# Pricing Mechanisms for Crowdsourcing Markets *

Yaron Singer
Google, Inc.
Mountain View, CA 94043 USA
yarons@google.com

Manas Mittal
UC Berkeley
Berkeley, CA 94709
mittal@cs.berkeley.edu

## ABSTRACT

Every day millions of crowdsourcing tasks are performed in exchange for payments. Despite the important role pricing plays in crowdsourcing campaigns and the complexity of the market, most platforms do not provide requesters appropriate tools for effective pricing and allocation of tasks.

In this paper, we introduce a framework for designing mechanisms with provable guarantees in crowdsourcing markets. The framework enables automating the process of pricing and allocation of tasks for requesters in complex markets like Amazon's Mechanical Turk where workers arrive in an online fashion and requesters face budget constraints and task completion deadlines. We present constant-competitive incentive compatible mechanisms for maximizing the number of tasks under a budget, and for minimizing payments given a fixed number of tasks to complete. To demonstrate the effectiveness of this framework we created a platform that enables applying pricing mechanisms in markets like Mechanical Turk. The platform allows us to show that the mechanisms we present here work well in practice, as well as to give experimental evidence to workers' strategic behavior in absence of appropriate incentive schemes.

## Categories and Subject Descriptors

H.0 [**Information Systems**]: General

## General Terms

Algorithms, Economics, Human Factors, Theory

## Keywords

Crowdsourcing, Mechanism Design, Human Computation, Mechanical Turk, Mechanical Perk

## 1. INTRODUCTION

The advancement of the internet in the past decade created a platform for a new form of labor markets known as *crowdsourcing markets* where cognitive work can be distributed to hundreds of thousands of geographically disparate workers. In contrast to traditional procurement markets that rely on specialized contractors, requesters in crowdsourcing markets typically outsource large quantities of simple tasks to anonymous, unspecialized workers. Typical examples of crowdsourcing tasks include image labeling, sentiment analysis, content generation, listing verification, image moderation, transcription, and other forms of tasks that are impossible, difficult or too expensive to automate.

There are several crowdsourcing platforms that provide workers with non-monetary incentives like entertainment [9], educational opportunities [5], information [27], and altruism [17] in exchange for their efforts. Despite the success of these platforms, it is often difficult to engineer non-monetary incentive schemes for tedious and repetitive work. Therefore an overwhelming majority of crowdsourcing tasks are performed in exchange for payments [1, 4, 8, 2, 6, 3, 7]. In such markets, implementing a campaign successfully requires pricing and allocating tasks effectively.

Designing effective pricing and allocation schemes presents a challenging problem due to requesters' constraints and the realities of crowdsourcing markets. Requesters often face task completion deadlines and budget constraints, and must account for dramatic elasticity in the workforce supply. Furthermore, there is a large variance in effort required to complete different tasks, which also largely depends on the skills and background of workers who are often based in multiple geographical locations. Despite this difficulty in pricing tasks, most crowdsourcing markets provide requesters surprisingly limited tools for pricing tasks effectively.

In this paper we address the problem of pricing and allocating tasks in crowdsourcing markets. We develop a theoretical framework and design mechanisms that work well in practice and have provable guarantees. In addition, we describe a platform which we implemented that enables requesters to automate the process of pricing in crowdsourcing markets using the mechanisms we present here as well as other pricing schemes. The framework is primarily designed for tasks where the quality of the worker's performance does not yield additional utility to the requester once above a certain quality threshold. Although there are crowdsourcing tasks that do not fall into this category like predicting future events, designing a logo, or writing an introduction to an academic paper, a large fraction of the work in crowdsourcing markets typically fits this criterion. For such tasks requesters often use various methods to ensure the threshold quality is met like injecting gold standards [29], majority voting, and more sophisticated cross-validation methods [31, 36, 30, 35, 28]. Since workers receive payments after requesters' approval in crowdsourcing platforms, we assume

*Preliminary results were presented at the *AAAI workshop on Human Computation (HCOMP) 2011*


*WWW 2013*, May 13–17, 2013, Rio de Janeiro, Brazil.
ACM 978-1-4503-2035-1/13/05.

requesters have access to such verification schemes and focus on efficiently pricing and allocating tasks, independent of their quality.

We take a mechanism design approach to the pricing problem and enable workers to bid on work by expressing their cost for performing tasks and the number of tasks they wish to perform. Although most crowdsourcing platforms do not provide workers such level of expressiveness, existing APIs make this feature easy to integrate into most platforms as we further describe. This relatively minor modification enables designing powerful mechanisms.

The mechanisms we present here are designed for two main objectives: maximizing the number of tasks performed under budget, and minimizing payments for a given number of tasks. We consider requesters that impose a deadline for the completion of tasks and workers who arrive i.i.d. according to some known distribution and can strategically misreport their cost or number of tasks they wish to perform. We therefore design *incentive compatible* mechanisms that ensure that the allocation and pricing are such that it is in every worker's best interest to bid truthfully.

## 1.1 Related Work

The problem of designing mechanisms for pricing tasks in crowdsourcing markets has been addressed using different models and different techniques, such as bargaining between requesters and workers to minimize work [26] and recently using bandit algorithms to maximize tasks [34]. While both are natural approaches, they leave room for frameworks that allow better theoretical guarantees as used here. In [24] the authors study an orthogonal problem and present an algorithmic framework for matching workers with with requesters based on their skills.

The approach in this paper advocates for eliciting workers costs via incentive compatible protocols. A different approach is that of developing a model for workers' effort and learning its parameters from data as done in [25, 13]. The problem of designing mechanisms for procurement has been extensively studied by the algorithmic game theory community over the past decade. The earlier line of frugality first suggested in [10] focuses on minimizing payments for complex objective functions. Recently, the budget feasibility framework has been initiated in [32], where the goal is to design incentive compatible mechanisms that maximize a requester's objective under a budget. The framework has been adapted to various settings [33, 19, 16, 15, 18] and we follow it in this paper.

In our model we account for the online arrival of workers, which raises a significant challenge. There is substantial literature on online mechanism design where workers arrive according to a given distribution [21, 22, 23, 12, 11]. In our case we consider mechanisms for *buying* items (rather then selling) from strategic agents which requires different machinery. In [14] the authors study online procurement though the emphasis there is on a different model of posted prices.

# 2. MODEL

In our model we consider a single requester and multiple workers. The requester has a task completion deadline $T \in \mathbb{N}_+$ by which his tasks need to be allocated. Each worker arrives at some time step $t \in \{1, \ldots, T\}$ i.i.d. according to some distribution. Each worker $a_i$ associates a cost $c_i \in \mathbb{R}_+$ for performing a single task posted by the requester and a number of tasks she can complete $v_i \in \mathbb{N}_+$. The requester does not know $c_i, v_i$, or the total number of workers that will appear by $T$. We assume the requester knows the distribution of the arrival of workers.[1] The requester either aims to maximize the number of tasks allocated under some budget $B \in \mathbb{R}_+$, or alternatively complete $L \in \mathbb{N}_+$ tasks while minimizing payments, in which case we assume $\sum_{i \in N} v_i \geq 100L$.

One can adversarially assume that the workers know $B, L, T$ and the objective of the requester. When allocated $\bar{w}_i$ tasks at $p_i$ per task, the worker's utility is simply $\bar{w}_i(p_i - c_i)$. We assume that workers are rational in that they wish to maximize their utility.

**The protocol.** At the time of her arrival, each worker $a_i$ submits a bid indicating her cost per task and the maximal number of tasks she wishes to perform, denoted as $b_i$ and $w_i$ respectively, which do not necessarily correspond to $c_i, v_i$. The mechanism must decide how many tasks to allocate to the worker and the price for each task upon her arrival, after processing her bid. The bid on the maximal number of tasks is abiding and if a worker completes less than the number of tasks allocated to her, the requester does not pay the worker.

## 2.1 Design Objectives

Our goal is to design mechanisms that perform well. A mechanism is simply an algorithm that decides how many tasks each worker performs and how she is paid. Since workers may report false costs, we will seek incentive compatible (truthful) mechanisms for which reporting the true costs is a dominant strategy. Formally, a mechanism is *incentive compatible* if for every $a_i \in N$ with cost and limit $c_i, v_i$ and bid $b_i, w_i$, and every set of bids by $N \setminus \{a_i\}$ we have $x_i(p_i - c_i) \geq x'_i(p'_i - c_i)$, where $p_i, p'_i$ are the payments and $x_i, x'_i$ indicate the number of tasks $a_i$ is allocated, when bidding $c_i, v_i$ and $b_i, w_i$, respectively. Since an incentive compatible mechanism guarantees that the bids are truthful, its performance over the bids can be compared against a theoretically optimal algorithm that knows workers' true values.

**Maximizing Tasks.** Under this objective we are given a fixed budget by the requester and seek to maximize the expected number of tasks performed without exceeding the budget, where the expectation is over the random arrival order of the workers and the randomization of the mechanism. This design objective is known as *budget feasiblity* where the mechanism must be designed so that the sum of its payments (and not workers' costs) does not exceed the budget [32]. To quantify the performance of the mechanism we compare its solution with the *optimal offline solution*: the solution that would have been obtainable if all workers' true values were known in advance and we could pay each worker exactly her cost. Note that this is the most demanding benchmark possible. A mechanism is $O(g(n))$-competitive if the ratio between the benchmark and the expected value guaranteed by the mechanism is $O(g(n))$. Ideally, we would like our mechanism to be $O(1)$-competitive.

**Minimizing Payments.** A complementary objective to maximizing expected number of tasks under budget is that of minimizing expected payments for a given number of tasks. Ideally, we would like the total payments the mechanism makes to be comparable to the minimal cost required

[1]For our performance guarantees we actually only need to know the *median* of this distribution.

for performing the task. It is easy to show however that no incentive compatible mechanism can perform well under this benchmark.[2] We therefore set our objective for minimizing payments as follows. Given a fixed number of tasks to perform, $L$, we say that a mechanism for minimizing payments is $\alpha$-competitive if it allocates $L$ tasks in expectation and is guaranteed to pay no more than the minimum cost required to complete $\alpha L$ tasks in the offline scenario when all costs are known. Here as well our goal is to design mechanisms that are $O(1)$-competitive.

# 3. THE PRICING MECHANISMS

The common approach to achieve desirable outcomes in online settings is to observe a fraction of the input and use it as a sample to make an informed decision on the rest of the input. We will use a similar strategy, though rather than rejecting workers whose bids are used as a sample, we will allocate tasks to workers in the sample. When our objective is to maximize the number of tasks performed, our mechanism dynamically learns an appropriate price for a task as it allocates tasks to workers. In the case of minimizing payments, the requester can specify a budget for allocating tasks during the sampling phase.

In essence, both mechanisms sample the input until the median time step – the time step at which each worker appears with probability 1/2 – and use the sample to estimate a *threshold price*. The mechanisms then use the threshold price to decide which bids to accept or reject. Estimating appropriate threshold prices is an important building block in both mechanisms, and we begin by describing this procedure and its properties.

## 3.1 Establishing Threshold Prices

Intuitively, the threshold price is the lowest single price we can offer which many workers will accept. The procedure for computing threshold prices presented below is a variant of the proportional share mechanism introduced in [32], which serves as the basis for designing procurement mechanisms under budget [33, 19, 16, 15, 18]. Note that this is an *offline* procedure that has access all bids.

**GetThreshold**

**input:** bids $\{(b_1, w_1), \ldots, (b_m, w_m)\}$, Budget $B$

1. initialize: sort bids s.t. $b_1 \leq b_2 \leq \ldots \leq b_m$; set $i = 1$
2. while $b_i \leq \frac{B}{\sum_{j<i} \bar{w}_j + 1}$
   set: $p = b_i$,
   $\bar{w}_i = \min\{w_i, \lfloor \frac{B}{p} \rfloor - \sum_{j<i} \bar{w}_j\}$,
   $i = i + 1$;

**output:** $p$

[2]Consider an instance where a single worker can complete all required tasks at some small cost $\epsilon$, while every other workers has some very high costs $P$. It is not hard to show that any incentive compatible mechanism that completes all required tasks must pay at least $P$ in such a case, thus making the ratio between minimal cost and the total payments of an incentive compatible mechanisms unbounded.

When tasks are priced at $p = b_i$ the remaining budget is $B - p\sum_{j=1}^{i-1} \bar{w}_j$, and therefore the number of tasks she could be allocated at this price without exceeding the total budget is $\bar{w}_i = \min\{w_i, \lfloor \frac{B}{p} \rfloor - \sum_{j<i} \bar{w}_j\}$ as used in the procedure. The desirable property of this threshold price is that it sets a single price which on the one hand is low enough so that it efficiently exhausts the budget and on the other hand is high enough so that enough workers accept. We formalize this in the following lemma. The proof in similar to [32].

LEMMA 3.1. *For a given sample of bids, let $L$ be the maximal number of tasks that can be allocated under a given budget. Then, at least $L/2$ tasks can be allocated under budget at the price computed by the* GETTHRESHOLD *procedure.*

PROOF. Let $a_i = (b_i, w_i)$ and $\{a_1, a_2, \ldots, a_k\}$ be the set of bids allocated by the procedure. First, observe that $\bar{w}_i = w_i$ for all $i < k$, as otherwise given an $i < k$ s.t. $w_i > \bar{w}_i = \lfloor \frac{B}{p} \rfloor - \sum_{j=1}^{i-1} \bar{w}_j$ we have:

$$\sum_{j=1}^{i} \bar{w}_j = \sum_{j=1}^{i-1} \bar{w}_j + \Big( \lfloor \frac{B}{p} \rfloor - \sum_{j=1}^{i-1} \bar{w}_j \Big) = \lfloor \frac{B}{p} \rfloor$$

and thus:

$$b_{i+1} \geq b_i = p > \frac{B}{\lfloor B/p \rfloor + 1} = \frac{B}{\sum_{j<i} \bar{w}_j + 1}$$

which implies that $b_{i+1}$ violates the condition in step 2 of the procedure, in contradiction to the assumption that it was allocated. For the purpose of this analysis, without loss of generality we can assume that $\bar{w}_k = w_k$. This is due to the fact that we can consider an input identical to ours except that $a_k$ is replaced with two bids $a_{k'} = (b_k, \bar{w}_k)$ and $a_{k''} = (b_k, w_k - \bar{w}_k)$. The value over this input is identical to the value obtained over the original input, and $a_{k''}$ is not allocated as it fails to meet condition in step 2 due to the same argument made above.

Given a fixed budget, to achieve the maximal number of tasks one must allocate to lowest bids until exhausting the budget, and thus $\{a_1, \ldots, a_k\}$ are included in the optimal solution. Let $\{a_1 \ldots, a_k, \ldots, a_r\}$ be the optimal solution. Assume for purpose of contraction that $2\sum_{i=1}^{k} w_i < \sum_{i=1}^{r} w_i$. Then, $\sum_{i=1}^{k} w_i < \sum_{i=k+1}^{r} w_i$. By this assumption and the definition of $b_{k+1}$ we have that:

$$b_{k+1} > \frac{B}{\sum_{i=1}^{k+1} w_i + 1} \geq \frac{B}{\sum_{i=k+1}^{r} w_i}$$

which implies:

$$B < b_{k+1} \cdot \Big( \sum_{i=k+1}^{r} w_i \Big) \leq \sum_{i=k+1}^{r} w_i \cdot b_i$$

but since the optimal solution cannot exceeds the budget, we have a contradiction. □

The above lemma suggests 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. As one might imagine, once the sample size is large enough, the threshold prices obtained on the sample will be a good estimate to the real threshold price as if they were computed offline. We formalize this argument and give a rigorous proof in the following section.

## 3.2 Maximizing Tasks

Given a distribution on the arrival of workers, we can easily compute every $2^i$ quantile (i.e. the time step $t$ s.t. the probability that a worker arrives before $t$ is $2^{-i}$). We therefore assume the mechanism is given $\{q_1, \ldots, q_\ell\}$ where $q_\ell$ is the first time step $t = 1$. We give a formal description of the mechanism below followed by a brief explanation in plain English. We use $S(t)$ to denote the set of all bids that arrived at and before time step $t$.

**MaximizeTasks**

**input:** Budget $B$, quantiles $\{q_1, \ldots, q_\ell\}$

1. initialize: $B' = 2^{-(\ell+1)}B, t = 1$;

2. For every quantile $q_j, j = \ell, \ell-1, \ldots, 1$ do:
   a. at time step $t = q_j$ set:
   $p = \text{GetThreshold}(S(t), 2B')$,
   $w^* = \min\{\max_{\{a \in S(t) | c_i \leq p\}} w_i, \lfloor \frac{2B'}{p} \rfloor\}$,
   $B' = 2B', A = \emptyset$;

   b. with probability 1/3 do:
   while $q_j < t \leq q_{j+1}$:
   for all $a_i$ who arrive at time $t$ s.t. $b_i \leq p$ do:
   allocate $\bar{w}_i = \min\{w_i, \lfloor \frac{B'}{p} \rfloor - \sum_{r \in A} \bar{w}_r\}$ at $p$,
   set $A = A \cup \{i\}$;

   with probability 2/3 do:
   for first $a_i$ arriving by $q_{j+1}$ s.t. $w_i \geq w^*$:
   allocate $\bar{w}_i = \min\{w_i, \lfloor \frac{B'}{p} \rfloor\}$ at $p$;

The mechanism iterates over $q_1, \ldots, q_\ell$ and at every time step $q_i$ it uses a budget of $B/2^i$ to allocate tasks, and decides whether to accept a bid using a threshold price which it computes on bids of all workers that arrived by time step $q_i$. At every time interval $q_i$ the mechanism randomly selects the procedure it will use on all workers that arrive until $q_{i+1}$. The first procedure pays each worker the threshold price per task, as long as her cost is below the threshold and the budget has not been exhausted. The second procedure allocates its entire budget to the worker that bid below the threshold price and is willing to perform at least as many tasks as $w^*$. The randomization between the two procedures handles extreme cases in which only a single worker can complete a large fraction of the tasks at the threshold price. The way which we find the worker that can complete the maximal number of tasks is an incentive compatible variant of Dynkin's celebrated algorithm to the problem of hiring the best secretary [20], tailored to our setting.

LEMMA 3.2. *The mechanism is incentive compatible, i.e. it is in every worker's best interest to bid her true cost for performing an assignment, and the number of assignments she wishes to perform.*

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 bid will not affect the allocation of the mechanism and thus she cannot benefit by reporting a false cost. Otherwise, assume there are remaining assignments by the time the worker arrives. In case $c_i \leq p$, bidding 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 being allocated, 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 though her utility will be negative. In the realization where all workers with costs smaller than $p$ are allocated until exhausting the budget, declaring a lower or higher number of tasks does not benefit the worker since her utility is linear. In the realization where we select the first worker with bid smaller than $p$ who will complete at least $w^*$ tasks, if the worker declares $w_i' \geq \min\{w, \lfloor B/p \rfloor\} > w_i$ the worker will be allocated more tasks than she can perform and will be paid 0, and will therefore not benefit. □

LEMMA 3.3. *The mechanism is budget feasible, i.e. the sum of the payments that the mechanism makes to all workers never exceeds the given budget.*

PROOF. At each stage $i = 1, \ldots, \ell$ the mechanism uses a budget of $B' = B/2^i$ and threshold price $p$ computed from the bids of the previous round, and allocates no more than $\lfloor B'/p \rfloor$ tasks. Therefore every iteration is budget feasible and in total $\sum_{i=1}^{\ell} B/2^i < B$. □

LEMMA 3.4. *The* MaximizeTasks *mechanism is* 360 *competitive and* 120*-competitive when using its entire budget in the median time step.*

PROOF. We will analyze the iteration of the mechanism, when the sample consisted of all workers who arrive by the median time step and the budget used for allocation was $B/2$. We will compare the expected number of tasks allocated with the number of tasks that would have been possible to allocate with the same budget on the entire set of workers at a single threshold price.

Let $S(T)$ be the set of all workers who arrive by time step $T$, and consider running the GetThreshold procedure on $S(T)$ using a budget of $B' = B/2$; let $A$ be the set of all workers who were allocated by this procedure, $k = |A|$, $W = \sum_{i \in A} \bar{w}_i$, $p$ be the threshold price obtained by running the procedure, and $OPT$ be the maximal number of tasks that can be performed under budget $B$ on $S(T)$. Note that the maximal number of tasks that can be performed under budget $B/2$ is at least $OPT/3$, and therefore from Lemma 3.1 we have that $W \geq OPT/6$.

Assume first that $\max_i w_i \leq W/10$. Consider the median time step $t$ and all workers bids sampled until this time step, $S(t)$. Let $A_1 = S(t) \cap A, A_2 = A \setminus A_1$, and $W_1 = \sum_{i \in A_1} \bar{w}_i, W_2 = \sum_{i \in A_2} \bar{w}_i$. Since each worker $a_i \in A$ arrives before the median with probability 1/2 we can associate a random variable $X_i$ that takes a value of $\bar{w}_i$ with probability 1/2 and 0 otherwise. To evaluate the expected value of $W_1$ and $W_2$, we can use the following version of the Chernoff bound:

THEOREM 3.5. *(Chernoff Bound) Let $X_1, \ldots, X_k$ be a set of $k$ independent random variables that take values in $[0, w_i]$*

*and* $\mu = \mathbb{E}[\sum_{i=1}^{k} X_i]$. *Then, for any* $\delta \in [0,1]$ *we have that:*

$$Pr\Big[\sum_{i=1}^{k} X_i > (1+\delta)\mu\Big] \leq \Big(\frac{e^{\delta}}{(1+\delta)^{(1+\delta)}}\Big)^{\frac{\mu}{\max_i w_i}}$$

$$Pr\Big[\sum_{i=1}^{k} X_i < (1-\delta)\mu\Big] \leq e^{\frac{-\delta^2\mu}{2\max_i w_i}}$$

This above bound implies that:

$$Pr\Big[W_2 < \frac{1}{4}W\Big] = Pr\Big[W_1 > \frac{3}{4}W\Big] \leq 59/100$$

$$Pr\Big[W_1 < \frac{1}{4}W\Big] \leq 21/100$$

Therefore, by union bound with probability at least 1/5 both $W_1, W_2 \geq W/4$. Now, let $p(T), p(t)$ be the thresholds computed using the GetThreshold procedure over $S(T)$ and the sample $S(t)$, respectively. Since $p(t) \geq p(T)$ as it is computed over a smaller subset, for each worker $a_i \in A_2$ it follows that $c_i \leq p(t)$ and they will be allocated if the budget has not yet been exhausted. If all workers in $A_2$ were allocated by the mechanism this implies that at least $W/4$ tasks were performed since $W_2 \geq W/4$. If the budget was exhausted before all workers in $A_2$ arrived, a total of $\lfloor B'/p(t) \rfloor$ were performed. Since

$$p(t) \leq \frac{B'}{W_1} \leq \frac{B'}{W}$$

this implies that at least $W/4$ tasks were performed with probability at least 1/5. Therefore, since this procedure is realized with probability 1/3 at least $W/60$ were completed in expectation when $\max_{i \in A} w_i \leq W/10$.

In case $\max_{i \in A} \bar{w_i} > W/10$, let $a = \text{argmax}_{i \in S(T)} w_i$, and $b = \text{argmax}_{i \in S(T) \setminus \{a\}} w_i$. With probability 1/4, $b$ appears at or before the median time step and $a$ arrives after the median time step, we therefore have at least one worker $a_i$ at stage $t > q_1$ s.t. $w_i \geq w^*$. In this case we have that:

$$\bar{w_i} = \min\{\max_i w_i, \lfloor B/p(t) \rfloor\} \geq W/10$$

since $p' \leq 4B/W$. Since this procedure is realized with probability 2/3 we have obtain at least:

$$\frac{W}{10} \cdot \frac{1}{4} \cdot \frac{2}{3} = \frac{W}{60}$$

tasks in expectation, as in the previous case. Since $W \geq OPT/6$ the mechanism is 360-competitive. If the entire budget is used in step $q_1$, $W \geq OPT/2$ and the mechanism is 120-competitive. □

Theorem 3.6. *The* MaximizeTasks *mechanism is incentive compatible, budget feasible and* $O(1)$*-competitive.*

While the constants may seem large, we emphasize that our goal is to show that the mechanisms are indeed $O(1)$-competitive, and thus that their guarantee is independent of the parameters of the problem that can be large (e.g. number of workers, their cost, the number of tasks they are willing to perform, etc.). We will later show that these mechanisms perform well in practice, implying that bounded competitive ratio serves as a good guide for designing such mechanisms.

## 3.3 Minimizing Payments

The idea behind the mechanism for minimizing payments is based on the following observation. Given a fixed number of tasks $L$, if we knew the minimal cost for performing $L' = 2L$ tasks, we could use a procedure similar to GetThreshold with this minimal cost as its budget. From Lemma 3.1 we know that the GetThreshold procedure finds a price s.t. at least $L'/2 = L$ tasks could be performed. Therefore, such a procedure would be a 2-approximation to the minimal cost in our case. The MinimizePayments mechanism is based on this idea: we compute the minimal cost for performing a constant blowup of the number of tasks required, and find an appropriate threshold price. We describe the mechanism formally below, followed by a brief description.

**MinimizePayments**

**input:** number of tasks $L$, budget $\beta$, price $\delta$, $q_1, T$

1. For all $a_i$ who arrive at time $t \leq q_1$, if $b_i \leq \delta$ do:
   allocate $\bar{w_i} = \min\{w_i, \lfloor \frac{\beta}{\delta} \rfloor - \sum_{r \in A} \bar{w_r}\}$ at $\delta$,
   set $A = A \cup \{i\}$;

2. At time step $t = q_1$ do:
   set $B = \text{FindMinCost}(S(t), 2L)$,
   set $p = \text{GetThreshold}(S(t), B)$,
   $w^* = \min\{\max_{\{a \in S(t) | c_i \leq p\}} w_i, \lfloor \frac{B}{p} \rfloor\}$

3. with probability 1/4 do:
   for all $a_i$ who arrive at time $t > q_1$ s.t. $b_i \leq p$ do:
   allocate $\bar{w_i} = \min\{w_i, \lfloor \frac{B}{p} \rfloor - \sum_{r \in A} \bar{w_r}\}$ at $p$,

   with probability 3/4 do:
   for first $a_i$ arriving by $T$ s.t. $w_i \geq w^*$:
   allocate $\bar{w_i} = \min\{w_i, \lfloor \frac{B}{p} \rfloor\}$ at $p$;

The above mechanism has two iterations. The first iteration samples the bids, and uses a given budget of $\beta$ and price $\delta$ specified by the requester to allocate to workers in the sample.[3] After the median time step, the $\text{FindMinCost}(S(t), 2L)$ procedure finds the minimal cost for performing $2L$ tasks (which can be done by a simple greedy algorithm which sorts workers according to their costs and allocates tasks until reaching the number of tasks required). The threshold price is then computed using the minimal cost as its budget, together with an estimate of the maximal number of tasks a worker can perform. Similarly to the MaximizeTasks mechanism, the mechanism then randomizes between a procedure which allocates tasks to workers with price smaller than $p$ and a procedure that allocates all tasks to a single worker.

The properties of the mechanism can be proven using similar ideas as in the proofs in the previous section. We state the theorem below, and leave the proof to the full version of the paper.

[3]In the MaximizeTasks mechanism we automatically used half of the budget for the sampling phase. Here, since we do not a priori know what workers' costs are, we leave $\beta$ and $\delta$ as a design choice to the requester.

THEOREM 3.7. *Given a fixed number of tasks $L$, the* MINIMIZEPAYMENTS *mechanism is incentive compatible, allocates $L$ tasks in expectation and is $O(1)$-competitive.*

# 4. EXPERIMENTS

To evaluate the performance of the mechanisms in practice and explore bidding behavior in crowdsourcing markets, we created the *Mechanical Perk* platform which enables us to conduct experiments with workers from Mechanical Turk and to collect real bidding data and observe their behavior.

## 4.1 Mechanical Perk

To enable implementing various mechanisms on Mechanical Turk (MTurk) we created the Mechanical Perk (MPerk) platform. The platform provides a service for requesters who wish to post Human Intelligence Tasks (HITs) with various automated pricing and incentive mechanisms. The platform receives the HIT from the requester as input and their choice for the mechanism they wish to use, along with additional information like the number of HITs to be posted, budget, expiration date, and other parameters for posting the HIT on MTurk. The platform then posts a HIT on MTurk that serves as a wrapper for the requester's original HITs. The HIT posted by MPerk enables workers to place bids and run a mechanism in the background.

In the Human Intelligence Task (HIT) workers are explained that a mechanism will decide how many assignments, if any, will be allocated to them based on their bid. We explain 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 explain to workers they would not be paid if their work will be found unsatisfactory. We also include 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 need to indicate their cost for performing an assignment and how many assignments they wish to perform. Their bids are collected by a mechanism which decides 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.

An important fact is that MPerk can use various pricing mechanisms that allow efficient allocation of tasks. In our experiments we used MAXIMIZETASKS as well as other simple pricing schemes to gain insight to bidding behavior in crowdsourcing markets. We describe these in detail in Section 4.5.

## 4.2 Experiments Objectives

We conducted two main sets of experiments on MPerk. The primary goals were to evaluate the performance of the online mechanism on real bids as well as to test workers' responses to different bidding mechanisms. We implemented the bidding mechanisms through MPerk and for Human Computation tasks, we used a batch of automatically generated assignments.

**Performance.** We conducted an experiment where we ran the MAXIMIZETASKS on MechanicalPerk with a modest budget. We primarily used this process to collect bids so we can run simulations with different budgets to observe the performance of the mechanism. In general, we found that the mechanism performs very well on real inputs and the threshold prices converge quickly.

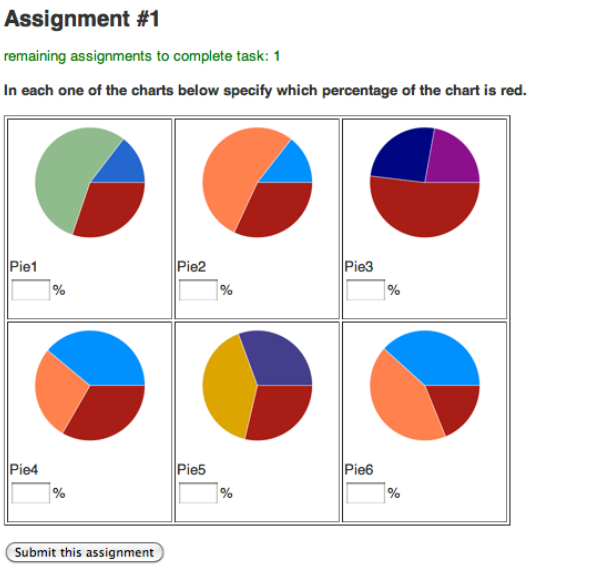


**Figure 1: Screenshot of example assignment used in the experiments.**

**Bidding behavior.** The main goal in these experiments was to examine workers' responses to various features in the mechanism which could serve as guidelines for future design of mechanisms in crowdsourcing markets. An important guideline in our design is incentive compatibility as we assume workers behave strategically in crowdsourcing platforms. To examine this we observed workers' response to different pricing schemes which suggest that they indeed strategize their bids. Another design principle is to avoid rejecting workers automatically. The main reasoning is that we believe that although rejecting workers automatically to obtain a sample will not hurt the mechanism during its iteration, rejecting bids automatically is not likely to be sustainable in crowdsourcing markets, as workers will avoid tasks that use such mechanisms for pricing. In our experiments we show evidence of this as well.

## 4.3 Experimental Setup

In the first experiment we ran the MAXIMIZETASKS mechanism described in Section 3 primarily to collect bids that we used to test its performance, and allowed workers to bid only once. In the second experiment we tested workers' responses to four different pricing mechanisms. We allowed workers to bid up to 15 times to observe their responses to various pricing schemes.

We limited the experiment to workers with approval rate on Mechanical Turk higher than 90%. We recorded workers' IP addresses and treated each IP session as a new worker. While this does not guarantee the worker is a different person, we used this as a reasonable proxy. In total we collected 1674 bids, from 764 different workers, allocated 3883 assignments and collected 23298 answers.

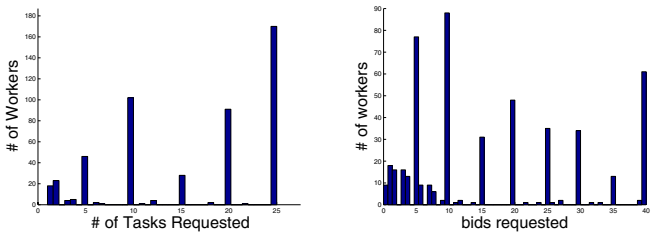


**Figure 2: (a) Histogram of price per assignment (0-40 cents) requested by bidders (b) Histogram of number of assignments requested by bidders**

**The Human Computation Tasks.** We used assignments that 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 and these area sizes were randomly generated. The reason for choosing this assignment is that it simulates a human computation task and allows to quantify a worker's performance objectively. We also gave workers an option to send us feedback about their experience. An example assignment is displayed in Figure 1.

## 4.4 Performance

To collect data for simulations we ran an experiment on Mechanical Perk where MaximizeTasks was used to allocate tasks. To collect a representative data set for the simulations, we allowed workers to bid once in this experiment. The maximal allowed bid was \$0.40 and the limit on the number of assignments was set to 25. We collected bids from 391 workers, each providing a single bid which indicates their cost and the number of assignments they wish to perform. The mean bid was 16.33 cents and the mean number of assignments was 16. We plot the distribution of bids and number of assignments in Figure 2.

To test the performance of the MaximizeTasks mechanism, we used the 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.

We compared MaximizeTasks against two benchmarks. The first benchmark is the optimal *offline* algorithm which has *full knowledge* about workers costs. The second benchmark is the GetThreshold procedure applied offline. This procedure is guaranteed to be within a factor of two of the first benchmark by Lemma 3.1 and is also the optimal incentive compatible solution due to a matching lower bound [32]. This mechanism does not have knowledge about workers' true 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. These benchmarks operate in simpler settings, where all the costs are known *a priori* and will therefore always outperform our mechanism. Ideally, we would be able to compare against other pricing methods such as those used by commercial platforms, though this data is difficult to obtain.

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 was to examine this ratio on descriptive inputs. Using the bids provided by workers, we simulated the different algorithms on budgets ranging from \$50 to \$1000 in increments of \$50. In Figure 3 we plot the resulting comparison between our mechanism and the benchmarks.

On the simulated data, the mechanism performs quite well. Analytically, we guarantee that the mechanism has a constant competitive factor in comparison to the optimal offline solution, and the experiments show that this ratio is almost as small as 2. In comparison to the best incentive compatible mechanism, this ratio is substantially smaller, and there is almost no difference in the performance of the two mechanisms. The simulations suggest that the MaximizeTasks has near optimal performance in practice.

To examine the change in the threshold prices as the number of workers increases in the sample, we simulated tasks with various budgets and ran the mechanism. We observed that in all simulations, the threshold prices converged quickly, and typically after running 16 and 32 bids varied by 1 to 2 cents. In Figure 3(a) we plot the value of the threshold price as a function of the stage of the mechanism (the number of workers that submitted their bids) on a logarithmic scale, during a simulation that used a budget of \$100. As one can see, the threshold price quickly stabilizes and remains almost constant throughout the run.

## 4.5 Bidding Behavior

To observe workers' responses to various features in pricing mechanisms we experimented with four simple mechanisms and allowed workers' to bid multiple times in order to observe how features of the mechanisms affect their bidding. All mechanisms required workers to bid how much they wish to be rewarded for each assignment they complete and the number of assignments for this bid. We allowed only 5 assignments to be performed per HIT (where the bid is per assignment), allowed bids no higher than \$0.50, and allowed workers to perform 15 HITs.[4] For each HIT performed the workers received a \$0.03 fixed reward for participating, even when their bid was rejected. All information was clearly indicated in the instructions. The workers were not notified what the pricing scheme was, and to avoid giving them information about the mechanism, if their bid was accepted, the payment they received was their bid. We used the following pricing schemes:

- **Always Win Mechanism:** This mechanism accepted workers' bids as long as the total payment to the worker did not exceed \$3.00. The fact that there is a budget or that it was exceeded was not revealed to the workers. Once meeting their budget workers were allowed to continue bidding but their bids we rejected.
- **Always Lose Mechanism:** This mechanism implements a fixed price mechanism with threshold price of \$0, i.e. workers were rejected regardless of their bids.

[4]The only exception was the Always Win mechanism described below where the limit on the bid was \$0.99 and we set a spending budget on each worker.

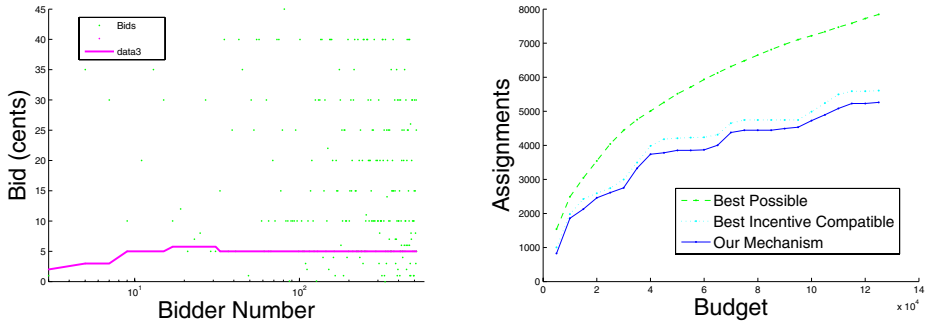


**Figure 3: (a) Variation of threshold price (pink line) over time (b) Comparative performance of algorithms**

- **Fixed Price Mechanism:** This mechanism uses a fixed threshold price that is not revealed to the workers. Each bid at or below the threshold price was accepted and otherwise rejected. In our experiment we set this threshold price to be \$0.04.
- **Random Price Mechanism:** This mechanism randomizes over different threshold prices each time a worker bids. If the bid is below the price the worker was allocated and otherwise their bid was rejected.

We also ran a control mechanism which presented workers with a fixed price of \$0.03 per task and required workers to reveal how many assignments they wish to perform. As in the above pricing schemes the limit was set to 5 assignments per HIT, and workers were allowed to perform 15 HITs.

There were 1033 bids (including 138 in the control) from 378 different workers in total (including 66 in control) in this experiment. Each bid consisted of the maximal number of assignments the worker is willing to perform in the bidding round and their cost for performing an assignment. In total, there were 952 valid bids, including 131 in the control (there were 81 *invalid* bids we discarded where the bid cost or number of assignments violated our instructions).

Recall that in this experiment a worker was given a chance to bid 15 times (rounds), and if their bid was accepted they worked on the tasks and received their bid as payment. Figure 4(a) plots the average bid at a given round for each pricing scheme. This figure is complemented by Figure 5 which gives a histogram of the number of workers that remained in each round. In the AlwaysLose pricing scheme, for example, there were only 2 workers after the eighth round, and the average price shown in Figure 4(a) is an average of these two bids. Up until the forth round there were 10 and 11 bidders in AlwaysLose and RandomPrice, respectively, and 16 in both AlwaysWin and FixedPrice. It therefore seems that the majority of the information in Figure 4(a) is the first 5 bids.

**Evidence of strategic bidding.** To examine whether workers include strategic considerations in their bidding, one can observe the obvious difference between the plots of the different responses to the pricing schemes as shown in Figure 4(a). Bidders in the AlwaysWin scheme, increased their bids as they got accepted (the following drop off is due to the budget constraint we enforced, for methodological reasons). In the AlwaysLose or RandomPrice schemes where workers bids were rejected, bids were lowered. We see this as clear evidence that when given an opportunity workers' will declare false costs if they believe this will increase their profit. We see this as strong support for insisting on *incentive compatible* mechanisms.

Interestingly, although a budget constraint was implemented in the AlwaysWin scheme and workers were automatically rejected, their bids were still significantly higher than those of other pricing schemes.

**Effects of rejection.** To observe this effect on workers, in Figure 4(b) we plot the mean of the success rate of workers (the number of bids that were accepted) vs. the number of bidding rounds they participated in for the RandomPrice and the FixedPrice mechanisms. [5] Although there seems to be a negative correlation between success and number of rounds, one must remember that there is very little data (in the sixth round there were 12 and 8 workers in the FixedPrice and RandomPrice mechanisms, respectively). Better evidence for whether rejection affects workers can be seen in Figure 5, where we plot the number workers in each bidding round for each pricing scheme. There are evident drop offs in the RandomPrice and AlwaysLose mechanisms, where almost no bidders stayed beyond 6 rounds. Note that this is despite the \$0.03 they received simply for placing a bid. This strengthens the claim that workers will avoid a HIT if they know they will be automatically rejected, even if they are paid to bid. Even when there is no monetary loss, there is a price associated with sampling in crowdsourcing platforms which can result in slower completion times for a batch of HITs.

**Quality of work.** 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).

[5]We only plot the results for these mechanisms as there are no successes in the AlwaysLose mechanism and also in the AlwaysWin mechanism once workers exceeded their budget.

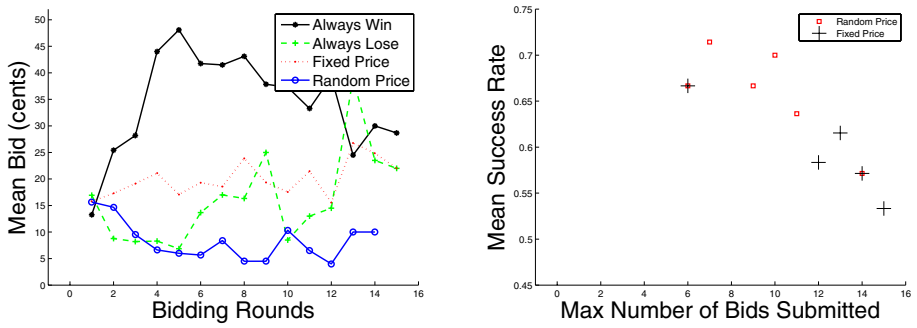


**Figure 4: (a) Bids as a function of number of rounds (b) Success rates of workers in Fixed Price and Random mechanisms. All workers who left before the sixth round had all their bids rejected and their success rate of zero is not shown.**

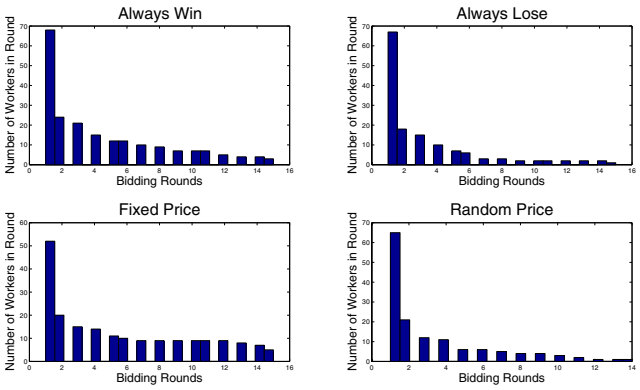


**Figure 5: Number of workers in each bidding round.**

The error distribution is presented in Figure 6. The error (vertical axis) is the difference between the workers guess and the actual marked percentage of red in the pie chart task. In general, workers performed well on the assignments. The worker mean error was 2.57, and almost all workers who were allocated assignments completed them. This was consistent with the control group of 66 workers who received a fixed price reward, where the mean error was 2.59. This implies that performance is not negatively affected by bidding.

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 mean error on the assignments performed against their bid. The mean 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 mean error in Figure 6(b). In our examination we found no significant correlation. We note that the data for this comparison involves 271 workers, since this is the total number of workers who were allocated assignments by the mechanism.

## 5. DISCUSSION

In this paper we present a framework that enables designing mechanisms for crowdsourcing markets with provable guarantees. The mechanisms we presented are easy to implement, have strong theoretical guarantees, and perform well in practice. From the experimentation on the platform, it seems there is evidence for strategic behavior and negative effects when automatically rejecting workers. We believe this evidence strengthens our model and assumptions and should be taken into account when designing pricing schemes in crowdsourcing markets.

We believe the model provides a good basis for designing pricing mechanisms for crowdsourcing markets, and that it can be further extended. A natural extension of this model could incorporate verification schemes and automatic quality control that could integrate with the pricing mechanisms presented here.

## 6. ACKNOWLEDGMENTS

We would like to thank Björn Hartmann for valuable discussions and advice. Part of this work was done while the first author was at UC Berkeley and supported by the Microsoft Research fellowship and the Facebook fellowship. We would also like to thank our anonymous reviewers for their thoughtful comments and suggestions.

## 7. REFERENCES

[1] Amazon mturk https://www.mturk.com.
[2] Clickworker. https://www.crowdflower.com.
[3] CloudCrowd. https://www.cloudcrowd.com.
[4] Crowdflower. https://www.crowdflower.com.
[5] duolingo. http://duolingo.com/.
[6] Microtask. http://microtask.com.
[7] MobileWorks. https://www.mobileworks.com.
[8] oDesk. https://www.odesk.com.
[9] L. v. Ahn. Games with a purpose. *Computer*, 39(6):92–94, June 2006.
[10] A. Archer and É. Tardos. Frugal path mechanisms. In *SODA*, pages 991–999, 2002.
[11] M. Babaioff, N. Immorlica, D. Kempe, and R. Kleinberg. A knapsack secretary problem with applications. In *APPROX-RANDOM*, pages 16–28, 2007.
[12] M. Babaioff, N. Immorlica, D. Kempe, and R. Kleinberg. Online auctions and generalized secretary problems. *SIGecom Exchanges*, 7(2), 2008.

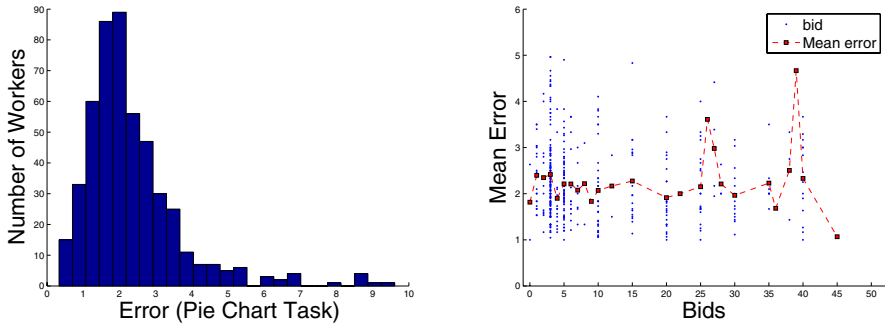


**Figure 6: (a) Histogram of errors. (b) Average error (Y) vs bid price in cents (X)**


[13] D. F. Bacon, D. C. Parkes, Y. Chen, M. Rao, I. A. Kash, and M. Sridharan. Predicting your own effort. In *AAMAS*, pages 695–702, 2012.

[14] A. Badanidiyuru, R. Kleinberg, and Y. Singer. Learning on a budget: Posted price mechanisms for online procurement. In *ACM Conference on Electronic Commerce*, pages 128–145, 2012.

[15] X. Bei, N. Chen, N. Gravin, and P. Lu. Budget feasible mechanism design via random sampling. *CoRR*, abs/1107.2994, 2011.

[16] N. Chen, N. Gravin, and P. Lu. On the approximability of budget feasible mechanisms. 2011.

[17] S. Cooper, F. Khatib, I. Makedon, H. L§, J. Barbero, D. Baker, J. Fogarty, Z. Popovic, and F. players. Analysis of social gameplay macros in the foldit cookbook. In M. Cavazza, K. Isbister, and C. Rich, editors, *FDG*, pages 9–14. ACM, 2011.

[18] P. Dandekar, N. Fawaz, and S. Ioannidis. Privacy auctions for recommender systems. *WINE*, 2012.

[19] S. Dobzinski, C. Papadimitriou, and Y. Singer. Mechanisms for complement-free procurement. In *EC*, 2011.

[20] E. B. Dynkin. The optimum choice of the instant for stopping a Markov process. *Soviet Math. Dokl*, 4, 1963.

[21] E. J. Friedman and D. C. Parkes. Pricing wifi at starbucks: Issues in online mechanism design. In *ACM Conference on Electronic Commerce*, pages 240–241, 2003.

[22] M. T. Hajiaghayi. Online auctions with re-usable goods. In *Proceedings of the 6th ACM conference on Electronic commerce*, EC '05, pages 165–174, New York, NY, USA, 2005. ACM.

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

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

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

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

[27] S. Jain, Y. Chen, and D. C. Parkes. Designing incentives for online question and answer forums. In *Proceedings of the 10th ACM conference on Electronic commerce*, EC '09, pages 129–138, New York, NY, USA, 2009. ACM.

[28] D. R. Karger, S. Oh, and D. Shah. Iterative learning for reliable crowdsourcing systems. In *NIPS*, pages 1953–1961, 2011.

[29] A. Kittur, E. H. Chi, and B. Suh. Crowdsourcing user studies with mechanical turk. In *Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, CHI '08, pages 453–456, New York, NY, USA, 2008. ACM.

[30] V. C. Raykar, S. Yu, L. H. Zhao, G. H. Valadez, C. Florin, L. Bogoni, and L. Moy. Learning from crowds. *J. Mach. Learn. Res.*, 11:1297–1322, Aug. 2010.

[31] V. S. Sheng, F. J. Provost, and P. G. Ipeirotis. Get another label? improving data quality and data mining using multiple, noisy labelers. In *KDD*, pages 614–622, 2008.

[32] Y. Singer. Budget feasible mechanisms. In *FOCS*, pages 765–774, 2010.

[33] Y. Singer. How to win friends and influence people, truthfully: Influence maximization mechanisms for social networks. In *WSDM*, pages 733–742, 2012.

[34] L. Tran-Thanh, S. Stein, A. Rogers, and N. R. Jennings. Efficient crowdsourcing of unknown experts using multi-armed bandits. In L. D. Raedt, C. BessiŕRe, D. Dubois, P. Doherty, P. Frasconi, F. Heintz, and P. J. F. Lucas, editors, *ECAI*, volume 242 of *Frontiers in Artificial Intelligence and Applications*, pages 768–773. IOS Press, 2012.

[35] P. Welinder, S. Branson, S. Belongie, and P. Perona. The multidimensional wisdom of crowds. In *NIPS*, pages 2424–2432, 2010.

[36] J. Whitehill, P. Ruvolo, T. Wu, J. Bergsma, and J. Movellan. Whose vote should count more: Optimal integration of labels from labelers of unknown expertise. *Advances in Neural Information Processing Systems*, 22:2035–2043, 2009.