Dynamic Pricing in Spatial Crowdsourcing: A Matching-Based Approach

Yongxin Tong[1], Libin Wang[1], Zimu Zhou[2], Lei Chen[3], Bowen Du[1], Jieping Ye[4]
[1] BDBC and SKLSDE Lab, Beihang University, Beijing, China, [2] ETH Zurich, Zurich, Switzerland,
[3] The Hong Kong University of Science and Technology, Hong Kong SAR, China, [4] Didi Chuxing Inc., Beijing, China
[1] {yxtong, lbwang, dubowen}@buaa.edu.cn, [2] zimu.zhou@tik.ee.ethz.ch, [3] leichen@cse.ust.hk,
[4] yejieping@didichuxing.com

ABSTRACT

Pricing is essential for the commercial success of spatial crowdsourcing applications. The spatial crowdsourcing platform prices tasks according to the demand and the supply in the crowdsourcing market to maximize its total revenue. Traditional pricing strategies seek a unified optimal price for a single global market. Yet spatial crowdsourcing needs to dynamically price for multiple local markets fragmented by the spatiotemporal distributions of tasks and workers and the mobility of workers. Dynamic pricing in spatial crowdsourcing is challenging because the supply in local markets can be limited and dependent, leading to global dependencies when pricing tasks in each local market. To this end, we define the *Global Dynamic Pricing* (GDP) problem in spatial crowdsourcing. We further propose a *MAtching-based Pricing Strategy* (MAPS) with guaranteed bound, which efficiently approximates the expected total revenue for markets with limited supply, effectively distributes the dependent supply and dynamically prices the tasks. Extensive evaluations on both synthetic and real-world datasets demonstrate the effectiveness and efficiency of MAPS.

CCS CONCEPTS

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

KEYWORDS

Spatial Crowdsourcing; Pricing Strategy

ACM Reference Format:
Yongxin Tong[1], Libin Wang[1], Zimu Zhou[2], Lei Chen[3], Bowen Du[1], Jieping Ye[4]. 2018. Dynamic Pricing in Spatial Crowdsourcing: A Matching-Based Approach. In *SIGMOD'18: 2018 International Conference on Management of Data, June 10–15, 2018, Houston, TX, USA.* Houston, TX, USA, 16 pages. https://doi.org/10.1145/3183713.3196929

1 INTRODUCTION

The development of mobile Internet and sharing economy brings the prosperity of spatial crowdsourcing. Nowadays, spatial crowdsourcing applications are ubiquitous: intelligent transportation (*e.g.*, Uber [3] and DiDi [5]), food delivery (*e.g.*, Seamless [1]), information collection (*e.g.*, Waze [6] and OSM [2]) and micro-tasks (*e.g.*, Gigwalk [4] and gMission [16]). These platforms provide a new way of organizing the crowd to complete spatial tasks.

Pricing the spatial tasks is one crucial step in the management and operation of spatial crowdsourcing. The typical pricing process in spatial crowdsourcing works as follows. First, requesters submit tasks to the crowdsourcing platform. Each task requires a crowd worker to travel a distance from his/her origin to a specific destination [15, 40]. Then the platform decides the price per unit distance for the tasks and reveals the prices to the requesters. After observing the unit price, the requesters decide whether to accept it or not, according to their expectations on the unit price. After receiving the requesters' decisions, the platform assigns workers for those requesters who accept the unit price, and gets a corresponding revenue. Proper unit prices are vital for the total revenue of the platform, because *(i)* low unit price may yield low total revenue when the number of workers is limited; *(ii)* high unit price may threaten requesters away, which also decreases the total revenue.

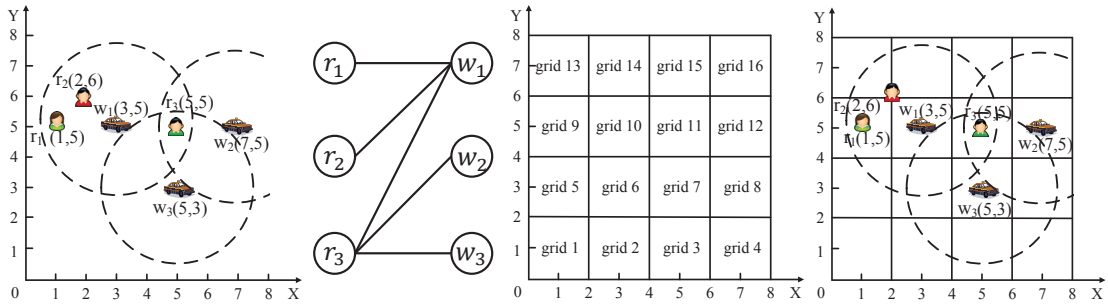
Yet pricing tasks in spatial crowdsourcing is non-trivial. Unlike traditional crowdsourcing where each worker can potentially perform *all tasks* posted on the platform [17, 25, 30, 31], workers in spatial crowdsourcing can only serve *a portion of spatial tasks*, since some tasks require a traveling distance beyond the capability of workers [15, 40]. Consequently, the *unified market* in traditional crowdsourcing tends to fragment into *multiple local markets* in spatial crowdsourcing. As a result of the spatiotemporal distributions of workers and tasks, each local market often varies in supply and demand, posing a need for *dynamic pricing for each local market*. Despite existing research on pricing in crowdsourcing [36, 37], dynamic pricing in spatial crowdsourcing is largely unexplored due to the following three challenges.

Unknown Demand. Only the requesters who accept the price will contribute to the total revenue of the crowdsourcing platform. But the decisions of requesters are unknown before the platform decides the unit prices. A common approach is to estimate the expectation of requesters on the unit prices. Thus the first question for pricing in spatial crowdsourcing is: *How to estimate the expectations of requesters on different prices* (i.e., *how much they are willing to pay)?*

Limited Supply. Unlike traditional crowdsourcing where workers are sufficient, spatial crowdsourcing applications can face a shortage of workers. For instance, near the stadium after a football match, there are usually insufficient taxis to drive people home, and passengers are willing to pay a higher price due to the imbalanced demand and supply. The variety of demand and supply across local

SIGMOD'18, June 10–15, 2018, Houston, TX, USA

ACM ISBN 978-1-4503-4703-7/18/06…$15.00
https://doi.org/10.1145/3183713.3196929



(a) Tasks and workers. (b) Bipartite graph. (c) Grids. (d) Tasks and workers in grids.

Figure 1: An illustration of the running example.

Table 1: Acceptance ratios for Example 1.

p	1	2	3
$S(p)$	0.9	0.8	0.5

markets poses a question: *How to formulate a pricing framework to meet the diverse demand-supply conditions in multiple local markets?*

Dependent Supply. Workers in real-world spatial crowdsourcing applications can be dependent. For example, a taxi may be able to pick up passengers living in multiple districts. But once it picks a passenger, it cannot serve other passengers, leading to a reduce in supply in the rest of the districts (local markets). As the platform aims to maximize the total revenue among all local markets, the dependency among local markets raises an additional question. *How to distribute the supply in multiple dependent local markets such that the unit prices in each local market maximize the total revenue?*

We illustrate the unique challenges of pricing in spatial crowdsourcing via the following example.

Example 1. *Imagine a taxi-calling platform and assume 3 tasks ($r_1 - r_3$) and 3 workers ($w_1 - w_3$) appear on the platform. The origins of the tasks and the locations of the workers are shown in Fig. 1a. To finish r_1, r_2 and r_3, a worker needs to travel from the task's origin to destination with a distance of 1.3, 0.7, and 1, respectively. The platform will offer a unit price for each task, and the tasks whose requesters accept the offered price will be served. Table 1 shows the probabilities of a requester to accept a given price. Note that Table 1 needs to be estimated in practice. If every worker can perform all the tasks, according to Table 1, a unit price of 2 will maximize the expected total revenue of the platform. In reality, workers can only move in a finite speed, imposing a range constraint on the tasks a worker can serve. We assume all workers have the same range constraint, which is a circle of a radius of 2.5. Based on the range constraint, we can construct a bipartite graph (Fig. 1b), to reflect the spatial distributions of tasks and workers. From Fig. 1b, at most two tasks can be served and at most one of r_1 and r_2 can be served. We can set the unit price of 3 to r_1 and r_2 , because the possibility of both r_1 and r_2 reject the price is low. And r_3 is assured to be served as long as the offered price is accepted, so we can offer the unit price of 2 to r_3, which can maximize the expected revenue contributed by r_3. We will see later that unit prices of 3, 3, 2 for r_1, r_2 and r_3 is optimal in this example. To get such an optimization, we need to estimate the acceptance ratios of requesters, and consider the spatial distribution of tasks and works, which indicates limited and dependent supply in the region.*

Contributions and Roadmap. Motivated by the above example, we propose the *Global Dynamic Pricing* (GDP) problem in spatial crowdsourcing (Sec. 2). It stems from real-world spatial crowdsourcing applications and aims to deal with *dynamic pricing in multiple local markets with (i) unknown demand, (ii) limited supply and (iii) dependent supply*. To solve the GDP problem, we first propose a *base pricing* strategy with guarantees to set a unified base price for all local markets with unknown demand (Sec. 3). Taking the base price as input, we design *MAtching-based Pricing Strategy* (MAPS), which *(i)* promptly learns the demand (probabilities to accept a given price), *(ii)* efficiently approximates the expected revenue with both sufficient and limited supply, and *(iii)* incrementally optimizes the dependent supply (Sec. 4). We verify the effectiveness and efficiency of the proposed pricing strategies on large-scale synthetic and real-world datasets (Sec. 5). Finally, we review previous works and conclude this paper (Sec. 6 and Sec. 7).

2 PROBLEM STATEMENT

This section introduces the important notations and defines the *global dynamic pricing (GDP)* problem in spatial crowdsourcing. All proofs involved in this paper are presented in the appendix.

2.1 Preliminaries

We assume the region of interest is partitioned in space as grids.

Definition 1 (Grid). *The entire spatial region of interest is partitioned into grid cells, indexed by* $1, \ldots, G$.

Spatiotemporal information is a central factor for pricing in spatial crowdsourcing. Platforms tend to set a single unit price (*i.e.*, price per unit distance) for the tasks in the same grid and the same time period, due to lack of additional data for personalized pricing. Note that the concrete space partitioning scheme is application-specific, and we adopt grid indices for simplicity.

Example 2. *We use grids with the side of length 2 and index from the bottom-left to partition the region in Example 1. w_3 is in grid 7. r_1 and r_2 are in grid 9.*

Definition 2 (Spatial Task). *A spatial task ("task" for short), denoted by $r =< t, ori_r, des_r >$, is issued by a requester*[1] *in a time period t. Each requester has a private valuation v_r representing the maximum unit price he/she is willing to accept.*

[1]In the rest of the paper, we also use r to denote the requester who issues the task r.

If a requester accepts the unit price set by the platform, he/she will be assigned a worker to complete his/her task. To complete the task, the worker travels from ori_r to des_r with a total distance d_r (*e.g.*, Euclidean or road-network distance). And the platform gets a revenue equal to d_r times the offered unit price.

Private valuations v_r are unknown to the platform. We assume that the private valuations in the same grid g are i.i.d. samples from an unknown distribution in a given time period [9, 10, 22, 34, 37]. For any requester in a grid g, the cumulative distribution function ("CDF" for short) of his/her v_r is defined by $F^g(p) = \Pr[v_r \leq p]$. We further define the acceptance ratio of requesters.

Definition 3 (Acceptance Ratio of Requesters). *For any requester in a grid g and a given unit price p, his/her acceptance ratio w.r.t. p is defined by $S^g(p) = \Pr[v_r > p] = 1 - F^g(p)$.*

Note that the acceptance radio is a generic indicator of the price level in the spatial crowdsourcing market. According to the efficient-market hypothesis (EMH) [24], the acceptance ratios have implicitly accounted for all available information such as time, routes, traffic conditions, weather, etc.

Definition 4 (Crowd Worker). *A crowd worker, denoted by $w =< t, l_w, a_w >$, is available on the platform from time period t at an initial location l_w. A worker w can complete any task r only if r's origin ori_r is located within the circle centered at l_w with the radius a_w (known as the range constraint).*

This work only considers workers without a constraint on his/her final destination. Studies have shown that most workers in crowdsourcing platforms tend to perform multiple tasks for a long time [33], *e.g.*, full-time taxi drivers. Workers who only perform tasks at their convenience (*e.g.*, temporary Uber drivers who take passengers on their way home) are beyond the scope.

2.2 Problem Definition

As stated in Sec. 1, the platform sets the unit prices for spatial tasks to maximize its potential total revenue. The total revenue is affected by *(i)* the tasks that accept the unit prices and *(ii)* the spatiotemporal relationships between tasks and workers. We define total revenue leveraging a probabilistic bipartite graph, which represents both the probabilistic acceptance of tasks (modeled by acceptance ratios in Definition 3), as well as the spatial constraints between tasks and workers (each worker may serve multiple grids but can only perform one task at a time).

We use *possible world semantics* [19] to identify the *probabilistic bipartite graph*, denoted by $\mathcal{B}^t =< R^t, W^t, E^t, S >$, with the distribution of sampled *possible bipartite graphs*, where R^t and W^t denote the sets of issued tasks and available workers in time period t, respectively, and S is the set of acceptance ratios of requesters. *(i)* The nodes on the left and right represent the tasks and workers, respectively. *(ii)* There is an edge $(r, w) \in E^t$ if the task r satisfies the range constraint of the worker w. *(iii)* The weight of an edge (r, w) is $d_r \times p_r$ where p_r is a specific unit price of r, and d_r is the distance from the origin ori_r to the destination des_r. At the end of t, a bipartite graph $B^t =< R'^t, W'^t, E'^t >$, which is an instantiation of the *probabilistic bipartite graph* $\mathcal{B}^t$, can be constructed. Specifically, $R'^t \subseteq R^t$ are the tasks that accept the unit price, *i.e.*, nodes on the left in $\mathcal{B}^t$ with $p_r \leq v_r$. W'^t is the same as W^t. E'^t is the edge set of the subgraph based on R'^t and W'^t. We then define the total revenue of all grids for time period t.

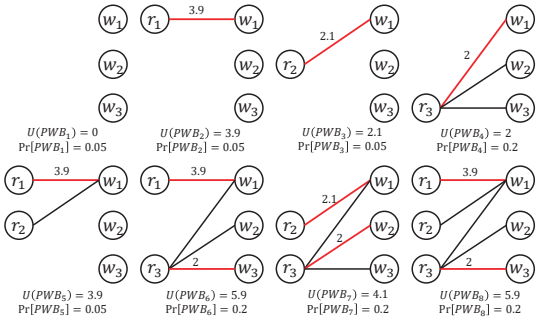


Figure 2: Possible bipartite graphs for Fig. 1b.

Definition 5 (Total Revenue). *At the end of a time period t, given a set of issued tasks R^t, a set of available workers W^t, and the corresponding prices for the tasks, an instantiated bipartite graph $B^t =< R'^t, W'^t, E'^t >$ can be constructed, and the total revenue $U(B^t) = \sum_{r \in R'^t, w \in W'^t, (r,w) \in M} d_r \times p_r$, where M is the maximum weighted bipartite matching in B^t.*

There can be $2^{|R^t|}$ possible bipartite graphs (and thus $2^{|R^t|}$ revenues). Denote the i-th possible bipartite graph by PWB_i. In PWB_i, we use $R^t_{PWB_i}$ as the set of tasks that accept the unit price p_r and thus $R^t \backslash R^t_{PWB_i}$ is the set of the tasks that reject the unit price. Since the private valuations of requesters v_r within the same grid g and time period are assumed to be i.i.d., thus the acceptance ratios of all requesters in a grid are the same probability $S^g(p_r)$. Then we can define the sampling probability of PWB_i:

$$\Pr[PWB_i] = \prod_{r \in R^t_{PWB_i}} S^g(p_r) \prod_{r \in R^t \backslash R^t_{PWB_i}} (1 - S^g(p_r)),$$

where the first continued product is the probability of requesters accepting the unit prices and the second continued product means the probability of requesters rejecting the unit prices. We use $PWB_i \sqsubseteq \mathcal{B}^t$ to represent that PWB_i is sampled from $\mathcal{B}^t$.

Now we define the expected total revenue for the platform.

Definition 6 (Expected Total Revenue). *For a time period t, given a set of issued tasks R^t, a set of available workers W^t, a multiset of unit prices P^t for each r, and a set of acceptance ratios w.r.t P^t for $|R^t|$ unit prices, the expected total revenue is*

$$\mathbb{E}[U(\mathcal{B}^t)|P^t] = \sum_{PWB_i \sqsubseteq \mathcal{B}^t} U(PWB_i) \Pr[PWB_i]$$

where $U(PWB_i)$ is the total revenue of the i-th bipartite graph.

Example 3. *Back to our Example 1, for each task/requester, we assume that the acceptance ratios are the same as in Table 1. The specific prices of the three tasks r_1, r_2, r_3 are 3, 3, 2, respectively. All possible bipartite graphs of Fig. 1b are shown in Fig. 2, where each possible bipartite graph is sampled with probability $\Pr[PWB_i]$. For example, the sampling probability of the second possible bipartite graph is calculated by $S(3) \times (1 - S(3)) \times (1 - S(2)) = 0.5 \times (1 - 0.5) \times (1 - 0.8) = 0.05$. For each possible bipartite graph, the maximum weighted bipartite matching is shown as red lines, and $U(PWB_i)$ is the corresponding total revenue. Hence, given the unit prices {3, 3, 2}, the expected total revenue of Fig. 1b is 4.1.*

Table 2: Summary of major notations.

Notation	Description
W^t,R^t	The set of workers and requesters
G	The number of grids
v_r	The evaluation of requester r
d_r	The travel distance of request r
a_w	The radius of the acceptance range of worker w
$F^g(p)$,$S^g(p)$	The CDF of v_r and the acceptance ratio of requester r
B^t	The bipartite graph between workers and requesters
E^t	The edge set in the bipartite graph
U^t	The total revenue in time period t

In real-world spatial crowdsourcing applications, *the platform does not know the acceptance ratios of requesters in advance*. The acceptance ratios of requesters are hidden variables to be estimated. And the platform always hopes to find a set of globally optimal prices to maximize its expected total revenue. The goal stated above can be formalized to the following problem.

Definition 7 (GDP Problem). *For a time period t, given a set of tasks R^t without knowing their acceptance ratios, a set of workers W^t, the GDP problem is to find a multiset of unit prices P^t for each r such that the expected total revenue $\mathbb{E}[U(\mathcal{B}^t)|P^t]$ is maximized.*

Theorem 1. *The GDP problem is NP-hard.*

Table 2 lists the major notations used throughout the paper.

3 BASE PRICING

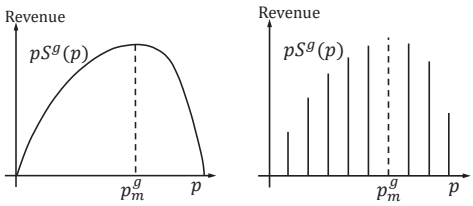
In this section we present base pricing, a baseline pricing strategy that efficiently estimates the unknown demand (*i.e.*, acceptance ratios of requesters), and sets the same unit price for all the grids, a.k.a. *base price*. A unified base price is reasonable when the supply is sufficient, *i.e.*, after each requester accepts the price, there is always an available worker to serve it. It reflects the long-term unit price, which is a conservative option when the platform has no prior knowledge of the market. It can also serve as the initial inputs for dynamic pricing, which we will elaborate on in Sec. 4.

3.1 Basic Idea

Base pricing first estimates the optimal unit prices that maximize the expected total revenue in each grid and then takes the average of these unit prices as the base price. When there is sufficient supply for the tasks in a grid, the optimal price that maximizes the expected total revenue in the grid is related to the Myerson reserve price [34], which we explain as follows.

3.1.1 Optimal Price for Each Grid and Myerson Reserve Price. For a given unit price p, a requester r in grid g will accept p with the probability $S^g(p) = \Pr[v_r > p] = 1 - F^g(p)$. If there is always a worker who can finish r (*i.e.*, sufficient supply), the expected revenue of r can be expressed as $d_r p S^g(p)$ (where ori_r locates in grid g). Thus the expected total revenue of the requests in grid g can be expressed as $\sum_r d_r p S^g(p) = p S^g(p) \cdot \sum_r d_r$. And we only need to find the price (*i.e.*, p_m^g) which maximizes $pS^g(p)$ because it will also maximize the expected total revenue in grid g.

Further assume the demand distribution $F^g(p)$ is Monotone Hazard Rate (MHR) distributions, meaning that $F^g(p)$ is twice differentiable and the hazard rate function $F'^g(p)/(1 - F^g(p))$ is monotone non-decreasing, where $F'^g(p)$ is the first-order derivative of the cumulative distribution function of v_r. Then the price $p_m^g = \arg\max_p pS^g(p)$ is often named as the *Myerson reserve price* [34], as shown in Fig. 3a. The function $pS^g(p)$ is concave, which can be deduced by the properties of MHR [9]. There exists a value p_m^g such that $pS^g(p)$increases when $p < p_m^g$ and decreases when $p > p_m^g$. Hence the Myerson reserve price p_m^g is the unique maximizer. Note that the assumption on MHR distributions is mild, because MHR distributions are common, which include normal, exponential, and uniform distributions [9, 11].



(a) Revenue curve. **(b) Price estimation.**

Figure 3: An illustration of base pricing.

We next introduce how to estimate p_m^g for each grid.

3.1.2 Estimating Optimal Price for Each Grid. Accurate calculation of the Myerson reserve price p_m^g for a give grid g relies on the *continuous* revenue curve (*i.e.*, $pS^g(p)$ given p). Yet it is difficult to obtain the revenue curve for each grid g. To estimate p_m^g efficiently and effectively, we choose a candidate set of prices (denoted by P_{cand}^g) to sample the acceptance probability $S^g(p)$ and apply the best sample price that maximizes $p\widehat{S}^g(p)$ as p_m^g, where $\widehat{S}^g(p)$ is the sample mean. The procedure is illustrated in Fig. 3b.

3.1.3 Determining Base Price. The base price p_b is the arithmetic mean of the estimated Myerson reserve prices $\{p_m^g\}$ of all the grids.

3.2 Algorithm Details

Algorithm 1 shows the procedure of base pricing. We select p_{min} and p_{max} as the bounds of the sample prices, and $(1 + \alpha)$ as the multiplier between two successive sample prices. The parameters ϵ and δ control the accuracy of the sampling, which will be discussed in Sec. 3.3. In line 1, k represents the number of candidates to estimate the Myerson reserve price. In lines 2-9, we estimate p_m^g for each grid g from 1 to G. Specifically, in lines 5-6, price p is offered to $h(p)$ requesters who recently have issued tasks. Then we observe their decisions of acceptance or rejection to update the acceptance ratio (the number of accepted requesters over $h(p)$). In line 7, we add the current price p and its observed acceptance ratio $\widehat{S}^g(p)$ into the candidate set P_{cand}^g. Then we generate the next sample price in line 8. In line 9, we choose the price which maximizes $p\widehat{S}^g(p)$. Ties are broken by choosing the smaller price, since it usually represents a higher acceptance ratio. Finally the base price p_b is output as the arithmetic mean of the estimated $\{p_m^g\}$.

Example 4. *Suppose $p_{min} = 1$, $p_{max} = 5$, $\alpha = 0.5$, $\epsilon = 0.2$, and $\delta = 0.01$. Then $k = 4$. The sample prices in P_{cand}^g are $1, 1.5, 2.25, 3.375$. For some g, we sample price $p = 1$ for $h(p) = 335$ times. Suppose when we offer the price of 1 to 335 requesters, 300 requesters accept the price. Then $\widehat{S}^g(1) = 0.9$. Further suppose we get all the $\widehat{S}^g(p)$ corresponding to each sample price as $0.9, 0.85, 0.75, 0.4$. Thus we set $p_m^g = 2.25$.*

Remarks. We choose successive sample prices based on a multiplier $(1+\alpha)$. The performance guarantee of such a sampling scheme is analyzed in Sec. 3.3. However, other step sizes also apply. If the

Algorithm 1: Base Pricing

input : $p_{min}, p_{max}, \alpha, \epsilon, \delta$
output: p_b

$k \leftarrow \lceil \frac{\ln(p_{max}/p_{min})}{\ln(1+\alpha)} \rceil$;
for $g \leftarrow 1, \ldots, G$ **do**
  $p \leftarrow p_{min}, P^g_{cand} \leftarrow \emptyset$;
  while $p \le p_{max}$ **do**
    $h(p) \leftarrow \lceil (2p^2/\epsilon^2) \ln(2k/\delta) \rceil$;
    Use the price p for $h(p)$ times and observe the acceptance ratio $\widehat{S^g}(p)$;
    $P^g_{cand} \leftarrow P^g_{cand} \cup \{(p, \widehat{S^g}(p))\}$;
    $p \leftarrow (1+\alpha)p$;
  $p^g_m \leftarrow \arg\max_{(p, \widehat{S^g}(p)) \in P^g_{cand}} p\widehat{S^g}(p)$;
return $p_b \leftarrow \sum_g p^g_m / G$

Myerson reserve price p^g_m falls outside of $[p_{min}, p_{max}]$, the algorithm returns p_{min} or p_{max}.

3.3 Algorithm Analysis

To estimate the Myerson reserve price for each grid g efficiently and effectively, we want to test the sample prices with a small number (*i.e.*, $h(p)$) of requesters while $\widehat{S^g}(p)$ is approximated with high probability. The efficiency and effectiveness of our algorithm is guaranteed by the following theorems. As the theorems are applicable for each grid g, we omit the superscript for ease of notation.

Theorem 2. *Let p' denote the best price among P_{cand},* i.e., $p' = \arg\max_{(p, \widehat{S}(p)) \in P_{cand}} pS(p)$. *With probability $1-\delta$, the base pricing algorithm finds p_m such that $p_m S(p_m) \ge p'S(p') - \epsilon$.*

We need to set ϵ to be small to get p' in P_{cand}. Yet an extremely small ϵ may lead to too many samples. Based on Theorem 2, ϵ can be set as $\alpha p_{min} \min_p S(p)$. This is because the absolute difference between the $pS(p)$ of two successive prices $|pS(p) - (1+\alpha)pS((1+\alpha)p)|$ is at least $\alpha p_{min} \min_p S(p)$, and we find price p such that $pS(p) \ge p'S(p') - \epsilon$, which ensures we find p'.

When we choose successive sample prices based on the multiplier $(1+\alpha)$, we have the following theorem that bounds the accuracy of our estimation compared with the optimal price (*i.e.*, Myerson reserve price) on the continuous interval.

Theorem 3. *Let p^* denote the optimal price on the continuous interval $[p_{min}, p_{max}]$. If $\alpha \in (0,1)$, we have $p_m S(p_m) \ge (1-\alpha)p^* S(p^*)$.*

Finally, we show the approximation guarantee when using a single base price p_b for all grids.

Theorem 4. *Let ALG represent the expected revenue achieved by using the base price p_b for all grids. Let OPT represent the maximum expected total revenue obtained by setting p^{*tg} for time period t and each grid g. Assume $Gp_{min} \ge p_{max}$. Then $ALG \ge \frac{1}{eG} OPT$.*

Complexity Analysis. It takes $O(|P_{cand}|)$ time to decide the price for each grid. Thus the time cost of base pricing is $O(G|P_{cand}|)$.

Taking the base price as initial inputs, we further propose a dynamic pricing strategy, which addresses changes from not only unknown demand, but also limited and dependent supply.

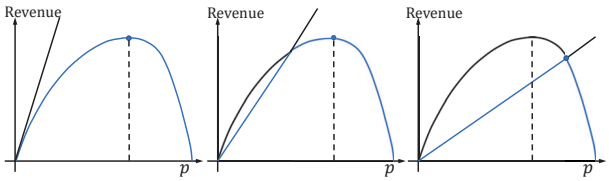


Figure 4: Expected revenue with limited supply.

4 DYNAMIC PRICING

The aim of *dynamic* pricing is to increase the profit by setting *diversified* prices for each grid based on the base price. In spatial crowdsourcing, each grid can be regarded as a *local market* where the demand (tasks) and the supply (workers) differ across grids. The supply of multiple grids can be dependent because *(i)* each worker can serve multiple grids yet *(ii)* each worker can only perform one task at a time. In this section, we present MAPS, a matching-based pricing strategy for spatial crowdsourcing to deal with the variety of supply-demand and the dependency of supply among grids. We elaborate on the basic idea, algorithm details and performance guarantee of MAPS in sequel.

4.1 Basic Idea

Unlike base pricing, which assumes *sufficient* supply, MAPS sets the optimal prices for grids with both *sufficient* and *limited* supply. Inspired by works in the economy [9, 22], we propose an efficient approximation of the expected revenue for grids with both sufficient and limited supply (Sec. 4). Based on the approximation, MAPS needs to estimate the unknown demand (*i.e.*, acceptance ratio of requesters) and determine supply to set the unit price for each grid that maximizes the expected revenue. MAPS adopts an upper confidence bound technique (UCB) [8] to boost the estimation of acceptance ratios (Sec. 4.2.2). Since the supply of multiple grids is dependent, MAPS incrementally optimizes the dependent supply in each grid and sets the prices efficiently to maximize the expected total revenue of all grids (Sec. 4.2.3).

4.2 Algorithm Details

We first present the approximation of expected revenue and the pricing framework for limited supply. Then we detail the MAPS scheme to estimate the demand and optimize the supply, respectively, which finally decides the unit price for tasks in each grid.

4.2.1 Pricing with Limited Supply. The base pricing strategy in Sec. 3 assumes sufficient supply, *i.e.*, after each requester accepts the price, there is always an available worker to perform the task. Yet it is common that the supply in certain grids is limited, *e.g.*, during rush hours at transportation hubs. We need an approximation of the expected revenue applicable to all supply situations.

In MAPS, we approximate the expected revenue using a demand curve and a supply curve with the price p^{tg} as the variable. The demand curve is determined by $\sum_{r \in R^{tg}} d_r p^{tg} S^g(p^{tg})$, where R^{tg} and $S^g(p^{tg})$ denote the set of tasks with their origins located in g and the acceptance ratio w.r.t. price p^{tg}, respectively. Without loss of generality, we assume the distances of tasks $d_{r_1} \ge d_{r_2} \ge \ldots d_{r_{|R^{tg}|}}$. The supply curve is calculated as $\sum_{i=1}^{n^{tg}} d_{r_i} p^{tg}$, *i.e.*, the sum of the top n^{tg} revenue $d_{r_i} p^{tg}$, where n^{tg} is the number of supply in grid g that we need to specify. The demand curve can be considered as the

expected revenue under the condition that the supply is sufficient. The supply curve represents the revenue that n^{tg} workers can yield at most. Given the demand and the supply curves, the expected revenue in time period t and grid g can be approximated as

$$L^g(n^{tg}, p^{tg}) = \min\Big(\sum_{r\in R^{tg}} d_r p^{tg} S^g(p^{tg}), \sum_{i=1}^{n^{tg}} d_{r_i} p^{tg}\Big). \quad (1)$$

The formal mathematical explanation of the approximation will be presented in the proof of Theorem 10.

Based on such an approximation, there can be three cases to derive the best unit prices, as illustrated in Fig. 4. In the first case, the supply curve has a large slope (*i.e.*, large amounts of workers specified to serve tasks in grid g), indicating sufficient supply. This is the case in Sec. 3 and the Myerson reserve price is the maximizer. In the second and the third cases, there is limited supply (*i.e.*, $n^{tg} < |R^{tg}|$). With a low unit price, there will be a shortage in supply. With a high unit price, requesters tend to reject the price, but the supply becomes sufficient. The second and the third cases differ in the price which maximizes $L^g(n^{tg}, p^{tg})$. Specifically, the Myerson reserve price is still the maximizer in the second case, while the price of the intersection applies in the third one. Note that these maximizers can be derived only when n^{tg} is specified.

Summary. Approximating the expected revenue in a grid using the demand and the supply curves reduces the overhead of searching for the optimal price (*i.e.*, enumerating all the possible bipartite graphs). Next we elaborate on how to *estimate the demand curve* and *adjust the supply curve* such that the approximate expected revenue of all grids $\sum_g L^g(n^{tg}, p^{tg})$ is maximized.

4.2.2 Boosting Acceptance Ratio Estimation. One prerequisite to set the price for each grid to maximize $\sum_g L^g(n^{tg}, p^{tg})$ is to estimate the acceptance ratio $S^g(p)$ in each grid g. In base pricing (Sec. 3), a sampling strategy is applied. In MAPS, we boost the estimation of the acceptance ratios via the upper confidence bound (UCB) [8], a technique for the multi-arm bandit (MAB) problem. UCB combines exploration (passively acquiring new information about $S^g(p)$) and exploitation (actively filtering prices p^{tg} that are unlikely to maximize the expected revenue given the current supply). Compared with the sampling process in base pricing, UCB adopts a different score function to choose the appropriate price, which only relies on a *rough* estimate of the acceptance ratios. It needs fewer samples to decide the best unit price for a grid and is more suitable when the acceptance ratios need frequent updating.

Mathematically, UCB is defined as the sample mean (for some probability) plus a confidence radius. In our context, instead of using the true acceptance ratio $S^g(p)$, we use $\widehat{S}(p) + \sqrt{\frac{2\ln N}{N(p)}}$ when setting a price given specific supply in a grid. As UCB is identical for each g, we omit the superscript. $\widehat{S}(p)$ is the sample mean. N is the number of requesters in g so far. $N(p)$ is the number of times we have used p in g. The radius $\sqrt{\frac{2\ln N}{N(p)}}$ is zero when $N(p)$ is zero.

As in base pricing, we still choose a price from a candidate set. Based on the UCB defined above, we set a numerical score (index) for each price and will choose the price with the largest index. The index, denoted by $\widetilde{I}(p)$, is $\min(p\widehat{S}(p) + p\sqrt{\frac{2\ln N}{N(p)}}, \frac{\sum_{i=1}^{n^{tg}} d_{r_i} p}{\sum_{r\in R^{tg}} d_r})$. It is designed such that after testing necessary number of requesters, we always find the price that maximizes $L^g(p^{tg})$ w.r.t. the current n^{tg} and the true probability $S^g(p)$ among all the candidate prices. We analyze the performance guarantee of the UCB-based acceptance ratio estimation in Sec. 4.3.1.

Note that the acceptance ratio of a grid may change over time and we need to notify this change. MAPS determines the change in the acceptance ratio $S^g(p)$ via the statistically-significant deviations [26]. Specifically, for some price, the number of accepted requesters in g follow the binomial distribution with the probability of $S^g(p)$ for each tested price. If there is no change in $S^g(p)$, the binomial random variable takes value around the expected value with high probability as long as there are enough samples. Thus we flag a change if the number of accepted requesters is not within $m\widehat{S}^g(p) \pm 2\sqrt{m\widehat{S}^g(p)(1-\widehat{S}^g(p))}$ for m requesters, where $\widehat{S}^g(p)$ is the acceptance ratio for the previous m requesters.

Summary. By replacing the estimation of the true acceptance ratio $S^g(p)$ with the UCB-based learning process, MAPS is able to get the optimal price given a specific supply accurately with a small number of sampling prices. The UCB-based strategy will be utilized as a building block to derive the maximum expected revenue of each grid given the supply (known) and the demand (unknown) of all grids, as will be detailed in Algorithm 3.

4.2.3 Optimizing Dependent Supply and Dynamic Pricing. Our goal is to maximize the approximate expected revenue of all grids $\sum_g L^g(n^{tg}, p^{tg})$, yet the supply in multiple grids can be dependent. Therefore, we propose to jointly adjust the slopes of the supply curves (see Fig. 4) for all grids to decide the appropriate n^{tg} and thus the corresponding p^{tg} that maximizes $\sum_g L^g(n^{tg}, p^{tg})$. Note that given a specific n^{tg}, there exists a unique p^{tg} maximizing $L^g(p^{tg})$, which will be found using the technique in Sec. 4.2.2. Eq. (1) can be rewritten as $\sum_g L^g(n^{tg})$. MAPS optimizes the dependent supply by first maintaining a bipartite graph which represents the range constraints and the dependency of supply, incrementally finding a worker that most increases $\sum_g L^g(n^{tg})$ without violating the bipartite graph, and deriving the corresponding price p^{tg} using the UCB techniques introduced in Sec. 4.2.2.

We initialize the number of supply in each grid n^{tg} as 0. In each iteration, we try to add one worker to each grid. If the addition for some g can be made, it corresponds to a match in the bipartite graph. We choose the match that introduces the largest increase in $\sum_g L^g(n^{tg})$ among all the grids. That is, each grid reports its increase if its n^{tg} can be added by 1, then we choose the grid with the largest increase, and truly add the corresponding n^{tg} by 1.

Another viewpoint is that for each g, we have the demand curve (*i.e.*, the first term in Eq. (1)), and we adjust the slope of the supply curve (*i.e.*, the second term in Eq. (1)), and calculate the increase in the expected total revenue. The increase in expected revenue for each g is the maximum value based on the new supply curve minus the one based on the old supply curve, which is the x-axis at first. The new slope of the supply curve is the sum of the top $n^{tg}+1$ distances and the old one is the sum of the top n^{tg} distances. At last, we will choose the grid with the largest increase to add one worker. Note that when the increase of some grid g becomes zero, we ignore g in the next iteration, because the slope can only be larger and the maximum will not change.

Algorithm 2: MAPS

input : $R^t, W^t, p_{min}, p_{max}, p_b$
output: P^t

1 Construct the bipartite graph $B(R^t, W^t)$;
2 Pre-matching $M' \leftarrow \emptyset$, Max-heap $H \leftarrow \emptyset$;
3 **for** $g \leftarrow 1, \ldots, G$ **do**
4 $\quad n^{tg} \leftarrow 0$ and insert $((g, 0, p_b), \infty)$ into H;
5 **while** H *is not empty* **do**
6 $\quad ((g, n^{tg}_{new}, p^{tg}_{new}), \Delta^g) \leftarrow$ the root of H;
7 $\quad$ remove the root from H;
8 $\quad$ **if** Δ^g *is not equal to* ∞ **then**
9 $\quad\quad n^{tg} \leftarrow n^{tg}_{new}$;
10 $\quad\quad$ Find an augmenting path for $r \in R^{tg}$ and add the match into M';
11 $\quad$ **if** Δ^g *is equal to 0* **then**
12 $\quad\quad p^{tg} \leftarrow p^{tg}_{new}$;
13 $\quad\quad$ **if** $p^{tg} > p_{max}$ **then**
14 $\quad\quad\quad p^{tg} \leftarrow p_{max}$;
15 $\quad$ **else**
16 $\quad\quad$ **if** $|R^{tg}|$ *is equal to 0 or there is not an augmenting path for any unassigned* $r \in R^{tg}$ **then**
17 $\quad\quad\quad$ insert $((g, n^{tg}_{new}, p^{tg}_{new}), 0)$ into H;
18 $\quad\quad$ **else**
19 $\quad\quad\quad n^{tg}_{new} \leftarrow n^{tg}_{new} + 1$;
20 $\quad\quad\quad$ calculate the p^{tg}_{new} and the new Δ^g using Algorithm 3;
21 $\quad\quad\quad$ insert $((g, n^{tg}_{new}, p^{tg}_{new}), \Delta^g)$ into H;
22 **return** $P^t \leftarrow \{p^{t1}, \ldots, p^{tG}\}$

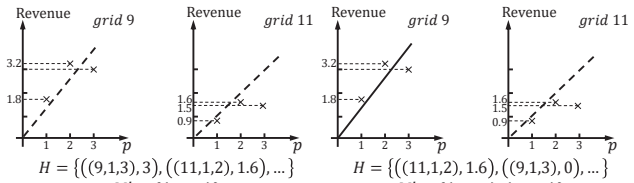
Algorithm 2 illustrates the main procedure. We use the base price p_b calculated by Algorithm 1 as the input. In line 1, we construct the bipartite graph according to the sets of tasks and workers by satisfying the range constraint. In line 2, we initialize the pre-matching M', which is used when we test if a new worker can be assigned to a task of each g (*i.e.*, the supply of g). The max-heap consists of tuples, each with a triple $(g, n^{tg}_{new}, p^{tg}_{new})$. The second term records the new number of supply for g, and the third term is the corresponding price. We use Δ^g to denote the key value of the tripe, which represents the increase in $L^g(n^{tg})$. The max-heap can output the tuple $(g, n^{tg}_{new}, p^{tg}_{new})$ with the largest Δ^g in $O(logG)$ time. In all but the iteration we finally set the price for each grid, the number of elements in H is always G. In line 4, we insert the tuple of each grid into the heap. Their keys are set to positive infinity so that we can update the key in the first G iterations. In lines 6-7, we take and remove the tuple with the largest Δ^g. Line 8 is used to avoid that the initialization of line 4 (where Δ^g is set to ∞) directly finds a match in the first G iterations. This case should be avoided because we have not updated the key using the true increase. In lines 9-10, given the grid g with the largest Δ^g, we admit this match and update the pre-matching M'. In line 11, if the largest Δ^g among all the grids is zero, we then set the final price for each grid g in line 12 or line 14 without inserting any new tuple into H. In lines 16-21, we try to add a new worker to grid g and calculate their increases in L^g, *i.e.*, Δ^g. Specifically in lines 16-17, if we cannot assign a task r in grid g to this worker, we set the increase to zero. Otherwise in lines 19-21, we get the maximum increase Δ^g and the corresponding new price p^{tg}_{new} using Algorithm 3, and insert the tuple for further comparisons. Note that when calling Algorithm 3, the latest statistics stored in P is used to set an optimal price. So the algorithm does not need to contact new requesters and wait for their responses. Algorithm 3 illustrates the procedure to calculate the maximizer discussed in Sec. 4.2.2. All the statistics of g are stored in P. We update $\widehat{S}(p)$ and $N(p)$ when receiving feedbacks of the current prices (which are set in lines 12-14 of Algorithm 2). In lines 2-3, we restore the number of requests shown so far in g. In lines 5-9, we iterate prices from big to small and choose the price with the maximum index (defined in Sec. 4.2.2).

Algorithm 3: Calculating the Maximizer

input : $g, R^{tg}, n^{tg}, P, p_{max}, p_{min}, \alpha$
output: p_{new}, Δ^g

1 $C \leftarrow \sum_{r \in R^{tg}} d_r p^{tg}, D \leftarrow \sum_{i=1}^{n^{tg}} d_{r_i} p^{tg}, N \leftarrow 0, \widetilde{I}_{new} \leftarrow 0, p \leftarrow p_{max}$;
2 **foreach** $(p, \widehat{S}(p), N(p)) \in P$ **do**
3 $\quad N \leftarrow N + N(p)$;
4 **while** $p \geq p_{min}$ **do**
5 $\quad c(p) \leftarrow p\sqrt{\frac{2 \ln N}{N(p)}}$;
6 $\quad$ **if** $\widetilde{I}_{new} < \min(p\widehat{S}(p) + c(p), \frac{D}{C}p))$ **then**
7 $\quad\quad \widetilde{I}_{new} \leftarrow \min(p\widehat{S}(p) + c(p), \frac{D}{C}p)$;
8 $\quad\quad p_{new} \leftarrow p$;
9 $\quad p \leftarrow p/(1+\alpha)$;
10 **return** $p_{new}, \Delta^g \leftarrow p_{new}\widehat{S}(p_{new}) - p_{old}\widehat{S}(p_{old})$

Example 5. *Back to our running example in Example 1. For simplicity, we assume we have obtained the statistics about the acceptance ratios as in Table 1. There are 16 grids in Example 1, where* r_1 *and* r_2 *are in grid 9, and* r_3 *is in grid 11. The bipartite graph is shown in Fig. 1b. For grid 9,* $\sum_{r \in R^{t9}} d_r p S^9(p)$ *is demonstrated by the three crosses in the first sub-figure in Fig. 5a.* $\sum_{r \in R^{t11}} d_r p S^{11}(p)$ *for grid 11 is shown in the second sub-figure in Fig. 5a. At first,* n^{t9} *and* n^{t11} *are both 0. Grid 9 reports its increase if* $n^{t9} = 1$*, which is 3. It can be also viewed as the maximum of the minor one of the line and the discretized demand curve (*i.e.*, the three crosses). The increase for grid 11 is 1.6. After the first 16 iterations, H is updated with the increase of each grid, as shown in the bottom-left in Fig. 5a. The omitted tuples belong to grids without any task and the increases are set to be 0 in lines 16-17 of Algorithm 2. Their prices are all set to the base price. In the 17th iteration, we take the root* $((9, 1, 3), 3)$ *from H in line 6 of Algorithm 2, admit its increase and find an augmenting path for* r_1 *in grid 9 in line 10.* M' *will be updated to* $\{r_1, w_1\}$*. Since* Δ^g *is not equal to 0, the algorithm goes to line 16. There is no augmenting path for* r_2 *in grid 9. So we insert* $((9, 1, 3), 0)$ *into H. And the dash line for grid 9 will become solid, meaning we admit that its increase is the largest one. At the beginning of 18th iteration, as showed in Fig. 5b, the root of H is* $((11, 1, 2), 1.6)$*. We admit its increase and update* M'



(a) 17th iteration **(b) 18th iteration**

Figure 5: An example of dynamic pricing.

accordingly. In the end, all the Δ^g is zero. We gradually remove tuples from H in line 7 and price the grids in line 12. The price for grid 9 is 3 and the price for grid 11 is 2, leading to a greater expected revenue.

Summary. MAPS dynamically sets the prices for each grid to maximize the expected total revenue by incrementally adding workers to the grid that induces the maximum increase in Eq. (1). MAPS applies concepts of bipartite graph matching to maintain the range constraints of supply and demand as well as the dependency of supply. Together with the approximation of the expected revenue and the UCB-based acceptance ratio learning procedure, MAPS is able to approach the optimal prices *(i)* without the need to enumerating all possible worlds in an uncertain graph, *(ii)* without accurately estimating the acceptance ratio beforehand and *(iii)* with considerations for both sufficient and limited supply.

We also make two practical notes. *(i)* MAPS tends to set a higher unit price for regions where workers are insufficient. This will motivate more drivers to move to these regions, as with many incentive mechanisms. *(ii)* A cap on the unit prices can be setting bounded prices. Spatial smoothing can also be integrated to reduce the gap of unit prices among neighbouring grids.

4.3 Algorithm Analysis

4.3.1 Analysis of UCB-based Acceptance Ratio Estimation. UCB ensures that after a small number of tests, we always choose the best possible price. Let p^* denote the optimal price in the continuous interval $[p_{min}, p_{max}]$ w.r.t $S(p)$ and the given n^{tg}, *i.e.*, $p^* = \arg\max_{p\in[p_{min},p_{max}]} L^g(p)$. Note that p^* must be in some interval with bounds that are two successive prices in the candidate set. Let p' and p'' denote the prices such that $p' \le p^* \le p''$. Let p_0 denote the larger price where the line and the curve intersect, *i.e.*, $p_0 S(p_0) = Dp_0/C$ and $p_0 > 0$. If it is the first case (*i.e.*, $n^{tg} \ge |R^{tg}|$) in Fig. 4, let $p_0 = 0$. We have the following theorem:

Theorem 5. *Provided that $N(p) > \frac{8p^2 \ln N}{(\max(p'S(p'),p''S(p''))-pS(p))^2}$ for any $p > p_0$ except p' and p'', with probability at least $1-O(N^{-4})$, we choose p' or p''.*

The requirements for $N(p)$ is inversely proportional to the square of p's badness, *i.e.*, $\max(p'S(p'),p''S(p'')) - pS(p)$. Meanwhile, $\ln N$ term ensures the quick convergence to the best possible choice.

The proof mainly uses two useful properties of the designed index, concluded with two lemmas below.

Lemma 6. *For any p, the probability that $pS(p) - c(p) < p\widehat{S}(p) < pS(p) + c(p)$ is at least $1 - 2N^{-4}$.*

Lemma 7. *If $N(p) > \frac{8p^2 \ln N}{(\max(p'S(p'),p''S(p''))-pS(p))^2}$ for any $p > p_0$ except p' and p'', the probability that $pS(p) < p\widehat{S}(p) + c(p) < \max(p'S(p'),p''S(p''))$ is at least $1 - 2N^{-4}$.*

Table 3: Synthetic datasets.

Factor	Setting
$\|W\|$	1250, 2500, **5k**, 7500, 10k
$\|R\|$	5000, 10k, **20k**, 30k, 40k
μ for temporal distribution	0.1, 0.3, **0.5**, 0.7, 0.9
$mean$ for spatial distribution	0.1, 0.3, **0.5**, 0.7, 0.9
μ for demand distribution	1.0, 1.5, **2.0**, 2.5, 3.0
σ for demand distribution	0.5, **1.0**, 1.5, 2.0, 2.5
T	200, **400**, 600, 800, 1000
G	5×5, 10×10, **15×15**, 20×20, 25×25
a_w	5, **10**, 15, 20, 25
Scalability	$\|W\| = \|R\|$ = 100k,200k, 300k,400k,500k

Since these inequalities all hold with high probability when the conditions are satisfied, we may state them as deterministic ones in the proof.

4.3.2 Analysis of MAPS. We will present bounds on the performance of MAPS.

Theorem 8. *MAPS always finds a feasible plan n'^t and prices p'^t such that $\sum_g L^g(n'^{tg}, p'^{tg})$ is at least $(1-\frac{1}{e})\max_{n^t,p^t} \sum_g L^g(n^{tg}, p^{tg})$.*

We first show Δ^g is decreasing with respect to g. Then we prove the submodularity of L, which is redefined on the set of workers. In each iteration we choose the maximum Δ^g among all the grids, and this gives us a near optimal solution.

Lemma 9. *For a fixed grid g, its Δ^g are decreasing each time inserted to the heap in the line 21 of Algorithm 2.*

Theorem 10. *Assume $(1-S^g(p^{tg}))\sum_{i=1}^{n^{tg}} d_{r_i} p^{tg} = O(d_{max}\sqrt{n^{tg}\log m})$, $ALG \ge (1-\frac{1}{e})OPT - O(d_{max}\sqrt{m\log m})$, where $d_{max} = \max_r d_r$, and $m = \max_g |R^{tg}|$.*

The assumption ensures that the distances in each grid do not vary too much, which often holds in practice. MAPS still functions without the assumption of i.i.d. private valuations. However, the performance guarantees might no longer hold. The i.i.d. assumption is mild because *(i)* the spatiotemporal factors that determine the private valuations in the same grid and time period are similar; and *(ii)* each individual requester tends to make decisions independently.

Complexity Analysis. Initializing the max-heap costs $O(GlogG)$. For each iteration in the while loop, we take the root in $O(logG)$ and find an augmenting path in $O(|E^t|)$, where $|E^t|$ denotes the number of edges in the bipartite graph. There will be at most $\min(|R^t|, |W^t|)$ iterations. To calculate the next increase, Algorithm 3 takes $O(|P_{cand}|)$ time to iterate through the prices. The total time will be $O(GlogG + \min(|R^t|, |W^t|)(logG + |E^t|)|P_{cand}|)$.

5 EXPERIMENTAL STUDY

This section presents the performance of MAPS on both synthetic and real-world datasets.

5.1 Experiment Setup

Synthetic datasets. Table 3 shows the parameters of the synthetic datasets. Default settings are marked in bold. All locations are generated within a square of 100 × 100 in the 2D coordinates. A time period is 1 minute long. The start times of tasks and workers are drawn from a normal distribution conditioned on the entire

Table 4: Real datasets.

	Duration	$\|W\|$	$\|R\|$	T	G	δ_w	a_w
#1	5pm-7pm	28210	113372	120	80	[5,10,15,20,25]	3km
#2	0am-2am	19006	55659				

time span of interest, which we call the *temporal distribution*. We vary the mean of the temporal distribution from 0.1 to 0.9. In each time period, the origins of tasks and workers are generated from a two-dimensional Gaussian distribution, which we call the *spatial distribution*. The mean of the spatial distribution is calculated by multiplying the values in Table 3 with a two-dimensional vector $(100, 100)$. The destinations of tasks are drawn from a uniform distribution within the 100×100 square. Note that varying the mean and the variance of the temporal and spatial distributions has similar impact, *i.e.*, affects the structure of bipartite graph. Hence we omit the experiments on varying the variance of these two distributions. We simulate the demand distribution via a normal distribution with its mean varying from 1 to 3. Then the valuations v_r are drawn from each normal distribution w.r.t. the mean of g. We restrict all the v_r to $[1, 5]$, so the distribution of v_r is a conditional probability distribution. We also experiment with other demand distributions such as an exponential distribution. The results are similar to those using a normally distributed demand (see Sec. D for more details). We vary the radius a_w of workers from 5 to 25. We also vary the number of workers $|W|$, the number of tasks $|R|$, the number of time periods T, and the number of grids G.

Real datasets. We use the taxi-calling data sampled from July 2016 to December 2016 in Beijing collected by a large-scale online taxi-calling platform in China. The data are sampled from two representative time of the day, *i.e.*, 5 p.m. to 7 p.m., and 0 a.m. to 2 a.m., when there are usually huge demand and light demand of taxis, respectively. The length of a time period is set to 60 second. We consider locations within a rectangle with the bottom-left coordinate of (116.30, 39.84) and the top-right coordinate of (116.50, 40.0). Each grid covers an area of 0.02 longitudes by 0.02 latitudes and there are $10 \times 8 = 80$ grids in total. The radius a_w for each worker is $3km$. Note that for the real dataset, we cannot obtain the exact valuations v_r. However, the valuation v_r should be higher than the price if the requester accepts the price, and such information can be obtained from the historical records. So we set v_r to be a random value greater than the set price, and vice versa. Since all the parameters are fixed, we test the pricing strategies varying the available duration of each worker.

Compared algorithms. We compare our MAPS pricing scheme with the following algorithms.

(1) BaseP. It is the strategy proposed in Sec. 3, which assumes the unlimited supply and sets the same base price p_b for all grids.

(2) SDR. It sets the price for a grid as the inverse of the supply-demand ration in the grid times a coefficient. We empirically optimize the coefficient on our datasets. SDR sets the price for a given time period t and a grid g as $0.5p_b|R^{tg}|/|W^{tg}|$ if $|R^{tg}| > |W^{tg}|$, and the base price p_b otherwise.

(3) SDE. It prices tasks via the supply-demand difference in exponential function. Specifically, SDE sets the price for a given time period t and a grid g as $p_b(1 + 2e^{|W^{tg}|-|R^{tg}|})$ if $|R^{tg}| > |W^{tg}|$, and the base price p_b otherwise.

(4) CappedUCB. It is the state-of-the-art pricing strategy proposed by [9] to tackle the problem of limited supply in just one market. We regard each grid as one single market and independently decides the price of each grid. Specifically, the price for grid g is $\arg\max_{p^g} \min(|R^{tg}|p^g S^g(p^g), |W^{tg}|p^g)$, equivalent to our Eq. (1) when $n^{tg} = |W^{tg}|$ and each $d_r = 1$.

Metrics and implementation. We assess the performance of the pricing strategies in terms of the output total revenue, running time, and memory cost. All the algorithms are implemented in C++ and the experiments are performed with Intel (R) Core (TM) i7 3.80GHz CPU and 4GB main memory.

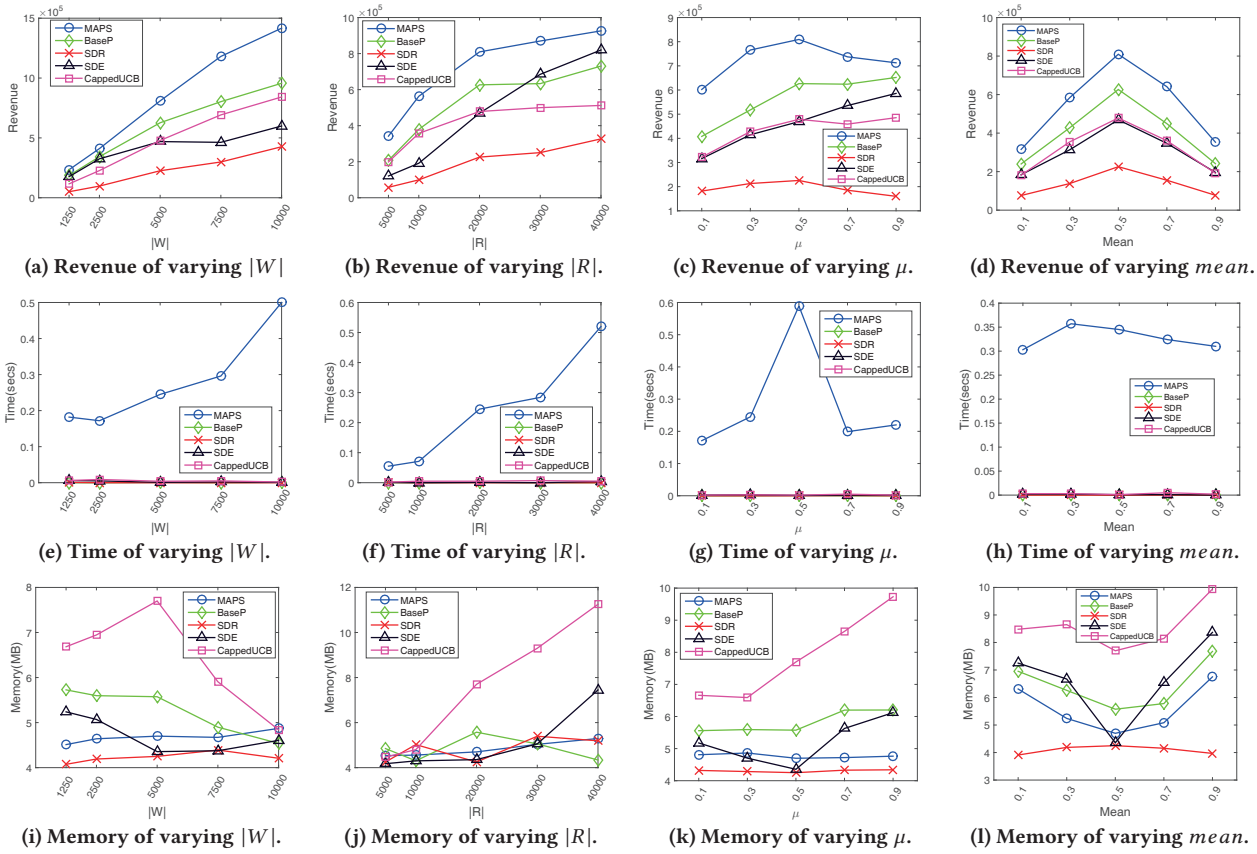
5.2 Experiment Results

We first present the results on the synthetic dataset with various parameters, and then show the performance on the real datasets.

Effect of $|W|$. The first column of Fig. 6 shows the results of varying the number of workers $|W|$. As $|W|$ increases from 1250 to 10000, the revenues of all the pricing strategies increase, because of the increasing amount of the finished requests,*i.e.*, the supply gradually matches the demand. Among the five strategies, MAPS yields the highest revenue. Base pricing outperforms the other three baselines, because it may already ensure the maximum expected revenue for each task, which is optimal in grids with sufficient workers. CappedUCB performs badly because it does not consider the grids globally. As for running time, strategies except MAPS take constant time. This is because MAPS needs to output a matching result, and with the increase of $|W|$, the calculation of the matching takes more time. However, its running time is acceptable, as the running time spans all the T time periods, and is negligible compared with the length of a time period. CappedUCB consumes the most memory and that of other four strategies is similar. The reason might be that CappedUCB needs to store more information such as the number of tasks and workers in each grid. The costs of all the five strategies are less than $10M$.

Effect of $|R|$. The second column of Fig. 6 presents the results of varying the number of requesters $|R|$. When $|R|$ increases, all strategies output a larger revenue, since there are more requests that can be performed. When $|R|$ is greater than 20000, the growth stabilizes. This is because the total number of workers is fixed, and it gradually becomes difficult to increase the revenue. Again MAPS achieves the highest revenue. MAPS costs the most running time because of the calculation of the matching result, and takes acceptable memory consumption. The other four strategies take constant running time. CappedUCB still costs the most memory.

Effect of μ of the temporal distribution for requests. The third column of Fig. 6 shows the effect of the mean of the temporal distribution for requests. The mean for the workers is fixed at $T/2$. As the mean for requests approaches 0.5 (*i.e.*, $T/2$), there is an increase of the revenues in all the strategies but SDE. This is because the mean values of tasks and workers gradually become close, and the tasks and workers overlap more in time. Hence more workers will satisfy the range constraint, contributing to more edges in the bipartite graph and thus potentially a larger revenue. MAPS is still the most effective one among the five pricing strategies. For running time, when μ approaches 0.5, the time cost of MAPS gets larger. This is because there are more edges in the bipartite graph. Yet the time cost is still small when considering the T time periods. For memory consumption, all consume less than 10M memory, and CappedUCB consumes the most.



(a) Revenue of varying $|W|$ (b) Revenue of varying $|R|$. (c) Revenue of varying μ. (d) Revenue of varying $mean$.

(e) Time of varying $|W|$. (f) Time of varying $|R|$. (g) Time of varying μ. (h) Time of varying $mean$.

(i) Memory of varying $|W|$. (j) Memory of varying $|R|$. (k) Memory of varying μ. (l) Memory of varying $mean$.

Figure 6: Results on varying $|W|$, $|R|$, μ in temporal distribution, and $mean$ in spatial distribution.

Effect of *mean* of the spatial distribution for requests.The fourth column of Fig. 6 shows the results of varying the mean of the spatial distribution. The mean of the spatial distribution is two-dimensional and the x-axis represents the mean on the diagonal. For example, the value of 0.1 on the x-axis means a vector (10, 10). Similar to the results when varying the mean of the temporal distribution, the revenue of all the strategies increases when the mean of the tasks' origin is close to that of the workers' origin. MAPS yields the largest revenue and costs reasonable time and memory.

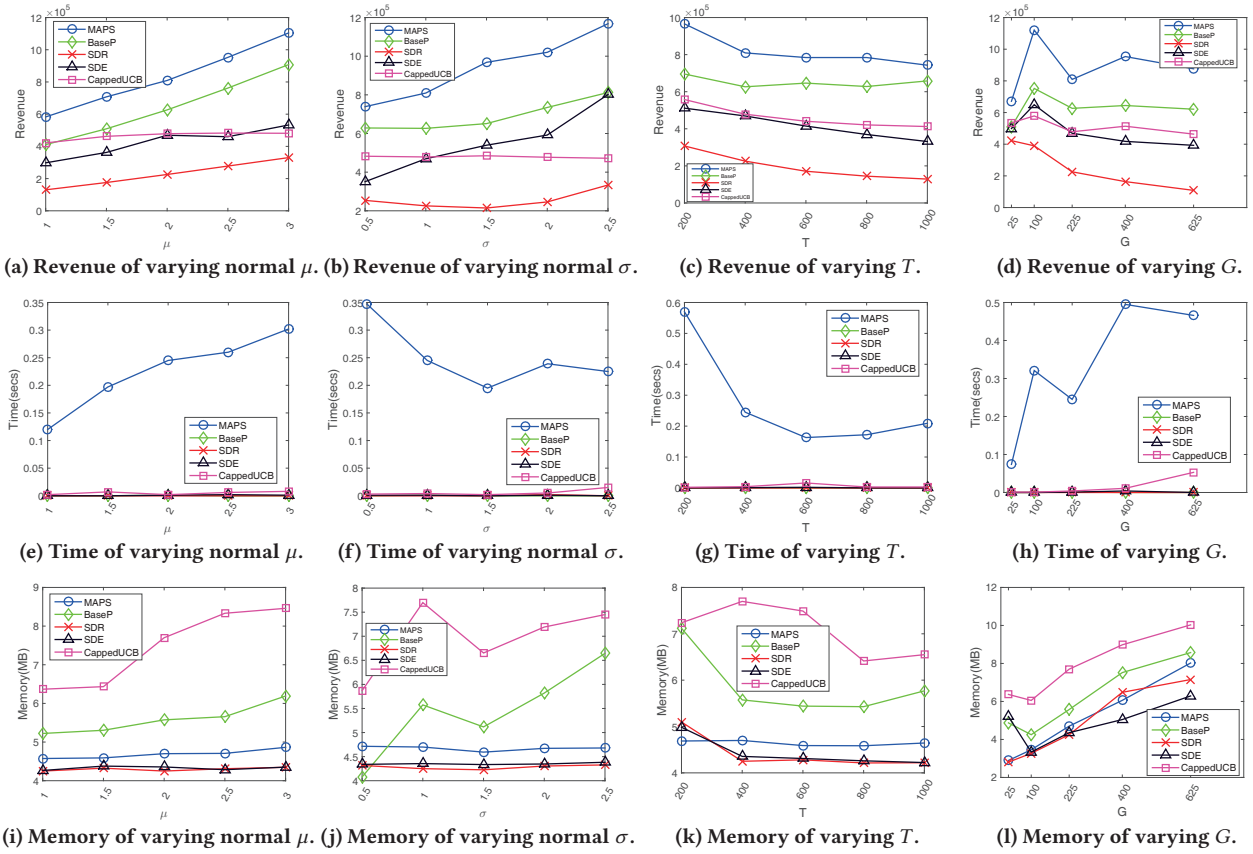
Effect of μ of the demand distribution for v_r. The first column of Fig. 7 shows the results with different mean values of the demand distribution. As the requesters' valuations increases (*i.e.*, they are willing to accept higher prices), the revenue also increases. For different means of the distribution, MAPS always achieves the highest revenue, validating the effectiveness of the UCB technique. MAPS costs more time when μ gets larger, because the acceptance ratios for some price increase and there is a need to assign more workers to finish the accepted requests. MAPS is still efficient when considering the average running time of each time period. For memory consumption, all strategies cost reasonable memory.

Effect of σ of the demand distribution for v_r. The second column of Fig. 7 presents the results of varying σ of the demand distribution. Note that the demand distribution is conditioned on the interval $[1, 5]$. If the mean of the distribution used to simulate the normal distribution is fixed at 2 (by default), the actual mean values are bigger as σ increases. Hence, the revenues of all the strategies increase. For running time and memory consumption, their fluctuations are normal and acceptable.

Effect of T. The third column of Fig. 7 shows the result of varying T. Since all the strategies optimize in each time period, technically we can observe the best performance if they are all in one time period. Hence, when the numbers of tasks and workers are fixed, these numbers in each time period decrease as T increases, and the revenues of all the strategies slightly decrease because the optimization becomes weaker. In general, the running time of MAPS decreases with the increasing of T. The reason might be that the numbers of tasks and workers in each time period decrease, making the calculation of the matching result easier.

Effect of G. The fourth column of Fig. 7 shows the results of varying the number of grids G. When G increases, each grid becomes smaller in size, because the size of the entire region of interest is fixed. Consequently, we can perform finer-grained optimization. The revenues first increase with G. G cannot be arbitrarily large, otherwise the assumption that the valuations in one grid are i.i.d samples may gradually become invalid, leading to inaccurate estimation of the acceptance ratios. Thus, when G is greater than 100, the revenues do not increase. When G increases, all strategies consume more memory because they need to store more data. MAPS performs the best and costs acceptable running time and memory.

Effect of a_w. The first column of Fig. 8 shows the results with different a_w, which determines the edge set of the bipartite graph. More edges result in more total revenue. MAPS is still the best in terms of revenue. Yet since the means of the spatial distributions



(a) Revenue of varying normal μ. (b) Revenue of varying normal σ. (c) Revenue of varying T. (d) Revenue of varying G.

(e) Time of varying normal μ. (f) Time of varying normal σ. (g) Time of varying T. (h) Time of varying G.

(i) Memory of varying normal μ. (j) Memory of varying normal σ. (k) Memory of varying T. (l) Memory of varying G.

Figure 7: Results on varying normal μ, normal σ, the number of time periods T, and the number of grids G.

of ori_r and l_w are close to each other, the number of edges may stop increasing when the radius reaches some value, leading to the revenues becoming stable when a_w is greater than certain value. With the increase of a_w, the time consumption of MAPS grows, as the bipartite graph has more edges. The memory consumption of the five strategies fluctuates within an acceptable range.

Scalability. The second column of Fig. 8 plots the results by increasing $|R|$ and $|W|$ simultaneously at the same scale. The running time of MAPS increases linearly and the other four strategies still take constant time. Note that we record the total running time of 400 time periods, and the average running time of MAPS in each time period is actually low. The memory costs of all strategies grow almost linearly and are acceptable.

Real datasets. The last two columns in Fig. 8 present the performance on the real-world datasets. The duration of workers is another factor in supply. As workers are available for more time periods, the quantity of supply increases. As we state in analyzing the results of varying $|R|$, there exists a limit on the revenue since the number of tasks is fixed. The revenues of all the strategies become stable when the duration becomes long. CappedUCB performs better than BaseP in the second dataset. This may be because the supply is more limited in this dataset, where CappedUCB still functions, while BaseP does not. MAPS is the best in effectiveness on both datasets. Similar to the results in the scalability test, all the strategies prove to be time-efficient and memory-efficient.

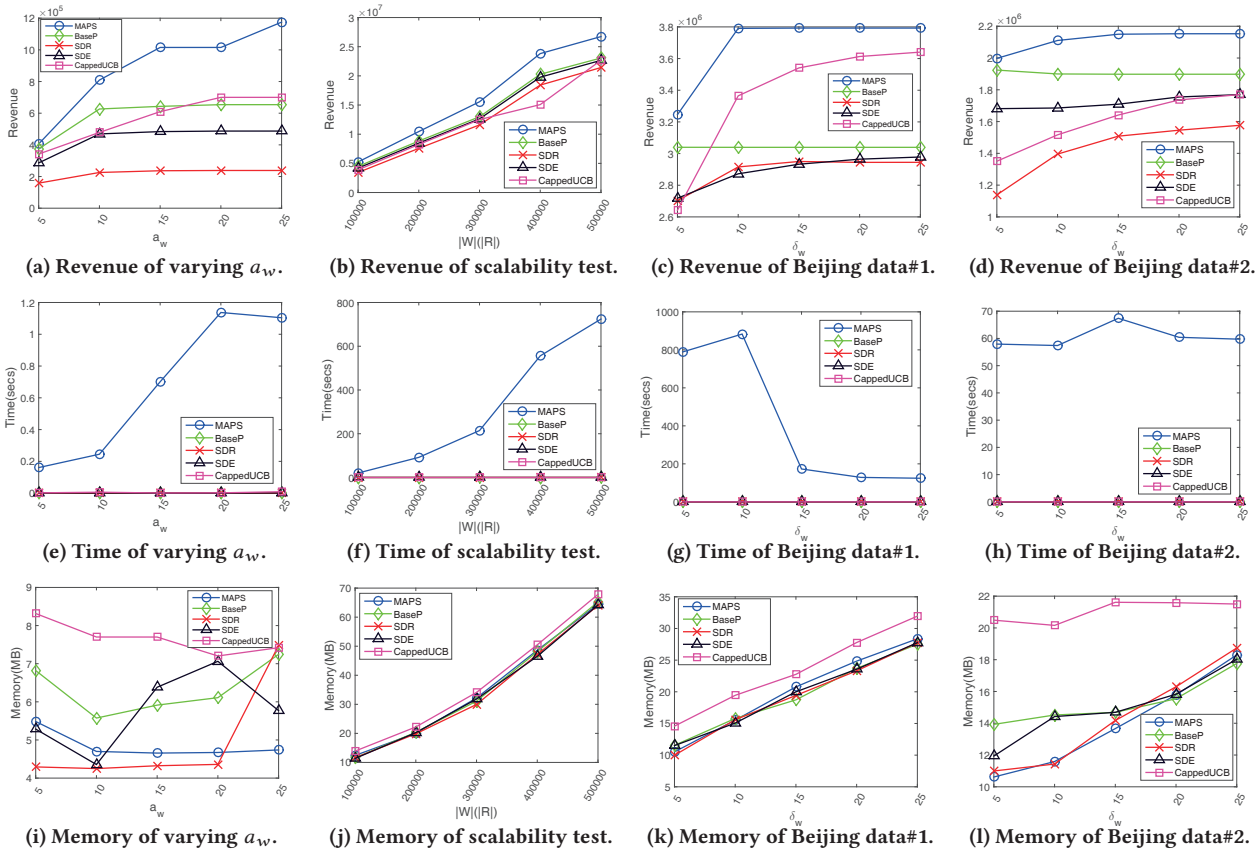
Summary of experimental results. *(i)* MAPS achieves the largest revenue in both real-world and synthetic datasets. Base pricing strategy outperforms some heuristics if we choose the base price appropriately. *(ii)* All the pricing strategies have acceptable time and memory costs. *(iii)* MAPS is scalable in time and space on the dataset of size of the order 10^6.

6 RELATED WORK

6.1 Pricing in Spatial Crowdsourcing

Tasks in spatial crowdsourcing can be processed in a batch at the end of one time period [20, 21, 27, 39, 44], or be immediately assigned [38, 41–43]. We focus on the former mode. Some pioneer works have considered the effect of prices in spatial applications [7, 32]. In [7], prices are obtained based on the profiles of workers and requesters, which are used as inputs to find a matching between workers and requests to maximize the revenue. However, we aim to optimize the prices to maximize the expected total revenue. In [32], an incentive mechanism in spatial crowdsourcing is proposed. However, it differs from our work in three-fold: *(i)* [32] is designed for crowdsensing, a special case of crowdsourcing, where a task is to collect data at a location. We focus on the more generic spatial crowdsourcing applications. *(ii)* The objective of [32] is to maximize the data quality, while we aim to maximize the expected total revenue. *(iii)* The number of workers in [32] is fixed, while we optimize the number of workers in each grid.

Dynamic pricing has been recently introduced in some spatial crowdsourcing companies such as Uber [3]. They prices tasks by considering the total number of drivers and requesters in a region



(a) Revenue of varying a_w. (b) Revenue of scalability test. (c) Revenue of Beijing data#1. (d) Revenue of Beijing data#2.

(e) Time of varying a_w. (f) Time of scalability test. (g) Time of Beijing data#1. (h) Time of Beijing data#2.

(i) Memory of varying a_w. (j) Memory of scalability test. (k) Memory of Beijing data#1. (l) Memory of Beijing data#2.

Figure 8: Results on varying the radius of workers a_w, scalability test, and real data.

during a time period. However, such a pricing strategy neglects the fact that a driver may serve requesters in multiple grids. By jointly optimizing the supply in multiple dependent grids, our solution tends to set prices that yield a larger expected total revenue.

6.2 Dynamic Price Mechanism Design

Online posted price mechanisms assume there is only one interaction between requesters and the platform; the platform declares its price and the requester decides whether to accept the price or not. Researchers have proposed competitive pricing strategies with fixed supply [9, 13, 29], which are inapplicable in our problem.

In the resource allocation markets [23, 28], requesters have preferences for different workers, which can be expressed by a bipartite graph. The tenet is to find the equilibrium prices or market clearing prices (*i.e.*, the demand equals supply), which is a different objective.

Our work is also related to revenue maximization in operation research [12]. Some works [35] assume nonparametric demand functions such as MHR distributions. Others [9, 22, 37] design techniques such as multi-arm bandit approaches to learn the demand function with limited supply. Our work adopts the parametric ones, which admit that the parameters (*e.g.*, the price in our problem) in the structure of the demand function. However, previous works only assume a single market with number of supply fixed, while we consider multiple markets with the number of supply adjustable.

7 CONCLUSION

In this paper, we propose the *Global Dynamic Pricing* (GDP) problem for spatial crowdsourcing , which is challenging due to *(i)* unknown acceptance probabilities of a given price for each task, *(ii)* limited and *(iii)* dependent supply among multiple grids. To solve the GDP problem, we first present a base pricing strategy that sets a unified base price by estimating the acceptance probabilities. We further develop MAPS, a matching based dynamic pricing strategy that optimizes the dependent supply and is able to approximately set the optimal prices for markets with limited supply and unknown acceptance probabilities of task requesters. We show through extensive evaluations that MAPS is both effective and efficient.

ACKNOWLEDGEMENT

We are grateful to anonymous reviewers for their constructive comments on this work. Yongxin Tong, Libin Wang and Bowen Du's works are partially supported by the National Science Foundation of China (NSFC) under Grant No. 61502021 and 61532004, National Grand Fundamental Research 973 Program of China under Grant 2014CB340300, the Base Construction and Training Program Foundation for the Talents of Beijing under Grant No. Z171100003217092, and the Science and Technology Major Project of Beijing under Grant No. Z171100005117001. Lei Chen's work is partially supported by the Hong Kong RGC GRF Project 16207617, the National Science Foundation of China (NSFC) under Grant No. 61729201, Science and Technology Planning Project of Guangdong Province, China, No. 2015B010110006, Webank Collaboration Research Project, and Microsoft Research Asia Collaborative Research Grant. Zimu Zhou and Bowen Du are the corresponding authors of this paper.

REFERENCES

[1] 1999. *Seamless.* https://www.seamless.com/.
[2] 2006. *OpenStreetMap.* https://www.openstreetmap.org.
[3] 2009. *Uber.* https://www.uber.com/.
[4] 2011. *Gigwalk.* http://www.gigwalk.com.
[5] 2012. *Didi Chuxing.* http://didichuxing.com/.
[6] 2013. *Waze.* https://www.waze.com/.
[7] Mohammad Asghari, Dingxiong Deng, Cyrus Shahabi, Ugur Demiryurek, and Yaguang Li. 2016. Price-aware Real-time Ride-sharing at Scale: An Auction-based Approach. In *GIS 2016.* 3:1–3:10.
[8] Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. 2002. Finite-time Analysis of the Multiarmed Bandit Problem. *Machine Learning* 47, 2-3 (2002), 235–256.
[9] Moshe Babaioff, Shaddin Dughmi, Robert D. Kleinberg, and Aleksandrs Slivkins. 2011. Dynamic Pricing with Limited Supply. *ACM Transactions on Economics and Computation* 3, 1 (2011), 4:1–4:26.
[10] Moshe Babaioff, Shaddin Dughmi, and Alex Slivkins. 2011. Detail-free, Posted-Price Mechanisms for Limited Supply Online Auctions. In *Workshop on Bayesian Mechanism Design 2011.*
[11] Richard E. Barlow, Albert W. Marshall, and Frank Proschan. 1963. Properties of Probability Distributions with Monotone Hazard Rate. *The Annals of Mathematical Statistics* (1963), 375–389.
[12] Omar Besbes and Assaf J. Zeevi. 2009. Dynamic Pricing Without Knowing the Demand Function: Risk Bounds and Near-Optimal Algorithms. *Operations Research* 57, 6 (2009), 1407–1420.
[13] Avrim Blum, Vijay Kumar, Atri Rudra, and Felix Wu. 2003. Online Learning in Online Auctions. In *SODA 2003.* 202–204.
[14] Gruia Călinescu, Chandra Chekuri, Martin Pál, and Jan Vondrák. 2007. Maximizing a Submodular Set Function Subject to a Matroid Constraint. In *IPCO 2007.* 182–196.
[15] Lei Chen and Cyrus Shahabi. 2016. Spatial Crowdsourcing: Challenges and Opportunities. *IEEE Data Engineering Bulletin* 39, 4 (2016), 14–25.
[16] Zhao Chen, Rui Fu, Ziyuan Zhao, Zheng Liu, Leihao Xia, Lei Chen, Peng Cheng, Caleb Chen Cao, Yongxin Tong, and Chen Jason Zhang. 2014. gMission: A General Spatial Crowdsourcing Platform. *PVLDB* 7, 13 (2014), 1629–1632.
[17] Anand Inasu Chittilappilly, Lei Chen, and Sihem Amer-Yahia. 2016. A Survey of General-Purpose Crowdsourcing Techniques. *IEEE Transactions on Knowledge and Data Engineering* 28, 9 (2016), 2246–2266.
[18] Fan R. K. Chung and Lincoln Lu. 2006. Survey: Concentration Inequalities and Martingale Inequalities: A Survey. *Internet Mathematics* 3, 1 (2006), 79–127.
[19] Nilesh N. Dalvi and Dan Suciu. 2007. Management of Probabilistic Data: Foundations and Challenges. In *PODS 2007.* 1–12.
[20] Dingxiong Deng, Cyrus Shahabi, and Ugur Demiryurek. 2013. Maximizing the Number of Worker's Self-selected Tasks in Spatial Crowdsourcing. In *GIS 2013.* 324–333.
[21] Dingxiong Deng, Cyrus Shahabi, and Linhong Zhu. 2015. Task Matching and Scheduling for Multiple Workers in Spatial Crowdsourcing. In *GIS 2015.* 21:1–21:10.
[22] Nikhil R. Devanur and Jason D. Hartline. 2009. Limited and Online Supply and the Bayesian Foundations of Prior-free Mechanism Design. In *EC 2009.* 41–50.
[23] Nikhil R. Devanur, Christos H. Papadimitriou, Amin Saberi, and Vijay V. Vazirani. 2008. Market Equilibrium via a Primal-dual Algorithm for a Convex Program. *J. ACM* 55, 5 (2008), 22:1–22:18.
[24] Eugene F. Fama. 1998. Market Efficiency, Long-term Returns, and Behavioral Finance1. *Journal of Financial Economics* 49, 3 (1998), 283–306.
[25] Hector Garcia-Molina, Manas Joglekar, Adam Marcus, Aditya G. Parameswaran, and Vasilis Verroios. 2016. Challenges in Data Crowdsourcing. *IEEE Transactions on Knowledge and Data Engineering* 28, 4 (2016), 901–911.
[26] Shawn R. Jeffery, Minos N. Garofalakis, and Michael J. Franklin. 2006. Adaptive Cleaning for RFID Data Streams. In *VLDB 2006.* 163–174.
[27] Leyla Kazemi and Cyrus Shahabi. 2012. GeoCrowd: Enabling Query Answering with Spatial Crowdsourcing. In *GIS 2012.* 189–198.
[28] Frank Kelly. 1997. Charging and Rate Control for Elastic Traffic. *European Transactions on Telecommunications* 8, 1 (1997), 33–37.
[29] Robert D. Kleinberg and Frank Thomson Leighton. 2003. The Value of Knowing a Demand Curve: Bounds on Regret for Online Posted-Price Auctions. In *FOCS 2003.* 594–605.
[30] Guoliang Li, Jiannan Wang, Yudian Zheng, and Michael J. Franklin. 2016. Crowdsourced Data Management: A Survey. *IEEE Transactions on Knowledge and Data Engineering* 28, 9 (2016), 2296–2319.
[31] Guoliang Li, Yudian Zheng, Ju Fan, Jiannan Wang, and Reynold Cheng. 2017. Crowdsourced Data Management: Overview and Challenges. In *SIGMOD 2017.* 1711–1716.
[32] JiaXu Liu, Yudian Ji, Weifeng Lv, and Ke Xu. 2017. Budget-Aware Dynamic Incentive Mechanism in Spatial Crowdsourcing. *Journal of Computer Science and Technology* 32, 5 (2017), 890–904.
[33] Mohamed Musthag and Deepak Ganesan. 2013. Labor Dynamics in a Mobile Micro-task Market. In *CHI 2013.* 641–650.
[34] Roger B. Myerson. 1981. Optimal Auction Design. *Mathematics of Operations Research* 6, 1 (1981), 58–73.
[35] Paat Rusmevichientong, Benjamin Van Roy, and Peter W. Glynn. 2006. A Nonparametric Approach to Multiproduct Pricing. *Operations Research* 54, 1 (2006), 82–98.
[36] Yaron Singer and Manas Mittal. 2013. Pricing Mechanisms for Crowdsourcing Markets. In *WWW 2013.* 1157–1166.
[37] Adish Singla and Andreas Krause. 2013. Truthful Incentives in Crowdsourcing Tasks Using Regret Minimization Mechanisms. In *WWW 2013.* 1167–1178.
[38] Tianshu Song, Yongxin Tong, Libin Wang, Jieying She, Bin Yao, Lei Chen, and Ke Xu. 2017. Trichromatic Online Matching in Real-Time Spatial Crowdsourcing. In *ICDE 2017.* 1009–1020.
[39] Hien To, Gabriel Ghinita, and Cyrus Shahabi. 2014. A Framework for Protecting Worker Location Privacy in Spatial Crowdsourcing. *PVLDB* 7, 10 (2014), 919–930.
[40] Yongxin Tong, Lei Chen, and Cyrus Shahabi. 2017. Spatial Crowdsourcing: Challenges, Techniques, and Applications. *PVLDB* 10, 12 (2017), 1988–1991.
[41] Yongxin Tong, Jieying She, Bolin Ding, Lei Chen, Tianyu Wo, and Ke Xu. 2016. Online Minimum Matching in Real-Time Spatial Data: Experiments and Analysis. *PVLDB* 9, 12 (2016), 1053–1064.
[42] Yongxin Tong, Jieying She, Bolin Ding, Libin Wang, and Lei Chen. 2016. Online Mobile Micro-Task Allocation in Spatial Crowdsourcing. In *ICDE 2016.* 49–60.
[43] Yongxin Tong, Libin Wang, Zimu Zhou, Bolin Ding, Lei Chen, Jieping Ye, and Ke Xu. 2017. Flexible Online Task Assignment in Real-Time Spatial Data. *PVLDB* 10, 11 (2017), 1334–1345.
[44] Luan Tran, Hien To, Liyue Fan, and Cyrus Shahabi. 2018. A Real-Time Framework for Task Assignment in Hyperlocal Spatial Crowdsourcing. *ACM Transactions on Intelligent Systems and Technology* 9, 3 (2018), 37:1–37:26.

A PROOF OF THEOREM 1

Proof. We will use the reduction from the classical 3-SAT problem to the decision version of GDP problem.

An instance of 3-SAT is a formula in conjunctive normal form (CNF), containing m clauses and n variables. Each clause has three literals, either positive or negative. The decision problem is whether there is an assignment of truth value for each variable such that the formula is satisfiable. To make a polynomial-time reduction, we first explain how to map the input of a 3-SAT instance to that of GDP, and then prove that the 3-SAT problem has a yes answer if and only if there is a pricing strategy such that the total revenue is m.

Formally, the formula $\phi = C_1 \wedge C_2 \wedge \cdots \wedge C_m$. For each clause C_i, there is a worker w_i. For each clause $C_i = a_i \vee b_i \vee c_i$, there will be three requests corresponding to the literals. For each positive one, the requester has the valuation $v_r = 1$ and the distance $d_r = 1$. For each negative one, the requester has the valuation $v_r = 2$ and the distance $d_r = 0.5$. Note that now the valuation v_r is a deterministic value (*i.e.*, requester will accept any price which is no more than v_r) and the revenue has no expectation. It is also easy to see that when we want to complete some request, pricing at its valuation will be the best strategy. If a_i and $\neg a_i$ both appear in the formula, the requesters representing them will locate in the same region (*e.g.*, grid in our problem), which also means that platforms must offer the same price to them. The worker w_i representing clause C_i locates in a position such that he/she can only complete the three requests which represent the literals C_i contains. The transformation can be done in polynomial time.

Given a satisfiable formula, we will show there is a pricing strategy such that the maximum total revenue is m. For clause C_i, there is at least one literal which is true. If a_i is true, we will price the requests representing a_i and $\neg a_i$ 1, and if $\neg a_i$ is true, we will price them both 2 (*i.e.*, same price for the grid contains a_i and $\neg a_i$). This will ensure that either a_i or $\neg a_i$ is true, worker for clause C_i can finish it and gain a revenue of 1 (*i.e.*, 1×1 when a_i is true or 2×0.5

when $\neg a_i$ is true.). They can not be true at the same time since we do offer the same price for requests representing a_i and $\neg a_i$. Each worker can finish one request and the maximum total revenue is m. The converse is also true. Since there are m workers, each must complete one request with the revenue of 1. If we price the grid containing a_i and $\neg a_i$ 1, we will assign true to a_i, and if we price it 2, we will assign false to a_i. Since each worker can complete one request, each clause must be true and the formula is satisfiable.

□

B PROOFS OF BASE PRICING

B.1 Proof of Theorem 2

Proof. We begin with hoeffding inequalities.

Fact 1. *(Hoeffding Inequality). Let $X_1, \ldots, X_n$ be independent random variables bounded by [0,1]. Let $\bar{X} = \sum_i X_i/n$.* $\Pr[\bar{X} - \mathbb{E}[\bar{X}] > \epsilon] = \Pr[\mathbb{E}[\bar{X}] - \bar{X} > \epsilon] \leq e^{-2\epsilon^2 n}$.

Let x_r be the random variable which is 1 if r accepts the price, and 0 otherwise. When we try price p, x_r can be seen as a Bernoulli random variable, which is 1 with probability $S(p)$ (if $p < v_r$), and 0 with probability $F(p)$ (if $v_r \leq p$). Notice that the expected value of $\widehat{S}(p)$ is $S(p)$. Using Hoeffding inequality, we sample the price p $h(p)$ times and obtain

$$\Pr[\widehat{S}(p) - S(p) > \frac{\epsilon}{2p}] \leq e^{-\epsilon^2 h(p)/2p^2} \leq \delta/2k.$$

. The last inequality is derived by replacing $h(p)$ with $\lceil (2p^2/\epsilon^2)\ln(2k/\delta)\rceil$.

For any price $p \in P_{cand}$ with $pS(p) < p'S(p') - \epsilon$, but p chosen as p_m, at least one of the two cases occur: $p\widehat{S}(p) > pS(p) + \epsilon/2$ or $p'\widehat{S}(p') < p'S(p') - \epsilon/2$. Otherwise we could chain the opposites of the two inequalities and the fact that $p\widehat{S}(p) > p'\widehat{S}(p')$ (for p chosen as p_m) to deduce the contradiction. In detail,

$$p\widehat{S}(p) \leq pS(p) + \epsilon/2 \leq p'S(p') - \epsilon/2 \leq p'\widehat{S}(p').$$

Due to the union bound, at least one of these two cases happen with a probability at most δ/k. Still using union bound, the event of at least one such price $p \in P_{cand}$ happens with probability at most δ. Taking the complement of the event, we complete the proof. □

B.2 Proof of Theorem 3

Proof. There exists some $p_0 \in P_{cand}$ such that $p_0 < p^* < (1+\alpha)p_0$. Theorem 2 tells us that with high probability, if $p_0 S(p_0) > (1+\alpha)p_0 S((1+\alpha)p_0)$, our choice p_m is p_0, otherwise it is $(1+\alpha)p_0$. So we have the following inequalities:

$$p_m S(p_m) \geq p_0 S(p_0) \geq p_0 S(p^*) \geq \frac{p^* S(p^*)}{1+\alpha} \geq (1-\alpha)p^* S(p^*).$$

□

B.3 Proof of Theorem 4

Proof. As it is sufficient to prove the ratio for any t, we omit the superscript t.

We first derive the upper bound of OPT. If we suppose all the requesters accept the prices, OPT must be smaller than the maximum weights of the matching. Let M^* and R^* denote the matching and the corresponding set of tasks. We have $OPT \leq \sum_{r \in R^*} d_r p^{*g}$.

For the lower bound of ALG, consider a matching algorithm which only ensures M^*; that is, if any requester $r \in R^*$ in the matching rejects the price p_b, its neighbor worker w w.r.t. M^* will not serve the other tasks. For each instance of the possible bipartite graph, since our algorithm achieves the maximum weights, ALG must be greater than the expected revenue of this algorithm. We have

$$\begin{aligned}
ALG &\geq \sum_g \sum_{r \in R^{*tg}} p_b S^g(p_b) d_r \\
&= \sum_g \sum_{r \in R^{*tg}} \frac{\sum_g p_m^g}{G} S^g(\frac{\sum_g p_m^g}{G}) d_r \\
&\geq \sum_g \sum_{r \in R^{*tg}} d_r \frac{\sum_g p_m^g S^g(p_m^g)}{G} \\
&\geq \frac{1}{e} \sum_g \sum_{r \in R^{*tg}} d_r \frac{\sum_g p_m^g}{G} \\
&\geq \frac{1}{eG} \sum_g \sum_{r \in R^{*tg}} d_r p^{*g} \\
&\geq \frac{1}{eG} OPT.
\end{aligned}$$

The third inequality is because of the concavity of the function $pS^g(p)$. The fifth one is because we assume $Gp_{min} \geq p_{max}$. The fourth one can be derived by the following fact.

Fact 2. *Let F be any MHR distribution with support on $[0, \infty]$. Let $p_m = \arg\max_p pS(p)$. $S(p_m) \geq 1/e$. [10]*

□

C PROOFS OF DYNAMIC PRICING

C.1 Proof of Lemma 6

Proof. By Hoeffding inequalities, $\Pr[\widehat{S}(p) - S(p) > \frac{c(p)}{p}] = \Pr[S(p) - \widehat{S}(p) > \frac{c(p)}{p}] \leq e^{-2c^2(p)N(p)/p^2} = N^{-4}$. The last equality is derived by replacing $c(p)$ with $p\sqrt{\frac{2\ln N}{N(p)}}$. □

C.2 Proof of Lemma 7

Proof. The first inequality can be derived by Lemma 6. For the second, we have $p\widehat{S}(p) + c(p) < pS(p) + 2c(p) = pS(p) + 2p\sqrt{\frac{2\ln N}{N(p)}} < \max(p'S(p'), p''S(p'')) - pS(p)$, where the last inequality can be obtained by replacing $N(p)$. □

C.3 Proof of Theorem 5

Proof. We discuss three cases showed in Fig. 4.

For the first one where $D \geq C$ and $p_0 = 0$, from Lemma 7, we have $p\widehat{S}(p) + c(p) < \max(p'S(p'), p''S(p''))$ for any p but p' and p''. For any p such that $p < p' \leq p^*$, since $Dp/C < Dp'/C < Dp''/C$ and $p\widehat{S}(p) + c(p) < \max(p'S(p'), p''S(p''))$, p's index is smaller than that of p' (or p'') and we cannot choose such p. For any p such that $p > p'' \geq p^*$, since we have $p\widehat{S}(p) + c(p) < \max(p'S(p'), p''S(p'')) < Dp/C$, we know the smaller term of its index is $p\widehat{S}(p) + c(p)$, and we will choose p' or p'', since $p\widehat{S}(p) + c(p) < \max(p'S(p'), p''S(p'')) \leq \max(p'\widehat{S}(p') + c(p'), p''\widehat{S}(p'') + c(p''))$, by Lemma 7.

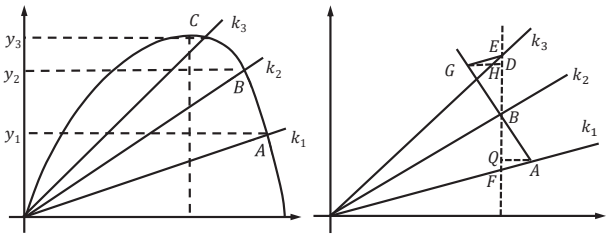


Figure 9: Proof of optimality.

For the second one and the third one, if $p < p_0$, Dp/C is the smaller term of its index due to the first inequality in Lemma 7. Furthermore, as $Dp/C < \max(p'S(p'), p''S(p'')) \le \max(p'\widehat{S}(p') + c(p'), p''\widehat{S}(p'') + c(p''))$, p' or p'' has larger index. For $p_0 \le p$, the analysis is similar to the first case; we cannot choose p for either $p < p'$ or $p > p''$. Note that there is no need to discuss $p < p'$ in the third case, since $p' < p_0 = p^*$.

□

C.4 Proof of Lemma 9

Proof. We will use coordinate geometry to give a proof. Some redefined variables may cause a little abuse of notation, but they will only be used in this proof.

Every two successive Δ^g is related to three lines. Without loss of generality, let k_1, k_2, k_3 denote their slopes. Since we add we add d_r in the decreasing order, we have $k_2 - k_1 > k_3 - k_2$.. They are showed with the concave function in Figure 9. The intersections of these lines and the concave function will be denoted by A, B, C, with their corresponding function values y_1, y_2, y_3. And note that A, B, C are all at the right of the maximizer of the concave function. $y_2 - y_1$ will be the old Δ^g and $y_3 - y_2$ is the new one. It is sufficient to prove that under the condition that $k_2 - k_1 > k_3 - k_2$, $y_2 - y_1 > y_3 - y_2$ holds.

Claim 1. *If $k_2 - k_1 = k_3 - k_2$, then $y_2 - y_1 > y_3 - y_2$.*

Suppose this claim is true. We claim that when $k_2 - k_1 > k_3 - k_2$, $y_2 - y_1 > y_3 - y_2$ holds as well. Indeed, if we reduce k_1 by a tiny small value so that $k_2 - k_1 > k_3 - k_2$, where k_2 and k_3 are fixed, y_1 can only be smaller since A moves downwards.

Now we will prove by contradiction when supposing $y_2 - y_1 \le y_3 - y_2$. Let A and B be fixed. Draw a line through B vertical to x-coordinate. The line intersects with the line of slope k_1 at F and with the line of slope k_3 at E. Through E draw a line parallel to the line with the slope k_1. The line intersects with the extension of line AB at G. Through G draw the line perpendicular to EF, with its foot denoted by D, and the line intersects with the line of slope k_3 at H. Through A draw the line perpendicular to EF, with its foot denoted by Q. Let the x value of F be c. $BF = c(k_2 - k_1)$ and $BE = c(k_3 - k_2)$. Since $k_2 - k_1 = k_3 - k_2$, $BE = BF$. $\angle AFB = \angle GEB$ because $EG \parallel AF$. With $\angle ABF = \angle GBE$, $\triangle BAF \cong \triangle BGE$. If $y_2 - y_1 \le y_3 - y_2$, the y value of C must be between the y value of E and the y value of D. Because C is on the line of slope k_3, C must be on the line segment EH. The slope of line BC is smaller than that of AB, which contradicts with the concavity of the curve, so $y_2 - y_1 > y_3 - y_2$. □

C.5 Proof of Theorem 8

Proof. For each feasible plan $n^t = (n^{t1}, n^{t2}, \ldots, n^{tG})$, it must correspond to a matching in the bipartite graph. This is because each worker becomes the supply of some grid g requires that there is an edge in the graph, and a worker can only be the supply of one grid. Let us denote the set of tasks which are assigned to workers by R'^t. The $\sum_g L^g(n^{tg})$ can be rethought as the function of R'^t, rewritten as $L(R'^t)$. The value is calculated by counting n^{tg} and deriving each p^{tg} for each g.

Lemma 9 tells us that in each grid g, each increase Δ^g is decreasing. And in each iteration, we use max-heap to find the largest increase among all grids. In a word, whenever there is a chance to increase $\sum_g L^g(n^{tg})$, we always choose the largest currently possible value.

If we show that $L(W'^t)$ is a monotone submodular set function, our algorithm will give us a $1 - \frac{1}{e}$ approximation by [14]. It is easy to prove that it is monotone increasing, since one more worker can only increase the function values. For submodularity, we need to prove that for any $R_1 \subseteq R_2 \subseteq R^t$ and $r \in R^t \backslash R_2$, $L(R_1 \cup \{r\}) - L(R_1) \ge L(R_2 \cup \{r\}) - L(R_2)$. Suppose r is w.r.t. g, since the slope in $L(R_1)$ is smaller than that of $L(R_2)$, we know the inequality holds by the concavity of the curve. □

C.6 Proof of Theorem 10

Proof. We will not consider tasks in R^t that is beyond any worker's reachability, because neither *ALG* nor *OPT* can complete these tasks. Given some p^t, we first focus on the revenue for each t. For some g, given price p^{tg}, the total revenue U^t is smaller than $\min(\sum_{r \in R^{tg}} d_r x_r p^{tg}, \sum_{i=1}^{n^{tg}} d_{r_i} p^{tg})$. Let random variables X and Y denote the two terms. Noticing that $\mathbb{E}[\min(X, Y))] \le \min(\mathbb{E}[X], \mathbb{E}[Y])$, we have

$$\mathbb{E}[U^t|p^t] \le \sum_g \min\Big(\sum_{r \in R^{tg}} d_r p^{tg} S^g(p^{tg}), \sum_{i=1}^{n^{tg}} d_{r_i} p^{tg}\Big) = \sum_g L(n^{tg}, p^{tg}).$$

On the other hand, given any p^t, we want to prove

$$\mathbb{E}[U^t|p^t] \ge \sum_g L(n^{tg}, p^{tg}) - O(d_{max}\sqrt{m \log m}).$$

For the first term in L, we use the general Chernoff bound [18]:

Fact 3. *Let $X_1, \ldots, X_n$ be independent Bernoulli random variables, which is 1 with probability S. Let $X = \sum_{i=1}^n d_i x_i$ with $d_i > 0$. Define $\nu = \sum_{i=1}^n d_i^2 S$. $\Pr[\mathbb{E}[X] - X > \epsilon] \le e^{\frac{-2\epsilon^2}{2\nu}}$.*

Taking the inequality for X, we have

$$U^t \ge \sum_g \sum_{r \in R^{tg}} d_r p^{tg} S^g(p^{tg}) - O(d_{max}\sqrt{m \log m})$$

with high probability. For the second term, we first observe that

$$\mathbb{E}[U^t|p^t] \ge \sum_g \sum_{i=1}^{n^{tg}} d_{r_i} p^{tg} S^g(p^{tg}).$$

This is because for the n^{tg}th largest d_{r_i}, they will be definitely finished if they accept the price. Here we also assume the following

inequality holds: $(1 - S^g(p^{tg})) \sum_{i=1}^{n^{tg}} d_{r_i} p^{tg} \leq O(d_{max}\sqrt{n^{tg}\log m})$, which is often true in practice. Combining these two, we will obtain the result for the second term.

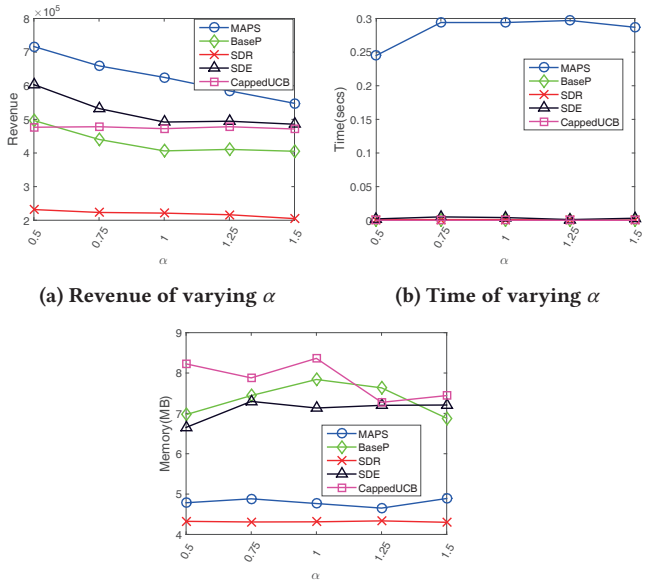
We know ALG chooses n'^t and p'^t which achieves at least $(1 - \frac{1}{e}) \max_{n^{tg}, p^{tg}} \sum_g L(n^{tg}, p^{tg})$. Let n^{*t} and p^{*t} denote the choices of OPT. We have

$$\begin{aligned} ALG &\geq \sum_g L(n'^{tg}, p'^{tg}) - O(d_{max}\sqrt{m\log m}) \\ &\geq (1 - 1/e) \sum_g L(n^{*tg}, p^{*tg}) - O(d_{max}\sqrt{m\log m}) \\ &\geq (1 - 1/e) \sum_g \mathbb{E}[U^t | p^{*t}] - O(d_{max}\sqrt{m\log m}) \\ &= (1 - 1/e)OPT - O(d_{max}\sqrt{m\log m}). \end{aligned}$$

Designing L this way gives a good approximation for the expected revenue. Another approximate expression could be

$$\sum_{i=1}^{\min(|R^{tg}|, S^g(p^{tg}), n^{tg})} d_{r_i} p^{tg} S^g(p^{tg}).$$

We leave the analysis in future work. □



(a) Revenue of varying α **(b) Time of varying α**

(c) Memory of varying α

Figure 10: Results on varying α of exponential distribution.

D EFFECT OF α OF THE DEMAND DISTRIBUTION OF v_r

Fig. 10 presents the results of varying α of the exponential demand distributions. As shown in the figures, MAPS still achieves the largest revenue with reasonable time and memory cost. The results are similar to those using demand following a normal distribution.