Proceedings of the Fifth Conference on
Human Computation and Crowdsourcing
(HCOMP 2017)

Revenue-Maximizing Stable Pricing in Online Labor Markets

Chaolun Xia, Shan Muthukrishnan
Rutgers University, New Brunswick, NJ, USA

Abstract

In online labor markets, millions of paid tasks are performed by workers every day. We solve the *stable pricing problem*, that is, given the information about tasks and workers, to find a revenue-maximizing mechanism – pricing and allocation – that is *stable* (where no worker or task is treated unfairly), and *truthful* (tasks reveal their true needs).

We propose two truthful, stable mechanisms named SMUP and SMNP. In SMUP, we use randomized *uniform* pricing, and prove that it has $(1 + \log h)$-guarantee on revenue where h is the maximum price of a task. In SMNP, we use randomized *non-uniform* pricing, and prove that it has $(3 + 3 \log h)$-guarantee on revenue, slightly worse than SMUP analytically. However our experiments show, SMNP has much less variance than SMUP. For the *online* setting when tasks arrive over time, we present a truthful online stable mechanism with $(2 + 2 \log h)$-guarantee on revenue.

Introduction

Online labor markets contribute significantly to Internet Economy. For example, in Amazon Mechanical Turk (AMT) and CrowdFlower, agents are employed for human intelligence tasks. Workers in TaskRabbit perform daily-life tasks, e.g. moving, cleaning. In Upwork, specialists are hired for more professional tasks, e.g. software development, translation.

For such markets, it is essential to price transactions and match tasks with workers suitably. In AMT, tasks are priced by owners. In Upwork, besides pricing the task, an employer needs to interview applicants which is burdensome. Recently, *automatic* pricing and matching is adopted in some online labor markets. For example, TaskRabbit provides an option called *Quick Assign*[1]. With this option, once the employer finishes defining the task by specifying requirements (e.g. in Fig 1), date and location, the system automatically charges a price [2] and allocates a qualified worker to the task.

[1] https://support.taskrabbit.com/hc/en-us/articles/205313120-What-is-Quick-Assign-

[2] To our best knowledge, the price only depends on the requirements and other facts like time and location. In other words, the task owner needs to pay the given price for any qualified worker. For example, $33/hr for any worker with a car, and $71/hr for any worker with a car.

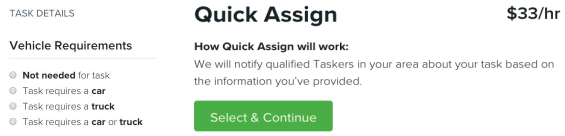


Figure 1: Specifying the requirement for a moving task

We study this automatic pricing and allocation problem. The natural goal is to optimize the revenue, but we require additional properties. In online labor markets, no worker or task should be treated unfairly by the pricing and allocation. We formulate this as a form of *stability* (similar to (Gale and Shapley 1962)), which is illustrated as follows:

Example 1. *We assume that any worker prefers a task with a higher price* [3]*. There are two workers, w_1 and w_2, and two tasks, t_1 and t_2. Only w_1 is qualified for t_1. w_1 and w_2 are both qualified for t_2, but the owner of t_2 prefers w_1 to w_2. Regardless of the pricing, the matching that $\{(w_1, t_1), (w_2, t_2)\}$ maximizes the revenue. However, this matching is unfair to w_1 when t_1 is priced less than t_2: if we announce the pricing and allow workers and tasks to match independently at their will, the matching will be $\{(w_1,t_2)\}$.*

In this paper, we solve the *stable pricing problem*. Our contributions are:

- We design a truthful, stable mechanism with randomized *uniform* pricing (SMUP), and prove that it has $(1 + \log h)$ factor approximation guarantee on revenue in expectation, where h is the maximum price for a task. In uniform pricing, all the tasks are priced equally. Given any uniform pricing, we show that one can compute the optimal allocation in polynomial time while preserving the truthfulness.
- We design a truthful stable mechanism with randomized *non-uniform* pricing (SMNP). Given an arbitrary non-uniform pricing, the allocation problem is NP-hard. We propose a $\frac{3}{2}$-approximation for the allocation problem by generalizing the (unweighted) maximum stable matching in (Mcdermid 2009; Király 2013). Therefore we show that SMUP is $(3 + 3 \log h)$ factor appoximation of the optimal

[3] For example, the price of a task is the hourly payment from the task owner, and a worker's hourly wage is a fixed fraction of the price of the task assigned to her.

revenue, which is slightly worse than SMUP above, but our experiments show that they have similar performance in practice, and SMNP is more robust.

- We consider *online* scenario where tasks arrive over time, and present a truthful online stable mechanism with randomized uniform pricing. Combined with a greedy matching strategy, we show that this mechanism is $(2 + 2\log h)$ factor approximation for the optimal revenue.

Our results are obtained by using combinatorial optimization techniques suitably with random choice of prices.

Related Work

Stable Matching. The stable matching problem was first introduced in the seminal paper (Gale and Shapley 1962), and has received much attention, e.g. (Gusfield and Irving 1989; Irving 1994). Among many variants (Manlove et al. 2002; Iwama and Miyazaki 2008), finding a maximum stable matching with ties and incomplete preference lists (MAX SMTI) (Iwama et al. 1999) is closely related to our work. This problem is known to be APX-hard (Halldórsson et al. 2003). (Iwama, Miyazaki, and Yamauchi 2007) gives a 1.875-approximation, and (Mcdermid 2009; Király 2013) improve the ratio to 1.5. Our allocation problem, which can be viewed as a weighted version of a special case of MAX SMTI, has not been addressed yet, despite many weighted variants were studied (Gusfield and Irving 1989; Iwama and Miyazaki 2008).

In two-sided markets, (Shapley and Shubik 1971) shows the equivalence of stable matching and core. In heterogeneous and competitive job markets, (Crawford and Knoer 1981; Kelso Jr and Crawford 1982; Hatfield and Milgrom 2005) consider the matching between workers and firms. In these pricing (and matching) problems, the wages are determined by equilibrium (Demange and Gale 1985). Their results apply for labor markets where the prices and matching are determined freely by employers and workers, but not for the market, considered in this paper, where a monopoly pricing and compelling matching are needed to maximize the revenue.

Envy-free Pricing Mechanisms. Pricing is a well-studied area in Economics. In particular, the envy-free pricing (Guruswami et al. 2005; Gul and Stacchetti 1999; Leonard 1983) in Walrasian Equilibrium (Nisan et al. 2007) is relevant to our work. In recent years, envy-free pricing is studied in various settings (Balcan, Blum, and Mansour 2008; Cheung and Swamy 2008; Feldman et al. 2012; Hartline and Yan 2011). (Guruswami et al. 2005) first addresses the computational issue of envy-free pricing. They show that the problem is NP-hard even for the two special cases where the buyers are either unit-demand or single-minded. However, the envy-free pricing they discussed is different from our stable pricing because in our market, not only the buyer, i.e. the task, will envy, but the item, i.e. the worker, will also envy.

Crowdsourcing Markets. In crowdsourcing markets, (Singer and Mittal 2011; 2013; Goel, Nikzad, and Singla 2014) consider online budget feasible mechanisms for a single employer with a budget and multiple workers. When the skill levels of workers are unknown, (Ho and Vaughan 2012; Ho, Jabbari, and Vaughan 2013) provide learning methods for online task assignment. (Yin, Chen, and Sun 2013) discusses that during a work session, additional rewards lead to higher effort of workers. (Yin and Chen 2015) presents an algorithm to decide when to offer bonuses to workers to improve the overall utility of the employer.

Preliminary

$[k]$ denotes the integer set $\{1, \ldots, k\}$. There are $\{1, \ldots, S\}$ skills, a set W of workers and a set T of tasks. We define $\mathcal{W} \triangleq |W|$ and $\mathcal{T} \triangleq |T|$. A worker is denoted by a vector $\mathbf{w} = (w_1, \ldots, w_S)$ where $w_s \in \mathbb{N}$ is her level of skill $s \in [S]$. For example, when $S = 3$, $\mathbf{w} = (4, 5, 0)$ denotes that this worker has skill 1 at level 4 and skill 2 at level 5. We use superscript to denote the index of a worker, e.g. $\mathbf{w}^i$ is the i-th worker where $i \in [\mathcal{W}]$. Task j is denoted by a triplet (s_j, l_j, c_j) where $s_j \in [S]$ is the required skill, $l_j \in \mathbb{Z}_+$ is the minimum level of the required skill, and $c_j \in \mathbb{R}_+$ is the maximum cost that the task is willing to pay. For example, (1, 2, \$5) denotes a task that is willing to pay at most \$5 for hiring a worker whose level of skill 1 is at least 2. We assume that 0 is the lowest level and a larger integer denotes a higher level, so worker $\mathbf{w}$ is said to be *qualified* for task j if $w_{s_j} \geq l_j$.

An allocation (or matching) A is the set of worker-task pairs. For any $(\mathbf{w}, j) \in W \times T$, $\mathbf{w}$ and j[4] are said to be matched with each other if $(\mathbf{w}, j) \in A$. $|A|$ denotes the matching size.

Definition 2. *A is* feasible *if (1) $\forall (\mathbf{w}, j) \in A$, $\mathbf{w}$ is qualified for j, i.e. $w_{s_j} \geq l_j$ and (2) any task (or worker) can be matched up to one worker (or task).*

$\mathbf{P}$ is the pricing function where $p(s, l) \in \mathbb{R}_+$ is the price of **any** task that requires skill s with minimum level l. Given s, it is natural to require the pricing to be non-decreasing over levels, i.e. $p(s, l) \geq p(s, l')$, $\forall l \geq l'$. Note that, the payment of a task only depends on what the task reports, i.e. task j pays $p(s_j, l_j)$ if it is matched with **any** qualified worker, otherwise 0. When the context is clear, we use p_j to denote $p(s_j, l_j)$. By default, the pricing-allocation pair ($\mathbf{P}$, A) is required to be *feasible* that A is feasible and $\forall (\mathbf{w}, j) \in A$, $p_j \leq c_j$.

We assume that any worker prefers a task with the higher price, and any task prefers a qualified worker with the higher level of the required skill. Based on this assumption, we can formulate *unfairness* as the existence of *blocking pairs*.

Definition 3 (Blocking Pair)**.** *Given W, T, $\mathbf{P}$ and A, a pair $(\mathbf{w}, j) \in W \times T$ is* blocking *if both are true:*

- *$\mathbf{w}$ is unmatched or matched with j' that $p_{j'} < p_j$*
- *j is unmatched or matched with $\mathbf{w}'$ that $w'_{s_j} < w_{s_j}$.*

In other words, $(\mathbf{w}, j)$ is blocking if $\mathbf{w}$ and j both prefer each other. We next define stability in our market.

Definition 4 (Stability)**.** *$(\mathbf{P}, A)$ is* stable[5] *w.r.t. W and T if $\forall (\mathbf{w}, j) \in W \times T$, $(\mathbf{w}, j)$ is not blocking.*

[4] When the context is clear, we use j to denote task j.

[5] The stability in this paper is equivalent to the weak stability in (Irving 1994).

Let $R(\mathbf{P}, A) \triangleq \sum_{(\mathbf{w},j)\in A} p_j$ be the revenue. We next define the pricing problem and allocation problem.

Definition 5 (Stable Pricing Problem)**.** *Given W and T, to compute the revenue-maximizing pricing-allocation pair ($\mathbf{P}$, A) subject to that ($\mathbf{P}$, A) is stable.*

Definition 6 (Allocation Problem)**.** *Given W, T and $\mathbf{P}$, to compute the revenue-maximizing allocation A subject to that ($\mathbf{P}$, A) is stable.*

We focus on the stable pricing problem, and in particular, we consider mechanism design approaches. A mechanism consists of a pricing and an allocation, but will have additional truthfulness as we describe below.

Definition 7 (Stable Mechanism)**.** *Let $\Phi(\mathbf{P}, A)$ be a mechanism with $\mathbf{P}$ and A. $\Phi(\mathbf{P}, A)$ is a* stable mechanism *if $(\mathbf{P}, A)$ is stable.*

A randomized (offline or online) mechanism $\Phi(\mathbf{P}, A)$ is said to have *α-guarantee on revenue* if $\alpha \cdot \mathbf{E}[R(\mathbf{P}, A)] \geq R(\mathbf{P}^*, A^*)$. $(\mathbf{P}^*, A^*)$ is the optimal solution to the offline stable pricing problem in Definition 5 where the true information of all the workers and tasks is assumed to be known by the market. In mechanism design, any task may not necessarily report its true information to the market if misreporting can increase its utility. Therefore, we consider *truthful* mechanisms, in which the dominant strategy of any task is to report all its private parameters truthfully[6] regardless whether other tasks report their parameters truthfully (Nisan et al. 2007). A mechanism is *individual-rational* (IR) if the utility of every task is non-negative. Next, we define the utility of a task.

Let $\phi = (s_j, l_j, c_j)$ be the parameters reported by j, while $\underline{\phi} = (\underline{s}_j, \underline{l}_j, v_j,)$ be the private parameters owned by j, where v_j is the true valuation if j is matched with any qualified worker $\mathbf{w}$ that $w_{\underline{s}_j} \geq \underline{l}_j$. Given A and $\mathbf{P}$, we consider the following utility function u_j for j:

$$u_j = \begin{cases} 0 & j \text{ is unmatched} \\ v_j \cdot \mathbb{1}(w_{\underline{s}_j} \geq \underline{l}_j) - p(s_j, l_j) & j \text{ is matched with } \mathbf{w} \end{cases}$$

$\mathbb{1}(w_{\underline{s}_j} \geq \underline{l}_j)$ is the indicator function returning 1 if $w_{\underline{s}_j} \geq \underline{l}_j$, otherwise 0.

Truthful Stable Mechanism

In this section, we design two offline truthful stable mechanisms. To warm up, we show the existence of feasible stable mechanisms. We first define *monotone allocation*.

Definition 8 (Monotone Allocation)**.** *A is* monotone *if: $\nexists(\mathbf{w}', \mathbf{w}, j) \in W \times W \times T$ that $(\mathbf{w}, j) \in A$, $\mathbf{w}'$ is unmatched and $w'_{s_j} > w_{s_j}$. A monotone allocation is* maximal *if $\nexists(\mathbf{w}, j) \in W \times T$ that $\mathbf{w}$ and j could be matched but not.*

A pricing is called *uniform* if all its entries are equal, otherwise *non-uniform*. When the context is clear, we use $\mathbf{p}$ to denote a uniform pricing.

[6]We do not consider truthfulness from the side of workers, i.e. whether workers report their skill/levels truthfully, because in many online labor markets, workers' proficiency in skills is publicly accessible, e.g. in term of ratings, reviews, certificates or experience.

Proposition 1. *If A is not monotone, $\nexists\mathbf{P}$ such that ($\mathbf{P}$, A) is stable. For any uniform pricing $\mathbf{p}$, ($\mathbf{p}$, A) is stable if A is a maximal monotone allocation.*

Proof. The first claim is obvious, so we only prove the second one. Given a uniform pricing $\mathbf{p}$ and a maximal monotone allocation A, suppose $(\mathbf{w}, j)$ is a blocking pair. First, it is impossible that $\mathbf{w}$ is matched because of the uniform pricing. $\mathbf{w}$ and j cannot be both unmatched because A is maximal. The remaining case is that $\mathbf{w}$ is unmatched but j is matched with some $\mathbf{w}'$ that $w'_{s_j} < w_{s_j}$, contradicting to Definition 8. □

Since it is straightforward to construct a maximal monotone allocation and generate a uniform pricing, we omit the proof for Corollary 2.

Corollary 2. *A stable mechanism always exists.*

Uniform Pricing

In this part, we present a truthful stable mechanism with $(1+\log h)$-guarantee on revenue based on randomized uniform pricing where h is the maximum price we can set for a task. We first solve the allocation problem for uniform pricing.

Theorem 3. *Given a uniform pricing, the allocation problem can be solved in polynomial time $O(S^2\mathcal{W} + \min\{\mathcal{W}, \mathcal{T}\}(S\mathcal{W} + \mathcal{T}))$.*

To prove Theorem 3, we first prove Lemma 4 and 5. Lemma 4 shows that for any uniform pricing $\mathbf{p}$, we can compute its revenue-maximizing allocation $\tilde{A}^*_{\mathbf{p}}$ (but $(\mathbf{p}, \tilde{A}^*_{\mathbf{p}})$ is not necessarily stable) in polynomial time. Lemma 5 shows that given $\tilde{A}^*_{\mathbf{p}}$, we can construct the optimal allocation $A^*_{\mathbf{p}}$ in polynomial time that $(\mathbf{p}, A^*_{\mathbf{p}})$ is stable and $|\tilde{A}^*_{\mathbf{p}}| = |A^*_{\mathbf{p}}|$.

Lemma 4. *Given uniform pricing $\mathbf{p}$, the revenue maximizing allocation $\tilde{A}^*_{\mathbf{p}}$ (but $(\mathbf{p}, \tilde{A}^*_{\mathbf{p}})$ is not necessarily stable) can be computed in polynomial time $O(\min\{\mathcal{W}, \mathcal{T}\}(S\mathcal{W}+\mathcal{T}))$.*

Proof. Let p be the uniform price. After removing any task j that $c_j < p$, we model this problem as a maxflow instance. Besides the source and sink, we create three types of nodes, corresponding to workers (x-type), skills/levels (y-type) and tasks (z-type) respectively. For each remaining task j, we introduce two nodes z_j and y_{s_j,l_j} (do not introduce y_{s_j,l_j} again if it exists); then introduce two directed edges with capacity 1, from y_{s_j,l_j} to z_j and from z_j to the sink. For each pair $y_{s,l}$ and $y_{s,l'}$, if $l' < l$ and $\nexists\, y_{s,\hat{l}}$ such that $l' < \hat{l} < l$, introduce a directed edge from $y_{s,l}$ to $y_{s,l'}$ with infinite capacity. This means that a skill at level l can be used as the same skill at a lower level l'. Finally, for each worker $\mathbf{w}^i$, introduce a node x_i and a directed edge with capacity 1 from the source to x_i. For every skill $w^i_s > 0$ that she has, introduce a directed edge with capacity 1 from x_i to $y_{s,l}$ if all the three conditions are true: (1) $l \leq w^i_s$, (2) $y_{s,l}$ exists and (3) any node $y_{s,l'}$ such that $l < l' \leq w^i_s$ does not exist. We remove any x_i from the network if she has no edge incident to y-type nodes.

Let f^* denote the maximum flow from the source to sink. It is easy to verify that $|\tilde{A}^*_{\mathbf{p}}| = f^*$. Thus the optimal

revenue is $f^* \cdot p$. The number of nodes corresponding to workers is $O(\mathcal{W})$, to skills/levels is $O(\mathcal{T})$, and to tasks is $O(\mathcal{T})$. Therefore, the total number of nodes is $O(\mathcal{W}+\mathcal{T})$. The number of edges from x-type nodes to y-type nodes is $O(S\mathcal{W})$, among y-type nodes is $O(\mathcal{T})$, and from y-type nodes to z-type nodes is $O(\mathcal{T})$. Thus, the total number of edges is $O(S\mathcal{W}+\mathcal{T})$. Since Ford-Fulkerson algorithm is $O(|E| \cdot f^*)$ and we have $f^* = O(\min\{\mathcal{W},\mathcal{T}\})$, the time complexity to run Ford-Fulkerson on this network is $O(\min\{\mathcal{W},\mathcal{T}\}(S\mathcal{W}+\mathcal{T}))$. □

If we directly compute $\tilde{A}^*_{\mathbf{p}}$ as maximum bipartite matching, i.e. without the y-type nodes, the time complexity increases to $O(\mathcal{W}^2\mathcal{T}+\mathcal{W}\mathcal{T}^2)$ because the total number of nodes is still $O(\mathcal{W}+\mathcal{T})$ while the total number of edges is up to $O(\mathcal{W}\mathcal{T})$. It is much worse because in practice, $S \ll \min\{\mathcal{W},\mathcal{T}\}$.

However, $\tilde{A}^*_{\mathbf{p}}$ is not necessarily monotone. For example, it is possible that the maxflow algorithm assigns a worker with skill s at level 1 to the task requiring s at level 1 but leaves a worker with s at level 2 unmatched. Thus, according to Proposition 1, $(\mathbf{p}, \tilde{A}^*_{\mathbf{p}})$ is not necessarily stable. In order to guarantee stability, intuitively, one may model the allocation problem as a minimum cost maxflow rather than the maxflow by additionally introducing costs on the edges from x-type nodes to y-type nodes. However, solving a mincost maxflow is much more expensive (see the survey (Kovács 2015)) than the efficient maxflow we proposed in Lemma 4.

Algorithm 1 Stability Adjustment

Input: W, T and $\tilde{A}$
Output: a monotone allocation A

```
1: W' ← the set of all the unmatched workers
2: A ← Ã
3: Let g(A, s) be the index of any s-marginal worker in A
4: while W' ≠ ∅ do
5:     Arbitrarily select and remove w from W'
6:     for s ∈ [S] do
7:         if g(A, s) exists and w_s > w_s^{g(A,s)} then
8:             Update A by matching w with the task assigned to w^{g(A,s)}
9:             W' ← W' ∪ {w^{g(A,s)}}
10:            break
11:        end if
12:    end for
13: end while
14: return A.
```

Therefore, we propose *stable-adjustment* in Algorithm 1, and prove in Lemma 5 that it can construct $A^*_{\mathbf{p}}$ from $\tilde{A}^*_{\mathbf{p}}$ efficiently. To present it, we first define *skill-marginal*.

Definition 9 (s-Marginal)**.** *In an allocation, we call a worker* s-marginal *if she is matched with a task requiring skill s and her level of s is no higher than all the other workers matched with tasks requiring s. If* $\mathbf{w}$ *is s-marginal, we call* w_s *the* marginal-level *of s.*

Lemma 5. *Given $\tilde{A}$ which is not necessarily monotone, Algorithm 1 computes a monotone allocation A in polynomial time $O(S^2\mathcal{W}+S\mathcal{W}\log\min\{\mathcal{W},\mathcal{T}\})$ that $|A|=|\tilde{A}|$.*

Proof. It is obvious that the adjustment will finally converge. W.l.o.g., we assume that after $M>0$ iterations, the while-loop (lines 4-13) stops. Let A_m be the allocation at iteration $m \in \{0,\dots,M\}$, e.g. $A_0=\tilde{A}$ and $A_M=A$. Let $g(A_m,s)$ be the index of any s-marginal worker in A_m. Let $\mathbf{L}_s = (w_s^{g(A_0,s)},\dots,w_s^{g(A_M,s)})$ be the sequence of the marginal-levels of skill s during the while-loop (lines 4-13). It true that $\forall s$, $\mathbf{L}_s$ is non-decreasing because the while-loop keeps replacing the s-marginal worker $\mathbf{w}^{g(A,s)}$ with a worker $\mathbf{w}$ that $w_s > w_s^{g(A,s)}$.

So once an s-marginal worker becomes unmatched, she will never be matched with any task requiring skill s because her level of s cannot exceed the marginal level of s. This means, for any skill s, every worker can be matched and unmatched with a task requiring s at most once respectively. So a worker will get matched and unmatched for at most S times, implying $M=O(S\mathcal{W})$. In each iteration (lines 5-12), the algorithm sends at most S queries to get the marginal workers of all the skills, and makes at most one update in $g(A,s)$ for some s. If we implement $g(A,s)$ by a min-heap, a single query is $O(1)$ and a single update is $O(\log|A|)=O(\log\min\{\mathcal{W},\mathcal{T}\})$. Therefore, the overall time complexity is $O(S^2\mathcal{W}+S\mathcal{W}\log\min\{\mathcal{W},\mathcal{T}\})$. Note that, the time complexity for preprocessing S min-heaps is dominated.

After the algorithm stops, it is true that for any unmatched $\mathbf{w}$ and skill s, $w_s \le w_s^{g(A,s)}$. Therefore, A is monotone according to Definition 8. From the algorithm, it is true that $|A|=|\tilde{A}|$, thus proving the Lemma. □

To compute $A^*_{\mathbf{p}}$, we run the stable-adjustment with $\tilde{A}^*_{\mathbf{p}}$ as the input allocation, which is output by the maxflow in Lemma 4. The stable-adjustment only introduces an extra additive term $O(S^2\mathcal{W})$ to the time complexity of the maxflow. In practice, $S \ll min(\mathcal{W},\mathcal{T})$, so it is efficient.

We next present the **S**table **M**echanism with **U**niform **P**ricing (SMUP) as follows.

Stable Mechanism with Uniform Pricing (SMUP)
Input T, W and h

1. Randomly pick up an integer k from $\{0,\dots,\lfloor\log h\rfloor\}$, and set every entry of $\mathbf{p}$ to 2^k;
2. Run the maxflow algorithm in Lemma 4 to get $\tilde{A}^*_{\mathbf{p}}$;
3. Run the stable-adjustment in Algorithm 1 to get $A^*_{\mathbf{p}}$.

Output: $\mathbf{p}$ and $A^*_{\mathbf{p}}$;

Theorem 6. *SMUP is polynomial, truthful and IR. SMUP has $(1+\log h)$-guarantee on revenue if $v_j \in [1,h]$, $\forall j \in [\mathcal{T}]$.*

Proof. It is trivial to verify individual rationality (IR) because any task j that $c_j < 2^k$ is removed according to Lemma 4. From Theorem 3, we know that the mechanism is polynomial time, and $(\mathbf{p}, A^*_{\mathbf{p}})$ is stable. We will prove truthfulness in Lemma 7 and the $(1+\log h)$-guarantee on revenue in Lemma 8. □

Lemma 7. *Reporting ($\underline{s}_j$, $\underline{l}_j$, v_j) truthfully is the dominant strategy of task j.*

Proof. We will show that j can never improve its utility by reporting some parameters (s_j, l_j, c_j) other than its private parameters $(\underline{s}_j, \underline{l}_j, v_j)$. First, it is obvious that reporting $s_j \neq \underline{s}_j$ cannot help, so the task will always report $\underline{s}_j$. We assume that j will only consider to report $l_j > \underline{l}_j$ because matching with $\mathbf{w}$ that $w_{s_j} < \underline{l}_j$ yields negative utility.

We next show that, for any $v > 0$, reporting ($\underline{s}_j$, l_j, v) cannot derive more utility than reporting ($\underline{s}_j$, $\underline{l}_j$, v) when $l_j > \underline{l}_j$. There are two cases. First, if reporting l_j and $\underline{l}_j$ both make j matched, it is obvious that reporting l_j derives no more utility than reporting $\underline{l}_j$ because $p(\underline{s}_j, \underline{l}_j) = p(\underline{s}_j, l_j)$. Next, we only need to prove that if reporting $\underline{l}_j$ fails to make j matched, reporting l_j also fails. There are two cases when reporting ($\underline{s}_j$, $\underline{l}_j$, v) fails. First, $v < 2^k$, so reporting ($\underline{s}_j$, l_j, v) fails too. Second, $v \geq 2^k$ but there is no more qualified worker $\mathbf{w}$ that $w_{\underline{s}_j} \geq \underline{l}_j$. So there is no $\mathbf{w}$ that $w_{\underline{s}_j} \geq l_j$, and reporting ($\underline{s}_j$, l_j, v) also fails in the second case.

We finally show that reporting $(\underline{s}_j, \underline{l}_j, c_j)$ derives no more utility than reporting $(\underline{s}_j, \underline{l}_j, v_j)$ for any $c_j \neq v_j$. There are two cases. First, if reporting c_j and v_j both make j matched, it is true that these two strategies derive the same utility for j because $p(\underline{s}_j, \underline{l}_j) = 2^k$ is independent of c_j. Next, we only need to prove that if reporting v_j fails to make j matched, reporting c_j also fails. There are two cases when reporting v_j fails. First, $v_j < 2^k$. In such cases, if reporting $c_j \in (0, 2^k)$, j still fails to match; if reporting $c_j \geq 2^k$, j might be matched, however, $u_j = v_j - 2^k < 0$. The second case is when $v_j \geq 2^k$. In such cases, j participates the allocation algorithm without being precluded. However, the allocation algorithm, i.e. the maxflow described in Lemma 4 and the stable-adjustment in Algorithm 1, is independent of c_j, so reporting any c_j will not change the final allocation. □

Lemma 8. *SMUP has $(1 + \log h)$-guarantee on revenue if $v_j \in [1, h]$, $\forall j \in [\mathcal{T}]$.*

Proof. Let $\mathbf{P}^*$ and A^* be the optimal solution to the pricing problem, and $n(s, l)$ be the number of matched tasks requiring skill s at minimum level l in A^*. Thus, the optimal revenue $R(\mathbf{P}^*, A^*) = \sum_{s=1}^{S} \sum_{l=1}^{L} p^*(s, l) \cdot n(s, l)$ where $L \triangleq \max\{l_j | j \in [\mathcal{T}]\}$. Let $\mathbf{p}$ be the uniform pricing with every entry as 2^k where k is sampled from $\{0, \ldots, \lfloor \log h \rfloor\}$. From $\mathbf{P}^*$ and A^* (but in fact we do not know $\mathbf{P}^*$ or A^*), we could construct an allocation $\tilde{A}$ such that ($\mathbf{p}$, $\tilde{A}$) is feasible as follows: (1) discard any task j that $c_j < 2^k$; (2) for all the remaining matched tasks in A^*, match them with the same workers as A^* does. Next, we show the lower bound of the expectation of $R(\mathbf{p}, \tilde{A})$:

$$
\begin{aligned}
\mathbf{E}[R(\mathbf{p}, \tilde{A})] &= \sum_{k=0}^{\lfloor \log h \rfloor} \frac{1}{1 + \lfloor \log h \rfloor} \sum_{s=1}^{S} 2^k \cdot m_s(2^k) \\
&= \frac{1}{1 + \lfloor \log h \rfloor} \sum_{s=1}^{S} \sum_{k=0}^{\lfloor \log h \rfloor} 2^k \cdot \sum_{l=l(s,k)}^{L} n(s, l) \\
&= \frac{1}{1 + \lfloor \log h \rfloor} \sum_{s=1}^{S} \sum_{l=1}^{L} n(s, l) \cdot (2^{k(s,l)+1} - 1) \\
&\geq \frac{1}{1 + \lfloor \log h \rfloor} \sum_{s=1}^{S} \sum_{l=1}^{L} n(s, l) \cdot p^*(s, l) \\
&= \frac{R(\mathbf{P}^*, A^*)}{1 + \lfloor \log h \rfloor}
\end{aligned}
$$

$m_s(2^k)$ is the total number of matched tasks (in A^*) that requires skill s and is priced no less than 2^k. $l(s, k)$ is the lowest level satisfying $p^*(s, l(s, k)) \geq 2^k$. Similarly, $k(s, l)$ is the integer satisfying $2^{k(s,l)} \leq p^*(s, l) < 2^{k(s,l)+1}$.

However, ($\mathbf{p}$, $\tilde{A}$) is not necessarily stable. We observe that if $(\mathbf{w}, j)$ is a blocking pair, $\mathbf{w}$ must be unmatched due to the uniform pricing. Therefore, there exists A' such that $|A'| \geq |\tilde{A}|$ and ($\mathbf{p}$, A') is stable and feasible. Let $A^*_{\mathbf{p}}$ be the optimal allocation given uniform pricing $\mathbf{p}$. It is obvious that $\mathbf{E}[R(\mathbf{p}, A^*_{\mathbf{p}})] \geq \mathbf{E}[R(\mathbf{p}, A')]$. Although we cannot compute A' because we do not know $\mathbf{P}^*$ or A^*, we can compute $A^*_{\mathbf{p}}$ in polynomial time according to Theorem 3, thus completing the proof. □

Nonuniform Pricing

Although SMUP provides an efficient solution with revenue guarantee, non-uniform pricing is usually more desirable to real markets. Therefore in this part, we consider a truthful stable mechanism with non-uniform pricing. For truthfulness, we still use randomized techniques to generate a non-uniform pricing. The allocation problem for non-uniform pricing has a close connection to the generalized stable matching where the preference lists not only contain ties but are also incomplete (Manlove et al. 2002). Although the existence of a generalized stable matching is NP-complete (Iwama et al. 1999), later we will show that it is polynomial to find an allocation A for any non-uniform pricing $\mathbf{P}$ that ($\mathbf{P}$, A) is stable. Maximum generalized stable matching (Max SMTI) is NP-hard (Manlove et al. 2002), and our allocation problem can be viewed as a weighted version of a special case (where the preferences lists of all the workers are the same) of Max SMTI. So the NP-hardness of Max SMTI cannot imply the NP-hardness of our allocation problem for non-uniform pricing, which we independently prove in Theorem 9.

Theorem 9. [7] *The allocation problem is NP-hard even if the pricing only contains two distinct values.*

We then propropose a $\frac{3}{2}$-approximation, in Algorithm 2, to the allocation problem by generalizing the state-of-arts (Mcdermid 2009; Király 2013) which both provide a $\frac{3}{2}$-approximation for (unweighted) maximum general stable matching. We first define the notion of *substitutable*.

Definition 10 (Substitutable)**.** *A matched (e.g. with task j) worker $\mathbf{w}$ is* substitutable *if there exists an unmatched worker $\mathbf{w}'$ that $w_{s_j} = w'_{s_j}$.*

[7] Due to space limit, all the missing proofs are available in the technical report (http://paul.rutgers.edu/~cx28/papers/stable_pricing.pdf).

Then we define *select*(j, Ψ), a function returning the worker in the worker set Ψ who is most preferred by task j. Let $\mathbf{w}$ be the worker returned by *select*(j, Ψ), so $\mathbf{w} = \text{argmax}_{\mathbf{w}' \in \Psi} w'_{s_j}$. We define how *select*$(j, \Psi)$ deals with ties as follows: (1) it prefers the worker with the highest level of s_j; (2) among workers with the same level, it prefers unmatched to matched workers; (3) among matched workers with the same level, it prefers substitutable workers. In other cases, *select*(j, Ψ) breaks the tie arbitrarily.

Ψ_j is initialized as the set of all the workers qualified for task j. We present the approximation in Algorithm 2. In each iteration, it picks an unmatched task j^* with the highest price (breaking ties arbitrarily), and tries to match j^* with workers in Ψ_{j^*}. For every selected worker $\mathbf{w}$ returned by *select*(j^*, Ψ_{j^*}), it will match $\mathbf{w}$ with j^* if one of the three cases is true: (1) $\mathbf{w}$ is unmatched; or (2) $\mathbf{w}$ is substitutable; or (3) j^* has the same price with j (which is now matched with $\mathbf{w}$), and j^* has selected all its qualified workers once (i.e. $r_{j^*} = 1$) but j has not (i.e. $r_j = 0$). Otherwise the algorithm will not match $\mathbf{w}$ with j^*. If a task has selected all its qualified workers but remains unmatched, we give it one more chance, i.e. set r_j from 0 to 1. If the task fails to match again, i.e. $r_j = 1$, it will never be considered. We will show that if we combine all these design together, this approximation will have $\frac{3}{2}$-guarantee, and with any one missing, the guarantee drops to 2.

We first prove the stability and analyze time complexity in Lemma 10, and then show its performance guarantee in Theorem 11.

Lemma 10. *Given any* $\mathbf{P}$*, Algorithm 2 outputs* $\hat{A}$ *in polynomial time* $O(\mathcal{W}^2\mathcal{T} + \mathcal{T}\log\mathcal{T})$ *that* $(\mathbf{P}, \hat{A})$ *is stable.*

Theorem 11. *Algorithm 2 is a* $\frac{3}{2}$*-approximation to the allocation problem.*

Proof. The proof contains two parts, Lemma 12 as the first part and the remaining part based on Lemma 12.

Lemma 12. *For any pricing* $\mathbf{P}$*, Algorithm 2 outputs* $\hat{A}$ *such that* $\frac{3}{2}|\hat{A}| \geq |A^*|$ *where* A^* *is the optimal allocation.*

Proof. Assuming we know A^*. Construct a graph as follows. For each task that is matched in either A^* or $\hat{A}$, create a task node, and similarly for each worker who is matched in either A^* or $\hat{A}$, create a worker node. For any pair of a task node and a worker node, if they are matched in A^*, introduce an undirected edge between them, and similarly, if they are matched in $\hat{A}$, also introduce an undirected edge between them. In the graph, there are at most two edges between two nodes. If we consider a component with exactly two nodes and two edges as a 2-cycle, each connected component in the graph is either a cycle or path.

It is obvious that in any cycle, any task matched in A^* is also matched in $\hat{A}$, so there is no revenue loss in cycles. Also, in any path with even number of edges, the number of tasks matched in A^* is the same with the number of tasks matched in A (but there might be some revenue loss). We call a path with odd number of edges as an *augmenting* path. It is obvious that in any augmenting path, there is at least one task matched in A^* but not matched in $\hat{A}$. Clearly, in an augmenting path with $2k+1$ edges ($k \in \mathbb{N}$), the number of task nodes and worker nodes are both $k+1$, and there are exactly k tasks matched in $\hat{A}$ and $k+1$ tasks matched in A^*. It is enough to prove this lemma by showing that the graph does not contain any augmenting path with 1 or 3 edges.

First, it is obvious that an augmenting path with 1 edge cannot exist. If there was, the worker node and task node would have formed a blocking pair in $\hat{A}$, contradicting with that $(\mathbf{P}, \hat{A})$ is stable.

Suppose that there is an augmenting path with 3 edges. As shown in Fig 2, the solid line denotes that $\mathbf{w}$ and j are matched in $\hat{A}$, while $(\mathbf{w}, j')$ and $(\mathbf{w}', j)$, represented by dashed lines, are matched respectively in A^* (but neither $\mathbf{w}'$ nor j' is matched in $\hat{A}$).

We first show $w_{s_j} > w'_{s_j}$. If $w_{s_j} < w'_{s_j}$, $(\mathbf{w}', j)$ is a blocking pair in $\hat{A}$, which is a contradiction. If $w_{s_j} = w'_{s_j}$, $\mathbf{w}$ is finally substitutable. However, j' has selected $\mathbf{w}$ twice, according to Claim 12.1, which is a contradiction. We next

Algorithm 2 Allocation for Non-uniform Pricing

Input: T, W and $\mathbf{P}$
Output: $\hat{A}$

$T' \leftarrow \{j | c_j \geq p(s_j, l_j), j \in [\mathcal{T}]\}$
$\forall j \in T', \Psi_j \leftarrow \{\mathbf{w} | w_{s_j} \geq l_j, \mathbf{w} \in W\}$
$\forall j \in T', r_j \leftarrow 0$
$\hat{A} \leftarrow \emptyset$
$t \leftarrow 0$ ▷ Only for indexing iterations
while $T' \neq \emptyset$ **do**
  $t \leftarrow t + 1$
  $j^* \leftarrow \text{argmax}_{j \in T'} p(s_j, l_j)$
  while $\Psi_{j^*} \neq \emptyset$ **and** task j^* is unmatched **do**
    $\mathbf{w} \leftarrow$ *select*(j^*, Ψ_{j^*})
    if $\mathbf{w}$ is unmatched **then**
      Let $\mathbf{w}$ match with task j^* in $\hat{A}$
      $T' \leftarrow T' - \{j^*\}$
    else (assume $\mathbf{w}$ is matched with task j)
      if $\mathbf{w}$ is substitutable **then**
        Let $\mathbf{w}$ re-match with task j^* in $\hat{A}$
        $T' \leftarrow T' \cup \{j\} - \{j^*\}$
      else if $p_{j^*} = p_j$ **and** $r_{j^*} > r_j$ **then**
        Let $\mathbf{w}$ re-match with task j^* in $\hat{A}$
        $T' \leftarrow T' \cup \{j\} - \{j^*\}$
        $\Psi_j \leftarrow \Psi_j - \{\mathbf{w}\}$
      else
        $\Psi_{j^*} \leftarrow \Psi_{j^*} - \{\mathbf{w}\}$
      end if
    end if
  end while
  if task j^* is unmatched **then**
    if $r_{j^*} = 0$ **then**
      $r_{j^*} = 1$
      $\Psi_{j^*} \leftarrow \{\mathbf{w} | w_{s_{j^*}} \geq l_{j^*}, \mathbf{w} \in W\}$
    else
      $T' \leftarrow T' - \{j^*\}$
    end if
  end if
end while
return $\hat{A}$.

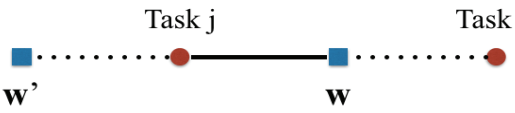


Figure 2: An augmenting path with 3 edges.

show $p_j > p_{j'}$. It is trivial to see that $p_j < p_{j'}$ is impossible. If $p_j = p'_j$, it is true that $r_j = 1$, otherwise j' would have been matched with $\mathbf{w}$. However, this contradicts with Claim 12.2 because $\mathbf{w}'$ is finally unmatched. Since we have $w_{s_j} > w'_{s_j}$ and $p_j > p_{j'}$, $(\mathbf{w}, j)$ is a blocking pair in A^*, which is a contradiction. Therefore, any augmenting path with 3 edges cannot exist.

Claim 12.1. *A worker is not substitutable after she is selected (line 10) twice by the algorithm.*

Claim 12.2. *$\forall j$, if $\exists \mathbf{w}$ such that (1) $\mathbf{w}$ is substitutable or unmatched and (2) $w_{s_j} \geq l_j$, we have $r_j = 0$.*

□

Now we show the second part based on Lemma 12. Suppose there are $K \leq \mathcal{T}$ unique values (i.e. prices) in $\mathbf{P}$, namely $\pi_1 \geq \ldots \geq \pi_K$. Let $\hat{A}_k$ (or A^*_k) be the partial approximate (or optimal) allocation which only includes all the matched worker-task pairs $(\mathbf{w}, j)$ that $p_j \geq \pi_k$. Let $\hat{A}(k)$ (or $A^*(k)$) be the approximate (or optimal) allocation if the input task set only includes all the tasks priced no less than π_k in T. Since the order for each task being chosen (line 8) for the first time depends on its price, it is true that $|\hat{A}_k| = |\hat{A}(k)|$, according to Claim 12.3.

Claim 12.3. *$\forall p_{j'} > p_j$, if task j' is matched (or unmatched) at the moment when task j is chosen by the algorithm (line 8) for the first time, task j' will still be matched (or unmatched) when the algorithm finishes.*

Proof of Claim. *It is easy to see that task j cannot affect j' if $p_{j'} > p_j$, except that $\mathbf{w}$ who is matched with j' is substitutable. In such a case, $\mathbf{w}$ will be re-matched with j, however, there is at least one unmatched worker available for j', so j' will still be matched.*

With Lemma 12, it is true that $\forall k \in [K]$, $\frac{3}{2}|\hat{A}_k| = \frac{3}{2}|\hat{A}(k)| \geq |A^*(k)| \geq |A^*_k|$. Introducing a dummy price $\pi_{K+1} = 0$, we have that $\forall k \in [K]$,

$$\frac{3}{2}(\pi_k - \pi_{k+1}) \cdot |\hat{A}_k| \geq (\pi_k - \pi_{k+1}) \cdot |A^*_k|$$

Summing over all the K inequalities, we have:

$$\begin{aligned} R(\mathbf{P}, \hat{A}) &= \pi_1 \cdot |\hat{A}_1| + \sum_{k=2}^{K} \pi_k \cdot (|\hat{A}_k| - |\hat{A}_{k-1}|) \\ &\geq \frac{2}{3}\Big(\pi_1 \cdot |A^*_1| + \sum_{k=2}^{K} \pi_k \cdot (|A^*_k| - |A^*_{k-1}|)\Big) \\ &= \frac{2}{3} R(\mathbf{P}, A^*) \end{aligned}$$

□

Proposition 13. *The ratio $\frac{3}{2}$ is tight for Algorithm 2.*

Based on the approximation, we present the **S**table **M**echanism with **N**non-uniform **P**ricing (SMNP) as follows.

Stable Mechanism with Non-uniform Pricing (SMNP)
Input T, W and h

1. $\forall s \in [S]$: (1) pick up an integer k_s uniformly and independently sampled from $\{0, \ldots, \lfloor \log h \rfloor\}$, and (2) $\forall l$: $p(s,l) \leftarrow 2^{k_s}$;
2. Run Algorithm 2 to compute $\hat{A}$;

Output: $\mathbf{P}$ and $\hat{A}$.

Theorem 14. *SMNP is polynomial, truthful and IR. SMNP has $(3 + 3\log h)$-guarantee on revenue if $v_j \in [1, h]$, $\forall j \in [\mathcal{T}]$.*

Proof. According to Lemma 10, $(\mathbf{P}, \hat{A})$ is stable, and $\hat{A}$ can be computed in polynomial time. SMNP is IR since task j will never be charged more than c_j (line 1 in Algorithm 2). The proof of truthfulness is similar to the proof that SMUP is truthful in Lemma 7, thus we omit it.

We next prove the $(3 + 3\log h)$-guarantee on revenue in two steps. In the first step, we show that if we knew (but in fact we do not know) the optimal stable pricing-allocation pair $(\mathbf{P}^*, A^*)$, we could have constructed $\tilde{A}$ satisfying that: given any $\mathbf{P}$ chosen at random (as SMNP does), (1) $(\mathbf{P}, \tilde{A})$ is feasible but not necessarily stable and (2) $\mathbf{E}[R(\mathbf{P}, \tilde{A})] \geq \frac{R(\mathbf{P}^*, A^*)}{2+2\log h}$. In the second step, we show that we could construct A' from $\tilde{A}$ such that $(\mathbf{P}, A')$ is stable and $R(\mathbf{P}, A') \geq \frac{R(\mathbf{P}, \tilde{A})}{2}$. Let $A^*_{\mathbf{P}}$ be the optimal solution to the allocation problem given $\mathbf{P}$, so we have $R(\mathbf{P}, A^*_{\mathbf{P}}) \geq R(\mathbf{P}, A')$. According to Theorem 11, we have $\frac{3}{2}R(\mathbf{P}, \hat{A}) \geq R(\mathbf{P}, A^*_{\mathbf{P}})$, so we have $\mathbf{E}[R(\mathbf{P}, \hat{A})] \geq \frac{R(\mathbf{P}^*, A^*)}{3+3\log h}$. Since according to Lemma 10, we can compute $\hat{A}$ in polynomial time, the proof is completed. Next, we show the proof for the two steps.

In the first step, we show that given $\mathbf{P}$, we could construct $\tilde{A}$ from A^* as follows: (1) discard any task j that $c_j < p_j$; (2) for all the remaining matched tasks in A^*, match them with the same workers as A^* does.

Claim 14.1. $\mathbf{E}[R(\mathbf{P}, \tilde{A})] \geq \frac{R(\mathbf{P}^*, A^*)}{1+\log h}$.

Proof of Claim. *Let $R_s(\mathbf{P}, \tilde{A})$ denote the revenue contributed by all the tasks requiring skill s, i.e. $R(\mathbf{P}, \tilde{A}) = \sum_{s=1}^{S} R_s(\mathbf{P}, \tilde{A})$. Since for any s, k_s is sampled independently, $\mathbf{E}[R(\mathbf{P}, \tilde{A})] = \sum_{s=1}^{S} \mathbf{E}[R_s(\mathbf{P}, \tilde{A})]$. The proof that $\forall s$, $\mathbf{E}[R_s(\mathbf{P}, \tilde{A})] \geq \frac{R_s(\mathbf{P}^*, A^*)}{1+\log h}$ is similar to the proof for the lower bound of expected revenue of the randomized uniform pricing in Lemma 8, thus omitted.*

However, $(\mathbf{P}, \tilde{A})$ is not necessarily stable because $\tilde{A}$ is not necessarily monotone. In the second step, we show that Algorithm 3 uses a simple greedy method to construct A' from $\tilde{A}$ and $\mathbf{P}$ such that (1) $(\mathbf{P}, A')$ is stable; and (2) $R(\mathbf{P}, A') \geq$

$\frac{R(\mathbf{P},\tilde{A})}{2}$. Note that, this method is merely for proving the existence of A', so instead of analyzing time complexity, we only need to prove that it will always halt in Claim 14.2.

Algorithm 3 Stability Adjustment 2

Input: $T, W, \mathbf{P}$ and $\tilde{A}$
Output: A'
1: $A' \leftarrow \tilde{A}$
2: **while** there is a blocking pair $(\mathbf{w}, j)$ in $(\mathbf{P}, A')$ **do**
3: Update A' by matching $\mathbf{w}$ with j
4: **end while**
5: **return** A'.

Claim 14.2. *Algorithm 3 always halts in finite iterations.*

Claim 14.3. $R(\mathbf{P}, A') \geq \frac{R(\mathbf{P},\tilde{A})}{2}$, *and* $(\mathbf{P}, A')$ *is stable*

Proof of Claim. *Since the algorithm always halts according to Claim 14.2,* $(\mathbf{P}, A')$ *is stable. We next show that* $R(\mathbf{P}, A') > \frac{R(\mathbf{P},\tilde{A}))}{2}$. *The loss of revenue might occur only when we re-match* $\mathbf{w}$ *and* j *(line 3) in* A' *if* $\mathbf{w}$ *and* j *were both matched (with others) before this re-match. In the worst case,* $\mathbf{w}$ *was matched with* j' *and* j *was matched with* $\mathbf{w}'$ *in* $\tilde{A}$*, but after the re-match,* j' *is finally matched in* A'*. In such a case, we lose the revenue* $p_{j'}$ *but guarantee the revenue* p_j*. Since* $p_{j'} < p_j$*, the total loss of revenue is less than* $\frac{R(\mathbf{P},\tilde{A}))}{2}$*, thus proving the claim.*

□

In this section, we presented SMUP and SMNP. Due to the NP-hardness of allocation problem for non-uniform pricing, the revenue guarantee of SMNP is slightly worse than SMUP. As we show through experiments, SMNP and SMUP generate nearly the same revenue but SMNP is more robust. We also tried to assign different prices over different levels of a skill, however, this strategy makes the guarantee worse, which is omitted here.

Online Truthful Mechanism

In this section, we present an online truthful stable mechanism. We assume that the market runs in M rounds, and the pricing and the allocation are static. That is, if we fix the price $p(s, l)$ in round m, we cannot change it in later rounds. Similarly, if we match a worker and a task in round m, we cannot re-match or unmatch them in later rounds. We require global stability. Let $\mathbf{P}$ be the pricing. Let A_m be the allocation in round m and A be the global allocation during all the M rounds, i.e. $A = \bigcup_{m=1}^{M} A_m$. We require not only ($\mathbf{P}$, A_m) to be stable $\forall m \in [M]$, but also ($\mathbf{P}$, A) to be stable.

Under these constraints, it is easy to see that if workers arrive online, there is no stable mechanism. So we study the case where all the workers are present in the first round but tasks arrive online. In this setting, task j needs to report two additional parameters a_j and d_j where $a_j \leq d_j \in [M]$ to the market. They denote the arrival and departure time respectively, i.e. the market can match j in any round $m \in \{a_j, \ldots, d_j\}$. Let m_j be the round that j gets matched, $\underline{a}_j$ and $\underline{d}_j$ be the private time parameters owned by j. The utility function becomes: $u_j = v_j \cdot \mathbb{1}(w_{\underline{s}_j} \geq \underline{l}_j \wedge m_j \in \{\underline{a}_j, \ldots, \underline{d}_j\}) - p(s_j, l_j)$ if j is matched with $\mathbf{w}$; otherwise $u_j = 0$. Besides misreporting $\underline{s}_j$, $\underline{l}_j$ and v_j, task j can strategically choose the time to appear in the market. As it is natural in online mechanism design, we adopt a restricted misreporting model where we assume no early arrival but unrestricted departure, i.e. j may report any $a_j \geq \underline{a}_j$. This assumption is practical because $\underline{a}_j$ can be viewed as the earliest time that the employer realizes he needs to solve a task.

We have not found any mechanism with non-uniform pricing that is simultaneously stable, truthful and with non-trivial revenue guarantee. So we present the **O**nline **S**table **M**echanism with **U**niform **P**ricing (OSMUP) as follows. Let *MaxFlow*(W, T, $\mathbf{p}$) denote the maxflow algorithm in Lemma 4 with inputs W as the worker set, T as the task set and $\mathbf{p}$ as the uniform pricing respectively. Let *StableAdjustment*($W, T, \tilde{A}$) denote the stable-adjustment in Algorithm 1 with W, T and $\tilde{A}$ as input. Let T_m be the set of tasks that arrive in round m (i.e. $a_j = m$).

Algorithm 4 OSMUP

Input: W and $T_1, \ldots, T_M$ in sequence
Output: $\mathbf{p}$ and $A_1, \ldots, A_M$
1: $W_1 \leftarrow W$
2: Sample k from $\{0, \ldots, \lfloor \log h \rfloor\}$ and $\forall s, l, p(s, l) \leftarrow 2^k$
3: **for** $m \in [M]$ **do**
4: $\tilde{A}_m \leftarrow$ *MaxFlow*($W_m, T_m, \mathbf{p}$)
5: $A_m \leftarrow$ *StableAdjustment*($W_m, T_m, \tilde{A}_m$)
6: $W_{m+1} \leftarrow W_m - \{\mathbf{w} | \mathbf{w}$ is matched in $A_m\}$
7: **end for**
8: **return** $\mathbf{p}$ and $A_1, \ldots, A_M$

Theorem 15. *OSMUP is polynomial, truthful and IR. OSMUP has* $(2 + 2 \log h)$*-guarantee on revenue if* $\forall j \in [\mathcal{T}]$, $v_j \in [1, h]$.

OSMUP is obviously polynomial and IR. We prove the stability in Lemma 16, the $(2+2 \log h)$-guarantee on revenue in Lemma 17 and the truthfulness in Lemma 18 respectively.

Lemma 16. $(\mathbf{p}, A)$ *is stable where* $\mathbf{p}$ *is the uniform pricing of* 2^k *where* k *is sampled from* $\{0, \ldots, \lfloor \log h \rfloor\}$.

Proof. Let $T(m)$ be the set of all the tasks that $a_j \in [m]$, i.e. $T(m) = \bigcup_{i=1}^{m} T_i$. Let $A(m)$ be the union of allocations for the first m rounds, i.e. $A(m) = \bigcup_{i=1}^{m} A_i$.

To prove the lemma, it is enough to prove the claim that $\forall m \in [M]$, $(\mathbf{p}, A(m))$ is stable w.r.t. W and $T(m)$. According to Theorem 3, it is true that $\forall m \in [M]$, $(\mathbf{p}, A_m)$ is stable w.r.t. W_m and T_m, which immediately implies the case when $m = 1$, i.e. $(\mathbf{p}, A(1))$ is stable w.r.t. W and $T(1)$. We next prove that for any $m = 1, \ldots, M-1$, if $(\mathbf{p}, A(m))$ is stable w.r.t. W and $T(m)$, $(\mathbf{p}, A(m+1))$ is also stable w.r.t. W and $T(m+1)$. For any $(\mathbf{w}, j)$ that $\mathbf{w} \in W$ and $a_j \leq m$, it cannot be a blocking pair. For any $(\mathbf{w}, j)$ that $\mathbf{w} \in W_{m+1}$ and $a_j = m + 1$, it cannot be a blocking pair. It is only possible that there exists a blocking pair $(\mathbf{w}, j)$ that $\mathbf{w} \in W \setminus W_{m+1}$ and $a_j = m+1$. However, $\forall \mathbf{w} \in W \setminus W_{m+1}$, $\mathbf{w}$ is matched

in A_m, so she cannot prefer a different task because of the uniform pricing, thus completing the proof. □

Lemma 17. *OSMUP has $(2+2\log h)$-guarantee on revenue if $\forall j \in [\mathcal{T}]$, $v_j \in [1, h]$.*

Proof. Assuming that we have randomly generated the uniform pricing $\mathbf{p}$. Let $A^*_{\mathbf{p}}$ be the optimal allocatio given $\mathbf{p}$ if all the tasks are present in the first round. So $(\mathbf{p}, A^*_{\mathbf{p}})$ is exactly the offline mechanism (SMUP), which has $(1+\log h)$-gurantee on revenue according to Lemma 8. It is enough to prove this lemma by showing that $|A| \geq \frac{1}{2}|A^*_{\mathbf{p}}|$ where $A = \bigcup_{m=1}^{M} A_m$, which is implied by the following claim (which is proved as Theorem 2.5 in (Manlove et al. 2002)).

Claim 17.1. *(Manlove et al. 2002) For any stable matching problem, the size of the largest matching is at most twice the size of the smallest.*

Since $(\mathbf{p}, A)$ and $(\mathbf{p}, A^*_{\mathbf{p}})$ are both stable w.r.t. W and T, according to Claim 17.1, we have $|A| \geq \frac{1}{2}|A^*_{\mathbf{p}}|$. □

Lemma 18. *Reporting $(\underline{a}_j, \underline{d}_j, \underline{s}_j, \underline{l}_j, v_j)$ truthfully is the dominant strategy for task j.*

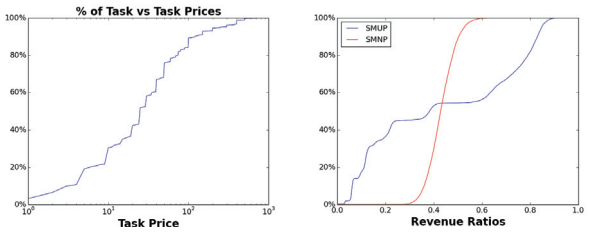
Proof. The proof of the truthfulness of $\underline{s}_j$, $\underline{l}_j$ and v_j is similar to the proof for Lemma 7, thus omitted here. We show the proof of the truthfulness of $\underline{a}_j$ and $\underline{d}_j$. First, reporting $\underline{d}_j$ truthfully cannot make the task worse off because the pricing and allocation are independent of d_j. Due to the assumption of no early arrival, we only consider reporting $a_j \in \{\underline{a}_j + 1, \ldots, \underline{d}_j\}$. Since the pricing is uniform, we only need to show that if reporting $\underline{a}_j$ cannot make j matched, reporting a_j cannot either, which is proved by the following arguments. Since j is not matched in round $\underline{a}_j$, it is true that in $W_{\underline{a}_j}$, there is no worker qualified for j. Since W_m is non-increasing over all the M rounds, there is no qualified worker in $W_{\underline{a}_j+1}, \ldots, W_M$ either. So reporting a_j cannot make j matched if reporting $\underline{a}_j$ cannot. □

Experiments

In this section, we use real price data from Amazon Mechanical Turk and synthetic skill data to evaluate the two offline stable mechanisms SMUP and SMNP, and the online stable mechanism OSMUP.

Dataset. We use the API[8] (Ipeirotis 2010) to crawl 46309 tasks from Amazon Mechanical Turk (HIT) and their prices. We discard the long tail ($< 5\%$), i.e. any task priced no less than 1000 cents. Let $\mathcal{D}$ be the *price pool* containing all the remaining 44043 tasks priced between 1 and 999 cents. We plot the price distribution of $\mathcal{D}$ in Fig 3a where (x, y) denotes that y is the accumulated percentage of tasks priced no more than x cents.

[8]http://www.mturk-tracker.com/#/general



(a) (b)

Figure 3: (a) Distribution of task prices; (b) Distributions of revenue ratios

However, the tasks from AMT do not contain skill or level information. So we create synthetic skills and levels. To randomly generate input instances of the pricing problem, we set h to 999, S to 100, and L to 10 where L is the maximum level of a skill. For each task j, we uniformly and independently sample s_j from $[S]$, l_j from $[L]$ and c_j from the price pool $\mathcal{D}$ with replacement. For each worker, we randomly assign her $k \in \{1, \ldots, 5\}$ skills, each of which is at the level sampled independently from $[L]$. We generate two types of instances, the instance of *small size* where $\mathcal{W} = \mathcal{T} = 1000$, and the instance of *large size* where $\mathcal{W} = \mathcal{T} = 100000$.

SMUP vs SMNP. In this part, we evaluate how the choice of pricing, i.e. uniform and non-uniform, impacts the revenue. In fact, we have not found any polynomial time solution to the pricing problem (i.e. the offline optimization problem without truthfulness). But we find that computing the optimal uniform pricing and its corresponding allocation is an $O(\log \min\{\mathcal{W}, \mathcal{T}\})$-approximation. Clearly, it is not truthful, but produces a revenue no less than the revenue of SMUP which uses randomized uniform pricing. However, this method, as an approximately optimal baseline, needs to run the maxflow in Lemma 4 and the stable-adjustment in Algorithm 1 by $O(\mathcal{T})$ times. So we generate 10000 instances of small size for evaluations.

For each case, we use R^*, R_u and R_n to denote the revenue generated by the baseline, SMUP and SMNP respectively. Let $r_u = \frac{R_u}{R^*}$, and $r_n = \frac{R_n}{R^*}$ be the *revenue ratios* of SMUP and SMNP respectively. We plot the CDFs of the two revenue ratios in Fig 3b where (x, y) denotes that there are y percentages of instances that $r_u \leq x$ (or $r_n \leq x$). The mean values of r_u and r_n over the 10000 cases are very close, i.e. $\mu(r_u) = 0.435$ and $\mu(r_n) = 0.431$, while their standard deviations differ significantly, i.e. $\sigma(r_u) = 0.31$ and $\sigma(r_n) = 0.06$. This means that, although theoretically SMUP has $(1 + \log h)$-guarantee on revenue while SMNP only has $(3 + 3\log h)$-guarantee, in practice, they produce nearly the same revenue. However, SMNP is more robust than SMUP. So we recommend SMNP.

Besides revenues, we also analyze matching size, i.e. the number of matched worker-tasks pairs. We have highly similar observations. Let m_u and m_n be the number of matched worker-task pairs in SMUP and SMNP respectively. We have $\mu(m_u) = 457$, $\mu(m_n) = 445$, $\sigma(m_u) = 382.3$ and $\sigma(m_n) = 39.2$, which again shows that they have similar performance in practice, but SMNP is more robust.

We also analyze the revenue and matching size over 1000 input instances of large size. In such cases, R^* is not able to compute, so we directly compare R_u and R_n. We observe nearly the same patterns: (1) the mean of revenue of SMNP, i.e. $\mu(R_n)$, is slightly ($\approx 0.8\%$) higher than that of SMUP, i.e. $\mu(R_u)$; and (2) the standard deviation of R_u is 9.7 times as the standard deviation of R_n.

SMUP vs OSMUP. In this part, we evaluate the online stable mechanism OSMUP. We use SMUP as the baseline. We consider 1000 instances of large size. Within each instance: we randomly generate the same uniform pricing for both OSMUP and SMUP; we assume that the market runs in 100 rounds, and randomly assign the arrival time $a_j \in [100]$ to each task j. Let R_o denote the revenue produced by OSMUP, and we are particularly interested in the relative ratio $\frac{R_o}{R_u}$ where R_u is revenue of SMUP. Among all the 1000 cases, the least revenue ratio is 0.733; there are 75.0% cases with the ratio at least 0.95; the average ratio is 0.948, much larger than the theoretical guarantee which is 0.5 according to Claim 17.1.

Conclusions

We addressed the pricing problem, in particular, revenue maximizing stable pricing in online labor markets. We presented three efficient truthful stable mechanism with provable guarantees on revenue. We believe that there is a real need to further study and improve mechanisms for pricing and allocation of workers and tasks based on fairness since many online labor markets rely on such mechanisms.

References

Balcan, M.-F.; Blum, A.; and Mansour, Y. 2008. Item pricing for revenue maximization. In *EC*.

Cheung, M., and Swamy, C. 2008. Approximation algorithms for single-minded envy-free profit-maximization problems with limited supply. In *FOCS*.

Crawford, V. P., and Knoer, E. M. 1981. Job matching with heterogeneous firms and workers. *Econometrica*.

Demange, G., and Gale, D. 1985. The strategy structure of two-sided matching markets. *Econometrica*.

Feldman, M.; Fiat, A.; Leonardi, S.; and Sankowski, P. 2012. Revenue maximizing envy-free multi-unit auctions with budgets. In *EC*.

Gale, D., and Shapley, L. S. 1962. College admissions and the stability of marriage. *The American Mathematical Monthly*.

Goel, G.; Nikzad, A.; and Singla, A. 2014. Mechanism design for crowdsourcing markets with heterogeneous tasks. In *HCOMP*.

Gul, F., and Stacchetti, E. 1999. Walrasian equilibrium with gross substitutes. *Journal of Economic theory*.

Guruswami, V.; Hartline, J. D.; Karlin, A. R.; Kempe, D.; Kenyon, C.; and McSherry, F. 2005. On profit-maximizinhard variants of stable marriageg envy-free pricing. In *SODA*.

Gusfield, D., and Irving, R. W. 1989. *The Stable Marriage Problem: Structure and Algorithms*.

Halldórsson, M. M.; Irving, R. W.; Iwama, K.; Manlove, D. F.; Miyazaki, S.; Morita, Y.; and Scott, S. 2003. Approximability results for stable marriage problems with ties. *Theoretical Computer Science*.

Hartline, J., and Yan, Q. 2011. Envy, truth, and profit. In *EC*.

Hatfield, J. W., and Milgrom, P. R. 2005. Matching with contracts. *The American Economic Review*.

Ho, C.-J., and Vaughan, J. W. 2012. Online task assignment in crowdsourcing markets. In *AAAI*.

Ho, C.-J.; Jabbari, S.; and Vaughan, J. W. 2013. Adaptive task assignment for crowdsourced classification. In *ICML*.

Ipeirotis, P. G. 2010. Analyzing the amazon mechanical turk marketplace. *XRDS: Crossroads, The ACM Magazine for Students*.

Irving, R. W. 1994. Stable marriage and indifference. *Discrete Applied Mathematics*.

Iwama, K., and Miyazaki, S. 2008. A survey of the stable marriage problem and its variants. In *ICKS*.

Iwama, K.; Miyazaki, S.; Morita, Y.; and Manlove, D. 1999. Stable marriage with incomplete lists and ties. In *ICALP*.

Iwama, K.; Miyazaki, S.; and Yamauchi, N. 2007. A 1.875: approximation algorithm for the stable marriage problem. In *SODA*.

Kelso Jr, A. S., and Crawford, V. P. 1982. Job matching, coalition formation, and gross substitutes. *Econometrica*.

Király, Z. 2013. Linear time local approximation algorithm for maximum stable marriage. *Algorithms*.

Kovács, P. 2015. Minimum-cost flow algorithms: an experimental evaluation. *Optimization Methods and Software*.

Leonard, H. B. 1983. Elicitation of honest preferences for the assignment of individuals to positions. *The Journal of Political Economy*.

Manlove, D. F.; Irving, R. W.; Iwama, K.; Miyazaki, S.; and Morita, Y. 2002. Hard variants of stable marriage. *Theoretical Computer Science*.

Mcdermid, E. 2009. A 3/2-approximation algorithm for general stable marriage. In *ICALP*.

Nisan, N.; Roughgarden, T.; Tardos, E.; and Vazirani, V. V. 2007. *Algorithmic game theory*, volume 1. Cambridge University Press Cambridge.

Shapley, L. S., and Shubik, M. 1971. The assignment game i: The core. *International Journal of game theory*.

Singer, Y., and Mittal, M. 2011. Pricing tasks in online labor markets. In *HCOMP*.

Singer, Y., and Mittal, M. 2013. Pricing mechanisms for crowdsourcing markets. In *WWW*.

Yin, M., and Chen, Y. 2015. Bonus or not? learn to reward in crowdsourcing. In *IJCAI*.

Yin, M.; Chen, Y.; and Sun, Y.-A. 2013. The effects of performance-contingent financial incentives in online labor markets. In *AAAI*.