Three-Dimensional Stable Matching Problem for Spatial Crowdsourcing Platforms

Boyang Li
Northeastern University
Shenyang, China
liboyang@stumail.neu.edu.cn

Yurong Cheng
Beijing Institute of Technology
Beijing, China
yrcheng@bit.edu.cn

Ye Yuan
Northeastern University
Shenyang, China
yuanye@mail.neu.edu.cn

Guoren Wang*
Beijing Institute of Technology
Beijing, China
wanggrbit@126.com

Lei Chen
The Hong Kong University of Science and Technology
Hong Kong SAR, China
leichen@cse.ust.hk

ABSTRACT

The popularity of mobile Internet techniques and Online-To-Offline (O2O) business models has led to the emergence of various spatial crowdsourcing (SC) platforms in our daily life. A core issue of SC platforms is to assign tasks to suitable crowd workers. Existing approaches usually focus on the matching of two types of objects, tasks and workers, and let workers to travel to the location of users to provide services, which is a 2D matching problem. However, recent services provided by some new platforms, such as personalized haircut service[1] and station ride-sharing[2], need users and workers travel together to a third workplace to complete the service, which is indeed a 3D matching problem. Approaches in the existing studies either cannot solve such 3D matching problem, or lack a assignment plan satisfying both users' and workers' preference in real applications. Thus, in this paper, we propose a *3-Dimensional Stable Spatial Matching* (3D-SSM) for the 3D matching problem in new SC services. We prove that the 3D-SSM problem is NP-hard, and propose two baseline algorithms and two efficient approximate algorithms with bounded approximate ratios to solve it. Finally, we conduct extensive experiment studies which verify the efficiency and effectiveness of the proposed algorithms on real and synthetic datasets.

CCS CONCEPTS

• **Information systems → Spatial-temporal systems**.

KEYWORDS

Spatial database, Crowdsourcing, Stable Matching

*This author is the corresponding author
[1]http://www.nanguache.com/.
[2]https://www.didiglobal.com/.

ACM Reference Format:
Boyang Li, Yurong Cheng, Ye Yuan, Guoren Wang, and Lei Chen. 2019. Three-Dimensional Stable Matching Problem for Spatial Crowdsourcing Platforms. In *The 25th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '19), August 4–8, 2019, Anchorage, AK, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3292500.3330879

1 INTRODUCTION

With the development of mobile Internet techniques and Online-To-Offline (O2O) business models, various spatial crowdsourcing (SC) platforms show their popularity in people's daily life where the workers are employed through the Internet and finish the tasks in the real world [21]. Famous SC platforms contains taxi travel service platform (e.g. DiDi[2]), sports venues and trainers service (e.g. InterestingSport[3]), customers and makeup artists service (Nanguache[1]), and event-participant service (e.g. Meetup[4]), etc. For these SC platforms, one of the typical tasks is to assign tasks to suitable workers. Existing studies usually evaluate the interests of the workers with respect to the tasks by calculating the utility score based on their historical records and the profile information over the platform. They model this problem as a maximum unweighted/weighted bipartite graph matching (shorten as 2D matching) problem [5][6][16][17][28]. The goal of these studies is to maximize the total numbers or the sum of utility score. In this model, the workers travel to the users' locations to provide services. In other words, the service place is the same with the location of users. However, some new O2O applications provide such services that require users and workers go to a third place together. In follows, we provide 3 real-world application examples for the above SC service.

Personalized haircut service. On the Nanguache[1] platform, each user would like to require a hairdresser (i.e., worker) and book a barber shop (i.e., workplace) to have a haircut. The user (e.g. u_2) and hairdresser (e.g. w_2) can depart from their living places, and go to a barber shop (e.g. p_1) together to make the haircut.

Station ride-sharing. The DiDi[2] platform provides a station ride-sharing service, in which several users go to a station (i.e., workplace) on foot, and a car/taxi (i.e., worker) drive to that station to pick these users.

[3]http://www.quyundong.com/.
[4]https://www.meetup.com/.

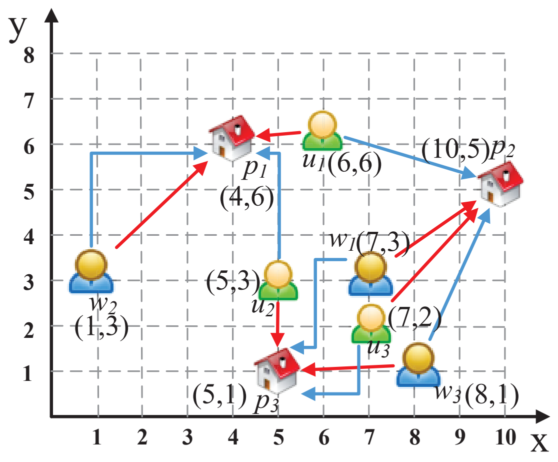


Figure 1: An Example of Crowdsourcing Platform

Sport Class. The InterestingSport[3] platform provide a sport class service. A tennis coach (i.e., worker) can teach one or several students (i.e. user) in one class. The coaches and students will go to a tennis court (i.e. workplace) to complete the class.

In all the above three applications, the platform needs to match all users, workers and workplaces, and provides a global plan. Essentially, this involves in a tripartite graph matching (shorten as 3D matching) problem. The methods [5][6][16][17][28] for old services solving the 2D matching problem cannot be simply applied to such 3D matching problem. Moreover, Song et al in [25] propose a 3D matching strategy, in which the platform makes plan by maximizing the total utility score of users and workers. This strategy just maximized the satisfaction of the platform, but failed to comprehensively consider the request of both users and workers. For example, in Fig. 1, there exist three types of points in the platform, users (e.g. u_1, u_2, u_3), workers (e.g. w_1, w_2, w_3), and workplaces (e.g. p_1, p_2, p_3), and we treat the reciprocal value of distance as the utility. Using the strategy in [25], we can obtain the global plan shown as the red lines. In this plan, u_3 and w_1 are matched together, and assigned to place p_2. However, there obviously exists another place p_3 which is nearer to both u_3 and w_1. Then neither u_3 nor w_1 will be happy: "Clearly, there is a workplace closer to us. Why the platform let us travel further?!"

We propose a new matching over the SC platforms for such 3D matching services, which considers the distance between workplaces to both users and workers comprehensively. We name this problem as *3-Dimensional Stable Spatial Matching* (3D-SSM). Under our 3D-SSM, the global plan in Fig. 1 is shown as the blue lines, which can overcome the problems of existing strategy.

In summary, we make the following contributions.

- We formulate a new stable matching problem called the *3-Dimensional Stable Spatial Matching* (3D-SSM). We prove that the 3D-SSM problem is NP-hard and propose two baseline algorithms.
- We develop a DSM algorithm and a Path Swap algorithm with bounded approximate ratios to efficiently solve the 3D-SSM problem approximately.
- We verify the effectiveness and efficiency of the proposed methods with extensive experiments on real and synthetic datasets.

The rest of the paper is organized as follows. Section 2 formally defines the 3D-SSM problem. Section 3 presents two baseline algorithms. We propose our approximate algorithms and analyze the complexity and approximation ratio in Section 4 and 5. Section 6 evaluates the proposed techniques by extensive experiments over both real and synthetic datasets. We review the related works in Section 7 and conclude the paper in Section 8.

2 PROBLEM STATEMENT

In this section, we first introduce some basic concepts, and then formally define the stable triple and 3D-SSM problem. Table 1 shows the symbols used in this paper and their corresponding descriptions.

Table 1: Summary of Symbols used in Our Paper

Symbols	Description
M	The matching set
W	The worker set
U	The user set
P	The workplace set
$\mathbf{l}_w, \mathbf{l}_u, \mathbf{l}_p$	The location of workers, users and workplaces
c_w	the capacity of workers
DW	The ordered distance set of workers to workplaces
DU	The ordered distance set of users to workplaces
DLW	The ordered distance set of workplaces to workers
DLU	The ordered distance set of workplaces to users
$\mathcal{P}$	The swap path

Given m users, n workers and t workplaces, we denote U as the user set, W as the worker set and P as the workplace set. Each user $u \in U$ or workplace $p \in P$ is described by a 2D space location $\mathbf{l}_u$ or $\mathbf{l}_p$. Each worker is described by a triple $w < \mathbf{l}_w, c_w >$. Here, $\mathbf{l}_w$ is the 2D space location; c_w is the number of users that he can matched, which is called the capacity of w. To be convenient, for each worker $w \in W$ (resp. user $u \in U$), we use a set DW_w (resp. DW_u) to store all workplaces $p \in P$ in a decreased order of the distance between p to w (resp. u). Meanwhile, for each place p, we use a set DLW_p (resp. DLU_p) to store all worker $w \in W$ (resp. users $u \in U$) in a decreased order of the distance between w (resp. u) to p. Without loss of generality, the distance can be extented to different kinds of distance, such as the road-network distance. We simply use the Euclidean distance in our paper.

Notice that one user can be matched by only one worker in one workplace. A worker can match several users no more than the capacity, but can match them at one workplace. A workplace can only accommodate one worker and users that the worker serves.

A match (w, u, p) is a triple which means that worker w and user u go to workplace p to complete the service. As stated in the Introduction previously, if there exists another workplace p' that is closer to both w and u than p, w and u would feel unsatisfiable to the plan provided by the platform. Thus, we define a new concept of stable triple as follows.

Definition 1 (Stable Triple). For a worker w, a user u and a workplace p, (w, u, p) is a stable triple if and only if there is no other *locations $p' \in P$ satisfies w is closer to p' than p and u is closer to p' than p, no matter whether p' is matched with other workers and users or not.*

Example 1. Tables 2(a), 2(b) and 2(c) show the information of workers, users, and workplaces respectively. A match (w_3, u_7, p_5) is an unstable triple because p_4 is closer to both w_3 and u_7 than p_5. It means that p_4 is a better choice of w_3 and u_7 than p_5. All matches shown in Table 2 (d) are stable triples.

Based on the stable triple concept, we define 3D-SSM as follows.

Definition 2 (3D-SSM Problem). Given a set of workers W, a set of users U and a set of workplaces P, the 3D-SSM problem is to find a matching set M among the workers, users and workplaces such that the number of matches in M is maximized under the following constraints:

- ***Stable Constraint:*** *Each matched triple $(w, u, p) \in M$ should be stable.*
- ***Capacity Constraint:*** *Each worker should be matched with users no more than the capacity.*
- ***Unique Constraint:*** *Each user should be matched in one triple only, each worker should be matched with one workplace only, and each workplace should be matched with one worker only.*

Example 2. For the 3 workers, 3 users, and 3 workplaces shown in Fig. 1, the capacity of the 3 workers are $c_{w_1} = c_{w_2} = c_{w_3} = 1$. A match set $M = \{(w_3, p_2, u_1), (w_2, p_2, u_2), (w_1, p_3, u_3)\}$ satisfies all the 3 constraints in Definition 2, and it contains 3 matches. Another match set $M' = \{(w_1, p_1, u_1), (w_1, p_3, u_2)\}$ can also satisfy all the 3 constraints in Definition 2. However, M' only contains 2 matches, since any other matches adding to M' would break at least one of the 3 constraints. M' cannot be the result, since it contains smaller number of matches than M.

Theorem 1. The 3D-SSM problem is NP-hard.

3 BASELINE ALGORITHM

In this section, we propose two baseline algorithms to solve 3D-SSM, Maximum Weight Independent Set (MWIS) algorithm and Greedy Stable Matching (GSM) algorithm.

3.1 The MWIS-based Algorithm

From the proof of Theorem 1, we can find that the special case of 3D-SSM can be converted to maximum independent set problem [27]. It stimulates us that we can solve 3D-SSM by constructing a graph and conducting the Maximum Weight Independent Set (MWIS) [14] [15].

The main idea is as follows. Firstly, we construct vertices using stable triples in 3D-SSM. When the capacity of each worker $c_w = 1$, a graph GM can be equivalently constructed for 3D-SSM. Each stable pair is a vertex of GM. When $c_{w_i} > 1$, the vertex will become a super vertex containing multiple stable triples. In the vertexes, worker and workplace are the same, but users are different. The number of stable triples in a vertex becomes its weight. Secondly, we link edges between vertices who contain the same workers, users, or workplaces. Finally, we conduct the algorithms solving

Table 2: Statistics for workers, users and workplaces

(a) Statistics for workers

W	DW	**capacity**
w_1	p_2, p_3, p_5, p_1, p_4	3
w_2	p_2, p_3, p_4, p_1, p_5	3
w_3	p_2, p_1, p_4, p_5, p_3	2
w_4	p_4, p_3, p_1, p_5, p_2	3
w_5	p_2, p_5, p_1, p_3, p_4	1

(b) Statistics for users

U	DU	U	DU
u_1	p_4, p_3, p_5, p_2, p_1	u_7	p_3, p_4, p_1, p_5, p_2
u_2	p_3, p_2, p_1, p_4, p_5	u_8	p_5, p_2, p_4, p_3, p_1
u_3	p_3, p_1, p_4, p_2, p_5	u_9	p_3, p_1, p_4, p_5, p_2
u_4	p_2, p_1, p_5, p_4, p_3	u_{10}	p_1, p_5, p_3, p_4, p_2
u_5	p_1, p_2, p_3, p_5, p_4	u_{11}	p_5, p_3, p_2, p_4, p_1
u_6	p_2, p_4, p_1, p_5, p_3	u_{12}	p_1, p_2, p_4, p_5, p_3

(c) Statistics for workplaces

P	DLW	DLU
p_1	w_3, w_5, w_2, w_1, w_4	$u_7, u_9, u_3, u_5, u_{12}, u_4, u_{10}, u_6, u_2, u_8, u_1, u_{11}$
p_2	w_2, w_5, w_3, w_1, w_4	$u_4, u_6, u_5, u_{12}, u_7, u_2, u_3, u_8, u_{11}, u_1, u_{10}, u_9$
p_3	w_2, w_1, w_4, w_5, w_3	$u_7, u_3, u_2, u_9, u_1, u_5, u_{11}, u_{10}, u_8, u_4, u_6, u_{12}$
p_4	w_4, w_3, w_2, w_5, w_1	$u_7, u_1, u_3, u_9, u_6, u_{12}, u_2, u_8, u_{10}, u_4, u_5, u_{11}$
p_5	w_5, w_1, w_3, w_2, w_4	$u_8, u_7, u_5, u_{10}, u_{11}, u_1, u_9, u_6, u_4, u_3, u_{12}, u_2$

(d) A stable matching

(w_1, u_7, p_3)	(w_2, u_5, p_2)	(w_4, u_1, p_4)
(w_1, u_9, p_3)	(w_2, u_6, p_2)	(w_4, u_2, p_4)
(w_1, u_{10}, p_3)	(w_3, u_{11}, p_5)	(w_4, u_3, p_4)
(w_2, u_4, p_2)	(w_3, u_8, p_5)	(w_5, u_{12}, p_1)

the Maximum Weight Independent Set (MWIS) problem to calculate the result [14] and [15].

When $c_{w_i} > 1$, all possible numbers of users that may be allocated to w_i should be enumerate. Suppose $c_{w_1} = 2$, w_1 is matched with workplace p_1, and 2 users can compose stable triples with w_1 and p_1. Specifically, $(w_1, u_1, p_1) \dots (w_1, u_2, p_1)$ are all stable triples. Then 3 vertices are constructed, which are $vm_1((w_1, u_1, p_1))$, $vm_2((w_1, u_2, p_1))$, and $vm_3((w_1, u_1, p_1), (w_1, u_2, p_1))$. vm_3 is a super vertex that contain 2 matches. We denote the capability of worker w_i as c_{w_i}, the number of users matched with w_i as x. If $x \geq c_{w_i}$ users can compose stable triples with w_i and workplace p_j, $\binom{x}{1} + \dots \binom{x}{c_{w_i}}$ vertices are constructed. If $x < c_{w_i}$, then $\binom{x}{1} + \dots \binom{x}{x}$ vertices are constructed. Given a vertex $vm_3((w_1, u_1, p_1), (w_1, u_2, p_1))$, all other vertices that contain any of w_1, u_1, u_2, or p_1 should be linked to vm_3.

The pseudocode of the baseline algorithm is shown in Algorithm 1. We initialize an empty graph GM as a pair of empty vertex set VM and edge set EM (Line 1). Then, for each worker $w \in W$ and each workplace $p \in P$, we initialize an empty set S. We enumerate all possible stable triples and put them into S (Lines 2-7). We state a counter i from 1 to $\min(c_w, |S|)$, and $\binom{|S|}{i}$ vertices with weight i would be constructed (Line 9). After constructing all vertices (Lines 2-9), we link the edges between vertices whose w or u or p are the same (Lines 10-12). Finally, we conduct algorithms for MWIS problem to solve the problem (Lines 13-14).

Theorem 2. The accuracy of the baseline algorithm is the same with that of the MWIS problem.

[14] and [15] can calculate exact result of the MWIS problem. Though the complexity is $O(2^{|V|})$, efficient pruning algorithms can help to fast calculate the exact results. Thus, the baseline algorithm can be treated to provide an exact answer.

3.2 GSM Algorithm

As discussed above, the MWIS-based algorithm can provide exact answers, but its computational complexity is too high. Thus, we consider another approximate algorithm called Greedy Stable Matching (GSM), which is more efficient. We propose an approximate algorithm, Greedy Stable Matching (GSM) algorithm. It directly finds the stable matching for users one by one according to the distance.

The main idea of GSM is to optimize the workplace and worker for a given user. Given a user u, we greedily check the nearest workplace p and the nearest worker w to p, until (w, u, p) is stable. Recursively repeat this method until no stable matches can be found.

The details of GSM are illustrated in Algorithm 2. We initialize an unmatched worker (resp. workplace) set AW (resp. AP) (Line 1). For each $u \in U$, it enumerates workplaces ordered by the distances from the smallest to largest. If $p \in AP$, it first enumerates workers in AW to find a stable triple, then deletes the worker (resp. workplace) from AW (resp. AP) and puts the triple into matching results (Lines 5-9). If $p \notin AP$, there is a worker w matched with p (Line 11). If (w, u, p) is stable and the capacity of w is satisfied, the triple is a matching result of u (Lines 10-14). If a stable triple is found for the user, the algorithm starts to process the next user (Lines 15-16).

3.3 Shortcomings of the Baseline Algorithms

Though the MWIS algorithm can provide an exact solution, the calculation is very inefficient. The computational complexity of constructing the graph is $O((|U| \times |W| \times |P|)^2 + 2^{c_w} \times (|W| \times |P|)$, where c_w is the maximal capacity of all workers. Moreover, the number of vertices and that of edges in the converted graph can be $O(2^{\min(c_w, x)} \times (|W| \times |P|)$ and $O(2^{2c_w} \times (|P|^2|W| + |W|^2|P|))$ respectively, which is much larger and denser than the normal simple graph. So it is hard for algorithms in [14] [15] to run efficiently.

Comparing with MWIS, the GSM algorithm is more efficient. The computational complexity of GSM is $O(|W| \times |U| \times |P|)$. However, the number of stable matches found by the GSM algorithm could be rather fewer than the exact solution. This is because in GSM, the matching result cannot be changed once a worker is matched with a workplace, even though he may match more users in other workplaces. The capacity of workers may be wasted and leads to a bad effectiveness.

According to the above analysis, we need to find algorithms that can solve 3D-SSM efficiently and accurately. Thus, in the next two sections, we propose two approximate algorithms with bounded approximate ratios to improve the 3D-SSM processing.

4 DYNAMIC STABLE MATCHING ALGORITHM

As discussed above, we propose an improved greedy algorithm, named Dynamic Stable Matching algorithm. It aims to overcome the shortcoming of GSM.

The main idea of DSM is to maximize the number of stable triples for a given user. Given user u, we find a stable triple from workers and workplaces that are not matched. Different from the GSM algorithm, if the triple does not exist, we consider to break some existing triples if more users can be matched in the new matching results.

The details of DSM are illustrated in Algorithm 3. We initialize an unmatched sets AW, AU and AP, and initialize a termination flag (Line 1). The algorithm executes until $Flag = 0$, which means it cannot find any other new stable triples (Lines 2-18). For each user $u \in AU$ (Line 4), we consider three different cases. In the first case, if we find a stable triple (w, u, p) where $w \in AW$ and $p \in AP$, we update the matching results, and set $Flag$ equals to 1 (Lines 5-7). Next, we find whether other users can be matched with w and p (Lines 8-10). If u is not matched in the first case (Line 11), the algorithm tries to find a stable triple where $w \in AW$ and $p \notin AP$. If w can match more users with p than the current worker matched with p, the current triples with p are deleted from M and new stable triples are put into M (Lines 12-14). If u is still not matched (Line 15), the last case is to find a stable triple where $w \notin AW$ and $p \in AP$. The matching result M will be updated if w can match more users with p than current workplace where he matches (Lines 16-18).

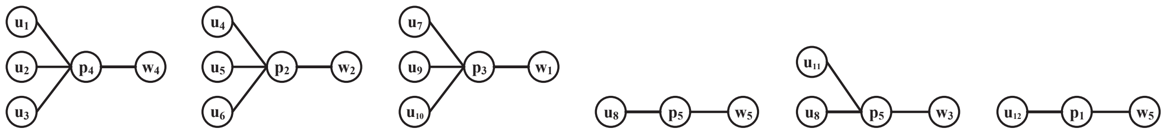
Example 3. Back to the running example in Table 2. At the beginning, we initialize AW, AU and AP. For $u_1 \in AU$, we find p_4 and w_4, where p_4 is the closest workplace to u_1 and w_4 is closest worker to p_4, and (w_4, u_1, p_4) is a stable triple and put into M. The capacity of w_4 is 3, so u_2 and u_3 can be matched. The matching results are shown in Fig. 2(a). Fig. 2(b)-2(d) show the matching results of u_4, u_7, u_8. For $u_{11} \in AU$, the only elements in AW and AP are w_3 and p_1, but (w_3, u_{11}, p_1) is not stable. For $p_5 \notin AP$, (w_3, u_{11}, p_5) is stable and (w_3, u_8, p_5) is also stable. One more user will be matched if w_3 is matched with p_5. Therefore, (w_5, u_8, p_5) is deleted, (w_3, u_{11}, p_5) and (w_3, u_8, p_5) are put into M. The result is shown in Fig. 2(e). Finally, (w_5, u_{12}, p_1) is matched in Fig. 2(f), and DSM terminates as AU is empty.

Approximate Ratio. Next, we analyze the approximate ratio of DSM.

Theorem 3. Suppose η is the percentage of users whose closest workplaces are different, the approximate ratio of DSM is η.

Complexity Analysis. For a worker set W, a user set U and a workplace set P, the time complexity of DSM is $O(\frac{|U|^2 \times |W| \times |P|}{c_w})$, where c_w is the maximal capacity of all workers. For each user $u \in AU$, the complexity of finding a stable triple is $O(|W| \times |P|)$. When a stable triple is found, it costs $O(|U|)$ time to verify whether more users can be matched and the size of AU decrease at most c_w in each loop. The space complexity is $O((|W|+|U|) \times |P|+|U|+|W|+|P|)$. The space complexity of DSM to store the distance set of workers/users and workplaces is $O(|W| \times |P| + |U| \times |P|)$, the space complexity of AW, AU and AW is $O(|W| + |U| + |P|)$.

We find that during the process of finding the stable triples, many intermediate results are repetitively scanned. Thus, we consider to design a structure that can effectively store these intermediate results, such that the algorithm can be more efficient. We illustrate the details of such algorithm in the next section.



(a) u_1 starts (b) u_4 starts (c) u_7 starts (d) u_8 starts (e) u_{11} starts (f) u_{12} starts

Figure 2: Example of DSM Algorithm

5 PATH SWAP ALGORITHM

In this section, we propose Path Swap algorithm. We build a data structure storing the stable triples that are either already in the result set M or has possibility to be put in M. We call this data structure "Swap Path". The calculation of 3D-SSM can be accelerated by constructing and updating the Swap Paths.

5.1 Swap Path

Before introducing the Path Swap algorithm, we introduce some concepts and lemmas related to the algorithm.

Definition 3 (Unit). If a worker w and workplace p are matched in the result set M, then (w, p) is a ***unit****.*

Definition 4 (Strong Candidate). For a unit (w, p) in M and a user u, if $(w, u, p) \notin M$ and (w, u, p) is stable, u is said to be a ***strong candidate*** *of (w, p) iff*

- u is not in any triple in M; **OR**
- if $(w', u, p') \in M$, (w', p') has at least one strong candidate.

Definition 5 (Weak Candidate). For a unit (w, p) in M and a user u, if $(w, u, p) \notin M$ and (w, u, p) is stable, u is said to be a ***weak candidate*** *of (w, p) iff*

- (w', p') does not have any candidate; **OR**
- if $(w', u, p') \in M$, (w', p') has weak candidates only.

Definition 6 (Fixed Unit). Given a unit (w, p) matched in M, if p is the closest workplace of w, in other words, p is the first element in DW_w, then (w, p) is a ***fixed unit****.*

Example 4. Take Fig. 3(a) as an example, there are two fixed units (w_1, p_2) and (w_4, p_4). $\{u_1, u_2, u_3\}$ are users matched with (w_4, p_4). The strong candidates of both (w_1, p_2) and (w_4, p_4) are $\{u_7, u_8, u_9, u_{10}, u_{11}, u_{12}\}$, these users are not matched in any triples. They have no weak candidates.

With the above basic concepts, we define our "Swap Path" structure.

Definition 7 (Swap Path). Given a matching result M, a worker u_1 and a workplace p_1 that are not matched. We define a swap path *for (u_1, p_1), denoted as $\mathcal{P} = \{(w_1, u_1, p_1), (w_2, u_2, p_2), \ldots, (w_{|\mathcal{P}|}, u_{|\mathcal{P}|}, p_{|\mathcal{P}|})\}$. A valid* swap path *follows the following rules.*

- *Each triple $\mathcal{P}_i$ is stable.*
- *For the fist triple, w_1 and p_1 are not matched in M, u_1 is matched in M or u_1 is available if $|\mathcal{P}| = 1$.*
- *Each u_i in $\mathcal{P}_i$ is a strong/weak candidate of (w_i, p_i) and is matched with the (w_{i+1}, p_{i+1}), where $i \in [1, |\mathcal{P}|)$.*
- *If $|\mathcal{P}| > 1$,$u_{|\mathcal{P}|}$ is not matched or $(w_{|\mathcal{P}|}, p_{|\mathcal{P}|})$ has neithor strong nor weak candidates.*

We can find swap paths through enumerating candidates, and consider two kinds of swap paths in this paper: (1) all the users in paths are strong candidates; (2) all the users in paths are weak candidates. The former kind can reduce the size of available users to improve the performance, the latter is aimed to adjust the current matching set, which will release the matched workers and workplaces.

After introducing the swap path, we are ready to describe how to update the swap paths. For each triple (w_i, u_i, p_i) in $\mathcal{P}$, where $i \in [1, |\mathcal{P}|)]$, u_i is matched with (w_{i+1}, p_{i+1}) in M, we break the triple (w_{i+1}, u_i, p_{i+1}) and put the new triple $(w_i, u_i.p_i)$ into M. If $u_{|\mathcal{P}|}$ is matched with a unit (w', p') and there are no users matched with it after $(w', u_{|\mathcal{P}|}, p')$ is broken, we delete the unit from M. If $u_{|\mathcal{P}|}$ is not matched with any units, we just put $(w_{|\mathcal{P}|}, u_{|\mathcal{P}|}, p_{|\mathcal{P}|})$ into M.

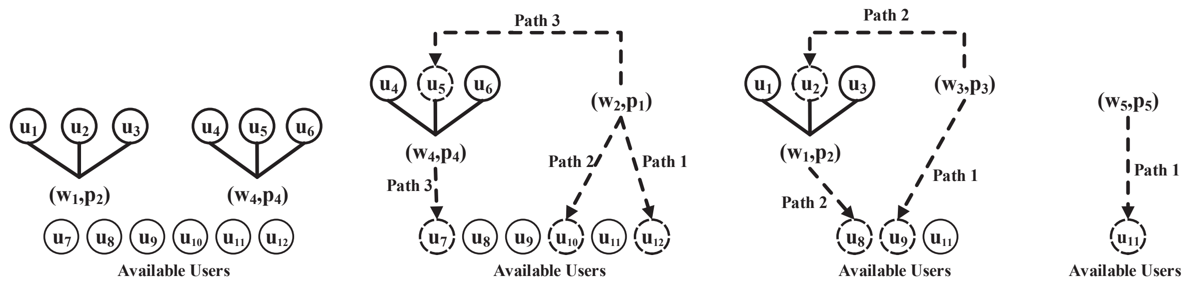
Example 5. Take Fig. 3(b) as an example, we can find three paths for (w_2, p_1), $Path1 = \{(w_2, u_{10}, p_1)\}$, $Path2 = \{(w_2, u_{11}, p_1)\}$ and $Path3 = \{(w_2, u_5, p_1), (w_4, u_7, p_4)\}$. And then, we start to update the matching result based on the paths. For $Path1$, u_{10} is not matched and we can put (w_2, u_{10}, p_1) into M. $Path2$ is updated in the same way. $Path3$ is different from the formers. We firstly break (w_4, u_5, p_4) and put (w_2, u_5, p_1) into M, and then match u_7 with (w_4, p_4), where u_7 is a strong candidate of (w_4, p_4).

5.2 Algorithm Description

The Path Swap algorithm has four steps.

- **Step 1(Assignment Initialization)** It firstly initializes fixed units and matches users as many as possible.
- **Step 2(Find swap paths)** For an available worker w and an available workplace p, it finds a swap path starting from the (w, p).
- **Step 3(Update the paths)** It re-assigns some matches in M and update the candidates of matched workplaces.
- **Step 4(Iterative Step)** It repeats **Step 2** and **3** until reaching the iteration limit or there is no available users.

The pseudocode is shown in Algorithm 4. At the beginning, we initialize an available worker (resp. user and workplace) set AW (resp. AU and AP) and a termination flag $Flag = 1$ (Line 1). Then, the algorithm initializes the fixed units (Lines 2-7). For each $p \in AP$ and $w \in AW$, it finds the candidates for the unit (w, p) (Line 12). It firstly finds and updates the paths in the strong candidates (Lines 13-15). If it does not find any swap paths, the loop breaks (Lines 16-17). Otherwise, it continues to find and update the paths in the weak candidates (Lines 18-20). After the above steps, some users are matched with (w, p), we delete w and p from AW and AP (Line 21).



(a) Initialize fixed workplaces (b) w_2 and p_1 (c) w_3 and p_3 (d) w_5 and p_5

Figure 3: Example of Path Swap Algorithm

Example 6. Back to our running example in Table 2. At the beginning, two fixed units are initialized shown in Fig. 3(a). And then, shown in Fig. 3(b), for w_2 *and* p_1*, three paths are found. Next, for* w_3 *and* p_3*, two paths are found, shown in Fig. 3(c). Finally, for* w_5 *and* p_5*, one path is found and the algorithm terminates, shown in Fig. 3(d).*

5.3 Algorithm Analysis

Approximate Ratio. Next, we analyze the approximate ratio of Path Swap.

Theorem 4. Suppose σ *is the percentage of workers whose closest workplaces are different, the approximate ratio of Path Swap is* σ*.*

Complexity Analysis. For a worker set W, a user set U and a workplace set P, the maximal capacity of all workers is c_w, the time complexity of Path Swap is $O(c_w \times |U| \times |W| \times |P| \times \min\{|W|, |P|\})$. The time complexity of finding stable triples for workers and workplaces is $O(|U| \times |W| \times |P|)$. The complexity of finding a path and updating from the path is $O(\min\{|W|, |P|\})$, each matched workplace is executed at most once in this process. The path finding and path update are called at most c_w times while finding triples. The space complexity of Path Swap is $O((|W| + |U|) \times |P| + \min\{|W|, |P|\} \times |U| + |W| + |P|)$. The space complexity to store the distance set of workers/users and workplaces is $O(|W| \times |P| + |U| \times |P|)$, the space complexity of AW, AU and AW is $O(|W| + |U| + |P|)$. Moreover, it costs $O(\min\{|W|, |P|\} \times |U|)$ space to index the candidates.

6 EXPERIMENT

6.1 Datasets and Experiment Environment

Datasets. We conduct our algorithms over two real-life datasets, Meetup [20] and DiDi Open Dataset[5]. Meetup is an event-based social network platform. We extract data over different cities and set the capacity of each worker from 1 to 3. DiDi is a popular taxi travel service platform in recent years. We draw some areas of different sizes according to latitude and longitude, and set the upper limit of capacity as 4, according to the actual passenger load situation of the taxies. Table 3 shows the statistics on the parameters over two real datasets.

We also use a synthetic dataset for testing the scalability of our algorithms. We generate the locations of workers, users and workplaces and the capacity of each worker. The statistics of synthetic dataset are illustrated in Table 4.

[5]https://outreach.didichuxing.com/research/opendata/.

Table 3: Statistics on Real Datasets

(a) Meetup Dataset

City	$\|W\|$	$\|U\|$	$\|P\|$	capacity
Beijing	10	12	8	[1,3]
Auckland	160	309	137	[1,3]
Singapore	419	1031	477	[1,3]
Vancouver	812	1408	513	[1,3]

(b) DiDi Dataset

Region	$\|W\|$	$\|U\|$	$\|P\|$	capacity
A	6	13	8	[1,4]
B	987	1537	618	[1,4]
C	2083	5471	2032	[1,4]
D	2861	7662	1955	[1,4]

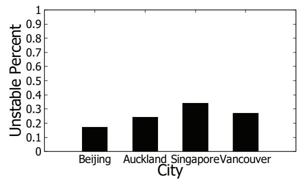
Table 4: Statistics on Synthetic Dataset

Factor	Setting
$\|W\|$	500,1000,**1500**,2000,**2500**
$\|U\|$	1000,2000,5000,8000,**10000**
$\|P\|$	500,1000,**1500**,2000,**2500**
capacity	1,2,3,4,**5**

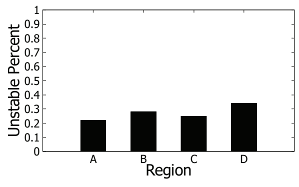
Experiment Environment. In each experiments, we repeat 10 times and report the average results. The algorithms are implemented in Visual C++ 2017, and the experiments are conducted in a Windows 10 machine with Intel(R) Core(TM) i5-6500 3.20GHz CPU and 8GB main memory.

6.2 Results on Real Datasets

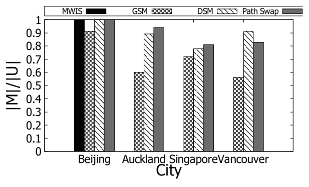
In this section, we conduct experiments using the two baseline algorithms (MWIS and GSM), DSM algorithm and Path Swap algorithm on real datasets, and test the effectiveness and efficiency of the algorithms. To test the effectiveness, we firstly conduct the offline version of TOM [25] and calculate the unstable triples in the matching result. We set the radius as the maximum distance among workers, users, and workplaces, and set the utility as the reciprocal value of distance. We compare the ratio of $|M|$ divided by $|U|$, where the matching size $|M|$ is the number of users who are matched and $|U|$ is the number of users. To test the efficiency, we compare the running time and memory cost of the algorithms.



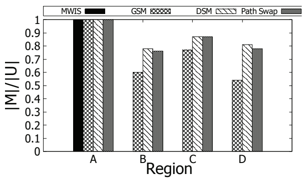
(a) Unstable percentage on Meet Up



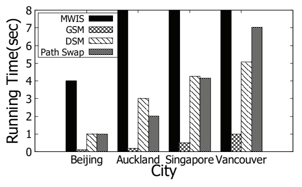
(b) Unstable percentage on DiDi



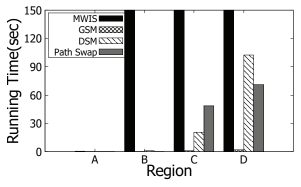
(c) Matching Result on Meet Up



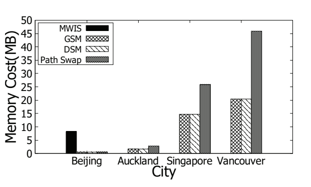
(d) Matching Result on DiDi



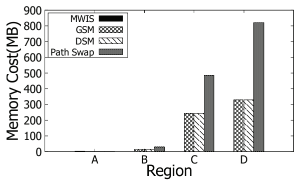
(e) Running Time on Meet Up



(f) Running Time on DiDi



(g) Memory Cost on Meet Up



(h) Memory Cost on DiDi

Figure 4: Experiment Results on Real Datasets

Effectiveness on Real Datasets. Fig. 4(a) and Fig. 4(b) show the matching results of offline TOM which focuses on minimizing minimize total distance. Many matches are unstable in the real datasets, even more than 30% in Singapore and Region D, while the matching results of our algorithms are always stable. Though algorithm of TOM minimizes the total distance, it is not suitable for our problem, because users will be dissatisfied with the unstable results in real applications.

The matching results of our algorithms are shown in Fig. 4(c) and Fig. 4(d). The MWIS algorithm can find an optimal solution that all the users can be matched on dataset of Beijing and Region A, but cannot output solutions within an acceptable period of time in other datasets. The matching result of GSM is the worst. The matching results of Path Swap is better than DSM in Meetup dataset, and both the algorithms reach the optimal solutions in Beijing. In DiDi dataset, DSM can match more triples than Path Swap.

Efficiency on Real Datasets. The running time of MWIS is unacceptable except in Beijing and Region A. GSM runs much faster than the other algorithm, however, the matching result is too bad as mentioned above. Path Swap runs faster than DSM in Auckland, Singapore and Region D, shown in Fig. 4(e) and Fig. 4(f). What is more, MWIS costs much more time and memory than DSM and Path Swap on Beijing dataset. The memory of GSM and DSM is similar. GSM, DSM and Path Swap cost more time and memory as the size of datasets increases on both the datasets. Path Swap costs more memory than two greedy algorithms, because it needs to index the candidates for each matched workplace, shown in Fig. 4(g) and Fig. 4(h).

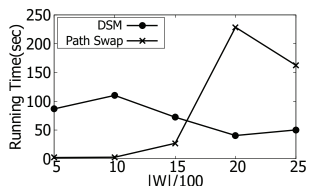
6.3 Scalability on Synthetic Dataset

In this section, we test the scalability of the two approximate algorithms. We conduct the algorithms on a synthetic dataset and test the running time, matching results and memory cost.

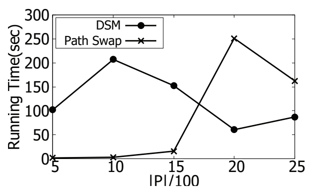
Scalability w.r.t $|W|$. In this experiment, we study the scalability *w.r.t* $|W|$ over the synthetic dataset. Firstly, we set $|U| = 10000$, $|P| = 2500$ and the capacity of each worker $c = 5$. And then, we change $|W|$ from 500 to 2500. The results are shown in the first column of Fig. 5. Fig. 5(a) shows the running time of the algorithms with different size of $|W|$. As $|W|$ increases, the running time of Path Swap gets longer. The reason is that it costs more time to enumerating stable triples as $|W|$ increase. When $|W| = 2500$, there are many fixed units after the initialization step, more users are matched so that the algorithms can terminate faster. Therefore, the time starts to decrease. The running time of DSM shows a different trend. DSM costs more time than Path Swap when $|W|$ is small, and the time reduces when $|W| = 2000$. The reason is that when $|W|$ is small, workers are soon matched, but there are still many unmatched users and workplaces, DSM spends much time in the third case in order to matching more users. When the workers are enough to match all the users, DSM terminates quickly because most of users are matched in the first case. When $|W|$ is large enough, the running time increase again due to the time of enumerating workers increases. The matching result of DSM and Path Swap is similar, as shown in Fig. 5(e). When workers are not enough to match all the users, the size of matching set $|M|$ is close to the theoretical maximum number of matches, which is $\min\{|W|, |P|\} \times c$. Fig. 5(i) shows the memory cost of the algorithms, both the two algorithms cost more memory as $|W|$ increase, and Path Swap costs more memory than DSM because it needs to index the candidate information.

Scalability w.r.t $|P|$. In this experiment, we study the scalability *w.r.t* $|P|$ over the synthetic dataset. Firstly, we set $|U| = 10000$, $|W| = 2500$ and the capacity of each worker $c = 5$. And then, we change $|P|$ from 500 to 2500. The results are shown in the second column of Fig. 5. As $|P|$ increases, the running time of Path Swap increases and the running time of DSM presents a trend of rising first and then falling, as shown in Fig. 5(b), which is similar to the time cost of scalability w.r.t $|W|$ with the similar reason. In Fig. 5(f), both of the algorithms still show a good performance and the Path Swap is better than DSM when $|P| = 1500$. The memory cost, shown in Fig. 5(j), gradually increases as $|P|$ increases.

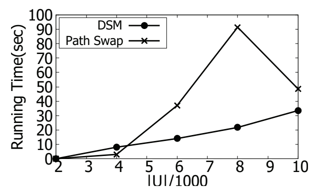
Scalability w.r.t $|U|$. In this experiment, we study the scalability *w.r.t* $|U|$ over the synthetic dataset. Firstly, we set $|W| = 1500$, $|P| = 1500$ and the capacity of each worker $c = 5$. And then, we change $|U|$ from 2000 to 10000. The results are shown in the third column of Fig. 5. As $|P|$ increases, the running time of Path Swap increases and then decreases when $|U|$ is greater than 8000, while the running time of DSM increases, as shown in Fig. 5(c). The reason



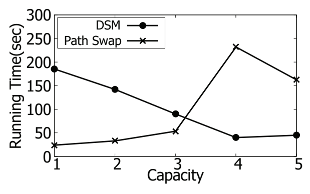
(a) Running Time w.r.t $|W|$



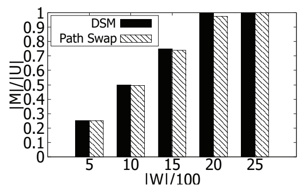
(b) Running Time w.r.t $|P|$



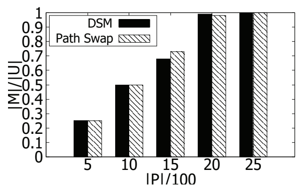
(c) Running Time w.r.t $|U|$



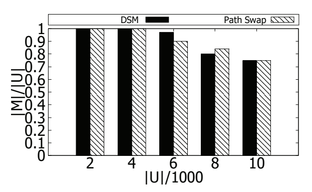
(d) Running Time w.r.t capacity



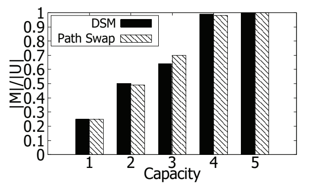
(e) Matching Result w.r.t $|W|$



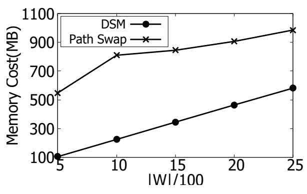
(f) Matching Result w.r.t $|P|$



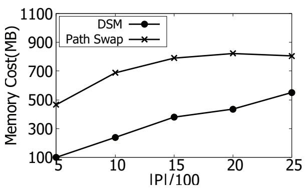
(g) Matching Result w.r.t $|U|$



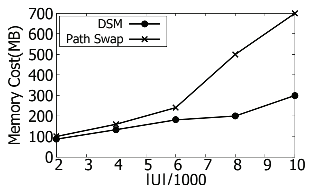
(h) Matching Result w.r.t capacity



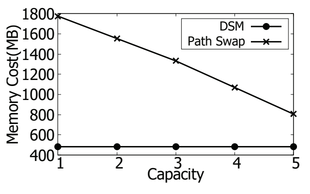
(i) Memory Cost w.r.t $|W|$



(j) Memory Cost w.r.t $|P|$



(k) Memory Cost w.r.t $|U|$



(l) Memory Cost w.r.t capacity

Figure 5: Scalability on Synthetic Dataset

is that the time of enumerating stable triples of both the algorithms and the cost of find and update paths of Path Swap increases as $|U|$ increases. But when $|U|$ is greater than 8000, there are many unmatched users while finding a path, it will save much time to find stable triples in these users than finding a long path and update. The value of $|M|/|U|$ decrease due to $|U|$ gradually surpasses the theoretical maximum number of matches. The memory cost of Path Swap gets larger due to the size of candidate set increases as there are more users, as shown in Fig. 5(k).

Scalability w.r.t capacity. In this experiment, we study the scalability *w.r.t* c over the synthetic dataset. Firstly, we set $|W| = 2500$, $|U| = 10000$ and $|P| = 2500$. And then, we change c from 1 to 5. The results are shown in last column of Fig. 5. As the capacity increases, workers can match more users, and the size of unmatched users decreases more quickly. When the capacity is greater than 4, the theoretical maximum number of matches is greater than $|U|$, therefore the time increases gently. As capacity increase, the path update costs more times because the update operator will be called more times in each matched unit. The matching result gets better as the theoretical maximum number of matches gets larger. The memory cost of Path Swap decreases linearly, as the size of candidate set decreases.

6.4 Summary of Experiment

The MWIS algorithm can output optimal solutions within an acceptable time on small datasets. As the size of dataset increases, the running time is unbearable compared with the other three algorithms. GSM shows a good efficiency but the matching result is not well. DSM and Path Swap show a good performance of the number of users that are matched, and Path Swap is better than DSM on real datasets. The running time of the algorithms are different in different data distribution. According to the experiment results, DSM is more suitable in the scenario that $|U|$ is close to the theoretical maximum number of matches and $|W|$ is close to $|P|$. In other scenarios, where $|U|$ is larger than the theoretical maximum number of matches and $|W|$ differs greatly to $|P|$, Path Swap is more efficient.

7 RELATED WORK

In this section, we review related works from two categories, spatial crowdsourcing and stable matching problem.

Spatial Crowdsourcing. Spatial crowdsourcing attracts more and more attention in recent years [3][6][12][32]. One of the typical task in spatial crowdsourcing is to assign tasks to suitable workers. Most of existing studies [4][26][31][33] aim to maximize the total number or total utility of assigned tasks. [16] and [17] studied an offline version of task assignment. [7] and [24] proposed algorithms for event participant arrangement in event-based social network. [13][29][30] focused on devising online algorithms to solve the dynamic task assignment problems in spatial crowdsourcing. These existing workers only studied the case of two sides (tasks and workers). Song et al. [25] proposed a 3-dimensional task assignment, called TOM problem, which is closely to our work but the aim is still to maximize the total utility score. They proved that even the offline version of TOM problem is NP-hard and gave some approximation algorithms with good performance. However, the offline solution of TOM leads to 40% triples in the matching set are unstable in the real datasets which cannot be accepted in real applications.

Stable Matching Problems. Stable marriage is a typical two-side stable matching problem, which is propose by Gale and Shapley [10]. In this problem, each man/woman has a preference order of the men/women, a stable marriage is an assignment that there are no such two couples that the man and the woman prefer each other than his/her partner. Roth [23] proved that the solution of stable marriage always exists and gave a man-optimal algorithm. The stable roommates and hospitals/residents problem are two major variants of stable marriage [19]. Gale and Sotomayor [11] studied an extension of stable marriage in which the preference lists are incomplete. Bansal et al. [2] studied a many-to-many extension of the stable marriage problem, in which men and women have capacity, and proposes an efficient algorithm for finding a many-to-many stable marriage with the minimum egalitarian. However, these existing technologies cannot be extended to the 3-dimensional version straightforward. Knuth [9] extended the stable marriage to a 3-dimensional version and it was proved the NP-completeness result in [1][22]. The complexity of another model, called cyclic 3D stable matching, is open but partial results can be found in Boros et al. [8] and Eriksson et al. [18]. Although these works propose some models to solve the 3-dimensional stable matching problem, the definitions are quite different from our problem.

8 CONCLUSION

In this paper, we formally design a novel spatial matching problem, called *3-Dimensional Stable Spatial Matching* (3D-SSM). We first analyze our differences with existing task assignment studies in SC and prove that the 3D-SSM is NP-hard. In order to solve this problem, we devise two baseline algorithms. And then, we give two approximation algorithm. The first algorithm is DSM, which has an approximate ratio of η, where η is the percentage of users who are closest to different workplaces, the performance is related to the spatial distribution of users. The second one is Path Swap, which has approximate ratio of σ, where σ is the percentage of workers who are closest to different workplaces, the performance is related to the spatial distribution of workers. Finally, we verify the effectiveness, efficiency and scalability of the proposed methods through extensive experiments on both synthetic and real datasets.

ACKNOWLEDGMENT

Yurong Cheng is supposed by the National Key Research and Development Program of China (Grant No. 2018YFB1004402), the NSFC (Grant No. U1811262) and the China Postdoctoral Science General Program Foundation (No. 2018M631358). Ye Yuan is supported by the NSFC (Grant No. 61572119 and 61622202) and the Fundamental Research Funds for the Central Universities (Grant No. N181605012). Guoren Wang is supported by the NSFC (Grant No. 61672145 and 61732003). Lei Chen is supported by the Hong Kong RGC CRF project C6030-18G, GRF Project 16207617, the NSFC (Grant No. 61729201), Science and Technology Planning Project of Guangdong Province, China (No. 2015B010110006), Hong Kong ITC ITF grants ITS/391/15FX and ITS/212/16FP, Didi-HKUST joint research lab project, Microsoft Research Asia Collaborative Research Grant and Wechat Research Grant. Guoren Wang is the corresponding author.

REFERENCES

[1] S. Ashok. 1994. *A New Approach to Stable Matching Problems.* Society for Industrial and Applied Mathematics. 671–701 pages.
[2] M. Baïou and M. Balinski. 2000. Many-to-many matching: stable polyandrous polygamy (or polygamous polyandry). *Discrete Appl. Math.* 101, 1-3 (2000), 1–12.
[3] Z. Chen, R. Fu, Z. Zhao, Z. Liu, L. Xia, L. Chen, P. Cheng, C. Chen Cao, Y. Tong, and C. J. Zhang. 2014. gMission: A General Spatial Crowdsourcing Platform. *PVLDB* 7, 13 (2014), 1629–1632.
[4] P. Cheng, X. Jian, and L. Chen. 2018. An Experimental Evaluation of Task Assignment in Spatial Crowdsourcing. *PVLDB* 11, 11 (2018), 1428–1440.
[5] P. Cheng, X. Lian, L. Chen, J. Han, and J. Zhao. 2016. Task Assignment on Multi-Skill Oriented Spatial Crowdsourcing. *TKDE* 28, 8 (2016), 2201–2215.
[6] P. Cheng, X. Lian, Z. Chen, R. Fu, L. Chen, J. Han, and J. Zhao. 2015. Reliable Diversity-Based Spatial Crowdsourcing by Moving Workers. *PVLDB* 8, 10 (2015), 1022–1033.
[7] Y. Cheng, Y. Yuan, L. Chen, C. G. Giraud-Carrier, and G. Wang. 2017. Complex Event-Participant Planning and Its Incremental Variant. In *ICDE*. 859–870.
[8] B. Endre, G. Vladimir, J. Steven, and K. Daniel. 2004. Stable matchings in three-sided systems with cyclic preferences. *Discrete Mathematics* 289, 1 (2004), 1–10.
[9] K. D. Ervin. 1997. *Stable marriage and its relation to other combinatorial problems: An introduction to the mathematical analysis of algorithms.* Vol. 10. American Mathematical Soc.
[10] D. Gale and L. S. Shapley. 2013. College Admissions and the Stability of Marriage. *The American Mathematical Monthly* 120, 5 (2013), 386–391.
[11] D. Gale and M. Sotomayor. 1985. Some remarks on the stable matching problem. *Discrete Appl. Math.* 11, 3 (1985), 223–232.
[12] D. Gao, Y. Tong, J. She, T. Song, L. Chen, and K. Xu. 2017. Top-k Team Recommendation and Its Variants in Spatial Crowdsourcing. *Data Science and Engineering* 2, 2 (2017), 136–150.
[13] U. Hassan and E. Curry. 2014. A Multi-armed Bandit Approach to Online Spatial Task Assignment. In *UIC/ATC/ScalCom*. 212–219.
[14] H. Jiang, C.-M. Li, Y. Liu, and F. Manyà. 2018. A Two-Stage MaxSAT Reasoning Approach for the Maximum Weight Clique Problem. In *AAAI*.
[15] H. Jiang, C.-M. Li, and F. Manyà. 2017. An Exact Algorithm for the Maximum Weight Clique Problem in Large Graphs. In *AAAI*.
[16] L. Kazemi and C. Shahabi. 2012. GeoCrowd: enabling query answering with spatial crowdsourcing. In *SIGSPATIAL*. 189–198.
[17] L. Kazemi, C. Shahabi, and L. Chen. 2013. GeoTruCrowd: trustworthy query answering with spatial crowdsourcing. In *SIGSPATIAL*. 304–313.
[18] E. Kimmo, S. Jonas, and S. Pontus. 2006. Three-dimensional stable matching with cyclic preferences. *Mathematical Social Sciences* 52, 1 (2006), 77–87.
[19] M. S. Krishnamoorthy. 1991. The Stable Marriage Problem: Structure and Algorithms (Dan Gusfield and Robert W. Irving). *SIAM Rev.* 33, 2 (1991), 323–324.
[20] X. Liu, Q. He, Y. Tian, W.-C. Lee, J. McPherson, and J. Han. 2012. Event-based social networks: linking the online and offline social worlds. In *KDD*. 1032–1040.
[21] M. Musthag and D. Ganesan. 2013. Labor dynamics in a mobile micro-task market. In *CHI*. 641–650.
[22] C. Ng and H. D. S. 1991. *Three-dimensional stable matching problems.* Society for Industrial and Applied Mathematics. 245–252 pages.
[23] A. E. Roth. 2008. Deferred acceptance algorithms: history, theory, practice, and open questions. *Int. J. Game Theory* 36, 3-4 (2008), 537–569.
[24] J. She, Y. Tong, L. Chen, and T. Song. 2017. Feedback-Aware Social Event-Participant Arrangement. In *SIGMOD*. 851–865.
[25] T. Song, Y. Tong, L. Wang, J. She, B. Yao, L. Chen, and K. Xu. 2017. Trichromatic Online Matching in Real-Time Spatial Crowdsourcing. In *ICDE*. 1009–1020.
[26] T. Song, F. Zhu, and K. Xu. 2018. Specialty-Aware Task Assignment in Spatial Crowdsourcing. In *AISC*. 243–254.
[27] Robert E Tarjan and Anthony E. Trojanowski. 1977. *Finding a maximum independent set.* Stanford University. 537–546 pages.
[28] H. To, C. Shahabi, and L. Kazemi. 2015. A Server-Assigned Spatial Crowdsourcing Framework. *TSAS* 1, 1 (2015), 2:1–2:28.
[29] Y. Tong, J. She, B. Ding, L. Chen, T. Wo, and K. Xu. 2016. Online Minimum Matching in Real-Time Spatial Data: Experiments and Analysis. *PVLDB* 9, 12 (2016), 1053–1064.
[30] Y. Tong, J. She, B. Ding, L. Wang, and L. Chen. 2016. Online mobile Micro-Task Allocation in spatial crowdsourcing. In *ICDE*. 49–60.
[31] Y. Tong, L. Wang, Z. Zhou, B. Ding, L. Chen, J. Ye, and K. Xu. 2017. Flexible Online Task Assignment in Real-Time Spatial Data. *PVLDB* 10, 11 (2017), 1334–1345.
[32] Yongxin Tong and Zimu Zhou. 2018. Dynamic task assignment in spatial crowdsourcing. *SIGSPATIAL Special* 10, 2 (2018), 18–25.
[33] L. Tran, H. To, L. Fan, and C. Shahabi. 2018. A Real-Time Framework for Task Assignment in Hyperlocal Spatial Crowdsourcing. *TIST* 9, 3 (2018), 37:1–37:26.

APPENDIX

A THEORIES AND PROOFS

A.1 Proof of Theorem 1

We consider a special case of this problem, where $|W| = |U| = |P|$ and the capacity of each user is 1, and the other constraints keep the same. The objection is to find the most number of matched stable triples. Its deterministic problem is to find such set of matched stable triples M whose size is α. We prove this problem is NP-hard by reducing the maximum independent set problem [27] (a well-known NP-complete problem) to it. Given a graph $G(V, E)$, the deterministic problem of the maximum independent set problem is to find a subset of V (denoted as $subV$) such that for any pair of vertices $\forall v_i, v_j \in subV$, $e(v_i, v_j) \notin E$, and the size of $subV$ equals to a constant β. To prove this theorem, we should construct a graph $GM(VM, EM)$ using the conditions of the special case of 3D-SSM problem. Firstly, we enumerate all possible triples $vm(w, u, p)$ for each $w \in W$, $u \in U$, and $p \in P$. If (w, u, p) is a stable triple, we put it into VM. In other words, each vertex in $vm \in VM$ is stable triple. The complexity of this step is $O(|W| \times |U| \times |P|)$, where $|W|$ (resp. $|U|$ and $|P|$) is the number of workers (resp. users and workplaces) of W (resp. U and P). For any $vm_i(w_i, u_i, p_i), vm_j(w_j, u_j, p_j) \in VM$, if $w_i = w_j$, or $u_i = u_j$, or $p_i = p_j$, we link an edge between vm_i and vm_j, i.e., $e(vm_i, vm_j) \in EM$. The complexity of this step is $\binom{|VM|}{2}$, where $|VM|$ is the size of VM.

Let $VM = V$, $EM = E$. If $subV$ exist and $|subV| = \beta$, since each vertex in $VM = V$ is a stable triple, the stable constraint satisfies. Since $subV$ is an independent set, no edges are between vertices in $subV$, which means that all workers, users, and workplaces are different. Thus, the unique constraint satisfies. Since the capacity of workers is 1, when the unique constraint satisfies, the capacity constraint is also satisfied. Thus, the result $M = subV$, and its size $\alpha = \beta$. On the other hand, if M exist, when all constraints of the simplified 3D-SSM problem are satisfied, all workers, users, and workplaces in M must be different. This means that no edges are between vertices. Thus, M presents an independent set of GM, i.e., $subV = M$ and $\beta = \alpha$. □

A.2 Proof of Theorem 2

Before conducting the MWIS algorithm, the transfer method in Algorithm 1 are exact. According to the proof of Theorem 1, if the solution of MWIS problem exist, there must be a corresponding solution of our 3D-SSM problem. So the accuracy of the baseline algorithm is the same with that of the MWIS problem. [14] [15] can calculate exact result of the MWIS problem. Though the complexity is $O(2^{|V|})$, efficient pruning algorithms can help to fast calculate the exact results. Thus, the baseline algorithm can be treated to provide an exact answer. □

A.3 Proof of Theorem 3

Since for each user, he can only be matched to one worker and one workplace. If the number of matched triples in the result set M is maximized, the matched number of users must also be maximized. Thus, we just use the number of matched users to present the size of M.

In this problem, not all the uses can be matched every time even in the optimal solution. The users are divided into two parts. For users in the first part, all of them can be matched. However, for users in the second part, only part of them can be matched. Now, we will analyze the second part.

Suppose that the percentage of users in $|U|$ who are the closest to a same workplace p with workers is ξ. These users can only be matched with p with one of the workers to whom p is the closest workplace. Otherwise, the triples are unstable. For each user has a capacity c, if $c \geq \xi|U|$, all these users can be matched. If not, only c users of them can be matched. And then, the optimal solution is $OPT = (|U| - \xi|U|) + \min\{c, \xi|U|\}$, where $|U| - \xi|U|$ is the size of users in the first, and $\min\{c, \xi|U|\}$ is the size of users who will be matched in the second part. The worst output of DSM is $\eta|U|$, where $|U|$ is the set of users. In this case, the user is matched with the closest workplace in each triple. but other users cannot be matched. Therefore, we have $GREEDY \geq \eta|U|$. Thus, $\frac{GREEDY}{OPT} \geq \frac{\eta|U|}{(|U|-\xi|U|)+\min\{c,\xi|U|\}} \geq \frac{\eta|U|}{|U|} = \eta$. □

A.4 Proof of Theorem 4

According to Theorem 3, the optimal solution is $OPT = (|U| - \xi|U|) + \min\{c, \xi|U|\}$, where $|U| - \xi|U|$ is the size of users in the first, and $\min\{c, \xi|U|\}$ is the size of users who will be matched in the second part. The worst output of Path Swap is $\sigma|W|c$, where $|W|$ is the set of workers. In this case, the worker is matched with the closest workplace in each triple and each user can match c users, but other users cannot be matched. Therefore, we have $PATH \geq \sigma|W|c$. Thus, $\frac{PATH}{OPT} \geq \frac{\sigma|W|c}{(|U|-\xi|U|)+\min\{c,\xi|U|\}} \geq \frac{\sigma\frac{|U|}{c}c}{(|U|-\xi|U|)+\min\{c,\xi|U|\}} \geq \frac{\sigma|U|}{|U|} = \sigma$. □

B PSEUDO-CODES OF ALGORITHMS

We include pseudo-codes of the algorithms proposed in this paper.

Algorithm 1: The MWIS Algorithm

```
Input: $W, U, P$
Output: $M$
1  $VM = \emptyset, EM = \emptyset$
2  foreach $w \in W$ do
3      foreach $p \in P$ do
4          $S = \emptyset$
5          foreach $u \in U$ do
6              if $(w, u, p)$ is stable then
7                  $S = S \cup \{(w, u, p)\}$
8          foreach $i$ from 1 to $\min(c_w, |S|)$ do
9              Let the weight of each vertex be $i$, construct $\binom{|S|}{i}$ vertices and put into $VM$
10 foreach $vm_i, vm_j \in VM$ do
11     if any $w$ or $u$ or $p$ are the same then
12         Link an edge between $vm_i$ and $vm_j$
13 $M$ = Conduct algorithms of the MWIS on $GM$
14 return $M$
```

Algorithm 2: GSM Algorithm

```
   Input: $W, U, P$
   Output: $M$
1  $AW = W, AP = P, M = \emptyset$
2  foreach $u \in U$ do
3      $Flag = 0$
4      foreach $p \in DU_u$ do
5          if $p \in AP$ then
6              foreach $w \in DLW_p$ && $w \in AW$ do
7                  if $(w, u, p)$ is stable then
8                      Update $AW$, $AP$ and $M$, $Flag = 1$
9                      break
10         else
11             $w$ = worker matched with $p$
12             if $c_w$ is satisfied && $(w, u, p)$ is stable then
13                 Update $M$, $Flag = 1$
14                 break
15         if $Flag == 1$ then
16             break
17 return $M$
```

Algorithm 3: DSM Algorithm

```
   Input: $W, U, P$
   Output: $M$
1  $AW = W, AU = U, AP = P, Flag = 1, M = \emptyset$
2  while $Flag$ do
3      $Flag = 0$
4      foreach $u \in AU$ do
5          Find a stable triple $(w, u, p)$ where $w \in AW$ and $p \in AP$
6          if $(w, u, p)$ exists then
7              Update $AW$, $AU$, $AP$ and $M$, $Flag = 1$
8              foreach $u' \in AU$ do
9                  if $c_w$ is satisfied && $(w, u', p)$ is stable then
10                     Update $AW$, $AU$, $AP$ and $M$
11         if $Flag = 0$ then
12             Find a stable triple $(w, u, p)$ where $w \in AW$, $p \notin AP$ and
                 $|U_{new}| > |U_{old}|$
13             if $(w, u, p)$ exists then
14                 Update $AW$, $AU$, $AP$ and $M$, $Flag = 1$
15         if $Flag = 0$ then
16             Find a stable triple $(w, u, p)$ where $w \notin AW$, $p \in AP$ and
                 $|U_{new}| > |U_{old}|$
17             if $(w, u, p)$ exists then
18                 Update $AW$, $AU$, $AP$ and $M$, $Flag = 1$
19 return $M$
```

Algorithm 4: Path Swap Algorithm

```
   Input: $W, U, P$
   Output: $M$
1  $AW = W, AU = U, AP = P, Flag = 1, M = \emptyset, units = \emptyset$
2  foreach $p \in P$ do
3      Find a worker $w$ with the maximal capacity to whom $p$ is the closest
4      if $w$ exists then
5          Match at most $c_w$ users with $(w, p)$
6          Update $AW$, $AP$ and $AU$, put $(w, p)$ into $units$
7  Set $AU$ as strong candidate for each $unit \in units$
8  while $Flag$ do
9      $Flag = 0$
10     foreach $p \in AP$ do
11         foreach $w \in AW$ do
12             Find strong and weak candidates for $(w, p)$
13             foreach $u \in (w, p).strong\_candidate$ do
14                 if $c_w$ is satisfied then
15                     Find a swap path and update, $Flag = 1$
16             if $Flag == 0$ then
17                 break
18             foreach $u \in (w, p).weak\_candidate$ do
19                 if $c_w$ is satisfied then
20                     Find a swap path and update, $Flag = 1$
21             Update $AW$, $AU$ and $AP$, put $(w, p)$ into $units$
22 return $M$
```