage and min-max workload cost in spatial crowdsourcing,” in *Proceedings of the IEEE UIC*, 2017.
[36] Y. Niu, L. Pan, M. J. Pérez-Jiménez, and M. R. Font, “A tissue p systems based uniform solution to tripartite matching problem,” *Fundamenta Informaticae*, vol. 109, no. 2, pp. 179–188, 2011.
[37] A. Krause and D. Golovin, “Submodular function maximization.” 2014.
[38] P. Raghavan and C. D. Tompson, “Randomized rounding: a technique for provably good algorithms and algorithmic proofs,” *Combinatorica*, vol. 7, no. 4, pp. 365–374, 1987.
[39] M. Piorkowski, N. Sarafijanovic-Djukic, and M. Grossglauser, “A parsimonious model of mobile partitioned networks with clustering,” in *Proceedings of the IEEE COMSNETS*, 2009.
[40] J. G. Jetcheva, Y.-C. Hu, S. PalChaudhuri, A. K. Saha, and D. B. Johnson, “Design and evaluation of a metropolitan area multitier wireless ad hoc network architecture,” in *Proceedings of the IEEE HotMobile*, 2003.
[41] L. Bracciale, M. Bonola, P. Loreti, G. Bianchi, R. Amici, and A. Rabuffi, “CRAWDAD dataset roma/taxi (v. 2014-07-17),” Downloaded from https://crawdad.org/roma/taxi/20140717, Jul. 2014.
[42] http://www.wi-fi.org/.html.



Ning Wang received his B.Eng. in Electrical Engineering from the University of Electronic Science and Technology of China, Chengdu, China in 2013. He is currently a 5^{th} year Ph.D. student in the Department of Computer and Information SciencesTemple University, Philadelphia, Pennsylvania, USA. His research focuses on mobile edge networks, data offloading and pub/sub systems.



Jie Wu is the Associate Vice Provost for International Affairs at Temple University. He also serves as the Director of the Center for Networked Computing and a Laura H. Carnell professor. He served as the Chair of Computer and Information Sciences from 2009 to 2016. Prior to joining Temple University, he was a program director at the National Science Foundation and was a distinguished professor at Florida Atlantic University. His current research interests include mobile computing and wireless networks, routing protocols, cloud and green computing, network trust and security, and social network applications. Dr. Wu regularly publishes in scholarly journals, conference proceedings, and books. He serves on several editorial boards, including IEEE Transactions on Service Computing and the Journal of Parallel and Distributed Computing. Dr. Wu was general cochair for IEEE MASS 2006, IEEE IPDPS 2008, IEEE ICDCS 2013, ACM MobiHoc 2014, ICPP 2016, and IEEE CNS 2016, as well as program cochair for IEEE INFOCOM 2011 and CCF CNCC 2013. He was an IEEE Computer Society Distinguished Visitor, ACM Distinguished Speaker, and chair for the IEEE Technical Committee on Distributed Processing (TCDP). Dr. Wu is a CCF Distinguished Speaker and a Fellow of the IEEE. He is the recipient of the 2011 China Computer Federation (CCF) Overseas Outstanding Achievement Award.