Front. Comput. Sci. China, 2008, 2(3)
DOI 10.1007/s11704-009-0000-0

REVIEW ARTICLE

Revenu-Maximizing Online Stable Task Assignment on Taxi-Dispatching Platforms

First name, LAST NAME[1,2]

1 Higher Education Press, Beijing 100029, China
2 Acadcmic Journal Division, Beijing 100029, China

Abstract With the growth of the demands of urban tripping the popularization of smartphones, taxi-dispatching platforms such as Uber and Didi Chuxing develop rapidly. In these platforms, a central issue is to assign drivers to passengers effectively, which is often modeled as bipartite matching problem. Existing studies either assume static scenario which is not practical or ignore the fairness of the assignment. A very recent work adopted the concept of stability to model the fairness of assignment. However, their definition on stability relies on restrict order of distances which influences the total revenue of the platforms. In this paper, by releasing the requirements on the priority of the distances, we define a new problem called the Revenue-Maximizig Online Stable Matching (RMOSM) problem. We first propose a baseline algorithm OnlineGreedy to deal with the problem. But it performs not well enough. Thus we introduce a concept called Substitutability and propose an improved algorithm called Equation-substitutable Online Matching (ESOM) algorithm. In ESOM, we use discrete distance and a mark about substitutability to provide tasks more chances to be matched to improve the overall profit. Experiments conducted on both real and synthetic data sets demonstrate that the proposed methods can achieve assignment with higher revenue efficiently while producing limited influence on the fairness of the results.

Keywords Up to 6 words separated by commas. We would like to encourage you to list your Keywords within the abstract section

1 Introduction

With the rapid development of mobile Internet, taxi-dispatching platforms become increasingly popular and important. For example, UBer and Didi are both famous O2O platform covering the dominating portion of the taxi-dispatching market over the world. A central issue in taxi-dispatching platform is task assignment, which assigns drivers to passengers on the platform.

Existing studies on task assignment in taxi-dispatching platforms often model it as bipartite graph matching problems. In a bipartite graph $G = (U; V; E)$, the set of nodes U and V can represent the workers (drivers) and tasks (passengers) respectively. The set of edges E can represent the utility or cost between different tasks and workers. Thus, the goal is to find matching on G to optimize different goals. To maximize the total utility, [1] reduces the bipartite problem into an instance of the maximum flow problem and obtain an exact result by Ford-Fulkerson algorithm. [2, 3] propose Greedy-Based methods to reduce the computation cost in actual scene. To minimize the total cost, which can be transferred to the minimum-cost maximum-flow problem [4], [5] devises the swap chain algorithm to find the optimal answer of matching with maximum carnality which minimizes the maximum travel cost among all the matching. However, the above studies only consider the static matching while the assignment in taxi-dispatching platforms is always dynamic, which means the nodes in U and V of G appears dynamically. There information cannot be known in advance.

There are also some works on task assignment in dynamic scenario and the problem is often formulated as online matching problem. For example, [6] studies the problem of maximizing the total utility and [7] addresses the problem of minimizing the total cost. Besides, [8] studies the trichromatic online matching problem where the influence of third-party workplaces on task assignment is also considered. [9] defines a new problem called flexible task assignment where workers can move in advance following the guiding of the platform to improve the total number of assigned tasks. However, their goal only considers the interest of one party of the platforms, the drivers and the passengers and ignores the fairness of the whole assignment.

Received month dd.yyyy; accepted month dd.yyyy

E-mail: ×××× @ ×××. ×××

By defining the preferences of passengers and drivers, [10] proposes a new problem called Online Stable Matching under Known Identical Independent Distributions(OSM-KIID) to address the task assignment in on-demand taxi-dispatching platform which considers both dynamic scenario and the stability of the matching. However, their definitions of distance to form stability is too strict euclidean distance, which is unnecessary in practice but influences the total profit. We use an example to expound it.

Example 1. Assume we have 3 tasks t_1, t_2, t_3 and 3 workers w_1, w_2, w_3 whose locations are shown in Figure 1 and the detailed attributes are listed in Table 1 and Table 2. For a task t, s_t is the time when t appears at the platform and l_t is the starting point. d_t is t's waiting time. In other words, t will disappear after $s_t + d_t$. p_t is the price that t will pay if matched, which is generally calculated according to the distance from l_t to t's destination; For a worker w, l_w is the current location and s_w is the time when w appears at the platform. t_w is the maximum distance that w can accept from l_w to l_t of an assigned task t. For convenience, here we assume that the assignment is performed in offline manner, and tasks prefer workers nearest to him while tasks prefer workers whose price is higher. Thus, we can get the only stable matching result $M = \{(t_1, w_1), (t_2, w_2)\}$ and the whole revenue is $p_{t_1} + p_{t_2} = 4 + 3 = 7$). If we redefine the distance $d(t, w)$ as follow: $d(t, w) = \lfloor(|t-w|)/\delta\rfloor \times \delta$. Here the $\lfloor x \rfloor$ is the integer part of x, $|t - w|$ is the strict euclidean distance between t and w while δ is a fixed given parameter which is set as 0.5 in this example. We temporarily call $d(t, w)$ as the relaxed distance between t and w. We replace the strict euclidean distance with relaxed distance, which reserves the euclidean distance's integer part as the corresponding distance, the answer set will change a lot. For instance, the distances between w_2 and t_2, w_3 and t_2 now are both 1 rather than approximately 1.13 and 1.41. With this definition, considering equal distances' existence, we can get three possible stable matching result $M' = \{(t_1, w_1), (t_2, w_3), (t_3, w_2)\}$, $M'' = \{(t_1, w_1), (t_2, w_2)\}$ and $M''' = \{(t_1, w_3), (t_2, w_2)\}$. It can be calculated that M' can earn the highest profit $p_{t_1} + p_{t_2} + p_{t_3} = 4+3+2 = 9$. Besides, if taking M' as the final result, as the distance still approximates the exact value, the workers will not move much more extra distance.

Table 1 Tasks

Task	l_t	s_t	d_t	p_t
t_1	(2.0 , 2.0)	0	2	4
t_2	(2.0 , 4.0)	1	2	3
t_3	(1.0 , 5.0)	1	2	2

Table 2 Workers

Worker	l_w	s_w	t_w
w_1	(1.0 , 1.2)	0	2
w_2	(2.8 , 4.8)	0	2
w_3	(3.1 , 3.0)	0	2

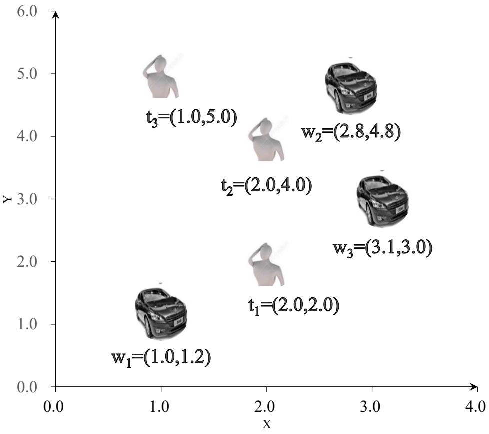


Fig. 1 Locations of tasks and workers

From the example we can see the necessarily of new distance definition. However, what shall we do to work out the matching with highest revenue? In this paper we adopt a concept called Substitutable [11], which provides tasks more chances to be matched and performs better with extended distance definition because of increased number of equal situation.

As discussed above all, we propose a new problem called *Revenue-Maximizing Online Stable Matching (R-MOSM)* problem. As Example 1 indicates, too strict distance definition might cause the loss of revenue in contrast relaxed distance. While in actual scene, passengers care little about assigned drivers' distance varying about 100 meters, which is within acceptable scale. Thus we redefine the distance as a mapping from euclidean distance to a discrete set and introduce the sustainability concept to create more chances for assignment. We build a new model and propose a new algorithm called *Equation-Substitutable Online Matching(ESOM)* to solve *RMOSM* problem. We make the following contribution:

- We redefine the distance between workers and tasks from strict euclidean distance to relaxed distance and propose a new problem called Revenue-Maximizing Online Stable Matching (RMOSM).
- We adopt the concept of Substituatable [11] to help the system to increase the overall number of matched pairs by offer more chances to tasks to be assigned with workers.
- We propose a new algorithm, Equation-Substitutable Online Matching(ESOM), which could solve the R-MOSM problem to an answer with higher overall profit against Baseline algorithm, but the running time performs at the same level.
- We verify the effectiveness and efficiency of ESOM algorithm with experiments on synthetic and real datasets.

The rest of the paper is organized as follow. In section 2, some work related to our problem is showed and simply analyses. In Section 3, we introduce some basic definition and formulate the problem, then put forward a baseline algorithm to deal with the problem. In Section 4, we present another algorithm ESOM with new concept Substitutable. In Section 5, by experiment we compare two algorithm's performance under different factor and prove ESOM's advantage to baseline algorithm. We finally conclude this paper in Section 6.

- RESEARCH articles describing original investigations relevant to Computer Science;
- REVIEW articles giving an in-depth overview of certain topic or a review of one's own work or one laboratory or a group of researchers;
- EDITORIALS providing overview on certain topic written by well known scientists in the field.

2 Related Work

Our problem can be viewed as the online version of Stable Marriage, which is combined of stable marriage problem and online matching problem, and we consult some related work as follow.

Stable Marriage. The stable marriage problem's first introduction is in [12], in which some classic concept like Stable was proposed. Since then, this matching model has been applied to lots of situations [13, 14]. In heterogeneous distributed systems, the concept of stability [12] is used to balance the efficiency and user satisfaction for optimality [15]. Some consider optimize the choice when matching, like using the concept of Substitutable [11, 16]. This concept's application makes it possible to offer objects more possibility to be matched from the overall perspective. However, the limit of feasible matching's number exits as before. Similar to our problem, online stable matching problem is studied in [15, 17]. In [17], it brings out an on-line weighted bipartite matching problem as an online stable marriage model. As for crowdsourcing, [11] considers pricing and revenue maximizing towards stable marriage-like problem, and also it uses the concept of Substituatable, while [11] focus mainly on pricing and maximizing revenue. In addition, some effort was done on house-roommates stable matching to maximize the social welfare [18].

Online Matching. The RMOSM problem is an online stable matching problem, which can be regarded as the online version of specified stable marriage [12]. The input of R-MOSM is a bipartite graph $G = (U, V, E)$, where U represents the offline set of workers while V represents the online set of tasks. Once a $v \in V$ arrives, the edges $(u, v) \in E$ incident to v are revealed. The goal of RMOSM problem is to maximize the overall profit from the matched tasks. At the same time, online matching platform need some principle to keep impartial when matching like the stable concept from offline stable marriage problem [12].

Batch Based Solution [1] is a classic method to reduce online scenario to offline scenario. The basic idea of batch based solution is to periodically match tasks with workers in the static scenario [19, 20]. Fix one time point and deal with all the existing tasks as an offline problem to work out a local assignment and remove the matched ones from corresponding sets. For every time step the batch mode will solve a local assignment out, and if we assume an end of time axis, the final answer matching set is just the combine of all the former local matching set. As reduced to offline problem, original problem is feasible and possible to be dealt with wildly-used methods on offline stable matching [12]. Kazemi use the technique to deal with maximum task assignment(MTA) [19, 20]. Heterogeneous distributed systems [15] uses batch mode to simplify the dynamic data sets. And it plays a vital role in dynamic spatial crowdsourcing [21–25].

3 Problem Statement

In this section we will introduce the basic concepts and definition of Revenue-Maximizing Online Stable Matching(RMOSM) Problem, and a baseline approach to deal with the RMOSM problem.

3.1 Preliminaries and Definition

Definition 1 (Task). *A task, denoted by*

$$t =< l_t, s_t, d_t, p_t > \quad (1)$$

released on the platform at time s_t and at location l_t in the 2D space, needs to be served with a delivery worth p_t within d_t time. In other words, the task t will leave from the platform if it's not assigned to a worker before the time $s_t + d_t$.

Definition 2 (Worker). *A worker, denoted by*

$$w =< l_w, s_w, t_w > \quad (2)$$

appears on the platform with an initial location l_w in the 2D space at time s_w, and only accepts the task to which the distance from the worker is no more than t_w. In other word, if there is a task r and the distance between r and w is more than t_w, r can not be assigned to w in any instance.

Normally, a task request is often put forward with a destination rather than the travel cost there, which is actually easy to be calculated with both location in the 2D space. We assume that workers only consider the travel cost related to the tasks, then the travel costs serve as the same role as the travel distance to workers, simplifying the following discussion.

Definition 3 (Distance). *The distance, denoted by d(t,w), depends on a certain mapping from $T \times W$ to R, which can be defined as*

$$d(t, w) = \lfloor (|t - w|)/\delta \rfloor \times \delta \quad (3)$$

and the δ is a given parameter in corresponding model.

Note that we don't define the distance simply as Euclidean Distance but a relaxed distance. For example, the Euclidean

Distance(d_1) between w_1 and t_1 is 11.0, while that(d_2) between w_2 and t_2 is 11.2, which is different from the former one. But as for our distance definition, it's possible that $d(t_1, w_1)$ and $d(t_2, w_2)$ are both classified to one batch denoted 11. Because in most situation, the passengers don't mind the little difference between 11.0 and 11.2, and ignoring the difference horizons the set of *stable matching*, which benefits to possibly improve the overall profit.

Definition 4 (Matching). *Given a set of tasks T and a set of workers W, a matching, denoted by M* $\in T \times W$*, insists of binary pairs* $< t, w >$*, . Certain t or w don't appear twice in different pairs.*

Definition 5 (Blocking Pair). *A blocking pair, denoted by* $< t, w > \in T \times W$*, satisfies following condition:*
There is another pair $< t^*, w^* >$*, and*

$$\begin{aligned} &p_{t^*} < p_t \; or \; w \; is \; unmatched \\ &d(t, w^*) < d(t, w) \; or \; t \; is \; unmatched \end{aligned} \tag{4}$$

Definition 6 (Stability). *A matching M is stable if* $\forall < t, w > \in T \times W, < t, w >$ *is not a blocking pair.*

Definition 7 (Time Window). *Given a set of tasks T, a set of workers W and the time window's length h, we denote* $min_{t \in T} s_t$ *as* h_0 *and [*h_0*,* h_0 *+ h) as* H_0*,*

$$H_i = [h_i, h_{i+1}) = [h_0 + i*h, h_0 + (i+1)*h), i = 1, 2, ... \tag{5}$$

In this way we divide time axis into several periods, which are denoted as time windows.

Definition 8 (Revenue). *Given an matching* M*,the revenue of M is denoted by*

$$R = \Sigma_{<t,w> \in M} p_t \tag{6}$$

Definition 9 (Revenue-Maximizing Online Stable Matching Problem). *Given a set of tasks T, a set of workers W and time windows* $\{h_i\}_0^\infty$ *based on T and W, we define*

$$T_0 = \{t \in T | s_t \in H_0\} \tag{7}$$

$$W_0 = \{w \in W | s_w \in H_0\} \tag{8}$$

Our problem begins with T_0 *and* W_0*. We deal with* T_i *and* W_i *to find a stable matching* M_i*, then turn to next time window* H_{i+1} *for another stable matching problem about* T_{i+1} *and* B_{i+1}*, and*

$$T_{i+1} = \{t \in T | s_t < h_{i+1}, (s_t + d_t) > h_{i+1}, t \; is \; unmatched.\} \tag{9}$$

$$W_{i+1} = \{w \in W | s_w < h_{i+1}, w \; is \; unmatched.\} \tag{10}$$

We get series of stable matching M_{i0}^{∞}. Then for each M_i we calculate the revenue R_i, and for certain fixed normal number n, define overall revenue R as $\Sigma_{i=0}^n R_i$.

The Revenue-Maximizing Online Stable Matching Problem is to find a algorithm which could give an stable matching with revenue as high as possible.

Online platform has a time axis with infinite length, so we fix a number m as the time scale for convenience of analysis. Then the online problem is divided into $m + 1$ offline problems.

3.2 A Baseline Approach

In this section, we'll simply introduce a baseline algorithm Online-Greedy, in which there is a sub-algorithm called Local-Match. The main procedures of those are presented in Algorithm 1 and Algorithm 2.

The idea of the baseline algorithm is based on Chain Algorithm [26, 27]. The Chain Algorithm's contribution is to solve a stable matching problem on Euclidean distance metric space about closet pairs. Applied to our problem, though the metric standard differs, the algorithm and the prove keeps almost the same.

For Algorithm 1, in lines 1-4, Online-Greedy divide time axis into n parts equally, and length of each part is all set with h. In lines 5, start a cycle for each batch. In line 6-7, at batch $[h_i, h_{i+1})$, we get tasks' and workers' set T_i and W_i in which members are capable at recent batch $[h_i, h_{i+1})$. In line 8, by sub-algorithm Local-Match(T_i,W_i), which will be illustrated later, the result of this period is denoted by M_i. In lines 9-12, if t $\in T_i$ or w $\in W_i$ appears at certain pair in M_i, remove t or w from T or W. In line 13, stop this batch and turn to the next one if there is. In line 14, with sub-problem of each divided period solved we get n results $M_0, M_1, ..., M_{n-1}$, and $M = \bigcup_0^{n-1} M_i$ is our final answer.

As for Algorithm 2 Local-Match, given parameters T and W, in line 1, the algorithm firstly initialize the M as empty set. In line 2-9, as long as T is not empty, select t from T as who offer the highest price, then set W_t containing all workers in W capable to t and w is the one who is closest to t in W_t.In line 6-8, add pair (t, w) to M, then delete t from T and w from W, then turn to line 2 for check. When the cycle ends, return M as answer in line 10.

Example 2. We have 3 tasks $t_1 - t_3$ and 3 workers $w_1 - w_3$ on certain taxi-dispatching platform, whose attributes are listed in TABLE 1 and TABLE 2, while locations are shown in Figure 1. Here the distance between t and w is defined as follow: $d(t, w) = \lfloor (|t-w|)/\delta \rfloor \times \delta$. Here the $\lfloor x \rfloor$ is the integer part of x, $|t - w|$ is the strict euclidean distance between t and w while δ is a fixed given parameter which is set as 0.5 in this example. And [0,1] is set as the first time window, (1,2] is set as the second one. With algorithm Online-Greedy, at first time window, t_1 and w_1,w_2,w_3 are dealt with Local-Match, And the matching start from t_1. Due to the existence of threshold, only w_1 and w_3 are available for t_1, and w_1 is assigned to t_1 because of its order. We add pair (t_1, w_1) to M_1, then turn to the second time window. As the price of t_2 ranks first, we assign w_2 to t_2 for the order and add $(t_2, 2_2)$ to M_2. While there are only t_2 and w_3 left but they are not able to be matched for threshold. Thus our algorithm ends with result $M = M_1 \bigcup M_2 = \{(t_1, w_1), (t_2, w_2)\}$, and the revenue is 7.

Algorithm 1 Online-Greedy(T, W, h, n)

Input: Tasks:T, Workers:W, Length of steps:h, Number of time windows:n

Output: Matching:M

$h_0 \leftarrow \min_{t \in T} s_t$
for $i = 1$ to n **do**
  $h_i \leftarrow h_0 + h \times i$
end for
for $i = 0$ to n **do**
  $T_i \leftarrow \{t \in T | s_t < h_i \; and \; (s_t + d_t) > h_i\}$
  $W_i \leftarrow \{w \in W | s_w < h_i\}$
  $M_i \leftarrow$ Local-Match(T_i, W_i)
  for each (t, w) in M_i **do**
    Remove t from T
    Remove w from W
  end for
end for
$M \leftarrow \bigcup_0^{n-1} M_i$
return M

Algorithm 2 Local-Match(T, W)

Input: Tasks:T, Workers:W

Output: Matching:M

$M \leftarrow \emptyset$
while $T \neq \emptyset$ **do**
  $t \leftarrow argmax_{t \in T} p_t$
  $W_t \leftarrow \{w \in W | d(t, w) \leq t_w\}$
  $w \leftarrow argmin_{w \in W_t} d(t, w)$
  Assert (t, w) into M
  Remove t from T
  Remove w from W
end while
return M

Complexity Analysis. For each time window, the time complexity of baseline is $O(|T_i||W_i|)$. As the largest number of time windows that one task covers is generally limited to a finite constant, with given T and W for the whole time period, the time complexity is $O(|T||W|)$.

However, as the result set is enlarged by relaxed distance, Online-Greedy algorithm has problem with solve out the result closer to the optimal one. In example 1 we analyse the possible matching of the given problem, and the best matching is $M^{'}$ with revenue 9, while the baseline algorithm's answer is only 7. It's because that Online-Greedy's strategy on selecting towards equal distance performs badly so that it's unable to find the optimal result.

In example 2, if t_2 is assigned to w_3 instead of w_2, as $d(t_2, w_3) = d(t_2, w_2)$, t_3 will be matched successfully with w_2 rather than remains unmathced, which inrcease the revenue of final matching directly. It's the shortcoming of Online-Greedy algorithm.

Thus we ought to device a new algorithm with new feature to deal with the equation situation properly.

4 ESOM Algorithm

In this section, we introduce an overview of a new algorithm to deal with *Revenue-Maximizing Online Stable Matching problem(RMOSM)* problem. As shown in the Baseline Approach, tasks are sorted in order of price to be matched. Therefore, revenue of taxi-dispatching platform is directly proportional to the number of matched pairs. Following this feature, to improve the Greedy algorithm, we focus our attention more on how to increase the possibility of tasks to be matched.

Note that the attribute t_w of w, which represents threshold, makes it possible that during one matching t_1 is matched leaving t_2 unmatched while $p_{t_1} < p_{t_2}$. Following Example 3 shows one kind of instance that threshold make sense in matching causing the former described situation.

From the example 2 we can see Online-Greedy has shortcoming so that we can get $\{(t_1, w_1), (t_2, w_3)\}$ rather than $\{(t_1, w_2), (t_2, w_1), (t_3, w_3)\}$ as result to gain higher profit. To increase the possibility of tasks to be matched, we introduce a new concept called Substitutable.

Definition 10 (Substitutable) [11]. *Given a matching M and matched pair (t,w) ∈ M, worker w is substitutable if there exits an unmatched worker $w^{'}$ that $d(t, w) = d(t, w^{'})$.*

This concept is presented in [11] to deal with a pricing problem also about matching. Algorithm 3 and Algorithm 4 illustrate the procedure of Equation-Substitutable Online Matching algorithm, simplified as ESOM algorithm.

The ESOM algorithm is based on traditional methods for stable matching like Gale-Shapley algorithm or Chain, and it reduces online scenario to offline scenario by batch-based solution, so that traditional methods are suitable here. Then the relaxed distance applied to our model creates more equaling situations when tasks and workers compete to be assigned while matching. In this way, substitutable concept offers workers more chances to be matched when equation happens and help to increase the number of matched pairs.

For Algorithm 3, in lines 1-4, ESOM divide time axis into n parts equally, and length of each part is all set with h. In lines 5, start a cycle for each batch. In line 6-7, at batch $[h_i, h_{i+1})$, we get tasks' and workers' set T_i and W_i in which members are capable at recent batch $[h_i, h_{i+1})$. While in lines 8-10, we set an attribute r_t of t as 0, for each $t \in T_i$, which is representative of whether t has failed in ever matching or not. In line 11, by sub-algorithm LE(T_i,W_i), which will be illustrated later, the result of this period is denoted by M_i. In lines 12-14, if t ∈ T_i or w ∈ W_i appears at certain pair in M_i, remove t or w from T or W. In line 16, stop this batch and turn to the next one if there is. In line 17, with sub-problem of each divided period solved we get n results $M_0, M_1, ..., M_{n-1}$, and $M = \bigcup_0^{n-1} M_i$ is our final answer.

For Algorithm 4, given parameters T_i and W_i, in line 1, the algorithm firstly initialize the M_i as empty set. In line

Algorithm 3 ESOM(T, W, h, n)

Input: Tasks:T, Workers:W, Length of steps:h, Number of time windows:n
Output: Matching:M
$h_0 \leftarrow \min_{t \in T} s_t$
for $i = 1$ to n **do**
  $h_i \leftarrow h_0 + h \times i$
end for
for $i = 0$ to n **do**
  $T_i \leftarrow \{t \in T | s_t < h_i \; and \; (s_t + d_t) > h_i\}$
  $W_i \leftarrow \{w \in W | s_w < h_i\}$
  for each t in T_i **do**
    $r_t \leftarrow 0$
  end for
  $M_i = LE(T_i, W_i)$
  for each (t, w) in M_i **do**
    Remove t from T
    Remove w from W
  end for
end for
$M \leftarrow \bigcup_0^{n-1} M_i$
return M

Algorithm 4 LE(T_i, W_i)

Input: Tasks:T_i, Workers:W_i
Output: Matching:M_i
1: $M_i \leftarrow \emptyset$
2: **while** $T_i \neq \emptyset$ **do**
3:   $t \leftarrow argmax_{t \in T_i} p_t$
4:   $W_t \leftarrow \{w \in W | d(t, w) \leq t_w\}$
5:   $M_i, T_i, W_i, W_t = SC(M_i, T_i, W_i, W_t, t)$
6:   **if** task t is unmatched **then**
7:     **if** $r_t = 0$ **then**
8:       $r_t \leftarrow 1$
9:       Reset W_t
10:     **end if**
11:   **else**
12:     Remove t from T_i
13:   **end if**
14: **end while**
15: **return** M_i

2-14, as long as T_i is not empty, and in line 3, select t from T_i as who offer the highest price. In line 4 we set W_t in which workers are able to accept the task, and in line 5 another sub-algorithm SC is performed. In line 6, check if t is not matched, if so, in lines 7-10, when $r_t = 0$, set it as 1 and reset W_t, which means t is gifted with another chance to be matched and own priority from certain perspective. In line 9, reset W_t for another matching of t; if not, remove t from T_i and t lose its matching chance at this batch. Until T_i is empty, in line 15, return M_i as this batch's answer set.

For Algorithm 5, given parameters M_i,T_i and W_i, in line 1, if W_i is not empty and t is not matched, run line 2-17. In line 2, select w from W_t as who is nearest to t. In lines 3-5, if w is unmatched, add (t, w) to M_i and remove w from W_t; if not, we assume w was assigned to t' before. In line 8, judge whether w is substitutable, and if so, rematch w with t rather than t' and replace t in T_i with t', then in line 11, if the price of t' and t is equal while $r_{t'} < r_t$, remove w from $W_{t'}$; if not, remove w from W_t. After finishing above, in line 17, turn back to line 1 for the next round. When the cycle ends, return M_i,T_i and W_i as answers.

Example 3. We have 3 tasks $t_1 - t_3$ and 3 workers $w_1 - w_3$ on certain taxi-dispatching platform, whose attributes are listed in TABLE 1 and TABLE 2, while locations are shown in Figure 1. Here the distance between t and w is defined as follow: $d(t, w) = \lfloor (|t - w|)/\delta \rfloor \times \delta$. Here the $\lfloor x \rfloor$ is the integer part of x, $|t - w|$ is the strict euclidean distance between t and w while δ is a fixed given parameter which is set as 0.5 in this example. And [0,1] is set as the first time window, (1,2] is set as the second one. With algorithm E-SOM, at first time window, t_1 and w_1,w_2,w_3 are dealt with LE, And the matching start from t_1. Due to the existence of threshold, only w_1 and w_3 are available for t_1, and w_1 is assigned to t_1 because of its order. We add pair (t_1, w_1) to M_1, then turn to the second time window. As the price of t_2 ranks first, we assign w_2 to t_2 for the order and add $(t_2, 2_2)$ to M_2. While there are only t_3 and w_3 left but they are not able to be matched for threshold. However, we find that w_2 is substitutable because of w_3, so we rematch w_2 with t_3. Then turn back to t_2 and the only choice is to assign w_3 to t_2. So $M_2 = \{(t_2, w_3), (t_3, w_2)\}$. Thus our algorithm ends with result $M = M_1 \bigcup M_2 = \{(t_1, w_1), (t_2, w_3), (t_3, w_2)\}$, and the revenue is 9.

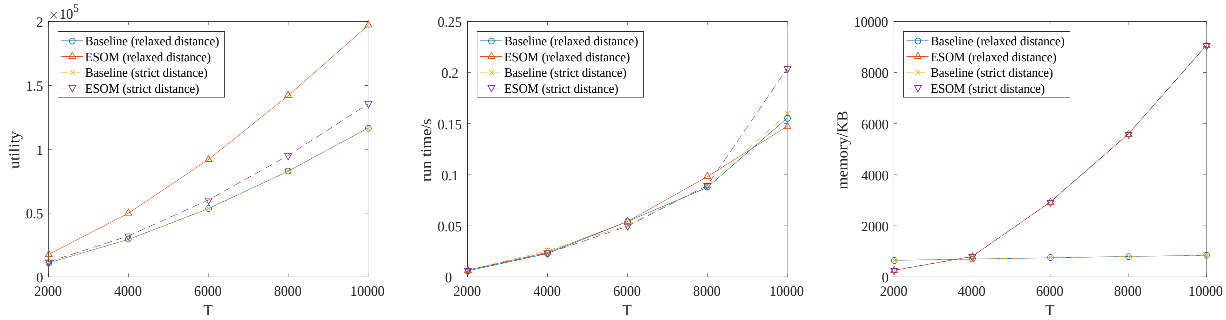
Complexity Analysis. For each time window, the time complexity of ESOM algorithm is $O(|T_i||W_i|)$. As the largest number of time windows that one task covers is generally limited to a finite constant, with given T and W for the whole time period, the time complexity is $O(|T||W|)$.

Definition 10 (Competitive Ratio). *For example, consider an online maximization problem, and ALG is a certain algorithm for the problem. We denote the expected performance of ALG on an input $\mathcal{L}$ as $ALG(\mathcal{L}) = \mathbb{E}_{\mathcal{I} \; \mathcal{L}}[ALG(\mathcal{I})]$,in which the $\mathcal{I}$ is a random arrival sequence. Let $OPT(\mathcal{L}) = \mathbb{E}[OPT(\mathcal{I})]$ denote the expected offline optimal algortihm, where $OPT(\mathcal{L})$ refers to the optimal value with observed full arrival sequence $\mathcal{I}$. Competitive ratio is defined as $min_{\mathcal{L}} \frac{ALG(\mathcal{L}}{OPT(\mathcal{L})}$.*

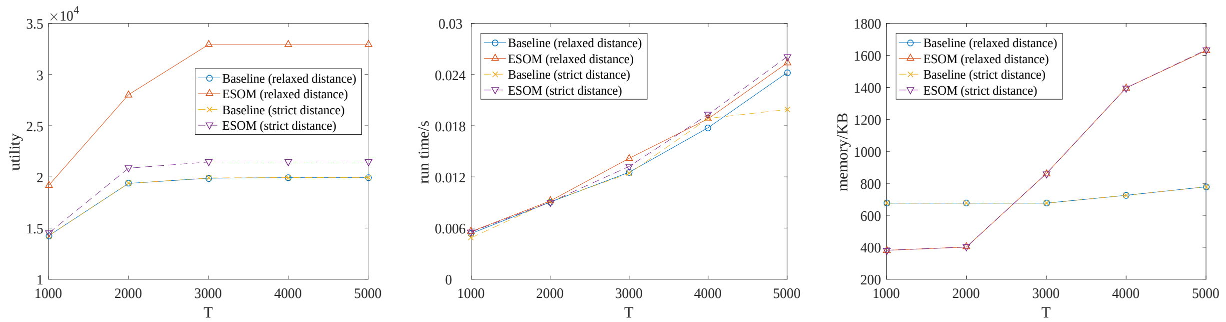
In RMOSM problem, we simpliy the competitive ratio as $\frac{M}{M^*}$, where M is the result of ESOM algorithm, and M^* is the optimal result.

Lemma 1. *For each time window, ESOM algorithm outputs M such that $|M| \geq \frac{2}{3}|M^*|$, where M^* is the matching with optimal profit.*

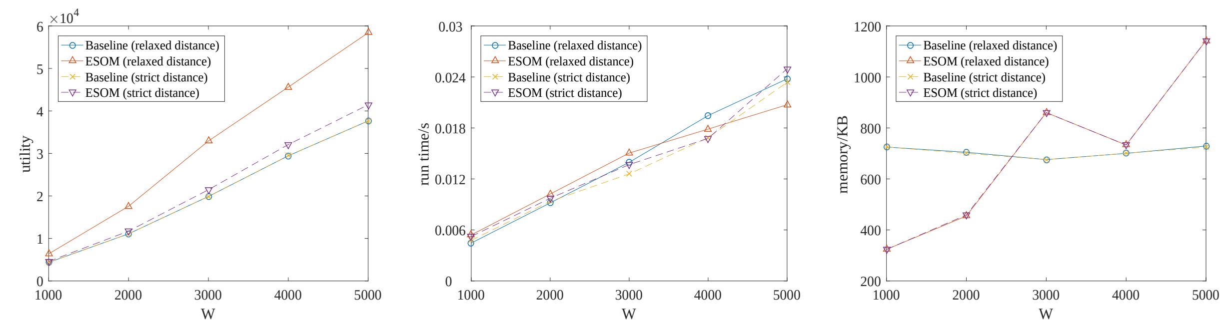
Proof. With both M and M^* known exactly, construct a graph as follows. Set a task node at the graph if and only if the task is matched in either M or M^*, similarly for worker nodes. For any pair of a task node and worker node at



(a) Utility, run time and memory of varying $|T|$ ($|W|$ = $|T|$)



(b) Utility, run time and memory of varying $|T|$



(c) Utility, run time and memory of varying $|W|$

Fig. 2 Results on varying $|T|$ and $|W|$

the graph, if they are matched in M, introduce an undirected edge between them, and if they are matched in M^*, similarly introduce an extra undirected edge. It's obvious that each connected component in the graph is either a cycle or path. For a cycle, it means a common matched pair of both M and M^*, so there is no loss for number of matched pairs. While for a path, an path with even number(2k) edges, as one task or worker can't matched twice or more in certain matching, consists of k matched pairs from M and another k pairs from M^*, which also causes no loss. An path with odd number(2k+1) of edges is called augmenting path. In an augmenting path, matched pairs' numbers of M and M^* are k and k+1 or k+1 and k. Thus if we prove that the graph doesn't contain augmenting paths with 1 or 3 edges then the lemma is proved naturally.

First, if there exists an augmenting path with only 1 edge(from M^*), the task and worker of the path is a blocking pair in M, which contradicts with the stability.

Second, suppose there exists an augmenting path with 3 edges($< w_1, t_1 >, < w_2, t_2 >$ from M^*, $< t_1, w_2$ from M), for p_{t_1} and p_{t_2}, if $p_{t_1} < p_{t_2}$, $< t_2, w_2 >$ is a blocking pair in M. So $p_{t_1} \geq p_{t_2}$. If $p_{t_1} = p_{t_2}$, as t_2 is not assigned to w_2 in M, r_{t_1} should be 1, but w_1 is not matched, which contradicts with $r_{t_1} = 1$, so $p_{t_1} > p_{t_2}$; For $d(t_1, w_1)$ and $d(t_2, w_2)$, it's easy to see that $d(t_1, w_1) \geq d(t_1, w_2)$, otherwise $< t_1, w_1 >$ is a blocking pair in M. If $d(t_1, w_1) = d(t_1, w_2)$, w_2 is substitutable, which contradicts with that t_2 is unmatched. So $d(t_1, w_1) > d(t_2, w_2)$. However, now $< t_1, w_2 >$ is a blocking pair in M^*. Therefore augmenting path with 3 edges can't exist. The lemma is proved.

Theorem 1. *For each time window, the profit R resulted from ESOM algorithm satisfies that* $R \geq \frac{2}{3}R^*$*, where R is the*

Algorithm 5 SC(M_i, T_i, W_i, t)

Input: Matching:M_i, Tasks:T_i, Workers:W_i, Available Workers:W_t,Task:t
Output: Matching:M_i, Tasks:T_i, Workers:W_i, Available Workers:W_t

while $W_t \neq \emptyset$ and t is unmatched **do**
  $w \leftarrow argmin_{w \in W_t} d(t, w)$
  if w is unmatched **then**
    Assert (t, w) into M_i
    Remove w from W_t
  else
    (Assume w was assigned to $t^{'}$)
    if w is substitutable **then**
      Replace $(t^{'}, w)$ in M_i with (t, w)
      Replace t in T_i with $t^{'}$
      if $p_{t'} = p_t$ and $r_{t'} < r_t$ **then**
        Remove w from $W_{t'}$
      end if
    else
      Remove w form W_t
    end if
  end if
end while
return M_i, T_i, W_i, W_t

optimal profit.

As in ESOM algorithm tasks with higher price have priority to be matched, the lemma 1 about number of matched pairs is easily to support Theorem 1. In another way, the competitive ratio is $\frac{2}{3}$.

5 Experimental Study

A. Experimental Setup

We use one real dataset, the New York City Taxi and Limousine Commission(NYC TLC) data set. NYC TLC is an agency of the New York City government that licenses and regulates the medallion taxis and for-hire vehicle industries. In the NYC TLC dataset, every passenger a location, a pickup time and trip distance. Since the information of drivers is not given in the dataset, we generate sets of drivers, of whom the locations follow uniform distribution. We also use synthetic datasets for evaluation. We generate the location, utility and appearing time following uniform distribution. Statistics of the synthetic dataset are shown in Table 3, where we mark our default settings in bold font. $|T|$ and $|W|$ are the capacity of Tasks and Workers, bound is the upper bound of the absolute number of objects' coordinate, β is the number of parts that time period is divided into and δ is that the distance of t and w is define as $d(t, w) = \lfloor (|t - w|)/\delta \rfloor \times \delta$ (Here the $\lfloor x \rfloor$ is the integer part of x). While α is the threshold of drivers.Note that here the thresholds of divers are all set as the same number.

We evaluate the Online-Greedy, ESOM algorithms with strict distance and relaxed distance in terms of total utility, running time and memory cost, and study the effect of varying parameters on the performance of the algorithms. The algorithms are implemented in Visual Studio Code, and the experiment were performed on a machine with AMD Ryzen 7 PRO 2700U 2.20 GHz CPU and 8GB main memory.

Table 3 Synthetic Dataset

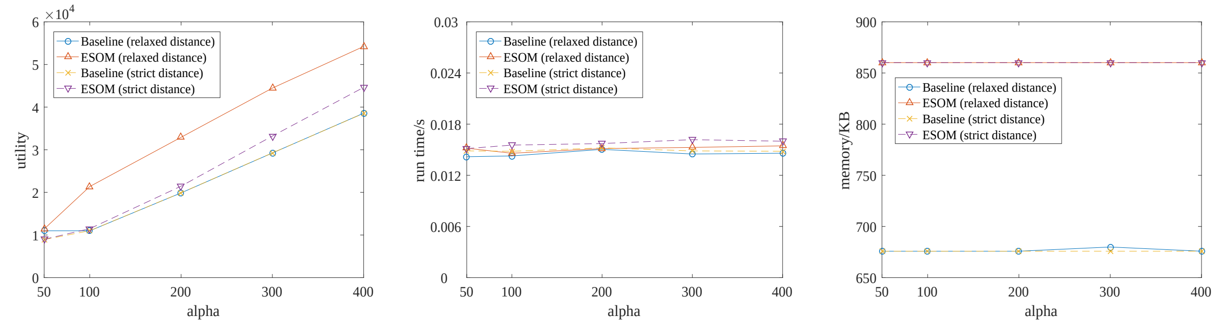
Factor	Setting
$\lvert T\rvert$	1000, 2000, **3000**, 4000, 5000
$\lvert W\rvert$	1000, 2000, **3000**, 4000, 5000
$\lvert T\rvert$ ($\lvert T\rvert = \lvert W\rvert$)	2000, 4000, 6000, 8000, 10000
α	50, 100, **200**, 300, 400
$\alpha\,(real\,data)$	500, 1000, **2000**, 3000, 4000
δ	1, 50, **100**, 150, 180
$\delta\,(real\,data)$	200, 400, **600**, 800, 1000
β	**50**
Bound	**3000**

B. Experiment Result

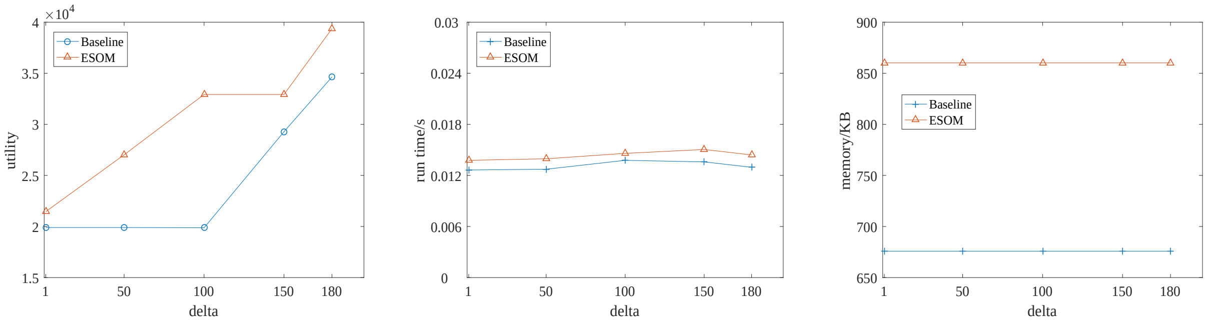
Effect of cardinality of both T **and** W. The results of varying both T and W are shown in (a) of Figure 2. In terms of total utility, we can first observe that the total utility increases as $|T|$ increases which is ought to be as the number of matched pairs increase. Second, we observe that ESOM algorithm with relaxed distance performs best, followed by ESOM with strict distance, Baseline with relaxed distance and Baseline with strict distance. As for running time, there is little difference between four algorithm, and at 10000 the ESOM algorithm with strict distance cost a little more than the other algorithms. Finally, as for memory consumption, we can see that it increase as $|T|$ increases. While ESOM algorithms are less efficient than the Baseline algorithm since they cost more storing space for the substitutable marks.

Effect of cardinality of T. The results of varying T are shown in (b) of Figure 2. In terms of total utility, we can observe that the value as $|T|$ increases at the first three data points while at the last three ones share the same value which is possibly for their corresponding matching' upper bounds. Also, we see that ESOM algorithm with relaxed distance ranks first among four algorithms. For running time, four algorithms perform likely and generally it increases as $|T|$ increase . Finally for memory cost, with low cardinality of $|T|$ ESOM algorithms cost less storing room but when $|T|$ increases they perform worse than baseline algorithms.

Effect of cardinality of W. The results of varying W are shown in (c) of Figure 2. In terms of total utility, we can first observe that the total utility increases as $|W|$ increases which is ought to be as the number of matched pairs increase. Second, we observe that ESOM algorithm with relaxed distance performs best, followed by ESOM with strict distance, Baseline with relaxed distance and Baseline with strict distance. For running time, four algorithms perform likely and generally it increases as $|W|$ increase. Finally for memory cost, with low cardinality of $|W|$ ESOM algorithms cost less storing room but when $|W|$ increases they perform worse than



(a) Utility, run time and memory of varying α



(b) Utility, run time and memory of varying δ

Fig. 3 Results on varying α and δ

baseline algorithms. when $|W|$ arrive at 4000 the running time decreases, and we think it's because that the proper distribution structure reduces the number of matched pairs but increase the total utility.

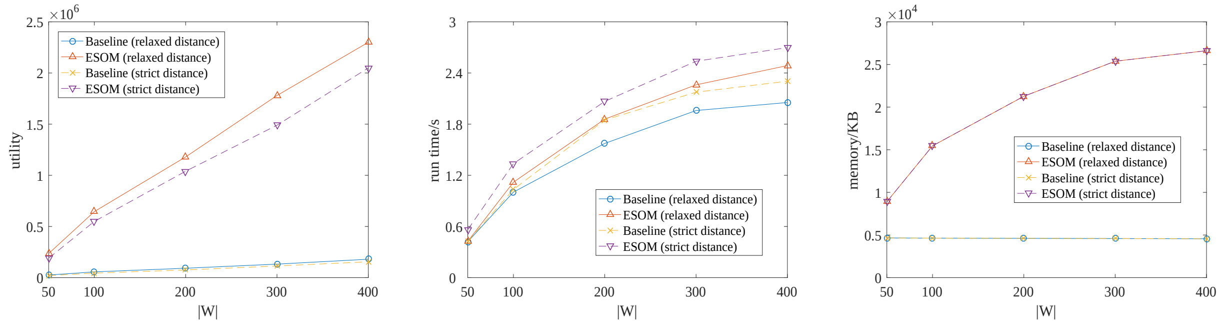
Effect of threshold. The results of varying α are shown in (a) of Figure 3. With respect to total utility, we can observe that it increases with the increment of $alpha$, and ESOM algorithm with relaxed distance ranks first. As for run time, the varying on threshold has no remarkable influence on it. And four algorithms also show little notable feature against each others. For memory cost, ESOM algorithms again cost most room, and we can see the change of threshold has no effect on memory cost.

Effect of distance's relaxation. The results of varying Bound are shown in (b) of Figure 3. The increase of δ as the capacities of workers and tasks on metric space. In terms of total utility, as δ varies from 1 to 180, the overall profits of ESOM and baseline keeps increasing, and at the beginning baseline doesn't increase which is probably due to its poor strategy towards equal distance. At the meanwhile the utilities of ESOM keep above that of baseline. For running time, ESOM costs a little more than baseline and the varying of δ influence little. And for memory cost, also ESOM algorithms cost most storing space, and we can see that the varying Bound has no influence on memory cost.

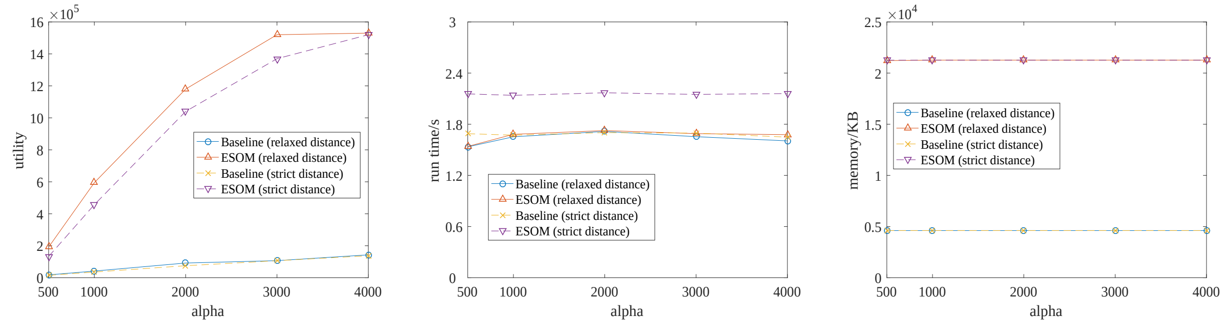
Real datasets. We finally present the experiment results on real dataset. In order to adapt to the actual scale based on real datasets, we adjust the factor of δ and α shown in Table 3. The results of varying $|W|$ are shown in (a) of Figure 4. In terms of total utility, it increases as $|W|$ increases, and ESOM algorithm with relaxed distance ranks first following ESOM with strict distance, Baseline with relaxed distance and Baseline with strict distance. As for run time, besides natural increment before the point 4000, there is a fall between 4000 and 1000 in $|W|$, which is possible that abundant workers make it easy for tasks to be matched. And also we can observe that ESOM with strict distance cost the most run time. Finally for memory cost, ESOM algorithms cost the most still in real datasets. While the results of varying t are shown in (b) of Figure 4. For total utility, it increase as t increases, and we can observe that obviously ESOM algorithms' rates of growth are higher than that of Baseline, and ESOM with relaxed distance ranks first among four. As for run time, ESOM algorithm with strict cost the most in contrast to other three. Finally for memory cost, we can observe that ESOM algorithms need more memory space. The results of varying δ are shown in (c) of Figure 4. The utilities of ESOM is absolutely higher than those of baseline, and from 80 to 100 ESOM's overall profit increases as the other parts keep flat. As for running time, ESOM cost more time against baseline while both of them is not sensitive to δ's change. For memory cost, δ influence little on it.

C. Experiment Summary

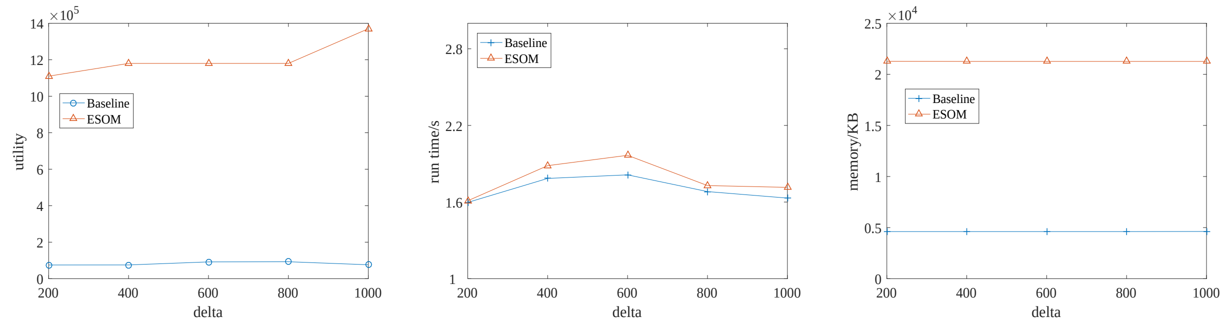
In terms of total utility scores, ESOM algorithm with relaxed distance always perform the best among four algorithms in both synthetic and real dataset. As for experiment, with ca-



(a) Utility, run time and memory of varying $|W|$



(b) Utility, run time and memory of varying t



(c) Utility, run time and memory of varying δ

Fig. 4 Results on varying $|W|$, α and δ on real datasets

pacity's increment, four algorithms' difference is mostly related to the number of matched pairs. And in the aspect of memory cost, although ESOM performs worst for the need of substitutable marks, it's efficient enough to be applied to real-time assignment with low memory cost.

From the Figure 2, Figure 3 and Figure 4 we can see the ESOM algorithm keep the advantage on overall profit as either of capacity of tasks or drivers grows and as both of them grow.

6 Conclusion

In this paper, we identify a problem about dynamic task assignment, called Revenue-Maximizing Online Stable Matching (RMOSM) Problem. We first analyze our difference with existing task assignment studies that assume offline scenario, dynamic bipartite matching problem without stability and traditional online stable matching for maximizing revenue. Then, we extend the definition of stability and distance and propose a baseline algorithm Online-Greedy. Although the baseline algorithm is able to work out a stable matching, it perform not well facing equal scenario. To find better solution, we introduce the concept Substitutable and design a novel algorithm Equation-Substitutable Online Matching (E-SOM), which offer more chances to tasks to be assigned. Finally we conduct experiments which verify the efficiency, effectiveness of the proposed approaches.

References

1. Leyla Kazemi and Cyrus Shahabi. Geocrowd: enabling query answering with spatial crowdsourcing. In *Proceedings of the 20th international conference on advances in geographic information systems*, pages 189–198. ACM, 2012.
2. Hien To, Liyue Fan, Luan Tran, and Cyrus Shahabi. Real-time task assignment in hyperlocal spatial crowdsourcing under budget constraints. In *2016 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pages 1–8. IEEE, 2016.
3. Luan Tran, Hien To, Liyue Fan, and Cyrus Shahabi. A real-time framework for task assignment in hyperlocal spatial crowdsourcing. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 9(3):37, 2018.
4. Ravindra K Ahuja, Thomas L Magnanti, James B Orlin, and K Weihe. Network flows: theory, algorithms and applications. *ZOR-Methods and Models of Operations Research*, 41(3):252–254, 1995.
5. Cheng Long, Raymond Chi-Wing Wong, Philip S Yu, and Minhao Jiang. On optimal worst-case matching. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, pages 845–856. ACM, 2013.
6. Yongxin Tong, Jieying She, Bolin Ding, Libin Wang, and Lei Chen. Online mobile micro-task allocation in spatial crowdsourcing. In *2016 IEEE 32Nd international conference on data engineering (ICDE)*, pages 49–60. IEEE, 2016.
7. Yongxin Tong, Jieying She, Bolin Ding, Lei Chen, Tianyu Wo, and Ke Xu. Online minimum matching in real-time spatial data: Experiments and analysis. *PVLDB*, 9(12):1053–1064, 2016.
8. Tianshu Song, Yongxin Tong, Libin Wang, Jieying She, Bin Yao, Lei Chen, and Ke Xu. Trichromatic online matching in real-time spatial crowdsourcing. In *2017 IEEE 33rd International Conference on Data Engineering (ICDE)*, pages 1009–1020. IEEE, 2017.
9. Yongxin Tong, Libin Wang, Zimu Zhou, Bolin Ding, Lei Chen, Jieping Ye, and Ke Xu. Flexible online task assignment in real-time spatial data. *Proceedings of the VLDB Endowment*, 10(11):1334–1345, 2017.
10. Boming Zhao, Pan Xu, Yexuan Shi, Yongxin Tong, Zimu Zhou, and Yuxiang Zeng. Preference-aware task assignment in on-demand taxi dispatching: An online stable matching approach. In *AAAI Conference on Artificial Intelligence (AAAI 2019)*, 2019.
11. Chaolun Xia and Shan Muthukrishnan. Revenue-maximizing stable pricing in online labor markets. In *Fifth AAAI Conference on Human Computation and Crowdsourcing*, 2017.
12. David Gale and Lloyd S Shapley. College admissions and the stability of marriage. *The American Mathematical Monthly*, 69(1):9–15, 1962.
13. Dan Gusfield and Robert W Irving. The stable marriage problem-structure and algorithms. foundations of computing series. *MIT Press*, 1(2):5, 1989.
14. Manlove David. *Algorithmics of matching under preferences*, volume 2. World Scientific, 2013.
15. Hyunyoung Lee. Online stable matching as a means of allocating distributed resources. *Journal of systems architecture*, 45(15):1345–1355, 1999.
16. Yuan Deng, Debmalya Panigrahi, and Bo Waggoner. The complexity of stable matchings under substitutable preferences. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.
17. Samir Khuller, Stephen G Mitchell, and Vijay V Vazirani. Online algorithms for weighted bipartite matching and stable marriages. *Theoretical Computer Science*, 127(2):255–267, 1994.
18. Guangda Huzhang, Xin Huang, Shengyu Zhang, and Xiaohui Bei. Online roommate allocation problem. In *IJCAI*, pages 235–241, 2017.
19. Leyla Kazemi and Cyrus Shahabi. Geocrowd: enabling query answering with spatial crowdsourcing. In *SIGSPATIAL*, pages 189–198, 2012.
20. Leyla Kazemi, Cyrus Shahabi, and Lei Chen. Geotrucrowd: trustworthy query answering with spatial crowdsourcing. In *SIGSPATIAL*, pages 304–313, 2013.
21. Yongxin Tong and Zimu Zhou. Dynamic task assignment in spatial crowdsourcing. *SIGSPATIAL Special*, 10(2):18–25, 2018.
22. Yongxin Tong, Zimu Zhou, Yuxiang Zeng, Lei Chen, and Cyrus Shahabi. Spatial crowdsourcing: a survey. *The VLDB Journal*, Aug 2019.
23. Yansheng Wang, Yongxin Tong, Cheng Long, Pan Xu, Ke Xu, and Weifeng Lv. Adaptive dynamic bipartite graph matching: A reinforcement learning approach. In *ICDE*, pages 1478–1489, 2019.
24. Yuxiang Zeng, Yongxin Tong, Lei Chen, and Zimu Zhou. Latency-oriented task completion via spatial crowdsourcing. In *ICDE*, pages 317–328, 2018.
25. Yongxin Tong, Yuxiang Zeng, Bolin Ding, Libin Wang, and Lei Chen. Two-sided online micro-task assignment in spatial crowdsourcing. *IEEE Transactions on Knowledge and Data Engineering*, 2019.
26. Raymond Chi-Wing Wong, Yufei Tao, Ada Wai-Chee Fu, and Xiaokui Xiao. On efficient spatial matching. In *Proceedings of the 33rd international conference on Very large data bases*, pages 579–590. VLDB Endowment, 2007.
27. Richard M Karp, Umesh V Vazirani, and Vijay V Vazirani. An optimal algorithm for on-line bipartite matching. In *Proceedings of the twenty-second annual ACM symposium on Theory of computing*, pages 352–358. ACM, 1990.