Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence

Novel Mechanisms for Online Crowdsourcing with Unreliable, Strategic Agents

Praphul Chandra[1,2], **Yadati Narahari**[1], **Debmalya Mandal**[3,*], **Prasenjit Dey**[4,†]
[1]Indian Institute of Science, Bangalore, India, [2]Hewlett Packard, Bangalore, India,
[3]SEAS, Harvard University, Cambrige, MA, USA, [4]IBM Research, Bangalore, India,
[*]Work done while at IISc, [†]Work done while at Hewlett Packard.

Abstract

Motivated by current day crowdsourcing platforms and emergence of online labor markets, this work addresses the problem of task allocation and payment decisions when unreliable and strategic workers arrive over time to work on tasks which must be completed within a deadline. We consider the following scenario: a requester has a set of tasks that must be completed before a deadline; agents (aka crowd workers) arrive over time and it is required to make sequential decisions regarding task allocation and pricing. Agents may have different costs for providing service and these costs are private information of the agents. We assume that agents are not strategic about their arrival times but could be strategic about their costs of service. In addition, agents could be unreliable in the sense of not being able to complete the assigned tasks within the allocated time; these tasks must then be reallocated to other agents to ensure on-time completion of the set of tasks by the deadline. For this setting, we propose two mechanisms: a DPM (Dynamic Price Mechanism) and an ABM (Auction Based Mechanism). Both mechanisms are dominant strategy incentive compatible, budget feasible, and also satisfy ex-post individual rationality for agents who complete the allocated tasks. These mechanisms can be implemented in current day crowdsourcing platforms with minimal changes to the current interaction model.

Introduction

We consider the problem of allocation and pricing of tasks in a crowdsourcing setting where agents arrive over time, so as to ensure that all tasks can be completed before a deadline. The costs of service are private information of the agents and the agents may be strategic about revealing this information. Furthermore, an agent assigned a task may not complete it (*unreliable agents*) and these incomplete tasks must then be reallocated to other agents to ensure on-time completion. Though motivated by crowdsourcing marketplaces like Amazon Mechanical Turk (AMT), this setting models has significance for any reverse (procurement) auction setting where agents commit to a time and may not deliver.

Allocation and pricing decisions in this setting are challenging due to the following reasons. *Unreliable Agents*: An agent who is allocated a task may not complete it; hence the task may need to be reallocated. *Online Arrivals*: Agents arrive over time and may leave; allocation and payment decision must be made without information about future arrivals. *Time Sensitive* : There is a deadline before which the set of tasks must be completed. *Strategic Agents*: An agent's cost is private information and he may not reveal it truthfully.

The strategic nature of agents clearly calls for a mechanism design approach. Consider a requester who posts time-sensitive tasks on a crowdsourcing platform. The arrival of agents who accept the task is a stochastic process. At any point before the deadline, the requester needs a mechanism to decide how many tasks to allocate to an agent and what payments to make for these allocated tasks. Even though we assume that only the costs of the agents are private information and the arrival time of the agents is common knowledge, the mechanism design problem becomes non-trivial due to the unreliable agents: when the requester values on-time completion, a mechanism must take into account the value of potential time lost due to the unreliability of an agent, along with the cost of paying the agent.

Related Work

A vast majority of the literature in mechanism design focuses on maximizing either social utility (allocative efficiency) or the seller's revenue (optimal) (Mas-Colell et al. 1995) - minimizing cost in reverse auctions. Singer (Singer 2010) introduced the scenario where the social planner's valuation depends on the allocation. In such settings, the mechanism seeks to maximize the social planner's value within a budget constraint. Such *budget feasible mechanisms* are relevant when the social planner is not indifferent to the allocation among agents. Singer and Mittal (Singer and Mittal 2011) design budget feasible auctions for crowdsourcing marketplaces. In crowdsourcing settings, when the requester utility depends on the capability of the agent to whom the task is allocated, a reverse auction which minimizes cost is not suitable. Instead, an auction which accounts for the value that a requester places on allocating a task to an agent, while ensuring that the budget constraint is satisfied, is more suitable. The online version of their mechanism relies on dividing time into epochs, splitting up the available budget into chunks and applying the proportional share allocation mechanism in each epoch so that the budget constraint is satisfied.

Our work is motivated by a similar setting where the social planner's utility depends on the allocation as well as the cost. However, our work is different in that we consider the challenges due to unreliable agents who may not complete the task allocated to them within a specified time. These tasks must then be reallocated to other agents. The time lost due to such failed allocations can lead to a failure to meet the deadline and lower requester value. Thus, the presence of unreliable agents introduces non-trivial challenges to the mechanism design problem here.

Singla and Krause (Singla and Krause 2013) also consider the design of truthful budget feasible mechanisms in a crowdsourcing marketplace. They formulate this problem as a multi-armed bandit problem where the requester must decide the posted price (arm) to use when the cost of each agent is private information and agents arrive over time. Their approach relies on designing a posted price mechanism strategy which minimizes regret within a budget constraint. They too do not consider the effect of unreliable agents who may not complete the tasks allocated to them and this distinguishes our work from theirs.

Fardani, Hartmann, and Ipeirotis (Faradani, Hartmann, and Ipeirotis 2011) propose a pricing mechanism for completing tasks by a deadline on crowdsourcing platforms. Their approach relies on establishing a relationship between price and completion time based on empirical observations on a crowdsourcing platform.

Gerding et. al. (Gerding et al. 2010) consider the setting where a social planner must procure services for completing a task within a deadline. Agents have duration uncertainty, that is, agents vary in the time they take to complete the task and this is private information of the agents. Their approach relies on redundant allocation to increase the probability of meeting the deadline. They propose an execution contingent VCG mechanism which pays agents only after they complete the task. Unlike our work, they do not consider the budget constrained setting where agents arrive over time.

Parkes and Singh (Parkes and Singh 2003) consider the online mechanism design problem where a social planner must make *sequential* decisions as agents arrive over time. In addition to the agents' valuations of the item being private, the exact arrival and departure times are also private information of the agent. Hence, agents have *temporal* strategies, that is, strategic agents can lie not only about their valuation but also about their arrival / departure time. We address a different challenge of sequential decision making in online settings. In our setting, agents do not have temporal strategies but the presence of unreliable agents requires the mechanism to take into account the potential time lost due to an allocation when working towards a deadline.

Our Contributions

We believe our work is the first one to consider allocation and pricing issues involving unreliable, strategic agents in an online setting when operating with a deadline constraint. For this setting, we propose two mechanisms - DPM (Dynamic Price Mechanism) and ABM (Auction Based Mechanism) - and requesters can choose to deploy either of the two.

The key insight of our work is that in this setting agent patience levels can influence the requester utility. When agents arrive sequentially and are not willing to wait (impatient agents), the mechanism must make allocation and payment decisions sequentially. DPM is suitable for this setting. When agents are willing to wait (patient agents), the mechanism can delay allocation and payment decisions. ABM is suitable for this setting. ABM allows a trade off between maximizing the probability of on time completion and minimizing the cost incurred. Assuming a quasilinear utility function, ABM seeks to maximize requester utility. We prove that both DPM and ABM mechanisms are dominant strategy incentive compatible (DSIC) and are ex-post individually rational (IR) for workers who meet the deadlines.

Outline of the Paper

The rest of the paper is organized as follows. In the next section, we describe our model of a crowdsourcing marketplace which captures many realistic phenomena observed in web-based crowdsourcing marketplaces. We propose an extension of the current interaction model which can be implemented with minimal changes to current crowdsourcing platforms. In the section entitled *Mechanism Design*, we propose two mechanisms: DPM and ABM. We establish the desirable properties of these mechanisms. The section on *Simulation* presents pertinent experimental results. We conclude with a summary and directions for future work.

Crowdsourcing System Model

Our crowdsourcing system model consists of the agents, the requester, and the interaction between them.

Agent Model

An agent $i \in \{1, 2, ..., n\}$ incurs a cost of c_i in completing each task. At time t, if he is allocated k_i^t tasks and paid r_i^t, his value (v_i) and utility (u_i) are:

$$\begin{aligned} v_i(k_i^t, c_i) &= -c_i k_i^t \\ u_i(k_i^t, r_i^t, c_i) &= r_i^t - c_i k_i^t \end{aligned} \quad (1)$$

We assume that agents are utility maximizers and the cost of servicing a task (c_i) is private information of the agent. Being strategic, an agent may lie about his true cost of service and misreport it. Each agent is characterized by his reliability (b_i) which refers to the probability that the agent will complete the task assigned to him within a specified timeout. Agent reliability can be estimated as the ratio of *responses submitted timely* to *tasks accepted* by the agent. Since the platform can empirically estimate this parameter and make it available, we treat it as common knowledge. Each agent is further characterized by his capability which refers to the probability that agent i's response to a task is correct. This can be estimated as the ratio of *correct responses* to *responses submitted* by the agent. Most platforms allow the requester to set a qualification threshold so that only agents having capability greater than this threshold can attempt the task. The requester can thus control the quality of responses and we do not consider this aspect of the agent further in this work.

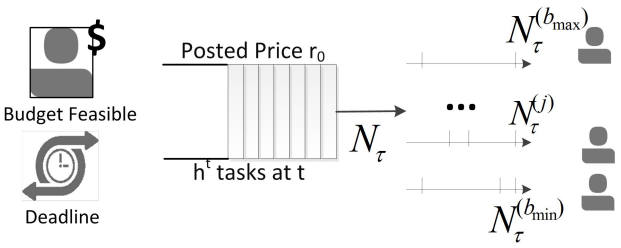

Figure 1: System Model

Requester Model

At $t = 0$, the requester ($i = 0$) has h^0 tasks that it aims to complete by a deadline T. With the above mentioned qualification process in place, the requester value depends on the number of tasks leftover at the deadline. Let h^T be the number of tasks leftover at the deadline T and let ϑ be the maximum possible requester valuation which is obtained if there are no leftover tasks at the deadline ($h^T = 0$). If C is the total cost incurred by the requester, her value and utility are given by:

$$\begin{aligned} v_0(h^T) &= \vartheta\delta(h^T) \\ u_0(h^T) &= v_0(h^T) - C \end{aligned} \tag{2}$$

where $\delta()$ is a delta function that takes the value 1 at 0 and 0 elsewhere. This captures the scenario where the requester has an *all or nothing* valuation for ontime completion. In general, any function which has a maximum at 0 and is non-increasing in h^T can be a valuation function. The requester's value and utility are revealed only at the deadline T. At any time $t < T$, the requester's value and utility are random variables (V_0, U_0) whose value can be estimated based on the number of pending tasks (h^t) and the probability of these tasks being completed in the remaining time ($T - t$). We use the notation $\mathbb{P}^t(V_0 = x)$ to refer to the probability that requester value at T would be x as estimated at time t. Similarly, we use the notation $\mathbb{E}^t(V_0)$ to refer to the expected requester value at T as estimated at time t.

Table 1: Notation Table

Symbol	*Term*
n_i	No. of tasks committed by agent i
t_i	Time committed for n_i tasks
b_i	Agent-i's ontime completion probability
c_i	Agent-i's cost per task
$\zeta_i = (n_i, t_i, b_i, c_i)$	Agent-i's contract
T	Deadline for all tasks to complete
r_0	Fixed posted task price
$\mu(r_0)$	Task assignment rate at posted price r_0
$N_{t_2-t_1}$	No. of tasks (contracts) assigned in $(t_1, t_2]$
$W_{t_2-t_1}$	No. of tasks completed in $(t_1, t_2]$
h^t	No. of tasks pending at time t
β^t	Bonus budget available at time t
ϱ_i^t	On-time bonus offered to agent i
(k_i^t, r_i^t)	Allocation and Payment rule at time t

Current Interaction Model

1. *Requester Posts Task Group*: At $t = 0$, the requester posts tasks on the platform at a posted price (r_0) and a posted timeout before which an agent who has accepted a task must submit it. Assume that the total number of tasks accepted by agents up to time t follows a Poisson process ($\{N(t)\}_{t\geq 0}$) with rate μ. The number of tasks accepted in the interval $(t_1, t_2]$ is given as:
$$N(t_2) - N(t_1) = N_{t_2-t_1} \sim \text{Poisson}(\mu(t_2 - t_1))$$
μ may be estimated by assuming a Gamma prior and updating the estimate based on empirical service times observed for an initial fixed period. We assume that the agent arrival process is homogeneous, so μ does not change with time. Previous work (Faradani, Hartmann, and Ipeirotis 2011) has shown that the posted price (r_0) is a dominant factor in determining the rate at which tasks get accepted by agents. Thus, the requester may leverage historical data for setting r_0 so that a task group would be completed by the deadline T in expectation, that is, $\mathbb{E}[h^T] = 0$ or equivalently $\mathbb{E}[N_{T-0}] = \mu(r_0)T = h^0$. However, there is a marked variation in completion times even for tasks that are priced the same ($Var(N_{T-0}) = \mu(r_0)T$) and thus, we propose an adaptive approach.
2. *Agent Accepts a Task*: When an agent accepts to work on a task, a predetermined timeout is started before which the agent must submit this task.
3. *Agent Submits*: Agent completes the task and submits her response to the task; alternatively, the timeout expires.
4. *Requester Pays / Reposts*: The requester verifies the agent response and if satisfied, pays the agent. If the timeout for the task has expired or if the requester is not satisfied by the agent's response, she may *reject* it (no payment is made to the agent) and repost the task on the platform.

Proposed Interaction Model

1. *Requester Posts Task Group*: Same as before.
2. Agent Proposes Contract: An agent i can propose to do several tasks at once (n_i) and can specify the timeout (t_i) by which he agrees to submit these n_i tasks. We treat this proposal as a contract for on time completion.
3. *Requester Accepts / Rejects Contract*: The requester decides whether or not to accept the contract.
4. *Agent Submits*: Same as before.
5. *Requester Pays / Reposts*: Same as before.

Mechanism Design

The contract proposed by an agent is modeled as a four-tuple $\zeta_i = (n_i, t_i, b_i, c_i)$. Contracts arrive over time and a mechanism must decide whether to accept the contract and the price. We assume that agent reliability (b_i) is either known or can be learned without any cost to the requester. All three mechanisms we discuss release the payments only after verifying that the agent has completed n_i tasks at t_i: so b_i, n_i, t_i are all known to the requester at the time the payment is released. However, a strategic agent may lie about his true cost of service, c_i, and misreport it as $\hat{c}_i$.

Fixed Price Mechanism

The current interaction model on crowdsourcing platform uses the Fixed Price (FP) mechanism with the allocation rule $k_i^t = 1$ and payment rule $r_i^t = r_0$. This mechanism allocates exactly one task to every qualified agent who is willing to work on the task at any time t and pays a fixed amount r_0 after the agent submits the response to the task. Since agents are unreliable, allocations may fail. If the agent population can be clustered into groups so that all agents belonging to cluster j have reliability $b^{(j)}$, then $N_\tau^{(j)}$ and $W_\tau^{(j)}$ represent the number of tasks allocated to and completed by agents of capability $b^{(j)}$ in time τ. The task service Poisson process can be considered as a merge of multiple Poisson processes and we have $N_\tau^{(j)} \sim \text{Poisson}(\mu^{(j)}\tau)$ and $W_\tau^{(j)} \sim \text{Poisson}(b^{(j)}\mu^{(j)}\tau)$. (Figure 1). Thus, at time t, the expected requester value can be calculated as:

$$W_\tau \sim \text{Poisson}(\sum_{j=b_{min}}^{b_{max}} b^{(j)}\mu^{(j)}\tau) \quad (3)$$

$$\mathbb{E}^t[V_0] = \vartheta\mathbb{P}^t(h^T = 0) = \vartheta\mathbb{P}(W_{T-t} \geq h^t) \quad (4)$$

Dynamic Price Mechanism

The dynamic price mechanism (DPM) seeks to operationalize our proposed interaction model where a requester may allocate more than one task to an agent. Here $N_\tau^{(j)} \sim \text{Poisson}(\mu^{(j)}\tau)$ is the number of contracts which arrive in time τ and since each contract can have more than one tasks allocated, we have $\mathbb{P}(W_\tau^{(j)} = m^{(j)}) =$

$$\sum_{s=1}^{m^{(j)}} \mathbb{P}(N_\tau^{(j)} = s) \sum_{x=m^{(j)}}^{h^t} \mathbb{P}(\sum_{i=1}^{s} n_i = x) p(m^{(j)}; x, b^{(j)})$$

$$\text{with} \quad p(m^{(j)}; x, b^{(j)}) = \binom{x}{m^{(j)}} (b^{(j)})^{m^{(j)}} (1 - b^{(j)})^{x - m^{(j)}}$$

The aggregate service rate for tasks is thus:

$$\mathbb{P}(W_\tau = m) = \mathbb{P}(\sum_{j=b_{min}}^{b_{max}} W_\tau^{(j)} = m) \quad (5)$$

The requester's expected value at time t if she accepts the contract ζ_i depends on the contract's outcome. With probability b_i, the agent completes the contract and we have $h^t - n_i$ tasks remaining to be serviced within the remaining time $T - t$. With probability $1 - b_i$, the agent reneges on the contract. The requester still has $h^t - n_i$ tasks remaining to be serviced within the remaining time $T - t$ but an additional n_i tasks to be serviced within $T - t_i$ when the requester realizes that agent has reneged on his contract and re-posts these tasks as an independent set of task. (Figure 2).

$$\begin{aligned}
\mathbb{E}^t[V_0|\zeta_i] &= \vartheta(\mathbb{P}^t(h^T = 0|\zeta_i)) \quad (6)\\
&= \vartheta(b_i\mathbb{P}(W_{T-t} \geq (h^t - n_i))\\
&+ (1 - b_i)\mathbb{P}(W_{T-t} \geq (h^t - n_i))\mathbb{P}(W_{T-t_i} \geq n_i))\\
&= \vartheta(b_i w + (1 - b_i)wz)\\
\omega(\zeta_i, t) &= \mathbb{E}^t[V_0|\zeta_i] - \mathbb{E}^t[V_0]\\
&= \vartheta(b_i w + (1 - b_i)wz - y) \quad (7)
\end{aligned}$$

where

$w \triangleq \mathbb{P}(W_{T-t} \geq (h^t - n_i)); y \triangleq \mathbb{P}(W_{T-t} \geq h^t); z \triangleq \mathbb{P}(W_{T-t_i} \geq n_i)$

We define the value of a contract $\omega(\zeta_i, t)$ as the increment in the expected value of the requester due to accepting a contract vis-a-vis relying on the default FP mechanism. This has a worst case interpretation: it is the contract value if no other contracts ($n_i > 1$) arrive in the future. For this approach, we use W_τ from (3) to evaluate w, y, z. An alternate approach is to define the value of a contract as the increment in the expected value of the requester due to accepting a contract vis-a-vis not accepting it and relying on the DPM mechanism. This has an expectation interpretation: it is the expected contract value if contracts arrive in the future at the same rate they have been arriving in the past. For this approach, we use W_τ from (5) to evaluate w, y, z.Having evaluated the the contract, we specify DPM as:

$$\begin{aligned}
k_i^t &= \begin{cases} n_i & \text{if } \omega(\zeta_i, t) > 0 \text{ and } r_i^t > n_i\hat{c}_i \\ 0 & \text{otherwise} \end{cases}\\
r_i^t &= \begin{cases} n_i r_0 + \varrho_i^t & \text{if } k_i^t > 0 \\ 0 & \text{otherwise} \end{cases} \quad (8)
\end{aligned}$$

DPM's allocation rule accepts all bids on a first-come first-serve basis as long as the contract offered by the agent increases the expected value of the requester ($\omega(\zeta_i, t) > 0$) and the bid is affordable ($n_i\hat{c}_i < r_i^t$). Thus, DPM is an online *myopic optimizer*. If the number of pending tasks is less than the contract offer ($h^t < n_i$), DPM allocates all pending tasks to the agent if the contract is accepted.

DPM's payment rule has a fixed price component (r_0) and an ontime completion bonus component (ϱ_i^t). The fixed price has been pre-decided by the requester at the time of posting the tasks. To incentivize agents to undertake larger contracts, DPM sets aside a bonus budget at initialization ($\beta^0 > 0$). It uses this budget to offer an additional bonus amount to agents who complete larger contracts. When offered a contract $(n_i, t_i, b_i, \hat{c}_i)$ at time t, it first evaluates the contract value as explained above. It then calculates a bonus payment (ϱ_i^t) that it is willing to offer to an agent's contract based on the value of the contract and the bonus budget available (β^t).

$$\varrho_i^t = \frac{\omega(\zeta_i, t)}{\vartheta}\beta^t$$

Once a contract is allocated the available budget for future contracts is reduced. Let a_i be the time at which ζ_i is offered and accepted. Thus,the leftover bonus budget at $t > a_i$ is $\beta^t = \beta^{a_i} - \varrho_i^{a_i}$. Only this leftover budget is used for calculating future contract bonuses. Even though the payment rule calculates the payment at t it defers the disbursement of payment till the agent has in fact submitted all n_i tasks. Thus, the payment is contingent on contract completion.

Theorem 1. *DPM is dominant strategy incentive compatible (DSIC), budget feasible, ex-post individually rational (EPIR) for agents who honor the contract, and for all* $t \in [0, T], \mathbb{E}_{DPM}^t[V_0] \geq \mathbb{E}_{FP}^t[V_0]$.

Proof. DSIC: The payment rule in DPM is independent of the agent's bid: r_i^t does not depend on $\hat{c}_i$. Also, the allocation

rule of DPM is monotone in the cost: a reduction in $\hat{c}_i$ can only increase k_i^t. These two conditions are sufficient for a mechanism to be DSIC (Mas-Colell et al. 1995).

EPIR: An agent will voluntarily participate and bid for a contract if his utility due to participation is greater than non participation, that is, $u_i(k_i^t(c_i, c_{-i}), r_i^t(c_i, c_{-i}), c_i) = (r_i^t - n_i c_i) \geq 0$. This follows easily from the DPM's allocation rule which allocates the contract iff $(r_i^t > n_i c_i)$ and when we note that $r_i^t \geq 0$ since $r_0 > 0$ and $\beta^0 > 0$. Thus, DPM is EPIR as long as the agent honors the contract. If the agent does not complete the tasks assigned to him, it is possible he may obtain a negative utility.

Budget Feasible: Let m be the total number of contracts accepted under DPM and let a_i be the arrival time of contract ζ_i. For budget feasibility, total payment under DPM $(\sum_{i=1}^{m} r_i^{a_i})$ must be less than the total budget $(\beta^0 + r_0 h^0)$. This constraint can be simplified as follows:

$$\begin{aligned} \beta^0 + r_0 h^0 &\geq \sum_{i=1}^{m} r_i^{a_i} = \sum_{i=1}^{m} r_0 n_i + \varrho_i^{a_i} \\ \Rightarrow \beta^0 &\geq \sum_{i=1}^{m} \varrho_i^{a_i} \quad (\because \sum_{i=1}^{m} n_i \leq h^0) \\ \Rightarrow \beta^0 &\geq \sum_{i=1}^{m} \beta^{a_i} \quad (\because \omega(\zeta_i, t) \leq \vartheta) \end{aligned}$$

$\sum_{i=1}^{m} n_i \leq h^0$ due to DPM's conservative approach regarding tasks allocated in a contract. DPM does not not re-allocate the tasks committed in a contract till the contract has been completed or it has expired even though the allocated tasks continue to show up as pending tasks (h^t is updated only at the time of successful contract completion). $\omega(\zeta_i, t) \leq \vartheta$ follows from equation (7) and the last inequality is true due to DPM's conservative approach regarding budget allocated in a contract: DPM updates the budget available for bonus after every contract allocation and does not re-allocate the budget that has been committed for a contract till the contract has been completed or it has expired.

More Efficient than FP: This result is an artifact of the way DPM treats tasks allocated in a contract: the requester does not change h^t till the contract has been completed but also does not re-allocate these tasks that she has committed in a contract. Formally, the expected requester value under DPM is a monotone submodular function since we ensure that the value of each contract is positive before accepting it $\mathbb{E}_{FP}^t[V_0] = \mathbb{E}^t[V_0|\{\phi\}] \leq \mathbb{E}^t[V_0|\{\zeta_1\}] \leq \mathbb{E}^t[V_0|\{\zeta_1, \zeta_2, \ldots, \zeta_m\}] = \mathbb{E}_{DPM}^t[V_0]$. The simulation section demonstrates this result more intuitively. □

Auction Based Mechanism

DPM increases the requester value while ensuring the budget constraint. However, it does not optimize the amount the requester has to pay: all available budget is used to incentivize on-time completion. ABM seeks to address this. In ABM, the available time T is divided into epochs of length l each indexed with τ. With a slight abuse of notation we also refer to the the time of the epoch ends as τ. Let $I(\tau)$

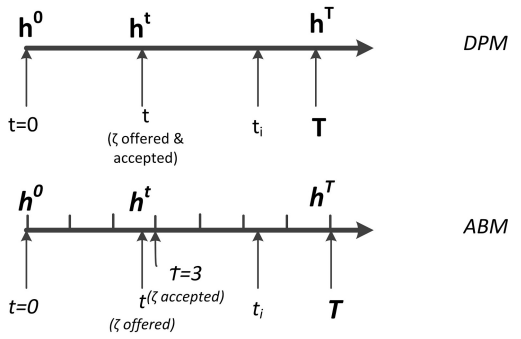

Figure 2: (i) DPM accepts or rejects contract when they are offered based on the number of pending tasks (h^t), deadline (T) and proposed contract time (t_i) (ii) ABM works in epochs and delays allocation and payment decisions

represent the set of positive valued contracts $(\omega(\zeta_i, \tau) > 0)$ available for consideration at the end of interval τ. Then, we specify ABM as follows:

$$\begin{aligned} j &= \arg\max_{i \in I(\tau)} c_i \\ I_{-j}(\tau) &= I(\tau) \setminus j \\ k_i^\tau &= \begin{cases} n_i & \text{if } i \in I_{-j}(\tau) \\ 0 & \text{otherwise} \end{cases} \\ r_i^\tau &= \begin{cases} \min(n_i c_j, \varrho_i^\tau + n_i r_0) & \text{if } k_i^\tau > 0 \\ 0 & \text{otherwise} \end{cases} \end{aligned} \quad (9)$$

If $|I(\tau)| = 1$, ABM's allocation rule operates as DPM i.e. accepts the contract and pays the maximum price as determined by the available budget. If $|I(\tau)| \geq 2$, ABM accepts all contracts in $|I(\tau)|$ except for the highest bid (highest per-task price). If $h^t < n_i$, ABM allocates all pending tasks to the agent if the contract is accepted. ABM's pricing rule uses the rejected bid to compute the payment for the accepted bids. All accepted contracts $(I_{-j}(\tau))$ are paid this same amount. ABM is thus a multi-unit auction with a budget constrained reserve price. Unlike the traditional multi-unit setting is that the number of contracts to be allocated is not pre-decided but determined based on the number of arrivals in in time epoch τ.

Theorem 2. *ABM is DSIC, budget feasible, ex-post IR for agents who honor the contract but not ex-post IR for agents who do not honor the contract.*

Proof. DSIC: Since agents are assumed not to have temporal strategies, we treat each time epoch as an independent auction. m^{th} price auctions in multi-unit settings are known to be DSIC (Mas-Colell et al. 1995).

EPIR: We have to show that $u_i(k_i^t(c_i, c_{-i}), r_i^t(c_i, c_{-i}), c_i) = (r_i^\tau - n_i c_i) \geq 0$. ABM's allocation rule ensures that for each accepted contract $(r_i^\tau > n_i c_i)$. Thus ABM is EPIR as long as the agent honors the contract.

Budget Feasible: ABM is budget feasible because DPM is Budget Feasible and ABM never pays more than DPM for any contract. □

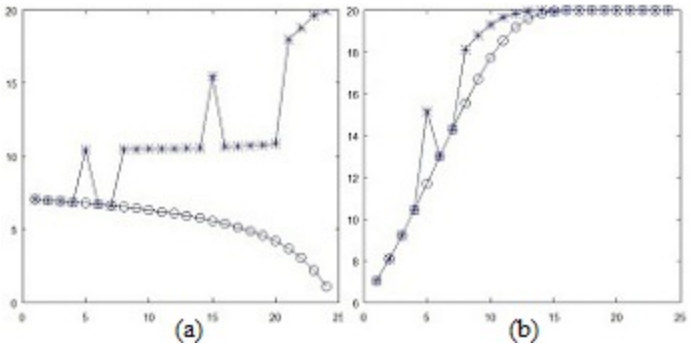

Figure 3: Expected Value vs. Time when (a) fewer than expected agents arrive and (b) higher than expected agents arrive

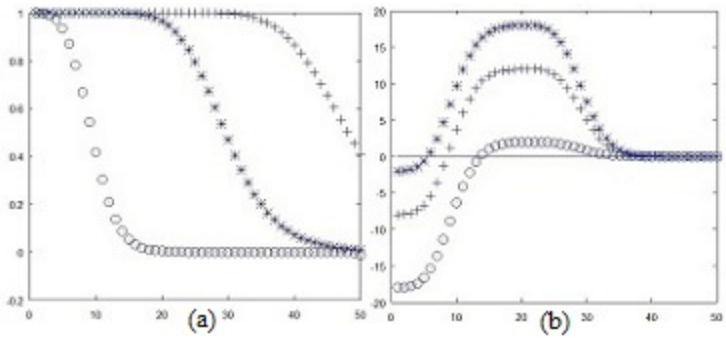

Figure 4: (a) Min. Agent Reliability for contract to be accepted vs. number of pending tasks at different points in time and (b) Contract Value vs. number of pending tasks for different agent capabilities

Comparison with DPM: It is clear that the expected requester value under ABM cannot be more than under DPM since the requester rejects a positive valued contract. However, the expected requester utility under ABM can be more than under DPM due to the lower costs incurred by the requester. The expected requester utility under ABM is:

$$\mathbb{E}^{\tau}_{ABM}[U_0] = \mathbb{E}^{\tau}[V_0|I_{-j}(\tau)] - \sum_{i=1}^{|I(\tau)|-1} c_j n_i \qquad (10)$$

Let $\mathbb{E}^{\tau}[V_0|\{\zeta_1,\ldots,\zeta_{|I(\tau)|}\}]$ be the expected valuation at time τ given that the mechanism accepted each contract at its *time of arrival*. Let $\mathbb{E}^{\tau}[V_0|I(\tau)]$ be the expected valuation at time τ given that the mechanism accepted all the contracts arrived *at time* τ. If l, the length of a time epoch, is small enough, then $\mathbb{E}^{\tau}[V_0|\{\zeta_1,\ldots,\zeta_{|I(\tau)|}\}] = \mathbb{E}^{\tau}[V_0|I(\tau)]$ (See Appendix). Since the expected requester value is a submodular function, $\mathbb{E}^{\tau}[V_0|I(\tau)] \leq \mathbb{E}^{\tau}[V_0|I_{-j}(\tau)] + \omega(\zeta_j,\tau)$ and we can bound the expected requester utility under DPM:

$$\mathbb{E}^{\tau}_{DPM}[U_0] \leq \mathbb{E}^{\tau}[V_0|I_{-j}(\tau)] + \omega(\zeta_j,\tau) - \sum_{i=1}^{|I(\tau)|} (\varrho_i^{a_i} + r_0 n_i) \qquad (11)$$

ABM outperforms DPM if $\mathbb{E}^{\tau}_{DPM}[U_0] < \mathbb{E}^{\tau}_{ABM}[U_0]$. Using equations (10) and (11), this condition reduces to:

$$\begin{aligned}
\omega(\zeta_j,\tau) &- \sum_{i=1}^{|I(\tau)|} (\varrho_i^{a_i} + r_0 n_i) < (- \sum_{i=1}^{|I(\tau)|-1} c_j n_i) \\
\omega(\zeta_j,\tau) &< \sum_{i=1}^{|I(\tau)|-1} (\varrho_i^{a_i} + r_0 n_i - c_j n_i) + \varrho_j^{a_j} + n_j r_0 \\
&< (r_0 - c_j) \sum_{i=1}^{|I(\tau)|-1} n_i + \sum_{i=1}^{|I(\tau)|-1} \varrho_i^{a_i} \\
&\leq (r_0 - c_j) \sum_{i=1}^{|I(\tau)|-1} n_i + \sum_{i=1}^{|I(\tau)|-1} \varrho_i^{\tau}
\end{aligned}$$

The last inequality holds because $(\tau \geq a_i) \Rightarrow (\omega(\zeta_i,\tau) \leq \omega(\zeta_i,a_i))$, that is, the value of a contract can only reduce by delaying allocation and if no allocations are made during this delay. Also $\beta^{\tau} = \beta^{a_i}$ since no allocations are made during $(\tau-1,\tau]$. These two conditions ensure $\varrho_i^{\tau} \leq \varrho_i^{a_i}$.

Thus, ABM outperforms DPM when the loss in expected value due to rejection of the highest bid contract (ζ_j) is less than the reduction in cost achieved due to rejection of the highest bid contract. A natural generalization of this mechanism is to drop the q highest bids. This reduces the consideration set by $|I(\tau)|-q$ but also potentially reduces the price to be paid to the allocated contracts. If $q = 0$, ABM reduces to DPM. As q increases, we sacrifice expected requester value for a potential reduction in payment. Another approach is to select q so as to maximize the requester expected utility. Unfortunately, unlike the expected requester value, the expected requester utility is not a submodular function. Hence, this approach requires the evaluation of the expected utility for all possible subsets ($2^{|I(\tau)|}$) of contracts in the consideration set. The choice of q also depends on the epoch length. $l = 0 \Rightarrow q = 0$ and this is suitable when agents are impatient. As $l \to T$, the maximum value that q can take increases. Realistically, l must be chosen taking into account the willingness of agents to wait after bidding.

Simulations

We use simulations to demonstrate the effectiveness of our proposed mechanisms with varying arrival rate and agent reliability. We compare DPM with FP which is currently the defacto mechanism on crowdsourcing platforms today. We have already derived the conditions under which ABM outperforms DPM. Real world deployments must also take into account how patient the agents are when choosing to deploy DPM or ABM. We consider the following setup. At $t = 0$, a requester posts $h^0 = 50$ tasks on the platform with a deadline of $T = 25$ hours. On-time completion of these tasks generates a requester value of $\vartheta = 20$. The posted price attracts qualified agents to service the task at $\mu = 2$ per hour (Poisson process) so that tasks will complete in expectation ($\mu T = 50 = h^0$). These values are representative of real world scenarios where more than 60% of task groups complete within 24 hrs (Faradani, Hartmann, and Ipeirotis 2011).

Figure 3(a) compares the requester's expected value under FP ('o') with the requester's expected value under DPM ('*') when 2 agents arrive per hour. At $t = 5$ an agent with reliability $b_i = 0.9$ offers to complete $n_i = 3$ tasks by $t_i = 8$. The requester's expected value at $t = 5$ hence shows an instantaneous spike. However, this gain in expected value is not taken into account till $t = 8$ when the agent actually submits the completed tasks. Thus, at $t = 8$ when this contract is fulfilled by the agent, the expected value is incremented and stays at this higher level. Subsequently a second contract arrives at $t = 15$ where an agent with reliability $b_j = 0.9$ offers to complete $n_j = 3$ tasks by $t_j = 21$. We see an instantaneous spike at the time the contract is offered with a permanent increase when the contract is completed.

Figure 3(b) compares requesters expected value under FP ('o') with the requesters expected value under DPM ('*') when 3 agents arrive per hour. The same two contracts are offered but the second contract is offered too late to offer a substantial gain in expected requester value.

The key criteria for accepting a contract ($\omega(\zeta_i, t) > 0$) translates to a bound on the agent reliability ($b_i > \frac{\frac{y}{w} - z}{1 - z}$). Figure 4(a) shows the minimum reliability that an agent must have for a contract to be accepted as a function of the pending tasks (h^t). We have used $n_i = 10$ and $t_i = 24$ for this simulation at $t = 1$ ('+'), $t = 10$ ('*') and $t = 20$ ('o'). At any time t, as the number of pending tasks increases, the minimum reliability threshold that the requester is willing to accept is lowered.

Figure 4(b) shows how the value of a contract of size $n_i = 20$ and $t_i = 24$ offered at $t = 20$ changes with number of pending tasks for various values of agent reliability $b_i = 0.1$ ('o'), $b_i = 0.6$ ('+') and $b_i = 0.9$ ('*'). At any time t, as the number of pending tasks increases, the contract value initially increases (due to the increasing risk to the requester of missing the deadline) and later decreases as the number of pending tasks increases due to the strict valuation function of the requester wherein even a single leftover task drives the requester value to 0.

Future work

We have assumed that agents do not strategically delay their arrival time. Relaxing this assumption is a key direction for future work. One approach is to assume that agents cost and arrival time are independent of each other. Even with this assumption and assuming the complete information setting, designing (Nash) Incentive Compatible mechanisms is non-trivial since the value that a requester places on an agent's contract depends in a non-trivial way on the number of left-over tasks, time to deadline and agent reliability. A second key direction of future work is to consider scenarios with unknown agent capabilities and reliabilities. These may need to be learned by the requester at a cost and leads to the problem of learning within a budget and time constraint.

References

Faradani, S.; Hartmann, B.; and Ipeirotis, P. G. 2011. What's the right price? pricing tasks for finishing on time. In *Human Computation*.

Gerding, E.; Stein, S.; Larson, K.; Rogers, A.; and Jennings, N. R. 2010. Scalable mechanism design for the procurement of services with uncertain durations. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1-Volume 1*, 649–656.

Mas-Colell, A.; Whinston, M. D.; Green, J. R.; et al. 1995. *Microeconomic theory*, volume 1. Oxford university press New York.

Parkes, D. C., and Singh, S. P. 2003. An mdp-based approach to online mechanism design. In *Advances in neural information processing systems*, None.

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

Singer, Y. 2010. Budget feasible mechanisms. In *Foundations of Computer Science (FOCS), 2010 51st Annual IEEE Symposium on*, 765–774. IEEE.

Singla, A., and Krause, A. 2013. Truthful incentives in crowdsourcing tasks using regret minimization mechanisms. In *Proceedings of the 22nd international conference on World Wide Web*.

Appendix

Theorem 3. $\mathbb{E}^\tau[V_0|\{\zeta_1, \ldots, \zeta_{|I(\tau)|}] = \mathbb{E}^\tau[V_0|I(\tau)]$

Proof. Suppose agents $j_1, \ldots, j_{|I(\tau)|}$ have arrived at times $a_1, \ldots, a_{|I(\tau)|} \in (\tau - 1, \tau]$. Then the expected valuation of DPM at time τ given it has accepted contracts $\zeta_1, \ldots, \zeta_{|I(\tau)|}$ is

$$\begin{aligned}
&\mathbb{E}^\tau[V_0|\{\zeta_1, \ldots, \zeta_{|I(\tau)|}\}] = \\
&\sum_{k=0}^{|I(\tau)|} \sum_{j_1,\ldots,j_k} \prod_{l=1}^{k} b_{j_l} \prod_{l \notin I(\tau)\setminus\{j_1,\ldots,j_k\}} (1 - b_l) \\
\times\ & \vartheta\mathbb{P}(h^T = 0|\{j_l \text{ finishes by } t_{j_l}, l = 1, \ldots, k\}) \\
=\ & \mathbb{E}^\tau[V_0|I(\tau)]
\end{aligned}$$

Here $\mathbb{E}^\tau[V_0|I(\tau)]$ is the expected valuation of DPM given all the accepted contracts arrived at time τ. The last equality holds for two reasons

- If agent i arrives at time τ instead of a_i where $a_i > \tau - l$, the agent will finish the task by time t_i with same probability b_i. This is true under the assumption that agents are patient for a duration of length at least l.
- $\mathbb{P}(h^T = 0|\{j_l \text{ finishes by } t_{j_l}, l = 1, \ldots, k\})$ is same in both the cases, since this probability is computed with respect to the arrival of new agents in the interval $(\tau, T]$.

□

Contract Variants: We have modeled the contract as a four-tuple $\zeta_i = (n_i, t_i, b_i, \hat{c}_i)$. Alternative contract specifications where the proposed mechanisms can also be applied are:

- $\zeta_i = (n_i, b_i, \hat{c}_i)$ and requester picks the t_i which optimizes her value ($\frac{d\omega}{dt_i} = 0$) and pays a price $r_i \geq n_i \hat{c}_i$.
- $\zeta_i = (t_i, b_i, \hat{c}_i)$ and requester picks the n_i which optimizes her value ($\frac{d\omega}{dn_i} = 0$.) and pays a price $r_i \geq n_i \hat{c}_i$.