Pricing Tasks in Online Labor Markets

Yaron Singer
University of California, Berkeley
yaron@cs.berkeley.edu

Manas Mittal
University of California, Berkeley
mittal@cs.berkeley.edu

Abstract

In this paper we present a mechanism for determining near-optimal prices for tasks in online labor markets, often used for crowdsourcing. In particular, the mechanisms are designed to handle the intricacies of markets like Mechanical Turk where workers arrive online and requesters have budget constraints. The mechanism is incentive compatible, budget feasible, and has competitive ratio performance and also performs well in practice. To demonstrate the mechanism's practical effectiveness we conducted experiments on the Mechanical Turk platform.

1 Introduction

Online labor markets have recently emerged as an effective way to crowdsource cognitive work. Marketplaces such as Amazon's Mechanical Turk (mtu) enable requesters to post tasks that are then performed by hundreds of thousands of geographically distributed workers. One of the main challenge requesters face is in providing appropriate incentives for workers. More concretely, requesters need to establish rewards that are attractive to workers, and yet enable outsourcing a large number of assignments without exceeding the budget.

To price tasks effectively, requesters need to address various challenges unique to online labors markets. For example, requesters are often motivated to pick simple tasks that have a large number of assignments, with an intent to maximize the number of assignments performed. This is in sharp contrast to traditional procurement markets dominated by large contractors who offer expensive, specialized services to address complex tasks and rely on a small number of workers. One step towards creating reasonable incentives is to estimate the workers' costs for performing these experiments. It is difficult to establish a representative cost distribution for these since the worker pool is typically geographically and economically diverse. Also, worker availability changes over time, and the task requester has to account for the online arrival of workers. Additionally, tasks typically don't require specialization but scale. The requesters specify a budget, and will typically wish to maximize the number of assignments performed under the budget.

In this paper we address the problem of pricing tasks in online labor markets by developing mechanisms that iteratively sample and allow work to the workers based on their bids for a given set of tasks. To establish a theoretical foundation, we first describe a natural model that captures the main intricacies of the problem. Our main contribution is a pricing mechanism that dynamically prices and allocates assignments to workers based on their cost. We show that the mechanism has desirable theoretical guarantees and that it also performs well in practice.

The main idea behind the mechanism is to use the bids solicited from workers in an intelligent manner. At every stage the mechanism allocates assignments to an arriving worker only if her cost is below a certain threshold price that has been computed using previous bids, and pays her the threshold price. The threshold price is computed in a manner that guarantees desirable performance properties of the mechanism. The properties of the mechanism are *incentive compatibility*, *budget feasibility* and *competitive ratio* performance, and are based on the theoretical foundations of mechanism design and online algorithms. Informally, *incentive compatibility* ensures workers bid their true costs, *budget feasibility* ensures the requesters budget constraint is not violated, and *competitive ratio* guarantees that in expectation over a random arrival order of workers the mechanism performs close to the optimal solution that would have been obtainable had the requester known the bidders true costs *a priori* to their arrival.

Related Work

There has been significant research in determining the appropriate price for crowdsourced-tasks. For example, (Horton and Chilton 2010) have developed and experimentally validated models for estimating the labor supply for a crowdsourcing task. The hagglebot(Horton and Zeckhauser 2010) haggles with workers to minimize the price paid for completion of tasks. Mechanical turk (mtu) like plaforms also compute and suggest the expected price per worker hour. Interestingly, research (Mason and Watts 2010) indicates that the quality of the work is invariant to the price paid, and that workers might be motivated by different incentives (Shaw, Horton, and Chen 2011). Prediction market approaches (Chen and Vaughan 2010) can also be useful for improving the quality of results.

The problem of designing mechanism for procurement auctions has been extensively studied by the algorithmic game theory community over the past decade. The earlier line of frugality first suggested in (Archer and Tardos 2002) focused minimizing payments for complex objective functions, and is not directly applicable to our setting. Recently, the budget feasibility framework has been initiated in (Singer 2010), where the goal is to design incentive compatible mechanisms that maximize a requester's objective under a budget. The framework has been adopted in to various online settings and we follow it in this paper, though in our model we account for the online arrival of workers, which raises a significant challenge. To address the online arrival of workers we assume that the arrival of workers is random as common in literature (Hajiaghayi, Kleinberg, and Parkes 2004; Babaioff et al. 2008). While there is a growing body of literature on algorithm design in this model for problems like knapsack (Babaioff et al. 2007), assortative matching (Babaioff et al. 2009) and submodular function maximization (Gupta et al. 2010), it has not been studied for procurement, as our setting requires. Since our goal is to design incentive compatible mechanisms, we cannot reward workers their cost, and must compute a price online in a manner that does not exceed the budget.

2 The Model

The task pricing problem can be formally described as follows. There is a single requester who posts a task in an online labor market, and n potential workers for that task, denoted $\mathcal{N}$. In each task there are m available assignments. The workers arrive online in a *random order*[1].

The workers. Each worker a_i associates a *private* cost $c_i \in \mathcal{R}_{\geq 0}$ for performing a single assignment as well as a limit t_i for the number of assignments she is willing to work on. That is, for a worker a_i we do not know *a priori* what c_i, at most t_i assignments can be allocated to a_i and the payment for each assignment must exceed c_i.

The requester. The requester has a public budget $B \in \mathcal{R}_{\geq 0}$ and utility function $f : 2^{[m]} \rightarrow \mathcal{R}_{\geq 0}$ over the subsets of *assignments* performed. In this paper we study the case where f is additive, i.e. $f(S) = \sum_{i \in S} v_i$, where v_i is the value the requester assigns for each assignment. The common case we will discuss most in the paper is where $v_i = 1 \ \forall i \in [m]$ (or equivalently, $f(S) = |S|$), i.e. the requester simply aims to maximize the number of assignments performed [2].

The mechanism. The goal in this setting is to design a mechanism that allocates assignments to workers in a manner that yields a good outcome. A mechanism $\mathcal{M} = (f, p)$ consists of an allocation function $f : \mathcal{R}^n_{\geq 0} \rightarrow 2^{[n]}$ and a payment function $p : \mathcal{R}^n_{\geq 0} \rightarrow \mathcal{R}^n_{\geq 0}$. The allocation function f maps a set of n bids to an allocation of assignments for a selected subset of workers. The payment function p returns a vector $p_1, \ldots, p_n$ of payments to the workers. We are interested in mechanisms where both the allocation and payments functions are polynomial-time computable. We seek normalized ($p_i > 0$ implies a_i is allocated), individually rational ($p_i \geq c_i$) mechanisms with no positive transfers ($p_i \geq 0$). In addition, our objective is to design a mechanism that is:

- **Incentive Compatible.** Since workers may report false costs if it is in their benefit, we seek incentive compatible (truthful) mechanisms for which reporting the true costs is a dominant strategy. Formally, a mechanism $\mathcal{M} = (f, p)$ is *incentive compatible* if for every $a_i \in \mathcal{N}$ with cost c_i and bid c'_i, and every set of bids by $\mathcal{N} \setminus \{a_i\}$ we have $p_i - s_i \cdot c_i \geq p'_i - s'_i \cdot c_i$, where (s_i, p_i) and (s'_i, p'_i) are the allocations and payments when the bidding is c_i and c'_i, respectively.
- **Budget Feasible.** We require the mechanism to be budget feasible: the mechanism's payments do not exceed the budget: $\sum_i p_i \leq B$.
- **Competitive.** The goal of the mechanism is to maximize the expected value of the requester's objective function, where the expectation is over the random arrival order of the workers. In the simple case we want to maximize the expected number of assignments performed. To quantify the performance of the mechanism we compare its solution with the *optimal solution*: the solution obtainable in the offline scenario where all costs are known. A mechanism is $O(g(n))$-competitive if the ratio between the *optimal* solution and the online mechanism is $O(g(n))$. Ideally, we would like our mechanism to be $O(1)$-competitive.

It might seem that achieving all three desiderata simultaneously is too much to hope for. As shown in (Singer 2010) where budget feasibility is introduced, designing incentive compatible mechanisms whose sum of payments are under the budget is not trivial, and there are simple instances where this objective cannot be achieved. In addition, we require our mechanism to perform well despite the random arrival order of the workers. Despite these challenges, the framework for designing mechanisms in this environment we present in this paper allows satisfying these conditions simultaneously.

3 The Mechanism

A pricing mechanism for online labor markets needs to overcome several nontrivial challenges. First, the workers' costs are unknown and need to be elicited in a manner that incentivizes truthful reporting, without having the total payments exceed the requester's budget. In addition, the mechanism needs to accommodate for the online arrival of the workers. The standard approach to achieve desirable outcomes in online settings is via *sampling*: the first batch of the input is

[1] Due to the impossibility results for arbitrary orderings, we use the standard secretary model where the assumption is that the order or arrival is not adversarial but random, as common in literature of dynamic mechanism design (Babaioff et al. 2007)

[2] Our model extends to general requester functions that are not necessarily additive, and we discuss these general cases further in the paper

rejected and used as a sample which enables making an informed decision on the rest of the input. Although the model we use here assumes the workers' arrival order is random and is not controlled by the workers, a standard sampling approach may be impractical: workers are likely to be discouraged to work on tasks knowing the pricing mechanism will automatically reject their bid.

To address the above challenges, we use the following approach. At each stage the mechanism maintains a *threshold price* which is used to decide whether to accept the workers' bids. The mechanism dynamically increases the sample size and updates the threshold price, while increasing the budget it uses for allocation. As a result, workers are not automatically rejected during the sampling, and are allocated when their cost is below the established threshold price. The threshold prices are set in such a way that ensures budget feasibility and incentive compatibility.

As a first step, we describe the procedure used to establish threshold price, and discuss some of its properties.

GetThreshold

Require: B, S

initialize: sort S s.t. $c_1 \leq \ldots \leq c_{|S|}$

while $c_i \leq \frac{B}{\sum_{j \leq i} t_j}$ **do:**

$i \longleftarrow i + 1$

$t_i \longleftarrow \min\left\{t_i, \left\lfloor \frac{B - c_i \cdot \sum_{j<i} t_i}{c_i} \right\rfloor\right\}$

Output: $\min\left\{\frac{B}{\sum_{j \leq i} t_j}, \frac{c_{i+1}}{t_{i+1}}\right\}$

The above procedure receives a workers' bid profile S and a budget B. The bid profile describes, for each worker, the minimal cost they associate for performing a single assignment and the maximal number of assignments they are willing to work on. The procedure computes a *threshold price*: a single price-per-task that can used to decide whether to accept or reject a bid. Since workers can work on fewer assignments then the maximal number of assignments they indicated, the procedure allows for *fractional* solutions: Inside the iterative loop, the procedure updates the number of assignments to be the minimum between the number of assignments the worker requested and the number of assignments that are available, given the worker's cost and the remaining budget.

Before we describe how the mechanism uses the above procedure, we prove a desirable property of the threshold prices. The proof of the lemma below can be easily derived from properties of the *proportional share* mechanism for the symmetric submodular case that are shown (Singer 2010), and is presented here for completeness.

Lemma 3.1. *Given a fixed budget and and bid profile of costs and maximal assignments workers are willing to perform, let t^* be the maximal number of assignments that can be performed. Then at least $\frac{t^*}{2}$ of the assignments have cost below the threshold price computed by the* `GetThreshold` *procedure.*

Proof. Given a fixed budget, the optimal solution is consider workers according to a minimal cost ordering and allocate t_i assignments to each worker a_i as long as the total cost does not exceed the budget and use the remaining budget to the last worker. For a given set of workers $a_1, \ldots, a_n$, assume, w.l.o.g. that $c_1 \leq c_2 \leq \ldots \leq c_n$. Let a_ℓ be the last worker allocated by the optimal solution and let a_k be the last worker who can be allocated by the above procedure. For purpose of this analysis we assume, w.l.o.g, that a_k and a_ℓ have been allocated all their assignments and that $t_{k+1} = 1$. We can assume this since otherwise, if only a fraction of t_k assignments are allocated to a_k, we can consider an equivalent instance where we have $n + t_k + t_{k+1} + t_\ell - 3$ workers, which consists of the original $\mathcal{N} \setminus \{a_k, a_{k-1}, a_\ell\}$ workers, and for each $i = k, k+1, \ell$ we replace a_i with t_i workers, each with cost c_i, willing to work on at most one assignment.

First observe that $c_{k+1} \cdot \sum_{i=k+1}^{\ell} t_i \leq \sum_{i=k+1}^{\ell} c_i \cdot t_i < B$ and therefore $c_{k+1} < \frac{B}{\sum_{i=k+1}^{\ell} t_i}$. One the other hand, we know that $c_{k+1} > \frac{B}{\sum_{i \leq k+1} t_i}$. We can therefore conclude that $\sum_{i=1}^{k+1} t_i > \sum_{i=k+1}^{\ell} t_i$ and in particular, since $t_{k+1} = 1$ we have that: $\sum_{i=1}^{k} t_i \geq \frac{1}{2} \sum_{i=1}^{\ell} t_i$. □

The above lemma implies that if we accept all assignments that have cost that is smaller than the threshold price, we will be able to complete at least half of the assignments we would have been able to complete if we paid each worker their cost. Intuitively, as one can imagine, once the sample size is large enough, the threshold prices obtained on the sample will be "close enough" to the real threshold price. We give a a formal description of our mechanism below.

An Online Mechanism for Task Pricing

Require: B, n

initialize:

$(n', B', p, S, i, A) \leftarrow (\frac{n}{2^{\lfloor \log n \rfloor}}, \frac{B}{2^{\lfloor \log n \rfloor}}, \epsilon, \{a_1\}, 1, \emptyset)$

while $i \leq n$ **do:**

if $i = n'$

$p^* \longleftarrow$ `GetThreshold`(B', S)

$(n', B', p, A) \longleftarrow (2 \cdot n', 2 \cdot B', p^*, \emptyset)$

if $c_i \leq p$

$t'_i \longleftarrow \min\left\{t_i, \left\lfloor \frac{B' - (\sum_{a_j \in A} t_j)}{p} \right\rfloor\right\}$

Allocate t'_i assignments, pay p for each

$S \longleftarrow S \cup \{(c_i, t_i)\}$

$i \longleftarrow i + 1$

Output: S

The mechanism initially sets a small threshold, sample size and budget. At each stage $i \in \{1, 2, \ldots, \lfloor \log n \rfloor\}$, the mechanism updates its threshold price by calling the `GetThreshold` procedure with the bids it has sampled

thus far. For every bidder that appears, the mechanism allocates tasks to the bidder as long as her cost is below the threshold price established, and the budget allocated for the stage hasn't been exhausted. In the last stage $i = \lfloor \log n \rfloor$, the threshold price has been computed for a sample of size $\frac{n}{2}$, and the budget used is $\frac{B}{2}$. We will now prove some of its desirable properties.

Lemma 3.2. *The mechanism is budget feasible.*

Proof. At each stage $i \in \{1, 2, \ldots, \lfloor \log n \rfloor\}$, we set a budget $B' = B_i$ and threshold price p_i. The condition of the mechanism is such that the number of assignments allocated at stage i does not exceed $\frac{B_i}{p_i}$, and since each assignment is rewarded p_i, we have budget feasibility in each stage i. Since $\sum_i B_i = \sum_{i=1}^{\lfloor \log n \rfloor} \frac{B}{2^i} < B$ the mechanism is budget feasible. □

Proposition 3.3. *The mechanism is incentive compatible.*

Proof. Consider a worker a_i with cost of c_i that arrives at some stage for which the threshold price was set to p. If by the time the worker arrives there are no remaining assignments, then the worker's cost declaration will not affect the allocation of the mechanism and thus cannot benefit for reporting a false cost. Otherwise, assume there are remaining assignments by the time the worker arrives. In case $c_i \leq p$, reporting any cost below p wouldn't make a difference in the worker's allocation and payment and her utility for each assignment would be $p - c_i \geq 0$. Declaring a cost above p would deny the worker from any tasks, and her utility would be 0. In case $c_i > p$, declaring any cost above p would leave the worker unallocated with utility 0. If the worker declares a cost lower than p she will be allocated. In such a case, however, her utility will be negative and she is thus better off reporting her true cost. □

To show the mechanism performs well, we will make the simplifying assumption that all workers have the same limit on the number of assignments they can perform.

Theorem 3.4. *For sufficiently large n, when all workers have the same assignment limit, the mechanism is $O(1)$-competitive.*

Proof. In the last stage of the mechanism the budget used is $\frac{B}{2}$ and the sampled subset is of size $\frac{n}{2}$. Since all workers have the same limit, we can assume the limit is 1 (otherwise normalize the budget). Let p be the threshold price computed by `GetThreshold` in the offline case (the price that is computed when all n workers are considered), and let T be the set of workers in $\mathcal{N}$ with cost below p. Note that $p \leq \frac{B}{|T|} \leq p'$ where p' is the threshold price computed from the *sample*. The number of workers from T in the sample follows a hypergeometric distribution, and for sufficiently large n, the probability that there are $\frac{|T|}{2}$ workers from T in the sample is close to $\frac{1}{2}$. In this case $p' \leq \frac{2B}{|T|}$, and we have $\frac{|T|}{2}$ workers not in the sample, who have cost below $p \leq p'$. If a worker from T who is not in the sample is not allocated by our mechanism, it can therefore only be because the budget has been exhausted. Since $p' \leq \frac{2B}{|T|}$ with probability close to $\frac{1}{2}$, at least $\frac{|T|}{2}$ workers have been allocated.

The mechanism thefore returns at least half of the offline solution that is computed with budget $\frac{B}{2}$. From the above lemma, and the fact that we are using a constant fraction of the budget, we get that expectation the mechanism provides a constant factor approximation to the optimal offline optimal solution, and is therefore $O(1)$-competitive. □

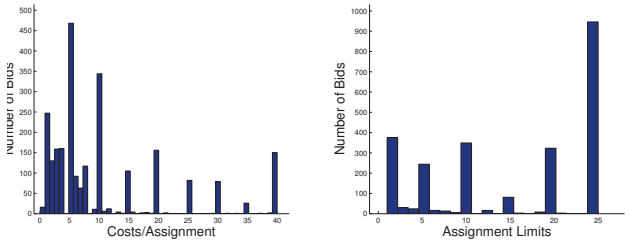

Figure 1: (a) Histogram of price per assignment (0-40 cents) requested by bidders(b) Histogram of number of assignments requested by bidders

4 Experimental Evaluation

Beyond its theoretical guarantees, we would like to show that the mechanism performs well in practice. In addition, we would like to show that allowing workers to express their costs is a feasible approach that can be easily implemented. For this purpose we created an experiment on Mechanical Turk where workers were asked to bid on assignments we created and the mechanism was used for allocation. We found that the mechanism does very well on real inputs and the threshold prices converge quickly. We also found that workers produced high quality results and importantly responded positively to the bidding method. We describe the experiment in detail below followed by analysis and discussion of the main results.

Description of the Experiment

The primary objective of the experiment was to collect bids in order to compare the mechanism's performance on descriptive inputs against various benchmarks. Our second objective was to test whether the bidding method has an effect on workers' performance. To conduct the experiment we created a web application that generates human computation assignments. We implemented the bidding mechanism in the application so that assignments were automatically generated to workers who were allocated. We used the Mechanical Turk to direct workers to the application, by posting it as a standard Human Intelligence Task (HIT) available to workers on the Mechanical Turk platform. This allowed us to collect bids from workers and measure the performance of the allocated workers on the assignments.

In the Human Intelligence Task (HIT) workers were explained that a mechanism will decide how many assignments, if any, will be allocated to them based on their bid. We explained to workers they would be paid through the

Mechanical Turk bonus payment system, which allows a requester to pay workers beyond the fixed price associated with the HIT. To encourage high quality work, we explained to workers they would be paid if their work will be found unsatisfactory. We also included a screenshot from an example assignment so that workers could assess their cost for performing the assignment prior to bidding. Following the set of instructions, workers had to indicate their cost for performing an assignment and how many assignments they wish to perform. Their bids were collected by our mechanism that then decided on their allocation. A worker that was allocated received assignments to work on, and based on the pricing decided by the mechanism was paid within a few days via the bonus payment system. Each worker that placed a bid was paid for participating in the HIT, independent of the payments made according to the mechanism's decision. In the control of the experiment, workers were only asked to indicate the number of tasks they wish to work on for a fixed price of $0.03.

The assignments required workers to estimate area sizes in pie charts. Each assignment included six pie charts, where each pie chart consisted of three colors, one of which was red. In each assignment, the workers were required to estimate the percentage of red color in each one of the six pie charts. The area sizes in the pie charts were randomly generated. We chose this assignment since it simulates a human computation assignment and allows us to objectively quantify a worker's performance. As common on the Mechanical Turk platform we ran the experiment in several HIT batches to promote it in the list of list of available HITs presented to workers. The payment for participation was $0.05. We also gave workers an option to send us feedback about their experience. We limited the experiment to workers with approval rate higher than 90%. At the time of writing, the application is accessible (Singer and Mittal).

Experimental Results

There were 2255 bids where each bid consisted of the maximal number of assignments they are willing to work on and their cost for performing an assignment. We restricted workers to 25 assignments and their bid per assignment had to be below $0.40. In total, there were 2215 legal bids (there were 40 *illegal* bids we discarded where the bid was below $0.01 or above $0.40). We plot the cost and requested assignment distributions in Figure 1 below.

Although the main measure of performance we consider in this paper is the number of assignments that can be performed under the budget, we examined the quality of the work performed as well. Showing that workers perform well on their allocated assignments helps exclude concerns regarding negative effects the bidding method may have. To examine the performance of workers we chose the percentage estimation assignment since it allows us to objectively quantify workers' performance by considering their errors from the true answer. In total, our mechanism allocated to 161 workers who, in aggregate, submitted 10870 answers (we count the number of answers submitted for each pie chart).

The error distribution is presented in Figure 2(a). In

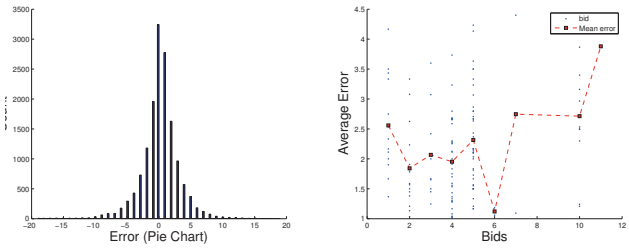

Figure 2: (a) Histogram of errors. (b) Average error (Y) vs bid price in cents (X)

general, workers performed well on the assignments. The worker mean error was 2.37, and almost all workers who were allocated assignments by the mechanism completed their assignments. This was consistent with the control group of 32 workers who received a fixed price reward, where the mean error was 2.64. This leads us to assume that the performance is not negatively affected by the bidding method.

A subject of ongoing debate in the crowdsourcing community is the relationship between performance and monetary incentives. To examine this in our context we compared a worker's average error on the assignments performed against their bid. The average error reflects on the quality of work, and the bid indicates the reward the worker expects to receive. We plot the worker's bid against their average error in Figure 2(b). In our examination we found no significant correlation. We note the data for this comparison involves only 161 workers, since this is the total number of workers who were allocated assignments by the mechanism.

To test the performance of the mechanism, we used the 2215 bids collected and compared our mechanism against several benchmarks. Note that in order to show how many assignments can be allocated given a specified budget, we only require the workers' bids, which is a much larger set than the subset of workers that were actually allocated and submitted their answers to the assignments. To simulate a task we use a random permutation of the bids we collected to model the random arrival of workers, and run our mechanism with a specified budget over this ordering.

First, to see how quickly the threshold prices of the mechanism converge, we simulated tasks with various budgets and ran the mechanism. In Figure 3(a) we plot the threshold price against the number of workers that appear on a logarithmic scale, when a budget of $1000 was used. As one can see, the threshold price quickly stabilizes and remains almost constant throughout the run.

We compared our mechanism against the following benchmarks. The first benchmark is the optimal *offline* algorithm which has *full knowledge* about workers costs. This is the benchmark we used to analytically compare our mechanism's performance in the above section. The second benchmark is the *proportional share* mechanism. This mechanism is the optimal offline incentive compatible and budget feasible mechanism which was introduced in (Singer 2010). Essentially, the proportional share allocation mech-

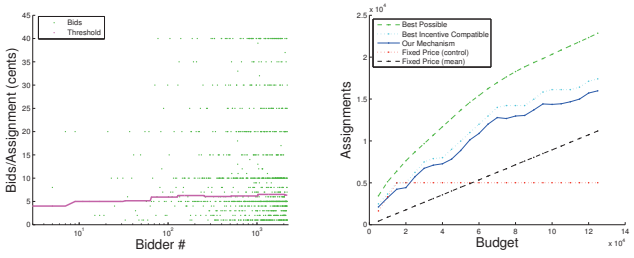

Figure 3: (a) Variation of threshold price over time (b) Comparative performance of algorithms

anism is the basis for `GetThreshold` procedure we use in our mechanism. This mechanism does not have knowledge about worker's costs, but it is an offline mechanism, i.e., all workers submit their bids to the mechanism and wait for the mechanism to collect all the bids and decide on an allocation. The other benchmarks are fixed price mechanisms where a single price is offered to agents, and if their cost is below that price they accept and otherwise reject. We used fixed price mechanisms with \$0.03 as used in the control, and \$0.01128 which is the mean of the bids.

The first two benchmarks operate in simpler settings, where all the costs are known *a priori* and will therefore always outperform our mechanism. We showed that the mechanism is guaranteed to be, in expectation over the arrival order of the workers, within a constant factor from the optimal offline solution. Our goal in this experiment is to examine this ratio on descriptive inputs. The fixed price benchmark that uses the mean of the bids also assumes the costs are known, and while it performs arbitrarily bad in the worst case, the a priori knowledge of the costs can potentially allow it to outperform our mechanism. In Figure 3(b) we plot the resulting comparison between our mechanism and the various benchmarks. Using the bids provided by workers, we simulated the different algorithms on budgets ranging from \$50 to \$1000 in increments of \$50.

The mechanism performs quite well in comparison with the various benchmarks. Although it is only guaranteed to be within a competitive factor of about 16 of the optimal offline solution, the experiments show that this ratio is almost as small as two. As compared to the proportional share mechanism, this ratio decreases below two. The mechanism that uses the mean of bids as its fixed price is a clear lower bound. Lastly, it appears the naïve solution of posting fixed prices even with knowledge of the workers' costs results in poor performance in practice as well. The fixed price benchmarks of \$0.03 levels off as the number of workers with bids smaller than \$0.03 is finite, and exhausted by a large enough budget.

5 Discussion

The mechanism we presented in this paper has provable theoretical guarantees and performs well in practice. Importantly, the pricing mechanism suggested here is simple to implement. While it can be implemented by a platform for online labor markets as an alternative to the commonly-used fixed-price method, it can be implemented by requesters as well. In a similar manner that we executed the experiment described in this paper, a requester can implement the mechanism and apply it to price its tasks. We intend to make this task simple for requesters and include an application through which requesters could post tasks on Mechanical Turk in a manner that will automatically implement the mechanism.

The model seems adequate for online labor markets, and we believe there is room for future that will use this model. In general, requesters may have objectives that are more complex, and it would be interesting to extend our results for such cases as well.

Acknowledgment

We would like to thank Björn Hartmann for valuable discussions and advise. Y.S. is supported by the Microsoft Research fellowship and the Facebook fellowship.

References

Archer, A., and Tardos, É. 2002. Frugal path mechanisms. In *SODA*, 991–999.

Babaioff, M.; Immorlica, N.; Kempe, D.; and Kleinberg, R. 2007. A knapsack secretary problem with applications. In *APPROX-RANDOM*, 16–28.

Babaioff, M.; Immorlica, N.; Kempe, D.; and Kleinberg, R. 2008. Online auctions and generalized secretary problems. *SIGecom Exchanges* 7(2).

Babaioff, M.; Dinitz, M.; Gupta, A.; Immorlica, N.; and Talwar, K. 2009. Secretary problems: weights and discounts. In *SODA*, 1245–1254.

Chen, Y., and Vaughan, J. W. 2010. A new understanding of prediction markets via no-regret learning. EC '10, 189–198.

Gupta, A.; Roth, A.; Schoenebeck, G.; and Talwar, K. 2010. Constrained non-monotone submodular maximization: Offline and secretary algorithms. In *WINE*, 246–257.

Hajiaghayi, M. T.; Kleinberg, R. D.; and Parkes, D. C. 2004. Adaptive limited-supply online auctions. In *ACM Conference on Electronic Commerce*, 71–80.

Horton, J. J., and Chilton, L. B. 2010. The labor economics of paid crowdsourcing. EC '10, 209–218.

Horton, J. J., and Zeckhauser, R. J. 2010. Algorithmic wage negotiations: Applications to paid crowsourcing. *CrowdConf*.

Mason, W., and Watts, D. J. 2010. Financial incentives and the "performance of crowds". *SIGKDD Explor. Newsl.* 11:100–108.

Amazon mechanicalturk https://www.mturk.com.

Shaw, A. D.; Horton, J. J.; and Chen, D. L. 2011. Designing incentives for inexpert human raters. CSCW '11, 275–284.

Singer, Y., and Mittal, M. Mechanical turk task (http://50.56.98.120:3000/bidders/new).

Singer, Y. 2010. Budget feasible mechanisms. In *FOCS*, 765–774.