Cost-Efficient Worker Trajectory Planning Optimization in Spatial Crowdsourcing Platforms

Ning Wang* and Jie Wu†
*Department of Computer Science, Rowan University, Glassboro, USA
†Center for Networked Computing, Temple University, Philadelphia, USA
Email: wangn@rowan.edu, and jiewu@temple.edu

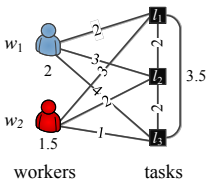
***Abstract*—With the progress of mobile devices and the successful usage of the wisdom of crowds, spatial crowdsourcing has attracted much attention from the research community. This paper addresses the efficient worker recruitment problem under the task coverage constraint. The efficiency of worker recruitment is measured by the total quality collected by a set of workers and the corresponding cost, e.g., proportional to the overall trajectory length of workers. Specifically, we consider two different scenarios, 1-D line topology and general 2-D topology, in which workers may have either homogeneous or heterogeneous crowdsourcing quality (e.g., the quality of videos or photos for an object at a particular location). In the 1-D scenario, we propose two dynamic programming approaches to find the optimal solution in both homogeneous and heterogeneous cases. In the general 2-D scenario, the proposed problem turns out to be NP-hard even in the homogeneous case. We first prove that the simple nearest assignment has an approximation ratio of $1/(2n)$, where n is the number of the workers. Therefore, the nearest assignment cannot be scalable. We further propose a novel assignment approach based on the minimum spanning tree. The proposed approach is proved to be close to the optimal solution in the homogeneous case and $1/\rho$ in the heterogeneous case, where ρ is the maximum quality ratio between two workers. The effectiveness of the proposed algorithm is verified using a real mobility trace: Uber pick-up trace in the New York City.**

***Index Terms*—Spatial crowdsourcing, worker recruitment, and trajectory planning.**

I. 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) over the past few years, all kinds of data have become widely available and large in amount. Traditional infrastructure-based computing approaches or systems have begun to show their limitations, i.e, high system implementation costs, difficult-to-handle dynamic environments, and hard-to-utilize kinds of big data [1]. To address the aforementioned three challenges, *spatial crowdsourcing* has emerged in the past few years [2].

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 task. Therefore, there is no need to build a specific system using the ubiquitous sensors in smart devices/vehicles, and we can enjoy the pay-as-you-go character. The new feature of the spatial crowdsourcing is that spatial crowdsourcing platforms consist of location-specific tasks. Therefore, workers who agree to participate in the spatial crowdsourcing have to physically be at specific locations to complete the tasks. There are many spatial crowdsourcing applications, e.g., pollution level monitoring [3], parking lot monitoring [4], and traffic congestion [5], etc. Spatial crowdsourcing has application in numerous domains such as transportation (e.g., Uber [6]), journalism [7], and food delivery (e.g., Grubhub [8]).



Schedule	w_1	w_2	Quality-Cost Ratio
Nearest	$\{l_1\}$	$\{l_2, l_3\}$	(3+2)/(5+4) = 0.56
Min dist.	{}	$\{l_1, l_2, l_3\}$	4.5/8 = 0.56
Max quality	$\{l_1, l_2, l_3\}$	{}	6/10 = 0.60
Optimal	$\{l_1, l_2\}$	$\{l_3\}$	(1.5 + 4)/(2 +7) =0.61

Fig. 1. An illustration of cost-efficient worker recruitment problem.

In this paper, we would like to address a fundamental problem in spatial crowdsourcing, that is, how to maximize the effectiveness of a crowdsourcing recruitment policy. In a typical crowdsourcing system, different workers might have different qualities for one task due to different sensors (e.g., cameras) and worker behaviors (e.g., following the instruction strictly or careless) [9, 10]. The cost of a recruited worker is usually related to the worker's trajectory length [11, 12]. To maximize the long-term benefit, a spatial crowdsourcing recruitment policy needs to have not only a high quality but also a low cost. That is, there is a quality-cost trade-off in the spatial crowdsourcing. Therefore, we propose using the *overall quality-cost ratio*, (i.e., the total collected qualities of divided by the total crowdsourcing cost), as the criteria.

In this paper, we propose the Cost-efficient Worker Recruitment Problem (CWRP), whose objective is to maximize the overall quality-cost ratio of a crowdsourcing policy under the constraint that all crowdsourcing locations should be covered. An illustration of the CWRP is shown in Fig. 1, where there are two workers, w_1 and w_2, and three crowdsourcing locations, l_1, l_2, and l_3. The links between these five locations are the corresponding crowdsourcing cost, e.g., distance. The crowdsourcing qualities of w_1 and w_2 to conduct tasks are 2 and 1.5, respectively. In Fig. 1, we show four different recruitment schedules and the detailed results are shown in the table in Fig. 1. The idea of the first schedule is to assign crowdsourcing locations to their nearest workers. Therefore, l_1 is assigned to w_1, since the cost between w_1 and l_1 is 2, and the cost between w_2 and l_1 is 3. By the same comparison, l_2 and l_3 are assigned to w_2. The shortest-round trip cost for w_1 to visit l_1, i.e., $w_1 \rightarrow l_1 \rightarrow w_1$, is 4. The shortest-round

trip for w_2 to visit l_2 and l_3, i.e., $w_2 \rightarrow l_2 \rightarrow l_3 \rightarrow w_2$. is 5. Therefore, the overall efficiency ratio is $(3+2)/(5+4) = 0.56$. Another schedule is to find one or multiple shortest round tour(s) so that the overall trajectory length is the minimum. In this example, since w_1's initial location is relatively far from these three crowdsourcing locations, through exhaustive searching, the shortest way to visit l_1, l_2, and l_3 is to recruit w_2 to visit all crowdsourcing locations. The shortest round-path length is 8 and the efficiency ratio is 0.56. However, these two schedules do not consider the worker's quality at all. The third schedule tries to maximize the achievable quality sum. Therefore, all crowdsourcing locations are assigned to the worker who has the largest corresponding quality, i.e., w_1 in this example, and the efficiency ratio is 0.60. Its performance is a little bit better than the first and second schedules. From this toy example, we observe that we need to leverage the crowdsourcing quality and crowdsourcing cost carefully to achieve an efficient schedule. The optimal schedule is to schedule l_1 and l_2 to w_1 and l_3 to w_2. The efficiency ratio turns out to be 0.61.

In this paper, we discuss the solution of the CWRP in homogeneous and heterogeneous worker quality settings under two different topologies, i.e., 1-D topology and general 2-D topology. In the 1-D topology, e.g., highway scenario, we propose two dynamic programming approaches to find the optimal solution in both homogeneous and heterogeneous cases with different complexities. Then, we extend the model into a general 2-D scenario, which turns out to be NP-hard even in the homogeneous case. We first prove that the simple nearest assignment has an approximation ratio of $1/(2n)$, where n is the total number of the workers in the network. It means the nearest assignment can be really bad when there are many workers and thus it cannot be applied to a scalable system. Then, we propose a novel assignment approach, which is proved to be close to the optimal solution in the Euclidean space in the homogeneous case and achieves an approximation ratio of $1/\rho$ in the heterogeneous case, where ρ is the maximum quality ratio between any two workers.

The contributions of this paper are summarized as follows:

- To our best knowledge, we are the first to consider the cost-effectiveness of spatial crowdsourcing under the task coverage constraint during the worker recruitment.
- In the 1-D topology scenario, we propose two dynamic programming approaches to find the optimal solution in the homogeneous and heterogeneous cases.
- In the 2-D topology scenario, we prove that the CWPR is NP-hard. A novel approach is proposed, and its performance is n times better than the nearest assignment approaches in terms of the approximation ratio.
- We verify the effectiveness of proposed approaches using real Uber pick-up trace in the New York City.

II. Related Works

a) Worker Trajectory Planning: In this category, the worker's trajectory is controlled by the server. Some applications are TaskRabbit and WeGoLook [13]. Previous researchers have produced many works with only one worker in their spatial crowdsourcing model. In [14], each task has a deadline and a feasible route of a worker should make sure that all tasks in the route can be finished before their deadlines. Later works considered the existence of multiple workers. In [15], the authors noticed that there are time conflicts for task assignments, which further complicates the planning problem. In addition, workers may have competing relationships. Therefore, the optimal trajectory for a particular worker might not be the optimal trajectory in terms of benefiting the whole network. To address this situation, a simple greedy collaborative trajectory planning scheme is proposed but the method in [15] did not have a performance analysis. In [16], they considered the online trajectory planning. In [11], the authors tried using the semi-bandit learning method to maximize task reliability and minimize travel costs.

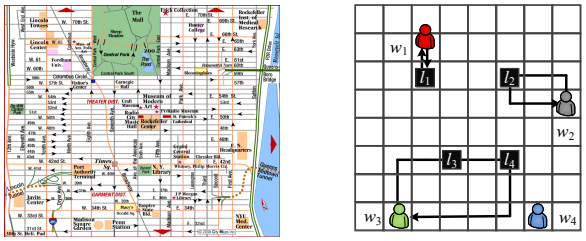
b) Trajectory Coverage: In this category, each worker's trajectory is predetermined. Therefore, the major problem is recruiting the most cost-efficient worker to maximize the crowdsourcing task. An application example is Waze Carpool [17]. There were many theoretical studies on task assignments and participant selection problems, playing trade-offs among the crowdsourcing budgets and the coverage range [18–20]. The difference between their models and proposed model in this paper is that they considered maximizing the coverage range; however, in many scenarios, ensuring the coverage in a certain area, such as traffic congestion monitoring or surveillance, with the minimum cost is very important. In [21], the authors considered the coverage requirement in crowdsourcing and minimized the overall recruiting cost; therefore, a greedy algorithm with a performance analysis was proposed. In [22], the authors considered the trade-off between the load balance of each worker and the utility maximization by modeling the recruitment as a Nash bargaining game. In [23], the authors considered both the number of assigned tasks and coverage area of workers.

III. Problem Statement

A. Network Model

In this paper, we address a centralized spatial crowdsourcing system in an offline manner. In a particular time slot, we assume there are n available workers who agree to participate in the crowdsourcing system, $\{w_1, w_2, \cdots, w_n\}$, and their initial positions are $W = \{p_1, p_2, \cdots, p_n\}$. Suppose there are m crowdsourcing locations, $\{l_1, l_2, \cdots, l_m\}$, whose locations are $C = \{p_{n+1}, p_{n+2}, \cdots, p_{n+m}\}$, and each location has one and only one crowdsourcing task that needs to be done. Without loss of generality, we assume that we have an $(n+m)\times(n+m)$ matrix, D, where d_{ij} is the shortest distance from p_i to p_j. The shortest distance between a pair of locations is symmetric, i.e., $d_{ij} = d_{ji}$. In addition, the 2-D crowdsourcing area is a metric space, namely, $d_{ij} + d_{jk} \geq d_{ik}$, for any i, j, k.

A unique feature of the crowdsourcing platform is that workers may have heterogeneous crowdsourcing qualities for a crowdsourcing task. For example, different smartphones have different camera resolutions or behaviors [9, 10], and



(a) Manhattan map (b) Crowdsourcing model

Fig. 2. Network model in the city environment.

different Uber drivers have different rating [24]. We use q_k to denote the earned quality for a crowdsourcing location by recruiting the worker w_k. In terms of the crowdsourcing system cost, each recruited worker has to physically move from its initial location to the crowdsourcing locations and finally goes back to the initial location. Given the assigned crowdsourcing locations for a worker, a tour with the shortest-distance can be calculated through shortest-path algorithms. Let us use a 2-D movement decision matrix X^k to denote the actual tour of the worker w_k, where x_{ij}^k is a decision variable. $x_{ij}^k = 1$ represents that worker w_k moves from p_i to p_j through the corresponding shortest path, otherwise $x_{ij}^k = 0$. Given the movement decision matrix X^k, the overall quality earned by recruiting w_k is $\sum_{i=1,j=1}^{n+m} x_{ij}^k q_k$, and the total cost of the w_k is $\sum_{i=1,j=1}^{n+m} c(d_{ij}) x_{ij}^k$, where $c(\cdot)$ is a cost function regarding the tour length [11, 12]. Therefore, in the remaining part of this paper, we use the tour length to represent the corresponding crowdsourcing cost without further explanation. In addition, we use u_i^k and n_k to denote the ith location and the total number of crowdsourcing locations visited by w_k, respectively. An illustration of the network model is shown in Fig. 2, where 3 workers are recruited to cover 4 crowdsourcing locations. Fig. 2(a) is a common setting in an urban area and Fig. 2(b) show a possible recruitment result, where w_1 covers l_1, w_2 covers l_2, and w_3 covers l_3 and l_4.

B. Problem Formulation

In this paper, we propose the Cost-efficient Worker Recruitment Problem (CWRP) in spatial crowdsourcing platforms. Given the workers' initial locations and the crowdsourcing locations at that time, we would like to generate a recruitment policy, $\mathbb{X}$, to recruit a set of workers to finish crowdsourcing tasks and determine the corresponding tour for each worker. The overall objective of CWRP is to maximize the ratio between the overall earned quality of recruited workers and the overall movement cost of them. The CWRP is mathematically formulated as follows:

$$\max \quad \frac{\sum_{k=1}^{n} \sum_{i=1,j=1}^{n+m} x_{ij}^k q_k}{\sum_{k=1}^{n} \sum_{i=1,j=1}^{n+m} c(d_{ij}) x_{ij}^k} \tag{1}$$

$$\text{s.t.} \quad \sum_{k=1}^{n} \sum_{i=1}^{m+n} x_{ij}^k = 1, \quad \forall i, j \quad x_{ij}^k \in \{0,1\}, \tag{2}$$

$$\sum_{i=1,i\neq j}^{m+n} x_{ij}^k = 1, \forall j, \quad \sum_{j=1,j\neq i}^{m+n} x_{ij}^k = 1, \forall i \tag{3}$$

$$u_i^k - u_j^k + n_k x_{ij}^k < n_k - 1 \quad \forall 2 \leq i \neq j \leq n_k \tag{4}$$

where the objective is to maximize the overall ratio between the crowdsourcing quality and the worker movement cost. We call it the recruitment effectiveness in the following of this paper. The first constraint is the coverage constraint, i.e., every crowdsourcing location should be assigned to a worker. The second constraint is that for each worker w_k, its trajectory is a tour, which covers a subset of crowdsourcing locations and it arrives and departs each assigned crowdsourcing location only once. The third constraint is the subtour elimination constraint, which ensures that each worker has only one tour rather than multiple isolated tours [25].

C. Hardness Analysis

Theorem 1. *The decision version of CWRP is NP-complete.*

Proof. To show the decision version of the CWRP $\in$ NP, we show that for a given recruiting policy, X, we can verify the correctness of $\mathbb{X}$ in polynomial time (i.e., constraints 2-4 are satisfied). Specifically, we check each crowdsourcing location l_i and check the number of worker assigned to l_i and the number of arrival and departure at this location. The complexity of the verification can be done in $O(nm^2)$ where m is the total number of crowdsourcing locations and n is the total number of workers.

To show that CWRP $\in$ NP-hard, we argue that the Travel Salesman Problem (TSP), which is NP-complete [26] can be reduced to the CWRP problem in polynomial time. The TSP is to find the shortest tour in the network to pass through each node exactly once. Mathematically, Consider there are n nodes (e.g., cities) belonging to the set $N = \{1, 2, ..., n\}$, with arc set $E = N^2$, and travel costs $t_{ij} \in E$, then the TSP is that what is if there is a 2D decision vector X, where x_{ij} means that the salesman travel from the city i to city j, so that the generated route is the shortest possible route and the that visit each city and return to the original cost.

We present a polynomial time reduction and construct an special instance of the CWRP, where there is only one worker, w_i in the network. In this special example, there are $m + 1$ location in the CWRP, i.e., the initial location p_1 and all crowd sourcing locations $\{p_2, p_3, \cdots, p_m + 1\}$. In this case, we can build $m + 1$ cities using the corresponding locations in the TSP problems. We claim that if there is a polynomial solution which can maximize the effectiveness, i.e., Eq.1, then it is an optimal route for the TSP problem. In this special case, we have to recruit that worker w_i and the overall quality by using that worker is a constant value, i.e., mq_i. To maximize the objective in Eq. 1, the CWRP has to find the shortest path for that w_i to all crowdsourcing locations, which is the same as the objective of the TSP.

We can observe that if the trajectory planning in the TSP exists, then we can use the same schedule in the CWRP. The solution is feasible since all crowdsourcing locations are visited. In addition, the overall length (ratio) is the minimum (maximum). This completes the proof. □

IV. 1-D Line Topology

In this section, we first discuss the CWRP in a 1-D line topology. An application scenario is the traffic monitoring in the main street and highway. An example is shown in Fig. 3.

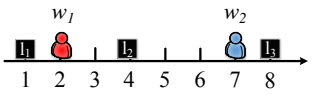


Fig. 3. An illustration of 1-D model.

A. Homogeneous Crowdsourcing quality

If all crowdsourcing workers have identical crowdsourcing abilities for all crowdsourcing locations, the CWRP becomes simple since the objective of the CWRP is equivalent to recruiting a set of workers which can minimize the overall movement distance. In such a case, we can solve the CWRP by using the dynamic programming technique. The dynamic programming works as follows: all the workers' and tasks' locations are sorted directionally, e.g., from the left to right. In Fig. 3, the sorted result is l_1, w_1, l_2, w_2, and l_3. Without loss of generality, we use p_1, p_2, p_3, p_4, and p_5 to represent them, respectively. Then, we use a vector $opt[\cdot]$ to store the optimal sub-solution, where the $opt[i]$ is the minimum total tour(s) length for the first i locations. Initially, all $opt[i]$ is ∞, except $opt[0]$, $opt[0] = 0$. Then, we can get the following optimal substructure as shown in Eq. 5.

$$opt[i] = \begin{cases} \min\{opt[j-1] + 2(p_i - p_j)\} & \forall p_j < p_i \\ \quad \text{if} \quad p_i \in C, p_j \in W & \\ \min\{opt[i], opt[j] + 2(p_i - p_j)\} & \forall p_j < p_i \\ \quad \text{if} \quad p_i \in W, p_j \in C & \end{cases} \tag{5}$$

There are two cases. (1) If the current location p_i is a crowdsourcing location, i.e., $p_i \in C$, we have to select a worker w_j which is before the crowdsourcing i to visit it and we would like to select the best worker w_j in the left of p_i to cover the crowdsourcing locations from p_j to p_i and the overall cost is the minimum. (2) If the current location p_i has a worker, i.e., $p_i \in W$, we would like to check if we can use this newly available worker to cover crowdsourcing locations from p_j to p_i and further reduce the overall cost. The optimal solution to the crowdsourcing location p_i, i.e., $opt[i]$, is calculated in the following method: we use the optimal solution to p_j plus the cost of recruiting a new worker to cover the crowdsourcing locations between p_j and p_i. The optimal value of $opt[i]$ is found after we try all p_j which is smaller than p_i.

Let us use Fig. 3 to illustrate the dynamic programming procedure. Note that we assume that each worker has the identical crowdsourcing ability. The first location is l_1, since there are no workers in l_1's left. Therefore, $opt[1]$ is ∞. The second location is w_1, and we would like to check if we can use w_1 to minimize the total tour length. Clearly, if we use the w_1 to cover l_1, the total tour length will reduce from ∞ to 2 for $opt[2]$. The third location is l_2, then we would like to find the best worker in its left to cover l_2. Since there is only one worker w_1 available, we update $opt[3]$ to $2 + 2 \times 2$. The fourth location is w_2 in this example, and we check if we need to change the coverage of l_1, l_2 or $\{l_1, l_2\}$ from worker w_1 to w_2. Through calculation, we found that it is better to keep the original assignment and thus w_2 is not recruited. The last position in this example is l_3, and it can be covered by w_1 or w_2, who are in its left. Through calculation, we found that w_2 is the best option to cover l_3. In the end, the minimum tour length is 8, where w_1 covers l_1 and l_2, and w_2 covers l_3.

The time complexity of the dynamic programming is as follows. We have $O(m + n)$ rounds at most. For each state, there is at most $O(m+n)$ calculation to get the optimal value so far. The update operation is $O(1)$. Therefore, the overall time complexity is $O((m + n)^2)$.

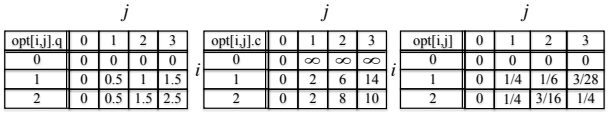


Fig. 4. An illustration of heterogeneous dynamic programming.

Algorithm 1 Dynamic Programing (DP) in 1-D topology

Input: The network matrix, D, and worker quality matrix, Q.
Output: The worker recruit decision, $\mathbb{X}$.

Initialize $opt[i, j] = 0$, $c_{max} = 0$.
for w_i from 1 to n **do**
  for l_j from 1 to m **do**
    for $w_{i'}$ from 1 to i **do**
      for $l_{j'}$ from 1 to j **do**
        Update $opt[i, j]$ based on Eq. 6.
Retrieve the recruitment in $opt[n, m]$.
Return the optimal worker recruitment $\mathbb{X}$.

Algorithm 2 Nearest Assignment (NA)

Input: The network matrix, D, and worker quality matrix, Q.
Output: The worker recruit decision, $\mathbb{X}$.

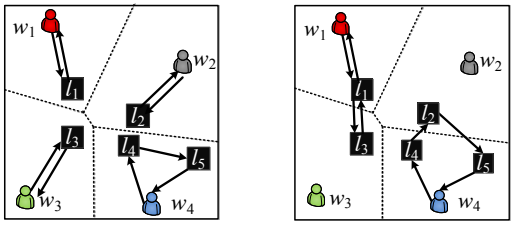
for l_j from 1 to m **do**
  Find out $w_i = \arg\min_{v \in V} d_{ij}$.
  Update X^i by adding l_j to its tour.
Return the optimal worker recruitment $\mathbb{X}$.

B. Heterogeneous Worker Crowdsourcing Quality

The heterogeneous crowdsourcing quality means different workers may have different crowdsourcing qualities for the same task. Note that a worker has the same crowdsourcing quality for all assigned tasks. In this scenario, we propose another dynamic programming method. As in the homogeneous case, we sort the workers and tasks directionally. The difference is that in this case, the workers and tasks are separately sorted. Without loss of generality, we can denote the ith crowdsourcing location and worker as l_i and w_i, respectively. The dynamic programming works as follows: We keep $O(nm)$ states, where $opt[i, j]$ is the highest effective ratio for first i workers ends with first j crowdsourcing tasks. $opt[i, j]$ is calculated by using two values, $opt[i, j].q$ and $opt[i, j].c$, which denote the corresponding total quality and the total cost to achieve $opt[i, j]$, i.e., $opt[i, j] = opt[i, j].q/opt[i, j].c$. Then, we can get the following optimal substructure,

$$opt[i, j] = \max \begin{cases} opt[i-1, j], \\ \dfrac{opt[i', j'].q + q_i(j - j')}{opt[i', j'].c + 2(\max\{l_i, l_j\} - \min\{l_i, l_{j'+1}\})} \\ \forall i', j', i' < i, j' < j. \end{cases} \tag{6}$$

The idea is that we gradually take workers into consideration based to their locations, e.g., from left to right. If the current



(a) Nearest assignment (b) Optimal assignment

Fig. 5. An illustration of the nearest assignment approach.

worker w_i is not recruited, the availability of worker w_i will not change the current optimal result for the first j crowdsourcing locations, i.e., $opt[i,j] = opt[i-1,j]$. If the worker w_i is recruited, we iterate all the possible assignments of w_i for first j crowdsourcing locations to check if we can get a better assignment.

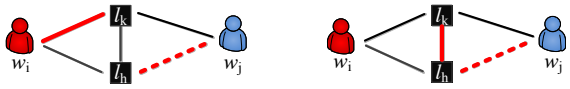
Fig. 4 shows an example of the dynamic programming procedure for Fig. 3. Let us assume that the crowdsourcing qualities of workers w_1 and w_2 are 0.5 and 1, respectively. All states is 0 initially. In this example, the $opt[1,1]$ is the maximal ratio end by using w_1 to cover l_1, since there is only one worker in such a scenario, $opt[1,1] = opt[1,1].q/opt[1,1].c = 0.5/2 = 1/4$, similarly, we can get $opt[1,2] = 0.5\times2/6 = 1/6$ and $opt[1,3] = 0.5 \times 3/14 = 3/28$, respectively. We also update the other two tables, $opt[1,1].q = 0.5, opt[1,2].q = 1, opt[1,3].q = 1.5$, $opt[1,1].c = 2, opt[1,2].c = 6$, and $opt[1,3].c = 14$. $opt[2,1] = \max\{opt[1,1], 1/(2 \times 6)\}$ = $1/4$, which is the comparison to use w_1 or w_2 to cover l_1. $opt[2,2] = \max\{(opt[1,0].q + 2)/(2 \times 6), (opt[1,1].q + 1)/(opt[1,1].c+3\times2) = 3/16, opt[1,2]\} = 3/16$. It represents three cases: (1) w_2 covering l_1 and l_2 (i.e., $w_2 \to l_2 \to l_1 \to w_2$) (2) w_1 covering l_1 and w_2 covering l_2, (i.e., $w_1 \to l_1 \to w_1$ and $w_2 \to l_2 \to w_2$), and (3) w_1 covering l_1 and l_2 (i.e., $w_1 \to l_1 \to l_2 \to w_1$). The corresponding $opt[2,2].q$ is updated to 1.5 and $opt[2.2].c$ is 8. Similarly, we can calculate the $opt[2,3]$ which is $\max\{opt[1,0]+3/(7\times2), (opt[1,1].q+1\times2)(opt[1,1].c+4\times2) = 1/4, (opt[1,2].q+1)(opt[1,1].c+1\times2) = 3/16, d[1,2]\}$ = $1/4$. The best assignment is that w_1 conducts the task at l_1 and w_2 conducts the task at locations at l_2 and l_3. We also update the corresponding $opt[2,3].q$ to 2.5 and $opt[2.3].c$ to 10.

The time complexity of the dynamic programming is as follows. We have $O(mn)$ states. For each state, there will be at most $O(mn)$ rounds to get the optimal value up-to-now. The update operation is $O(1)$. Therefore, the overall time complexity is $O(m^2n^2)$. The space complexity is $O(mn)$. Note that we can also use this method in the homogeneous case, but it leads to a higher complexity.

V. 2-D Topology

A. Homogeneous Crowdsourcing quality

If every worker has identical crowdsourcing quality, the CWRP can be reduced to a problem where we would like to plan at most n tours and the total movement distance of these tours is maximized. Similar to the 1-D scenario, where we first discuss the scenario where all workers have the same



(a) Nearest assignment (b) Optimal assignment

Fig. 6. A comparison between two approaches.

crowdsourcing ability. Even in such a case, the CWRP is NP-hard according to the proof of the Theorem 1. Therefore, a natural idea is to propose efficient heuristic approaches.

1) Nearest Assignment Algorithm: One natural idea is that we can partition the 2D space into a Voronoi graph [27], based on the workers' locations. After that, crowdsourcing locations that fall into the same region in the Voronoi graph are assigned to the same worker. The idea behind this approach is that each crowdsourcing location is assigned to its nearest worker. An illustration of this approach is shown in Fig. 5(a). However, sometimes it is not always optimal to assign a crowdsourcing location to its nearest worker, as shown in Fig. 5(b). Note that some workers might not be recruited by this approach.

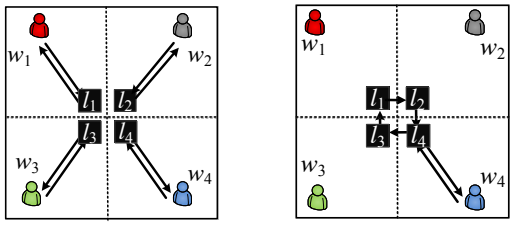
Theorem 2. *The nearest assignment achieves an approximation ratio of* $2n$ *in the homogeneous case.*

Proof. Without loss of generality, let us assume that we find a crowdsourcing location l_k which is assigned to w_i in the nearest assignment. However, in the optimal solution l_k is assigned to w_j through connection with l_h, possibly with multiple hops. An illustration is shown in Fig. 6, where the solid line is one-hop and the dotted line may be multiple hops. Note that l_k can also connect to w_j directly. The optimal solution is better, that is, $d_{ik} + d_{hj} \geq d_{kh} + d_{hj}$, which is equivalent to $d_{ik} \geq d_{kh}$. On the one hand, based on the greedy property of nearest assignment, we also know that $d_{ik} \leq d_{kj}$. Otherwise, l_j will be assigned to w_j. On the other hand, based on the property of metric space $d_{kj} \leq d_{ih} + d_{hj}$. We have $d_{kh} + d_{hj} \leq d_{ik} + d_{hj} \leq d_{kj} + d_{hj} \leq d_{ih} + 2d_{hj} \leq 2d_{ih} + 2d_{hj} = 2opt$.

For each worker w_i, the number of possible w_j which can reduce the overall length at most $n-1$, which is the largest number of adjunct regions for a region in the Voronoi graph. Therefore, the nearest assignment has an approximation ratio of $2n$ in the worst case. □

Note that in the 1-D scenario, every region is adjunct by two regions and thus nearest assignment achieves an approximation ratio of 2 by using the same proof. The idea behind the Theorem 2 is that the nearest assignment doesn't consider the crowdsourcing location distribution. A tight example of Theorem 2 is shown in Fig. 7, where the nearest assignment will assign each worker a crowdsourcing location. However, in the optimal solution, even through crowdsourcing locations l_1, l_2, and l_3 have larger distance to w_4, it is better to assign them to worker w_4, since after w_4 reaches l_4, there is small extra distance for w_4 to reach l_1, l_2 and l_3.

2) MST-based Algorithm: To address the problem of the nearest assignment, that the crowdsourcing location may not be assigned to the nearest worker, we propose a Minimum Spanning Tree (MST) based solution.



(a) Nearest assignment (b) Optimal assignment

Fig. 7. An worst-case analysis of the nearest assignment.

Algorithm 3 MST-based (MST) Algorithm

Input: The network matrix, D, and worker quality matrix, Q.
Output: The worker recruit decision, $\mathbb{X}$.

1: Construct the graph $\bar{G}(E,V)$.
2: Solve the minimum spanning tree.
3: Generate spanning forest by removing the edges between dummy nodes and worker nodes.
4: Generate the shortest-path tour(s) based on the minimum spanning forest.
5: Return the optimal worker recruitment $\mathbb{X}$.

(a) We construct a graph $\bar{G}(E,V))$, where V includes m crowdsourcing locations, n initial locations of workers and a dummy node. There is an edge between any pair of crowdsourcing locations and any pair of the initial worker location and the crowdsourcing location. The edge weight is the corresponding shortest path distance between them. In addition, there is an edge between the dummy node and any initial worker location, where the corresponding edge weight is 0.
(b) Based on the graph $\bar{G}(E,V))$, we find a MST, where the sum of degrees of the cervices denoting the initial worker locations is at most $2n$. The detailed method will be explained later.
(c) If we remove all edges between the dummy node and nodes which represent the initial worker locations, we generate a forest consisting of at most n non-trivial trees.
(d) We can construct a tour for each of the worker based on the edges of the non-trivial trees.

An illustration of the MST-based solution is shown in Fig. 8. The MST can be formulated as follows:

$$\min \quad \sum_{ij\in E} d_{ij}y_{ij} \tag{7}$$

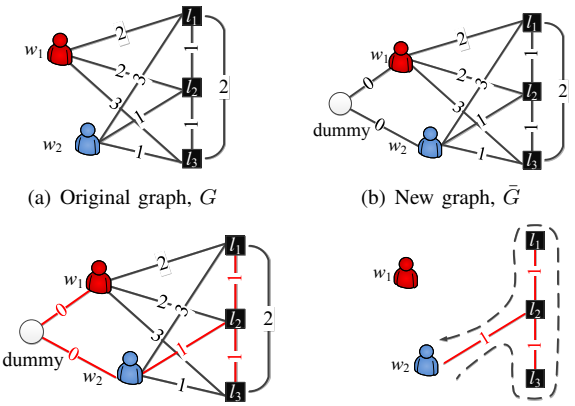
$$\text{s.t.} \quad \sum_{ij\in E} y_{ij} = n+m, \tag{8}$$

$$\sum_{ij\in E, i\in S, j\in S} y_{ij} \le |S|-1 \qquad \forall S \subseteq V \tag{9}$$

$$y_{ij} \in \{0,1\}, \forall i,j \in E \tag{10}$$

where y_{ij} is a recruitment decision variable. $y_{ij}=1$ means the corresponding edge is used to build the MST. The first constraint ensures two endpoints of any edge cannot be selected at the same time, i.e., $y_{ij} \le 1$. The second constraint ensures the number of edges is the total number of nodes minus 1. The first two constraints capture the properties of a spanning tree, namely that it contains no cycle.

An illustration of the proposed MST-based algorithm is shown in Fig. 8. In Fig. 8(b), we construct a new graph and add a dummy node, which has connection to workers w_1 and w_2. The corresponding edge weight between the dummy node and any worker is 0. After that, we can run MST algorithm to generate a MST, as shown in Fig. 8(c). Then, we remove all the edges between the dummy nodes and workers and the graph becomes a minimum spanning forest. Note that it is possible that some of the workers are not selected in the minimum spanning forest. In Fig. 8(d), the worker w_1 is not selected. Also, the total number of spanning tree in the spanning forest cannot exceed n. Then, we can generate a tour based on the spanning tree of each selected worker.



(a) Original graph, G (b) New graph, $\bar{G}$

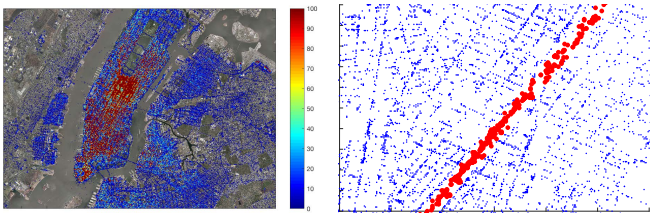
(c) Find the MST (d) Find the TSP tour(s)

Fig. 8. An illustration of the MST-based solution.

Theorem 3. *The proposed MST-based algorithm algorithm has an approximation ratio of* $1-\epsilon$ *in Euclidean space, where* ϵ *is an arbitrarily small positive number.*

Proof. In the steps (a) and (b), we generate a minimum spanning tree in $\bar{G}(E,V)$. In step (c), we generate a minimum spanning forest by removing all edges between dummy nodes and initial worker locations, which have the edge weight of 0. Clearly, the overall edge sum in the minimum is less than the optimal solution for CWPR. Otherwise, we can remove an arbitrarily edge in each tour of the CWPR solution to get a better minimum spanning forest. For each spanning tree, we can generate a corresponding tour which can double the total length. How to generate the shortest tour has been widely researched. The most state-of-the-art result algorithm achieves an approximation ratio of $1+c$ in the Euclidean space [28], where c can be an arbitrarily small positive number, by using the fully polynomial-time approximation scheme. Therefore, the overall performance of the MST-based algorithm has an approximation ratio of $1/(1+c)$, which is equivalent to $1-\epsilon$, where ϵ is an arbitrarily small positive number. □

The time complexity of MST-based solution is as follows: the graph construction is $O((m+n)^2)$, and solving the matroid intersection problem is $O((m+n)^3)$, according to [29]'s algorithm. The native TSP tour construction is $O(m+n)$. Therefore, the overall time complexity is $O((m+n)^3)$. However, if we use the nearest assignment algorithm, the overall time complexity is $O(n \log n)$, which is the time complexity



(a) New York City (b) Broadway, Manhattan

Fig. 9. Uber pick-up trace visualization.

of building the Voronoi graph.

B. Heterogeneous Crowdsourcing Quality

If we ignore the heterogeneous crowdsourcing quality of workers, we can directly apply the MST-based solution as discussed in Section V-A. However, in this case, it is hard to balance the two aspects of crowdsourcing operation cost and the total distance.

Theorem 4. *The proposed MST-based algorithm has an approximation ratio of $1/\rho - \epsilon$ in the heterogeneous case, where ρ is the maximal quality ratio between any two workers.*

Proof. Without loss of generality, let us assume that apply the MST-based solution in the heterogeneous case and denote the result as ALG. In addition, we can assume that each worker has the minimum/maximum crowdsourcing quality, and thus we can get two result, ALG' and ALG''. Clearly, we can get the following result, $ALG' \leq ALG \leq ALG''$. If we denote the optimal solution as OPT. Clearly, $OPT \leq (1+c)ALG''$, since some workers may have lower qualities. Then, we have the following result, $OPT \leq (1+c)ALG'' = (1+c)\rho ALG' \leq (1+c)\rho ALG$. Therefore, $ALG \geq 1/\rho(1+c)OPT > 1/\rho - \epsilon OPT$. This completes the proof. □

VI. Performance Evaluation

A. Traces

It is hard to get the people's location data due to the privacy issue. Therefore, we used a Uber pick-up trace from the New York City Taxi & Limousine Commission (TLC) [30]. This directory contains data on over 4.5 million Uber pickups in New York City from April to September 2014, and 14.3 million more Uber pickups from January to June 2015. In the experiments, we picked the April data, which has $564,516$ records. An illustration of the pick-up distribution is shown in Fig. 9(a), where a point represents a Uber pick-up event. Fig. 9(a) is a heat graph, where the hot color means a high density.

B. Experiment Setting

In the experiment, we picked up two areas, one is in Manhattan, i.e., $[40.72N, 40.74N, 73.98W, 74.01W]$ and the other one is outside of Manhattan, i.e., $[40.70N, 40.72N, 73.94W, 73.97W]$. We split the time into 24 hours and used all the events within one hour as the crowdsourcing location setting. In 1-D setting, we pick all pick-up events in the Broadway, Manhattan. An illustration is shown in Fig. 9(b). We use the statistic result of Uber driver rating distribution from [24] to generate a worker's

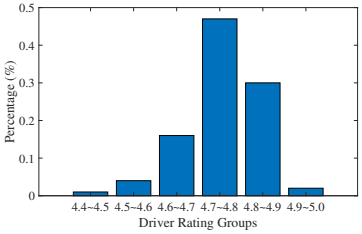


Fig. 10. An illustration of Uber driver rating distribution.

crowdsourcing quality and there are 6 different quality values in the experiment. Fig. 10. The quality of each worker was randomly picked from one of the six quality values. In each round of experiment, we randomly picked-up a certain number of pick-up events and randomly selected some of the locations as the workers' initial locations, and the remaining locations are crowdsourcing locations. All experiments were repeated for 1000 to 2000 times.

C. Algorithm Comparison

We compare four algorithms in the 1-D scenario:(1) Dynamic Programming (DP) Algorithm. We have explained it in Section IV. (2) Nearest Assignment (NA) Algorithm. Each crowdsourcing location is assigned to the worker who is the nearest. (3) Minimum Distance (MD) Algorithm. We try to find the shortest tour(s) from workers so that they can finish all crowdsourcing tasks. (4) Maximum Quality (MQ) Algorithm. We try to find the worker(s) with the highest quality, and recruit them to finish all crowdsourcing tasks. If there are multiple such workers, each crowdsourcing task is assigned to the nearest worker. In the 2-D scenario, the dynamic programming doesn't work. The MST-based algorithm tries to find the shortest-path by using all workers. In each scenario, we can always find the optimal solution by using the Brute Force (BF) Algorithm in a small scale. We use the effectiveness defined in Eq. 1 during the problem formulation to evaluate the performance of proposed algorithms.

D. Running time Analysis

In the experiments, codes were implemented in Matlab and were executed on a laptop with a 2.2 GHz Intel Core i7 2450M processor and 16 GB Memory. First, we show the running time results for algorithms in 1-D scenarios. From Figs. 11(a) and 11(b) We did not try larger scale. It is because that we observe that the exhaustive search algorithm has a really high time complexity and thus cannot be applied on a large scale. In the experiments, when the number of tasks is larger than 8 and the number of workers is larger than 5, the running time becomes unacceptable. Note that Fig. 11(a) has a logarithmic scale. The DP algorithm has the same performance as the exhaustive search but the time complexity is much lower than the exhaustive search. To clearly show the time difference between DP and the NA algorithms, we also provide Fig. 11(b) in the linear scale. Similarly, we conduct experiments in the 2-D scenario, and the result is shown in Fig. 11(c).

E. Results in 1-D scenario

1) The influence of task amount: Fig. 12(a) shows the results of four algorithms in terms of different task amount

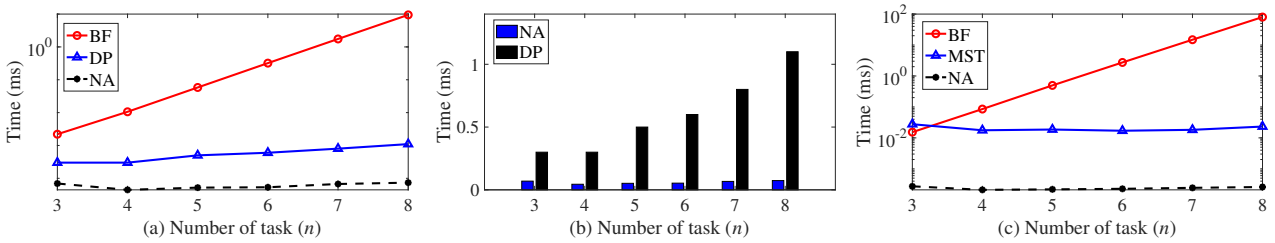


Fig. 11. Running time results.

in the 1-D scenario. The results show that when the task amount is small, the tour distance dominates the result. That is the reason why DP, NA, and ST achieve similar results, but the MQ has a much worse performance, due to the long tour length. When the task amount is much larger than the number of workers, the advantage of the proposed DP algorithm become very clear and it achieves a much better performance than other three algorithms. When number of task is 100, the DP achieves more than 20% higher effectiveness compared with other there algorithms.

2) The influence of worker amount: Fig. 12(b) shows the results of four algorithms with different worker amount in the 1-D scenario. We observe there is a turning point in terms of the performance between the proposed algorithm and the NA/ST algorithm. The turning point also demonstrates the trade-off between tour length and the quality. When the total number of workers is small, in order to get the high efficiency, it is more important to find the workers which have good qualities. When the total number of workers is large, we can generate a real good assignment to reduce the overall tour length. That is, the tour length dominates the result when the number of workers is large. From Fig. 12(b), the MQ's efficiency is less than 30% of that of NA/ST's, and less than 50% of that of DP's when the number of workers is 20.

3) The influence of quality value: Fig. 12(c) shows the results of four algorithms with different quality values in the 1-D scenario. When there is only one quality value, the overall tour length determines the crowdsourcing efficiency and thus the MQ achieves the worst performance. With the increase of the possible quality values, selecting good workers with high quality becomes very important and the DP algorithm can catch this trend automatically.

F. Results in 2-D scenario

1) The influence of task amount: Fig. 13(a) shows the results of three algorithms with different task amount. The proposed MST algorithm achieves the best performance, followed by the MA and NA algorithm, respectively. The results show the influence of task amount clearly. Given a certain number of workers, the performance gap between MST algorithm and the other two algorithm decreases with the increase of total task amount. When the task amount is small, the advantage of the MST algorithm is clear. When the task amount is large, the overall effectiveness is dominated by the total number of tasks. As a result, different algorithms achieve the similar performance since there is no short tour(s) available.

2) The influence of worker amount: Fig. 13(b) shows the results of three algorithms with different worker amount in the 2-D scenario. We observe there is a turning point in terms of the performance between the MST algorithm and the NA algorithm. This turning point also demonstrates the trade-off between the tour length and the quality. That is, the tour length dominates the result when the number of workers is large. The efficiency ratio of the MST algorithm is 300% more than the efficiency ratio of NA's and MQ's when $m = 20$.

3) The influence of quality value: Fig. 13(c) shows the result of three algorithms with different quality values in the 2-D scenario. When there is only one quality value, the overall tour length determines the crowdsourcing efficiency and thus the MQ achieves the worst performance. However, with the increase of possible quality values, selecting good workers with high quality becomes very important. The MST algorithm has the highest performance increase speed, and the NA algorithm is the worst since it might lead to a long tour length and low total quality.

VII. Discussion

The proposed solution can be extended to some other cases easily. The first potential extension is that we only considers the case that different workers may have different crowdsourcing qualities for the same task in this paper. For a single worker, the quality for different tasks are all the same, however in reality, a worker may have different qualities for different tasks. However, the proposed solution in Section V-B can be used to solve the problem in that case with the same performance bound. Another extension is that the worker does not need to come back to his/her initial location after visiting the last assigned crowdsourcing location. We can still use the proposed algorithm in this paper with an additional approximation of 2, since the proposed algorithm at most double the overall tour length. In addition, this paper does not consider the load balance of workers. However, we can address this problem by setting a maximum load constraint during the problem formulation.

VIII. Conclusion

We address the efficient worker recruitment problem under the coverage constraint in the spatial crowdsoucing. Specifically, we consider two different scenarios, 1-D line topology and general 2-D, in which workers may have either homogeneous or heterogeneous crowdsourcing quality. In the 1-D scenario, we propose two dynamic programming approaches to find the optimal solution, respectively. Then, we extend the

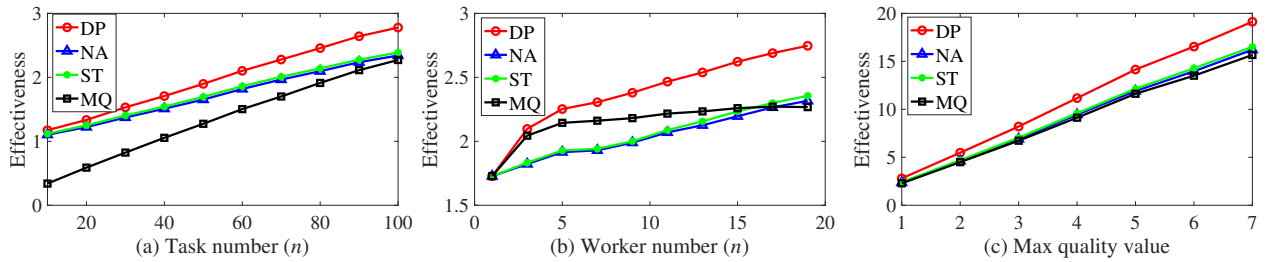


Fig. 12. Results in 1-D topology.

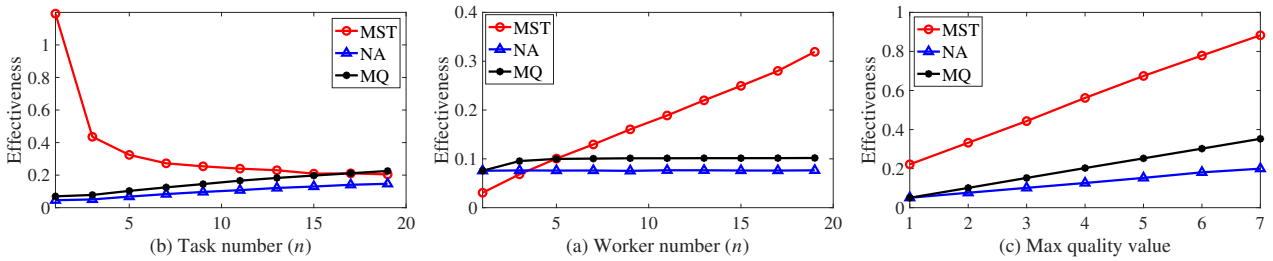


Fig. 13. Results in 2-D topology.

model into a general heterogeneous case. In the heterogeneous setting, the CWRP problem turns out to be NP-hard even when all workers have the same crowdsourcing quality. The effectiveness of the proposed algorithm is verified by the real mobility trace. The results show that the proposed approaches can greatly improve the crowdsourcing effectiveness.

IX. Acknowledgement

This research was supported in part by NSF grants CNS 1824440, CNS 1828363, CNS 1757533, CNS 1629746, CNS-1651947, CNS 1564128.

References

[1] J. Yick, B. Mukherjee, and D. Ghosal, "Wireless sensor network survey," *Computer Networks*, vol. 52, no. 12, pp. 2292–2330, 2008.

[2] Y. Zhao and Q. Han, "Spatial crowdsourcing: current state and future directions," *IEEE Communications Magazine*, vol. 54, no. 7, pp. 102–107, 2016.

[3] P. Dutta, P. M. Aoki, N. Kumar, A. Mainwaring, C. Myers, W. Willett, and A. Woodruff, "Common sense: participatory urban sensing using a network of handheld air quality monitors," in *Proc. of the ACM SenSys*, 2009.

[4] S. Mathur, T. Jin, N. Kasturirangan, J. Chandrasekaran, W. Xue, M. Gruteser, and W. Trappe, "Parknet: drive-by sensing of road-side parking statistics," in *Proc. of the ACM MobiSys*, 2010.

[5] P. Mohan, V. N. Padmanabhan, and R. Ramjee, "Nericell: rich monitoring of road and traffic conditions using mobile smartphones," in *Proc. of the ACM SenSys*, 2008.

[6] https://www.uber.com.

[7] J. Lehmann, C. Castillo, M. Lalmas, and E. Zuckerman, "Finding news curators in twitter," in *Proceedings of the ACM WWW*, 2013.

[8] https://www.Grubhub.com.

[9] S. Yang, F. Wu, S. Tang, X. Gao, B. Yang, and G. Chen, "On designing data quality-aware truth estimation and surplus sharing method for mobile crowdsensing," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 4, pp. 832–847, 2017.

[10] J. Li, Y. Zhu, J. Yu, Q. Zhang, and L. M. Ni, "Towards redundancy-aware data utility maximization in crowdsourced sensing with smartphones," in *Proc. of the IEEE ICDCS*, 2015.

[11] 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.

[12] H. Yu, C. Miao, Z. Shen, and C. Leung, "Quality and budget aware task allocation for spatial crowdsourcing," in *Proc. of the IFAAMAS AAMAS*, 2015.

[13] 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.

[14] D. Deng, C. Shahabi, and U. Demiryurek, "Maximizing the number of worker's self-selected tasks in spatial crowdsourcing," in *Proc. of the ACM SIGSPATIAL*, 2013.

[15] J. She, Y. Tong, and L. Chen, "Utility-aware social event-participant planning," in *Proc. of the ACM SIGMOD/PODS*, 2015.

[16] Y. Tong, J. She, B. Ding, L. Chen, T. Wo, and K. Xu, "Online minimum matching in real-time spatial data: experiments and analysis," *Proc. of the VLDB Endowment*, 2016.

[17] https://www.waze.com.

[18] Y. Chon, N. D. Lane, Y. Kim, F. Zhao, and H. Cha, "Understanding the coverage and scalability of place-centric crowdsensing," in *Proc. of the ACM Ubicomp*, 2013.

[19] H. Xiong, D. Zhang, G. Chen, L. Wang, and V. Gauthier, "Crowdtasker: Maximizing coverage quality in piggyback crowdsensing under budget constraint," in *Proc. of the IEEE PerCom*, 2015.

[20] 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.

[21] X. Zhang, Z. Yang, Y. Liu, and S. Tang, "On reliable task assignment for spatial crowdsourcing," *IEEE Transactions on Emerging Topics in Computing*, no. 1, pp. 1–1, 2016.

[22] Z. He, J. Cao, and X. Liu, "High quality participant recruitment in vehicle-based crowdsourcing using predictable mobility," in *Proc. of the IEEE INFOCOM*, 2015.

[23] 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.

[24] https://www.businessinsider.com/leaked-charts-show-how-ubers-driver-rating-system-works-2015-2.

[25] L. Gouveia and J. M. Pires, "The asymmetric travelling salesman problem and a reformulation of the miller–tucker–zemlin constraints," *European Journal of Operational Research*, vol. 112, no. 1, pp. 134–146, 1999.

[26] M. R. Garey and D. S. Johnson, "Computers and intractability: A guide to the theory of *np*-completeness," *Bulletin (New Series) of the American Mathematical Society*, vol. 3, no. 2, pp. 898–904, 1980.

[27] F. Aurenhammer, "Voronoi diagramsa survey of a fundamental geometric data structure," *ACM Computing Surveys*, vol. 23, no. 3, pp. 345–405, 1991.

[28] S. Arora, "Polynomial time approximation schemes for euclidean traveling salesman and other geometric problems," *Journal of the ACM*, vol. 45, no. 5, pp. 753–782, 1998.

[29] E. L. Lawler, *Combinatorial optimization: networks and matroids*. Courier Corporation, 1976.

[30] http://www.nyc.gov/tlc.