Hindawi
Journal of Sensors
Volume 2017, Article ID 8568613, 10 pages
https://doi.org/10.1155/2017/8568613

Research Article

Finding Optimal Team for Multiskill Task Based on Vehicle Sensors Data

Bowen Du, Qian Tao, Feng Zhu, and Tianshu Song

State Key Laboratory of Software Development Environment, School of Computer Science and Engineering and IRI, Beihang University, Beijing, China

Correspondence should be addressed to Qian Tao; qiantao@buaa.edu.cn

Received 28 July 2017; Accepted 12 September 2017; Published 31 October 2017

Academic Editor: Jia Hu

These days, with the increasingly widespread employment of sensors, particularly those attached to vehicles, the collection of spatial data is becoming easier and more accurate. As a result, many relevant areas, such as spatial crowdsourcing, are gaining ever more attention. A typical spatial crowdsourcing scenario involves an employer publishing a task and some workers helping to accomplish it. However, most of previous studies have only considered the spatial information of workers and tasks, while ignoring individual variations among workers. In this paper, we consider the Software Development Team Formation (SDTF) problem, which aims to assemble a team of workers whose abilities satisfy the requirements of the task. After showing that the problem is NP-hard, we propose three greedy algorithms and a multiple-phase algorithm to approximately solve the problem. Extensive experiments are conducted on synthetic and real datasets, and the results verify the effectiveness and efficiency of our algorithms.

1. Introduction

These days, with the development of sensors (especially vehicle sensors and mobile sensors) [1–3], it is increasingly simple to acquire spatial and temporal information [4, 5].

Many studies based on vehicle sensors data have been conducted in recent years [6–8]. As a result, many applications now provide services based on users' real-time spatial information and these are becoming ever popular. Among these applications, some focus on crowdsourcing services that use spatial information. These applications usually require some workers to help an employer to accomplish a task. For example, Uber (https://www.uber.com) organizes drivers and provides users with a convenient taxi service, whereas +Meituan (http://www.meituan.com) provides a credible and fast food-delivery service. This area, called spatial crowdsourcing, is attracting significant attention.

The task assignment problem is one of the fundamental concerns in spatial crowdsourcing. For example, real-time taxi-calling platforms, such as Uber and Didi Chuxing [9], always need to assign each taxi-calling task to a suitable taxi (i.e., a crowd worker). An incorrect assignment may cause taxis to be dispatched to far-away places, which results in a slow response time and the loss of the platform. Many studies on the task assignment problem have been published in recent years [10–12]. However, most of them only consider the spatial information of tasks and workers, while ignoring the individual variations among workers. Namely, different people may excel or struggle with different tasks, and tasks also contain certain requirements for which some workers may be inadequate.

Take Figure 1 as an example. Suppose a website development task requires coders skilled in .NET, SQL, and HTML to assemble at the location of the origin, and there are three coders available (whose skills are presented in Figure 1). Although coder c_3 is located closer to the origin than c_1 and c_2, hiring c_3 will not help finish the task. In other words, it is necessary to further consider individual variations among different workers and special requirements of tasks.

As in [13, 14], each worker is associated with a set of skills representing their strengths. Tasks are also associated with a set of skills representing their special requirements.

R-trees [15] are a classical index structure for multidimensional data. Derived from B-tree, the data in an R-tree are stored in leaf nodes and all leaves are located in the same level of the tree. Every internal node contains between m and

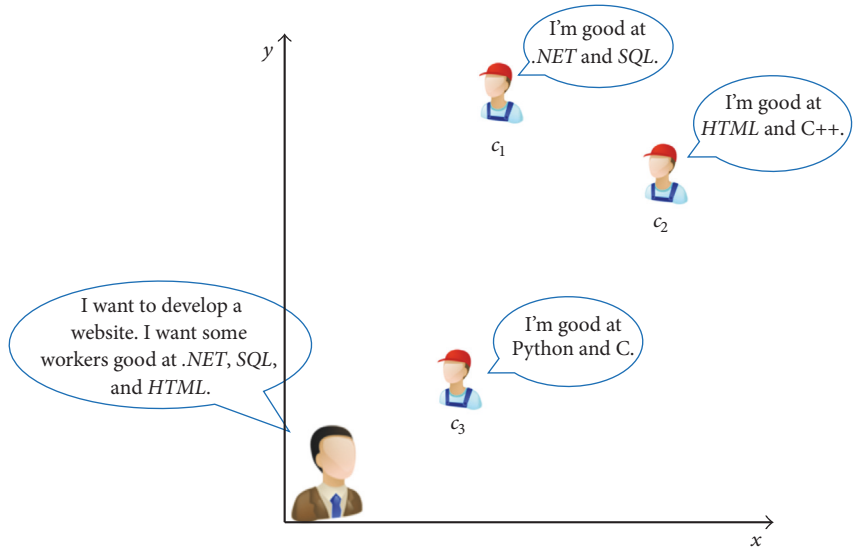


Figure 1: Example: variations among workers.

M child entries, and every leaf node contains between m and M data entries, where M is usually related to the size of disk pages, and m is predefined such that $m \leq M/2$. The tree is specially structured such that the children of a node overlap with few data from other nodes. Using an R-tree, we can dynamically insert/update/delete nodes, and rapidly search for all nodes located in a given rectangle.

The objective of our problem consists of two parts. First, workers need to move to the location of the task but receive no reward for this movement. In consideration of the workers, we attempt to reduce the gratuitous moving distance. Second, the employer wishes spend the minimum amount necessary to accomplish the task. In consideration of employers, we attempt to obtain a team at the lowest cost, on condition that the skill requirement is satisfied. As the problem definition in Section 2 shows, the objective of our work contains not only the distance between the task and workers, but also the total cost.

Contributions. In summary, our contributions are as follows:

(i) We propose a new Software Development Team Formation (SDTF) problem and prove that it is NP-hard.

(ii) Three greedy algorithms are provided to solve the SDTF problem.

(iii) We employ a multiphase algorithm based on R-trees.

(iv) We verify the effectiveness and efficiency of the proposed algorithms through extensive experiments on synthetic and real datasets.

Compared with our previous work [16], we propose a novel multiple-phase algorithm by using the index structure of R-trees. Additional experiments are also conducted on synthetic and real datasets.

The remainder of this paper is organized as follows. In Section 2, the problem is formally defined and proved to be NP-hard. In Section 3 three greedy algorithms are provided to solve the SDTF problem. In Section 4, we propose a multiple-phase algorithm based on R-trees. Extensive experiments on real datasets are described in Section 5. Previous work related to our problem is presented in Section 6, and the conclusions to this study are presented in Section 7.

2. Problem Statement

First, we introduce the two basic concepts of a task and a coder. We then formally define the Software Development Team Formation (SDTF) problem.

Definition 1 (task). A task t is defined as $\langle S, L\rangle$, where $t.S$ is a set of skills that are indispensable to complete the software development task t, and $t.L$ is the location specified to meet up and talk about task t, which, for example, can be described by longitude and latitude.

Similar to the definition of a task, a coder is formally defined as follows.

Definition 2 (coder). A coder c is defined as $\langle S, L, P\rangle$, where $c.S$ is a set of skills mastered by coder c, $c.L$ is the location of coder c, described similarly to that of a task t, and $c.P$ is the price of coder c.

Briefly, a team of coders is feasible for a task if the coders in the team can collaboratively accomplish the task.

Definition 3 (feasible team). A team T is defined as a set of coders $\{c_1, c_2, \ldots, c_{|T|}\}$. T is a feasible team for task t, if $\bigcup_{c_i \in T} c_i.S \supseteq t.S$.

Example 4. Suppose that we have a task t concerning website development, where $t.S$ = {LINUX, DATABASE, CSS, HTML} and a universal set of coders $C = \{c_1, c_2, c_3, c_4, c_5, c_6\}$.

Table 1: Coder profile.

Coder	*Skill*	*Price*	*Distance*
c_1	{PYTHON, CSS, C++}	70	1000
c_2	{CSS, LINUX, HTML}	60	5000
c_3	{LINUX, HTML}	50	7000
c_4	{DATABASE, JAVA}	55	2000
c_5	{PYTHON, DATABASE, LINUX}	65	6000
c_6	{HTML, C#}	60	3000

The skill set of every $c \in C$ is listed in Table 1. Team $T = \{c_2, c_4\}$ is a feasible team because $\bigcup_{c \in T} c.S = \{$CSS, LINUX, HTML, DATABASE, JAVA$\}$, which is a superset of $t.S = \{$LINUX, DATABASE, CSS, HTML$\}$.

We finally define our problem as follows.

Definition 5 (SDTF problem). Given a task t with a set of skills $t.S$ and a location $t.L$ and a universal set of coders $C = \{c_1, c_2, \ldots, c_{|C|}\}$, each with skill set $c_i.S$, location $c_i.L$, and price $c_i.P$, $1 \le i \le |C|$, we wish to find a team $T \subseteq C$ satisfying $\bigcup_{c \in T} c.S \supseteq t.S$, and we minimize Cost $= \alpha \cdot \max_{c \in T} |c.L, t.L| + (1 - \alpha) \cdot \sum_{c \in T} c.P$, where $|c.L, t.L|$ represents the distance between the location of coder c and task t and $\alpha \in (0, 1)$ is a parameter to adjust the weight of distance and price.

Theorem 6. *The SDTF problem is NP-hard.*

Proof. We prove the theorem by showing that a special case of the SDTF problem can be reduced from the weighted set cover problem. An instance of a weighted set cover problem consists of a set $U = \{1, 2, \ldots, n\}$ and a set $S = \{S_1, S_2, \ldots, S_{|U|}\}$, where $S_i \subseteq U$ for $1 \le i \le |U|$. Each S_i is associated with a positive value W_{S_i}, which can be viewed as the weight of S_i. The weighted set cover optimization problem aims to find a subset S^* of S satisfying $\bigcup_{S_i \in S^*} S_i \supseteq U$ minimizing $\sum_{S_i \in S^*} W_{S_i}$.

We consider a special case of the SDTF problem in which the task and coders are located in the same position, and the skill set of the task is the universal set of all skills. To reduce the weighted set cover problem to the special-case SDTF problem, we observe that each element in U corresponds to a skill in $t.S$, each element in S_i corresponds to a skill in $c_i.S$, and the weight of S_i corresponds to the price of c_i. As the task and all coders are at the same location, for every team T, $\max_{c \in T} |c.L, t.L| = 0$, and we need only minimize $\sum_{c \in T} c.P$. Obviously, there exists a solution to the weighted set cover problem if and only if there exists a solution to the special-case SDTF problem, and we can obtain an instance of the special-case SDTF problem from the instance of weighted set cover problem in polynomial time. Therefore, the general case of the SDTF problem is NP-hard. □

3. Greedy Solutions for SDTF

In this section, we present three greedy algorithms to solve the SDTF problem. The first two algorithms greedily choose the nearest/cheapest coder who can cover at least one uncovered skill. Because they only consider optimizing part of the objective function, the solution is sometimes not good enough. Thus we propose a third greedy algorithm that considers both price and distance when choosing a new coder.

input: Task t, Coders $C = \{c_1, c_2, \ldots, c_{|C|}\}$
output: A feasible team T
(1) $T \leftarrow \emptyset$;
(2) **while** T *is not feasible* **do**
(3) $\quad c \leftarrow \text{argmin}_{c \in C \&\& c.S \cap t.S \neq \emptyset} (c.P / |c.S \cap t.S|)$;
(4) $\quad T \leftarrow T \cup \{c\}$;
(5) $\quad t.S \leftarrow t.S - c.S$;
(6) **return** T

Algorithm 1: PF-SDTF.

3.1. Price First-SDTF Greedy Algorithm. The idea of the first greedy algorithm is to repeatedly add the cheapest coder to the team until the team is feasible. The whole procedure of this price first- (PF-) SDTF is illustrated in Algorithm 1. We assume that there exists at least one feasible team.

Considering that skills not in the skill set of the task contribute nothing to the accomplishment of the task, the term "cheapest coder" must be treated carefully. Here, we define the Average Price on Uncovered Intersecting Skills to describe how a coder contributes to the price part of the objective function:

$$\text{APUIS}(t, c) = \frac{c.P}{|c.S \cap t.S_u|}, \tag{1}$$

where S_u is the uncovered skill set of task t. We can see that APUIS describes how a coder influences the price part of the objective function if we add him/her to the final team. Choosing a coder with lower APUIS means we can satisfy the requirement of the skills with a lower total price. Note that when there is no intersection between the skill set of the worker and the uncovered skill set, APUIS will be infinity. Because we greedily choose the worker with the lowest APUIS, we omit this special case in (1).

In line (1) of Algorithm 1, we initialize an empty team T. In lines (2)–(5), when T is not feasible, we find a coder c who can cover at least one uncovered skill of task t and has the lowest $c.P/|c.S \cap t.S|$ value, add c to team T, and update $t.S$. Ties are broken by distance first, then arbitrarily. In line (6), we return the resulting feasible team T.

3.2. Distance First-SDTF Greedy Algorithm. The idea of distance first- (DF-) SDTF is to repeatedly add the nearest coder to the team until the team is feasible. The framework of DF-SDTF is similar to that of PF-SDTF. In each iteration, we find the nearest coder c_n who can cover at least one uncovered skill of task t; that is,

$$c_n = \underset{c \in C \&\& c.S \cap t.S_u \neq \emptyset}{\text{argmin}} |c.L, t.L|, \tag{2}$$

input: Task t, Coders $C = \{c_1, c_2, \ldots, c_{|C|}\}$
output: A feasible team T
(1) $T \leftarrow \emptyset$;
(2) **while** T *is not feasible* **do**
(3) $\quad c \leftarrow \text{argmin}_{c \in C \&\& c.S \cap t.S \neq \emptyset} |c.L, t.L|$;
(4) $\quad T \leftarrow T \cup \{c\}$;
(5) $\quad t.S \leftarrow t.S - c.S$;
(6) **return** T

Algorithm 2: DF-SDTF.

input: Task t, Coders $C = \{c_1, c_2, \ldots, c_{|C|}\}$
output: A feasible team T
(1) $T \leftarrow \emptyset$;
(2) **while** T *is not feasible* **do**
(3) $\quad c \leftarrow \text{argmax}_{c \in C \&\& c.S \cap t.S \neq \emptyset} \text{Utility}(c, t, T)$;
(4) $\quad T \leftarrow T \cup \{c\}$;
(5) $\quad t.S \leftarrow t.S - c.S$;
(6) **return** T

Algorithm 3: DP-SDTF.

where S_u is the uncovered skill set of task t. The whole procedure of DF-SDTF is illustrated in Algorithm 2. We assume that there exists at least one feasible team.

In line (1), we initialize an empty team T. In lines (2)–(5), when T is not feasible, we find the nearest coder c who can cover at least one uncovered skill of task t, add c to team T, and update $t.S$. Ties are broken by price first, then arbitrarily. In line (6), we return the resulting feasible team T.

3.3. Distance Price-SDTF Greedy Algorithm. The aforementioned two greedy algorithms are not effective, because they only try to optimize part of the objective function. To optimize both distance and price at every iteration, we design a utility function Utility. Given a task t, current team T, and coder c, the definition of Utility is

$$\text{Utility}(c, t, T) = \frac{|c.S \cap t.S|}{\alpha \cdot \Delta D(c, t, T) + (1 - \alpha) \cdot |c.P|}, \quad (3)$$

where $\Delta D(c, t, T)$ represents the increment in the maximum distance if c is added to team T, that is, $\Delta D(c, t, T) = |c.L, t.L| - \max_{c' \in T} |c'.L, t.L|$ if $|c.L, t.L| > \max_{c' \in T} |c'.L, t.L|$, and $\Delta D(c, t, T) = 0$ if $|c.L, t.L| \leqslant \max_{c' \in T} |c'.L, t.L|$. In fact, the value of $\alpha \cdot \Delta D(c, t, T) + (1 - \alpha) \cdot |c.P|$ in (3) is the increment in the objective function.

Using this utility function, we have a third greedy algorithm, Distance Price- (DP-) SDTF. The whole procedure of DP-SDTF is illustrated in Algorithm 3. We assume that there exists at least one feasible team.

In line (1), we initialize an empty team T. In lines (2)–(5), when T is not feasible, we find a coder c who gives the highest utility. Ties are broken by distance first, then arbitrarily. In line (6), we return the resulting feasible team T.

4. Multiple-Phase R-Tree Algorithm

In this section, we introduce an algorithm based on the R-tree data structure. Considering that some previous work has applied R-trees in Nearest Neighbor (NN) searching [17, 18], a naïve idea is to use an R-tree to accelerate the NN search in the DF-SDTF algorithm proposed in Section 3.2. However, this simple use of R-trees can only accelerate the search speed and does not help optimize the final cost. As our experiments will show, the DF-SDTF algorithm performs worse than the DP-SDTF algorithm proposed in Section 3.3. The above situation requires us to find an algorithm that is both efficient and effective in solving the SDTF problem.

Our original algorithm derives from an intuitive observation: if we query all nodes located in the square whose centroid is at the location of the task and whose side length is $2 \cdot l$, the distance between the task and the nodes in the result set will be at most $\sqrt{2} \cdot l$. This characteristic provides an applicable tool for the distance part of our objective function. By choosing a rectangle with suitable sides, we obtain a set of candidate coders who are close to the location of the task. The price part of the objective function can also be optimized if we employ a proper strategy to choose the next coder from the candidate coder set.

Based on the above observation, we propose the Multiple-Phase R-tree (MPR) algorithm. The main idea of our algorithm is as follows.

(1) Initialize a new R-tree and insert all coders into the tree.

(2) In each phase, obtain a candidate set of coders by querying all nodes located in the square whose centroid is at the location of the task.

(3) Sort all coders in the candidate set in descending order of APUIS. For each coder, add him/her to the final team T if his/her skills can cover at least one uncovered skill in the task.

(4) If team T is not feasible, return to step (2) and use a square with longer sides.

In detail, we generate the list of side lengths by uniformly dividing the maximum distance between the coders and the task. Given a parameter n_p denoting the number of phases, we first scan the whole set of coders and calculate the maximum distance between the coders and the task, maxDis. Then, we iteratively start a phase by using a square with side length $\text{maxDis}/n_p, 2 \cdot \text{maxDis}/n_p, \ldots, \text{maxDis}$, until we obtain a feasible team.

The pseudocode of our MPR algorithm is shown in Algorithm 4. First, we initialize the team T and find the maximum distance between the coders and the task in lines (1)-(2). Then, we calculate the step size of the sides between two phases in line (3). In each phase (iteration in lines (5)–(9)), we first query all nodes located in the square whose centroid is at the location of the task and whose side length is $2 \cdot i \cdot \text{step}$. We then alternately add coders with the minimum APUIS (lines (7)–(9)). Similarly, ties are broken by distance first, then arbitrarily. In line (10), we return the resulting feasible team T.

input: Task t, R-tree R, Coders
$C = \{c_1, c_2, \ldots, c_{|C|}\}$, number of phases n_p
output: A feasible team T
(1) $T \leftarrow \emptyset$;
(2) maxDis $\leftarrow$ maximum distance between coders and the task;
(3) step $\leftarrow$ maxDis$/n_p$;
(4) **for** $i \leftarrow 1$ *to* n_p **do**
(5) $\quad C_{\text{can}} \leftarrow R$.search$(t.L, 2 \cdot i \cdot \text{step})$;
(6) $\quad$ **while** T *is not feasible* **do**
(7) $\quad\quad c \leftarrow \text{argmin}_{c \in C_{\text{can}} \&\& c.S \cap t.S \neq \emptyset}(c.P/|c.S \cap t.S|)$;
(8) $\quad\quad T \leftarrow T \cup \{c\}$;
(9) $\quad\quad t.S \leftarrow t.S - c.S$;
(10) **return** T

Algorithm 4: MPR.

Table 2: Synthetic dataset.

Factor	Setting
α	0.1, 0.2, 0.3, 0.4, **0.5**, 0.6, 0.7, 0.8, 0.9
$\lvert t.\text{Skills}\rvert$	5, 6, 7, 8, 9, **10**, 11, 12, 13, 14, 15
$\lvert \bigcup_{c \in C} c.\text{Skills}\rvert$	100, 110, 120, 130, 140, **150**, 160, 170, 180, 190, 200
$\lvert C\rvert$	1 W, 2 W, 3 W, 4 W, **5 W**, 6 W, 7 W, 8 W, 9 W, 10 W

5. Evaluation

We applied our four algorithms to synthetic and real datasets. The algorithms were implemented in C++, and the experiments were performed on a machine with an Intel i7-4710mq 2.50 GHZ 4-core CPU and 8 GB memory.

5.1. Datasets. We use real and synthetic datasets to evaluate our algorithms. The real dataset is taken from CSTO (http://www.csto.com/) and includes 2033 active coders. In the CSTO dataset, each task is associated with a set of skills needed to complete a software development task, and each coder is associated with a set of skills and an average price that can be deduced from the history data. As few coders have associated price information (because many coders have not any completed tasks), we analyze the price distribution using coders associated with price information. Except for some expensive coders, the price of a coder is uniformly distributed in the range 0–5000 and is unrelated to the number of mastered skills. As the CSTO data are not associated with location information, we generate coordinates for each coder according to a uniform distribution.

For the synthetic data, based on our observations of the real dataset, we generate the price $c.P$ of coder c following a uniform distribution. We assume that each coder has 5–25 skills, which is common in practice. The distance from each coder to the task is generated according to a uniform distribution. The statistics and configuration of synthetic data are illustrated in Table 2, where the default settings are marked in bold font.

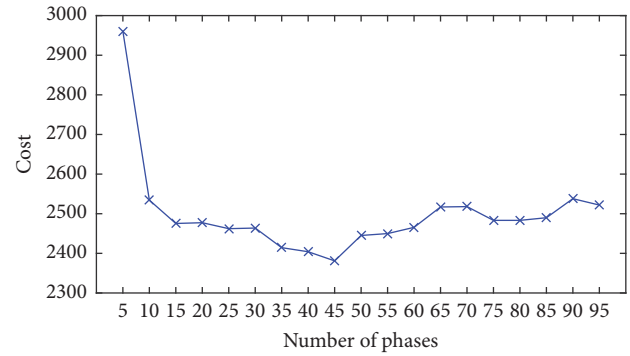


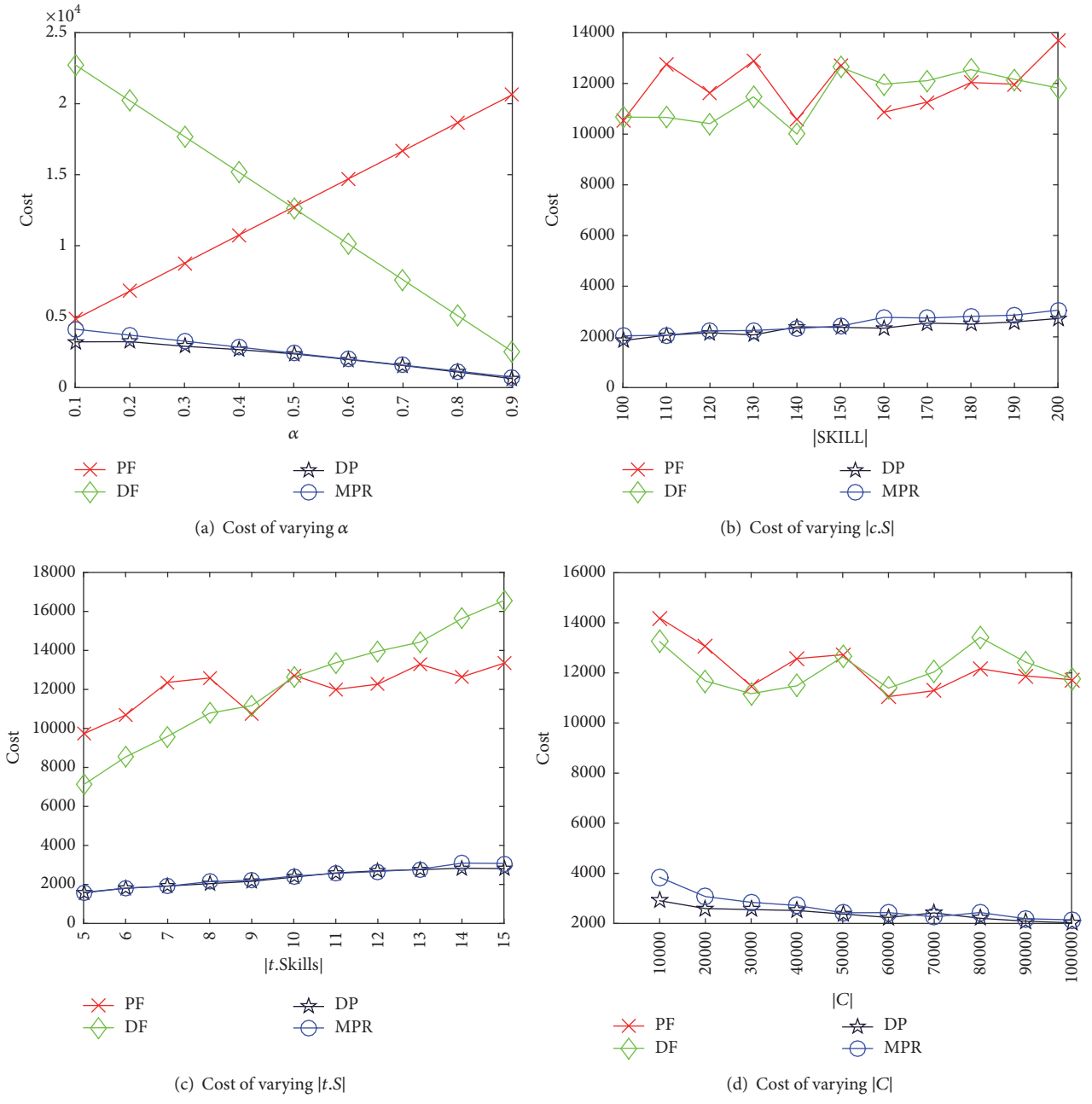
Figure 2: Cost of varying n_p.

5.2. Number of Phases in MPR Algorithm. In the MPR algorithm, we introduce a new parameter representing the total number of phases, n_p. Before conducting experiments on the synthetic and real data, we determined an appropriate value of n_p to ensure better performance of the MPR algorithm. We first generate a synthetic dataset with the default settings to preexamine how n_p affects the performance of the MPR algorithm. The results are shown in Figure 2 for n_p from 5 to 100. According to these results, we use $n_p = 45$ in all subsequent MPR experiments.

5.3. Experiments on Synthetic Datasets. The experimental results using the synthetic data are shown in Figures 3 and 4. In this section, we measure the effectiveness and efficiency of these four algorithms and analyze how various parameters affect the results given by each algorithm.

Effectiveness of Proposed Algorithms. Figure 3 shows the effectiveness of our four algorithms. The DP-SDTF and MPR algorithms offer similar performance and outperform both DF-SDTF and PF-SDTF.

Efficiency of Proposed Algorithms. Figure 4 shows the efficiency of our four algorithms. We can observe that although DP-SDTF and MPR have similar cost results, MPR is faster



(a) Cost of varying α

(b) Cost of varying $|c.S|$

(c) Cost of varying $|t.S|$

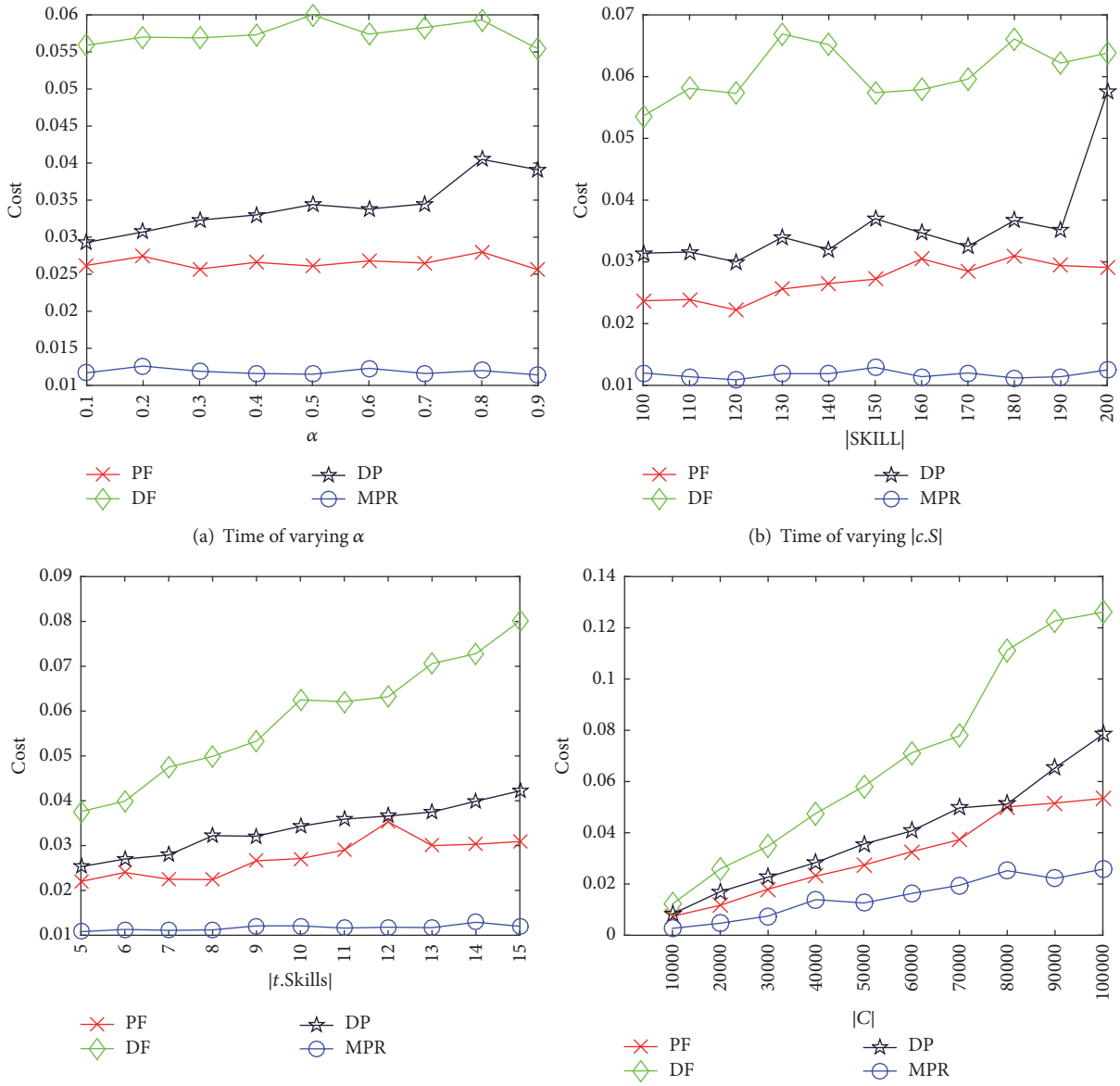
(d) Cost of varying $|C|$

Figure 3: Results on synthetic data.

than DP-SDTF. This is because we use the R-tree to prune some unvalued nodes and accelerate the process of the query. We can also observe how the restriction of skill satisfaction affects the running time of four algorithms. Although PF-SDTF, DF-SDTF, and DP-SDTF all use greedy strategy and their structures are similar, DF-SDTF algorithm consumes more time than that of PF-SDTF and DP-SDTF algorithms. This is because DF-SDTF algorithm only considers the effect of the distance. As a result DF-SDTF needs more coders to make the team feasible, resulting in more iterations than the PF-SDTF and DP-SDTF algorithms.

Effect of α. Figure 3(a) shows the effectiveness of varying α. As α varies from 0.1 to 0.9, the cost of DP-SDTF decreases smoothly, indicating that $\sum_{c \in T} c.P$ contributes more than $\max_{c \in T} |c.L, t.L|$. Because the DF-SDTF (PF-SDTF) algorithm only considers distance (price), when α is high (low), the performance is similar to that of the DP-SDTF and MPR algorithms. However, as α decreases (increases), the performance of DF-SDTF (PF-SDTF) becomes worse.

Effect of $|c.S|$, $|t.S|$, *and* $|C|$. The effect of varying $|c.S|$, $|t.S|$, and $|C|$ is illustrated in Figures 3(b), 3(c), and 3(d). Because



(a) Time of varying α

(b) Time of varying $|c.S|$

(c) Time of varying $|t.S|$

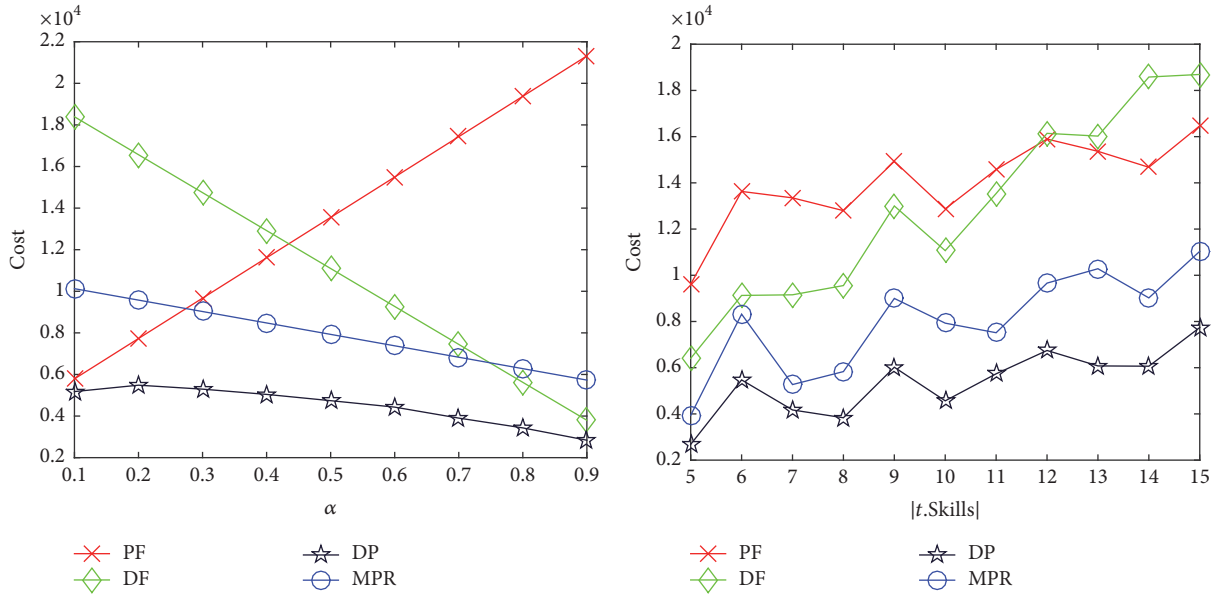
(d) Time of varying $|C|$

Figure 4: Running time on synthetic data.

the default setting of α is 0.5, finding a good team requires distance and price to be considered simultaneously. We can observe that the DP-SDTF and MPR algorithms perform better, with the DF-SDTF and PF-SDTF costing 3 to 4 times more.

5.4. Experiments on the Real Dataset. The experimental results using the real dataset are shown in Figure 5. Figure 5(a) shows the effects of varying α, and Figure 5(b) shows the effects of varying $|t.S|$. Varying α produces a similar effect as with the synthetic dataset. When varying $|t.S|$, the costs of the four algorithms oscillates, probably because of the structure of the CSTO dataset. Unlike the experiments on synthetic data, the MPR algorithm performs worse than DP-SDTF but still outperforms DF-SDTF and PF-SDTF. This is probably because, in real datasets, different skills may make different contributions, leading to a gap between results with synthetic data and real data.

Comparison with the Exact Result. Because the SDTF problem is NP-hard, we only conduct small-size experiments to compare the output of our DP-SDTF and MPR algorithms with the exact solution. The setting is $t.S = 5$ and $|C| = 300$, where coders are randomly chosen from the real dataset. The experimental results are shown in Figure 6. We can observe that the performance of DP-SDTF is similar to that of the exact algorithm, but the cost of the MPR algorithm is 1.25 to 1.5 times the exact minimum cost.



(a) Cost of varying α

(b) Cost of varying $|t.S|$

Figure 5: Results on real data.

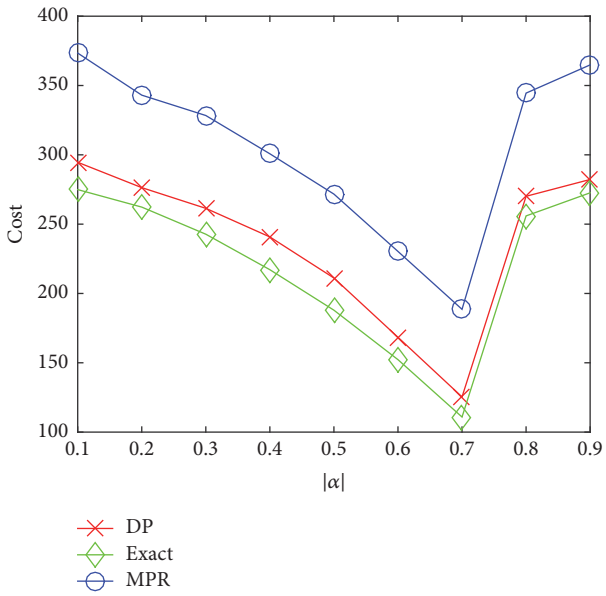


Figure 6: DP-SDTF versus MPR versus exact results.

Conclusion. From the extensive experiments conducted on both real and synthetic data to validate our four algorithms, we found that DF-SDTF (PF-SDTF) algorithm, which focuses on the distance (price) part of the objective function, performs better with larger (smaller) values of α. The DP-SDTF algorithm gives the best performance among the four algorithms discussed here because it considers both parts of the objective function. The fourth algorithm, MPR, accelerates the query process with little increase in the cost, which is more applicable in practice.

6. Related Work

The SDTF problem tackled in this paper covers the domains of *Team Formation* and *Spatial Crowdsourcing*. On the one hand, the SDTF problem can be simplified to the task assignment problem if we ignore the skill constraint. On the other hand, it is exactly the most distinctive requirement that the skills of a team must cover the skills of the task. Previous work related to these two domains is introduced in the following subsections.

6.1. Team Formation. The team formation problem was first proposed in [19]. The problem requires a team of workers that (1) its skills satisfy the requirement of the task; (2) the overall communication cost is minimum. In this paper, the NP-hard nature of this problem is also proved. The problem has been extended by associating each worker with a capacity [20], which is the maximum number of tasks assigned to the worker. To solve the capacitated team formation problem, two approximation algorithms with proved guarantees were proposed. Different from [19, 20], which only include a single task, the team formation problem has been considered with multiple tasks and workers in both offline and online scenarios [21]. While the above-mentioned studies attempt to optimize the overall communication cost, the workload can be balanced among workers by treating the communication cost as a restrictive constraint [22]. As the above shows, most studies on team formation focus on skills satisfaction in communicative graphs, while ignoring the influence of spatial information.

6.2. Spatial Crowdsourcing. The problem studied in this paper is an extension of the task assignment problem in spatial crowdsourcing, known as the server-assigned task assignment problem [10, 11], in which workers cannot reject

the assigned tasks. Recently, task assignment in real-time spatial crowdsourcing has also been studied by the online algorithmic model [12, 23]. Based on the original task assignment problem, both [24, 25] study the conflict-aware task assignment problem, in which tasks may conflict with each other and thus cannot be assigned to the same worker. In addition, the work [26] not only considers spatiotemporal conflicts of tasks but also schedules the plan that each worker complete tasks [26]. Furthermore, Kazemi et al. propose the quality-based task assignment problem [27], which utilizes majority voting techniques to guarantee the quality of task assignment results [28–30].

Although [13, 14] integrate the task assignment problem and team formation problem and propose a two-level-based framework to solve the problem, there are two main differences between [13, 14] and our work: (1) there is no capacity constraint in our work, which means that there are more candidates in the search space; (2) the objective of our work considers both the distance between the task and workers and the overall cost, whereas [13, 14] only attempt to minimize the overall cost.

7. Conclusion

With the development of sensors, particularly vehicle sensors and mobile sensors, spatial crowdsourcing is gaining ever more attention. In this paper, we propose a novel spatial crowdsourcing problem called Software Development Team Formation (SDTF). We prove that SDTF is NP-hard and design three greedy algorithms and an index-based algorithm to solve the SDTF problem. The first two greedy algorithms, DF-SDTF and PF-SDTF, only consider part of the optimization objective, and the performance is therefore below expectations. To overcome the shortcomings of these two algorithms, we design a third greedy algorithm, called DP-SDTF, which considers both parts of the optimization goal. In addition, we develop a multiple-phase algorithm based on R-trees called MPR. The MPR algorithm can accelerate the query process with little increase in cost. We conduct extensive experiments to evaluate the performance of our algorithms. The results show that our DP-SDTF algorithm achieves similar performance to the exact algorithm.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work is supported in part by National Grand Fundamental Research 973 Program of China under Grant 2015CB358700, NSFC Grant no. 71531001, and SKLSDE Open Program SKLSDE-2016ZX-13.

References

[1] G. Liang, J. Ca, X. Liu, and J. Liang, "Smart world: a better world," *Science China Information Sciences*, vol. 59, no. 4, Article ID 043401, 2016.

[2] B. Zhu, L. Xie, D. Han, X. Meng, and R. Teo, "A survey on recent progress in control of swarm systems," *Science China Information Sciences*, vol. 60, no. 7, 070201, 24 pages, 2017.

[3] J. Hu, B. B. Park, and Y.-J. Lee, "Transit signal priority accommodating conflicting requests under Connected Vehicles technology," *Transportation Research Part C: Emerging Technologies*, vol. 69, pp. 173–192, 2016.

[4] J. Hu, M. D. Fontaine, and J. Ma, "Quality of private sector travel-time data on arterials," *Journal of Transportation Engineering*, vol. 142, no. 4, Article ID 04016010, 2016.

[5] C. Ding, D. Wang, X. Ma, and H. Li, "Predicting short-term subway ridership and prioritizing its influential factors using gradient boosting decision trees," *Sustainability*, vol. 8, no. 11, article no. 1100, 2016.

[6] H. Jiang, J. Hu, S. An, M. Wang, and B. B. Park, "Eco approaching at an isolated signalized intersection under partially connected and automated vehicles environment," *Transportation Research Part C: Emerging Technologies*, vol. 79, pp. 290–307, 2017.

[7] C. Ding, X. Wu, G. Yu, and Y. Wang, "A gradient boosting logit model to investigate driver's stop-or-run behavior at signalized intersections using high-resolution traffic data," *Transportation Research Part C: Emerging Technologies*, vol. 72, pp. 225–238, 2016.

[8] C. Ding, C. Liu, Y. Zhang, J. W. Yang, and Y. P. Wang, "Investigating the impacts of built environment on vehicle miles traveled and energy consumption: Differences between commuting and non-commuting trips," *Cities*, vol. 68, pp. 25–36, 2017.

[9] Y. Tong, Y. Chen, Z. Zhou et al., "The simpler the better: a unified approach to predicting original taxi demands based on large-scale online platforms," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1653–1662, 2017.

[10] L. Kazemi and C. Shahabi, "GeoCrowd: enabling query answering with spatial crowdsourcing," in *Proceedings of the 20th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (GIS '12)*, pp. 189–198, ACM, November 2012.

[11] H. To, C. Shahabi, and L. Kazemi, "A server-assigned spatial crowdsourcing framework," *ACM Transactions on Spatial Algorithms and Systems*, vol. 1, no. 1, p. 2, 2015.

[12] Y. Tong, J. She, B. Ding, L. Wang, and L. Chen, "Online mobile Micro-Task Allocation in spatial crowdsourcing," in *Proceedings of the 32nd IEEE International Conference on Data Engineering, ICDE '16*, pp. 49–60, IEEE, Helsinki, Finland, 2016.

[13] D. Gao, Y. Tong, J. She, T. Song, L. Chen, and K. Xu, "Top-k team recommendation in spatial crowdsourcing," in *Proceedings of the 17th International Conference on Web-Age Information Management, WAIM '16*, vol. 9658 of *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, pp. 191–204, June 2016.

[14] D. Gao, Y. Tong, J. She, T. Song, L. Chen, and K. Xu, "Top-k Team Recommendation and Its Variants in Spatial Crowdsourcing," *Data Science and Engineering*, vol. 2, no. 2, pp. 136–150, 2017.

[15] A. Guttman, "R-trees:a dynamic index structure for spatial searching," in *Proceedings of the the 1984 ACM SIGMOD international conference*, pp. 47–57, June 1984.

[16] Q. Tao, T. Song, and K. Xu, "Finding optimal team for multi-skill task in spatial crowdsourcing," in *Proceedings of the 2th WORKSHOP on Data-Driven Crowdsourcing (DDC 2017) in conjunction with APWeb-WAIM 2017*, 2017.

[17] N. Roussopoulos, S. Kelley, and F. Vincent, "Nearest Neighbor Queries," *ACM SIGMOD Record*, vol. 24, no. 2, pp. 71–79, 1995.

[18] G. R. Hjaltason and H. Samet, "Distance browsing in spatial databases," *ACM Transactions on Database Systems (TODS)*, vol. 24, no. 2, pp. 265–318, 1999.

[19] T. Lappas, K. Liu, and E. Terzi, "Finding a team of experts in social networks," in *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '09*, pp. 467–475, July 2009.

[20] S. Datta, A. Majumder, and K. V. M. Naidu, "Capacitated team formation problem on social networks," in *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2012*, pp. 1005–1013, August 2012.

[21] A. Anagnostopoulos, L. Becchetti, C. Castillo, A. Gionis, and S. Leonardi, "Power in unity: Forming teams in large-scale community systems," in *Proceedings of the 19th International Conference on Information and Knowledge Management and Co-located Workshops, CIKM'10*, pp. 599–608, October 2010.

[22] A. Anagnostopoulos, L. Becchetti, C. Castillo, A. Gionis, and S. Leonardi, "Online team formation in social networks," in *Proceedings of the 21st Annual Conference on World Wide Web, WWW'12*, pp. 839–848, fra, April 2012.

[23] Y. Tong, J. She, B. Ding, L. Chen, T. Wo, and K. Xu, "Online minimum matching in realtime spatial data: Experiments and analysis," in *Proceedings of the 42nd International Conference on Very Large Data Bases, VLDB 2016*, pp. 1053–1064, ind, September 2016.

[24] J. She, Y. Tong, L. Chen, and C. C. Cao, "Conflict-aware event-participant arrangement and its variant for online setting," *IEEE Transactions on Knowledge and Data Engineering*, vol. 28, no. 9, pp. 2281–2295, 2016.

[25] Y. Tong, J. She, and R. Meng, "Bottleneck-aware arrangement over event-based social networks: the max-min approach," *World Wide Web*, vol. 19, no. 6, pp. 1151–1177, 2016.

[26] J. She, Y. Tong, and L. Chen, "Utility-aware social event-participant planning," in *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2015*, pp. 1629–1643, June 2015.

[27] L. Kazemi, C. Shahabi, and L. Chen, "GeoTruCrowd: trustworthy query answering with spatial crowdsourcing," in *Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, ACM SIGSPATIAL GIS 2013*, pp. 304–313, November 2013.

[28] C. C. Cao, Y. Tong, L. Chen, and H. V. Jagadish, "Wisemarket: a new paradigm for managing wisdom of online social users," in *Proceedings of the the 19th ACM SIGKDD international conference*, pp. 455–463, 2013.

[29] Y. Tong, L. Chen, and Y. Cheng, "Mining frequent itemsets over uncertain databases," *Proceedings of the VLDB Endowment*, vol. 5, no. 11, pp. 1650–1661, 2012.

[30] Y. Tong, L. Chen, and B. Ding, "Discovering threshold-based frequent closed itemsets over probabilistic data," in *Proceedings of the IEEE 28th International Conference on Data Engineering, ICDE 2012*, pp. 270–281, April 2012.