Online Assignment with Heterogeneous Tasks and Deadlines

Sepehr Assadi, Justin Hsu, Shahin Jabbari
Department of Computer and Information Science, University of Pennsylvania
{sassadi, justhsu, jabbari}@cis.upenn.edu

February 22, 2015

Abstract

We investigate the problem of *tasks assignment with deadlines* in crowdsourcing markets from the point of view of the requester. The requester has a collection of tasks, each with some deadline time, and workers arrive online one at a time. Each worker declares a price at which she is willing to do each task. The requester has to decide on the fly which task (if any) to assign to the worker; workers can only be assigned to tasks with deadlines after their arrival time. The goal is to maximize the number of tasks completed before their deadlines under a given budget.

We provide upper and lower bounds on the competitive ratio of any online algorithm for this problem assuming a worst-case sequence of bids, as long as the workers are *small*—their bids are small compared to the total budget. We also show algorithms achieving improved competitive ratio when the group of workers is fixed, with workers arriving in a random order.

1 Introduction

Suppose you are managing a ride-sharing platform, and as customers arrive online, you must decide how to dispatch drivers. In principle any driver can be sent to any customer, but there are some hard constraints in our setting. First, the customers each come with a *deadline*: they need to be served by some time. Second, the drivers are not always free to drive customers; say, they arrive at different times, and can only serve customers after they become available. Finally, drivers request payment, and the ride-sharing platform has a total budget B that it cannot exceed. What is the best strategy for arranging drivers and customers?

Looking more generally, we can also think of this problem in the context of crowdsourcing. Crowdsourcing markets are online labor markets where requesters can outsource large amount of simple *Human Intelligence Tasks* (HITs) to a set of workers usually in exchange of monetary payment ($e.g.$, *Amazon Mechanical Turk*[1] or *Crowdflower*[2]). Again, tasks may have deadlines: a task requester may need an answer by a certain time. If requesters have a fixed budget, how should they decide which workers to hire in an online fashion?

Singer and Mittal [2011] were the first to study this problem, when there is just a single deadline. In their setting, each worker has a single value for all the tasks; all the tasks are treated *homogeneously*. We generalize their setting in two significant ways: First, we allow the tasks to have different deadlines. Second, we work in a setting with *heterogeneous tasks*: workers can have different preferences over tasks. This is a realistic feature of many crowdsourcing markets: A requester might have different tasks that require different skill sets, appealing to different groups of workers; in the context or ride-sharing, taxi drivers may require different amounts of payment depending on how far they are from each customer.

It is not clear whether allowing heterogeneous tasks rather than homogeneous tasks changes our problem significantly. As we will show, however, this heterogeneity significantly complicates the task assignment problem. Algorithms for the heterogeneous case may look nothing like their counterparts in the homogeneous case.

[1] https://www.mturk.com.
[2] https://www.crowdflower.com.

We first warm up by providing two algorithms for the offline problem: first, the optimal algorithm via min-cost flows, and second, a fixed-threshold algorithm inspired by the algorithm of Singer and Mittal [2011]. Then, we move to the online setting, where we draw on techniques from the online knapsack literature Zhou et al. [2008].

As is typical for online algorithms, we measure the performance of our algorithm via the *competitive ratio*, i.e., the ratio of the number of tasks assigned by online algorithm to the number of tasks assigned by the best offline algorithm on the same problem. When the bids are bounded in $[1, R]$, the bidders are small compared to the budget, and the order is worst-case, we give an algorithm that achieves an $O(R^{\epsilon} \ln(R))$ competitive ratio when $R \leq \epsilon B$. The central idea is to use a moving threshold, and take all workers with bid below the threshold. As the budget is depleted, the threshold steadily decreases. Our algorithm is inspired by Zhou et al. [2008], who give algorithms for the online knapsack problem. We also show a lower bound: for any (possibly randomized) online algorithm, there exists an input that forces competitive ratio of at least $\Omega(\ln(R))$. Our worst-case instance actually has just a single deadline with homogeneous tasks.

Then, we consider the *random permutation* setting where the bids are adversarial, but where the order of bidders is chosen uniformly at random. We restrict ourselves to the single deadline case like Singer and Mittal [2011], but with heterogeneous tasks. Then, we measure the performance of an online algorithm by comparing the number of tasks assigned *averaged* over all permutations to the number of tasks assigned by the offline optimal (which is independent of the permutation, in the single deadline case).

2 Framework

Let us begin by modeling the setting. We will use j to index tasks and i to index workers throughout. We model the problem from the perspective of a requester who has a collection of m tasks, where each task has a deadline time $t_j \in [0, T]$.

Each worker i has an arrival time $s_i \in [0, T]$, and a numeric bid b_{ij} for solving task j. Workers arrive *online*: the sequence of the workers and their bids are initially unknown to the requester but once a worker arrives, her bids for all the tasks are revealed to the requester. We assume that each worker can be assigned to at most one task; we can relax this assumption by having multiple copies of each worker. We denote the total number of workers by n. As customary in the crowdsourcing setting we assume workers are abundant so we assume n is large (see Sections 3 and 4 for more details).

We model the sequence of workers (their bids and arrival times) in two different ways. First, we consider the case that we have no assumptions on the sequence of workers. This worst case scenario has been referred to as an *adversarial setting* in the literature. Second, we consider the *random permutation model* in which we still make no assumptions on the sequence of the workers but we assume the sequence is randomly permuted before being presented to the requester. We define these two settings in more detail in Sections 3 and 4.

The requester has a budget of B and has to decide on the fly which task to assign to the worker who arrives; of course, a worker can only be assigned to a task with deadline after her arrival time. If task j is assigned to worker i, then the requester pays worker i the price b_{ij}. The goal of the requester is to maximize the number of tasks he assigns to the workers while spending at most a budget of B and satisfying all the deadline constraints.

We compare the performance of any online algorithm to the performance of the best *offline* algorithm OPT, which can see all the bids when allocating tasks to workers. As is standard, we measure performance via the *competitive ratio*: the ratio of the number of tasks assigned in the offline optimal solution to the number of tasks assigned by the online algorithm (so competitive ratio is at least 1, and smaller competitive ratio is more desirable).

3 Adversarial Setting

When comparing an online algorithm with an offline algorithm, we first need to precisely specify the inputs on which we make the comparison. Let's first consider the worst case for the requester: *adversarial inputs*. In this setting, there is a fixed input and order, and we compare an online algorithm in this single input to an offline algorithm on the same input via the competitive ratio. We are interested in bounding this ratio in the worst case, i.e., the max over all inputs.

Of course, if we really do not make any assumption on the input, we cannot hope to compete with an offline algorithm—the competitive ratio may be arbitrarily high [see Zhou et al., 2008, Section 1.2].

So, we restrict the adversary's power by making one main assumption on the relationship between worker bids and the budget: the ratio of the largest bid to the smallest bid should be small compared to the budget. More precisely, we can scale worker bids so that $b_{ij} \in [1, R]$, and we write $R = \epsilon B$. We will frequently consider ϵ to be small—this is appropriate for the crowdsourcing problems we have in mind, where (i) the scale of the budget is much larger than the scale of payments to bidders and (ii) the tasks are not extremely difficult, so no worker charges an exorbitantly high price. We refer to this assumption as *large market* assumption because when the bids are small compared to the budget, the requester can hire a large number of workers before exhausting his budget. This is also in line with the assumption we made in Section 2 for n to be large.

Before diving into the technical details, let's consider which range of parameters and competitive ratios is interesting. Since the bids are restricted in $[1, R]$, achieving a competitive ratio of R is trivial. So, we will be mainly interested in the following question: "Can we design an algorithm that has a competitive ratio much smaller than R for any sequence of workers?"

The rest of this section is organized as follows. We first investigate the offline problem, describing how the problem can be solved optimally. Then we propose a simpler algorithm for the offline setting; the performance is a constant factor off from optimal, but the simpler algorithm will be useful in Section 4, when we will work in the *random permutation* setting. We then move to the online setting, giving an algorithm with competitive ratio of $O(R^{\epsilon} \log(R))$ for any sequence of workers. Finally, we give a lower bound showing that any algorithm has competitive ratio of at least $\Omega(\log(R))$ in the worst case.

3.1 The Offline Problem

Let us start by investigating offline problem. Since the sequence of workers and their bids is known up front, the problem can be converted to an instance of the *min-cost flow* problem. In this problem, given a capacitated graph with costs on the edges and a demand value of flow, the goal is to route this amount of flow from a specific source to a specific target while minimizing the total flow cost over the edges, where the flow cost over an edge is equal to the amount of the flow passing the edge multiply by the cost of the edge.

In our problem, we want to maximize the number of assigned tasks given a fixed budget. So, we construct the following bipartite graph with workers on one side, and tasks on the other. First, draw an edge from a worker to a task if the worker arrives before the deadline of the task and let the edge cost denote the bid of the worker for the task. Then, attach a source node pointing to all the workers and connect all the tasks to an added target node. All edges have capacity 1.

In terms of the min-cost flow construction, we want to maximize the amount of flow routed from the source to the target and we want the cost of this flow to be at most our budget, B.[3] However, the amount of flow corresponds to the number of assigned workers, which we do not know up front; indeed, we are trying to maximize this number. Nevertheless, we can search over all such possible amounts, because the search space is bounded by the number of tasks and workers and we only need to consider integral flows. The offline optimal solution is, then, the largest amount of integral flow which has a min-cost of at most B.

While this approach achieves the optimal solution, we will see in Section 4 that using a *fixed threshold* to decide which workers to hire can be very useful. Hence, we provide a simpler algorithm with this feature; while it does not output the optimal solution, its solution is within a factor of 4.

The algorithm has two components. In the first component, given a threshold value p, the algorithm goes over workers one by one and assigns a task to an unassigned worker if the bid of the worker for that task is not bigger than p. In case there are more than one unassigned task that the worker bids p or less for, the algorithm break ties by assigning the task with the earliest deadline (smaller s_j) to the worker. This algorithm which we refer to as *Fixed Threshold Policy* (FTP) is described in Algorithm 1.

In the second component, the algorithm searches over all possible values of p to find the proper threshold. Although the search space is continuous, it is sufficient for the algorithm to restrict its search to the bids of the workers. The

[3] Since the amount of flow routed from the source to the target and the capacity of the edges are integral, the min-cost flow problem has an integral solution.

Algorithm 1 FTP

Input: Threshold price p, set of tasks J with deadlines $t_j \in [0, T]$, budget B.
Online input: Worker bids $b_{ij} \in [1, R]$, arrival time $s_i \in [0, T]$.
  While $B > 0$, on input i:
    Let $C_i := \{j \in J \mid b_{ij} \leq \min(p, B)$ and $s_i \leq t_j\}$.
    If $C_i \neq \emptyset$:
      Output: $a(i) \in \arg\min_{j \in C_i} t_j$.
      Let $B := B - b_{i,a(i)}$.
      Let $J := J \setminus \{a(i)\}$.
    else:
      Output: $a(i) := \perp$.

algorithm, which we refer to as *Offline Approximation Algorithm* (OA), is given in Algorithm 2. We show that the OA assigns a tasks to at least a quarter of the workers assigned by the offline optimal solution.

Algorithm 2 OA

Input: Worker bids $b_{ij} \in [1, R]$, set of tasks J with deadlines $t_j \in [0, T]$, budget B.
  Let $Q := 0$.
  Foreach b_{ij}:
    Let $q := \mathrm{FTP}(b_{ij}, J, B)$.
    If $q > Q$ **then**: Update $Q := q$.
Output Q (number of assignments) and $p^* := B/Q$ (the threshold price).

Theorem 1. *Let* $\mathrm{ALG}_{\mathrm{OA}}(\sigma)$ *and* $\mathrm{OPT}(\sigma)$ *denote the number of tasks assigned by the* OA *and the offline optimal algorithm for a sequence of workers* σ*, respectively. Then for any sequence of workers* σ *that satisfies the large market assumption,* $\mathrm{OPT}(\sigma) \leq 4 \cdot \mathrm{ALG}_{\mathrm{OA}}(\sigma)$.

To prove this, we use the following useful lemma.

Lemma 1. *Let* $a_1, \ldots a_n, a_{n+1}$ *be a sorted sequence of* $n + 1$ *numbers in an increasing order such that* $\Sigma_{i=1}^{n} a_i \leq B$ *and* $\Sigma_{i=1}^{n+1} a_i > B$*. Then* $a_{\lfloor n/2 \rfloor} \cdot n/2 \leq B$*. [see Singer and Mittal, 2013, Lemma 3.1]*

The proof of Lemma 1 is straightforward knowing the sum of the second half of the numbers is at most B and each of the numbers in the second half is at least as big as median.

Proof. Consider all the (worker, task) pairs assigned by the offline optimal and sort them in an increasing order of the bids. Let p^* denote the median of the bids. By Lemma 1, we can assign half of these (worker, task) pairs, and be sure not to exceed the budget. However, there are two problems: *(i)* we do not know which (worker, task) pairs are in the optimal solution and *(ii)* we do not know the value p^*.

To deal with problem *(i)*, suppose we know p^* and let us assign each worker to the task with the earliest deadline she can solve and continue until *(a)* we exhaust the budget or *(b)* there are no more workers or tasks left.

In case *(a)*, the algorithm made at least B/p^* assignments and by Lemma 1, we know this number is at least $\mathrm{OPT}/2$. For case *(b)*, consider the bipartite graph between the tasks and the workers described earlier for the min-cost flow problem construction. Since the algorithm does not assign any task to the workers with bid bigger than p^*, we can remove all such edges from the graph. Since p^* is the median of the bids in the optimal solution, we know that in this graph, there exists a matching M of size at least $\mathrm{OPT}/2$ between the workers and the tasks. On the other hand, since there are no tasks or workers left in this case, we know that OA has arrived at a *maximal* matching. Finally, since

the size of any maximal matching is at least half of the size of the maximum matching and consequently the matching M, the number of assignments of the algorithm is at least $\mathrm{OPT}/4$.

To deal with problem *(ii)* (not knowing p^*), we run the FTP for all the values that p^* can take and return the maximum number of assignments as our solution. Note that this might result in the algorithm using a p which is different than p^* but since the algorithm picks a p that maximizes the number of assignments we know the number of assignments made using p is at least as big as the number of assignments made using p^*. □

A brief detour: the homogeneous case. As we discussed in the introduction, we make a strong distinction between the heterogeneous tasks and homogeneous tasks. In the homogeneous case, a simple algorithm computes the offline optimal: sort all the workers by their bids and (if possible) assigns a task with the earliest deadline (smallest t_j) to the sorted workers until the budget is exhausted or there are no more tasks or workers left.

It is easy to verify that this greedy construction indeed computes the best offline assignments when the workers are homogeneous. However, this method will not result in the optimal offline assignment when the tasks are heterogeneous. For example consider the following toy problem with two workers and two tasks with the single deadline. Let the bids of worker 1 and 2 to be $(0.4, 0.5)$ and $(0.45, 0.7)$ for the two tasks, respectively. It is easy to see that with a budget of 1, the optimal offline assignment is to assign task 1 to worker 2 and task 2 to worker 1. However, the homogeneous greedy algorithm will assign task 1 to worker 1 and will not have enough budget left to assign a task to worker 2.

3.2 The Online Problem

Let's now move to the online setting, where workers arrive online and our algorithm must decide which (if any) task to assign the worker before seeing the remaining workers.

Our *Online Heterogeneous Algorithm* (OHA) is inspired by an algorithm from the online knapsack literature proposed by Zhou et al. [2008], with an analysis modified for our setting. The idea is to use a potential function $\phi : [0, 1] \to [1, R]$ based on the fraction of budget spent so far as an input. The ϕ function acts as a price threshold: the algorithm assigns a task to a worker only if the bid of the worker for any of the remaining unassigned tasks is below the value of the potential function—intuitively, as the budget shrinks, the algorithm becomes pickier about which workers to hire. Once a worker is selected, the algorithm greedily assigns the task with the earliest deadline to the worker. See Algorithm 3 for the code.

Algorithm 3 OHA

Input: Tasks J with deadlines $t_j \in [0, T]$, budget B.
Online input: Worker bids $b_{ij} \in [1, R]$, arrival time $s_i \in [0, T]$.
  Define $\phi(x) = \min((Re)^{1-x}, R)$.
  Let $x := 0, f := B$.
  While $B > 0$, on input i:
    Let $C_i := \{j \in J \mid b_{ij} \leq \min(f, \phi(x)) \text{ and } s_i \leq t_j\}$.
    If $C_i \neq \emptyset$ **then**
      Output $a(i) \in \arg\min_{j \in C_i} t_j$.
      Let $x := x + b_{i,a(i)}/B$.
      Let $f := f - b_{i,a(i)}$.
      Let $J := J \setminus \{a(i)\}$.
    else:
      Output: $a(i) :=\perp$.

Theorem 2. *Let* $\mathrm{ALG}_{\mathrm{OHA}}(\sigma)$ *and* $\mathrm{OPT}(\sigma)$ *denote the number of tasks assigned by the* OHA *and the offline optimal algorithm for a sequence of workers* σ*, respectively. Then for any sequence of workers* σ *with bids in* $[1, R]$ *such that*

$R \leq \epsilon B$, the competitive ratio is at most

$$\frac{\mathrm{OPT}(\sigma)}{\mathrm{ALG}_{\mathrm{OHA}}(\sigma)} = O(R^{\epsilon}\log(R)).$$

Proof. Let $S = \{(i,j)\}$ be the set of (worker, task) pairs assigned by the OHA where i and j index workers and tasks respectively, and let $S^* = \{(i^*, j^*)\}$ be the offline optimal. We want to bound $|S^*|/|S|$.

Let x_i and X denote the fraction of the budget used by the OHA when worker i arrives and upon termination, respectively. We will analyze the (worker, task) pairs in three stages. First, consider the common (worker, task) pairs which we denote by $(i,j) \in S \cap S^*$. Let $W = \sum_{(i,j)\in S\cap S^*} b_{ij}$ denote the total bid of such workers. Since each worker i in the common part who is assigned to task j is picked by the OHA, it must be that $b_{ij} \leq \phi(x_i)$. Therefore,

$$|S \cap S^*| \geq \sum_{(i,j)\in S\cap S^*} \frac{b_{ij}}{\phi(x_i)}. \tag{1}$$

Second, consider $(i,j) \in S^* \setminus S$. This can happen either because *(i)* the worker i is assigned to a task different than j by OHA, or *(ii)* the worker i is not assigned to any task by OHA. We know the number of pairs that satisfy the condition *(i)* is at most $|S|$ because all such workers are assigned to a task by both OHA and the offline optimal.

For the pairs that satisfy condition *(ii)*, either *(a)* $b_{ij} \leq \phi(x_i)$ or *(b)* $b_{ij} > \phi(x_i)$. It is easy to see that in case *(a)*, OHA assigned task j to some other worker i' who arrived before i. Again we know this can happen at most $|S|$ times. For case *(b)*, since ϕ is non-increasing then $b_{ij} > \phi(x_i) \geq \phi(X)$ for all such pairs (i,j). Since the offline optimal spent a budget of W for hiring workers in $S^* \cap S$, then it has a budget of at most $B - W$ to hire workers in case *(b)*. Since all the pairs in case *(b)* have $b_{ij} > \phi(x_i) \geq \phi(X)$, the offline optimal can hire at most $(B-W)/b_{ij} \leq (B-W)/\phi(X)$ workers. Adding up cases *(i)* and *(ii)*, we can bound

$$|S^* \setminus S| \leq 2|S| + \frac{(B-W)}{\phi(X)}. \tag{2}$$

Finally, consider $(i,j) \in S \setminus S^*$. Since $b_{ij} \leq \phi(x_i)$ for all such pairs, we know

$$|S \setminus S^*| \geq \sum_{(i,j)\in S\setminus S^*} \frac{b_{ij}}{\phi(x_i)}. \tag{3}$$

Putting all the pieces together, Equations (1) to (3) yield

$$\frac{\mathrm{OPT}(\sigma) - 2\cdot\mathrm{ALG}_{\mathrm{OHA}}(\sigma)}{\mathrm{ALG}_{\mathrm{OHA}}(\sigma)} = \frac{|S\cap S^*| + |S^*\setminus S| - 2|S|}{|S\cap S^*| + |S\setminus S^*|} \leq \frac{\sum_{(i,j)\in S\cap S^*} b_{ij}/\phi(x_i) + |S^*\setminus S| - 2|S|}{\sum_{(i,j)\in S\cap S^*} b_{ij}/\phi(x_i) + |S\setminus S^*|}. \tag{4}$$

To see why the inequality holds, first note that we know $|S\cap S^*| \geq \sum_{(i,j)\in S\cap S^*} b_{ij}/\phi(x_i)$ by Equation (1). Now if $\mathrm{OPT} = |S^*| < 3|S|$ then we are done. Otherwise,

$$\begin{aligned} |S^*| &\geq 3|S| \\ |S\cap S^*| + |S^*\setminus S| &\geq |S\setminus S^*| + |S\cap S^*| + 2|S| \\ |S^*\setminus S| - 2|S| &\geq |S\setminus S^*|. \end{aligned}$$

Hence, the inequality holds because if $a \geq b$ and $c \geq d$ then $(a+c)/(a+d) \leq (b+c)/(b+d)$.

We bound the right hand side of Equation (4) as follows.

$$\begin{aligned}
\frac{\sum_{(i,j)\in S\cap S^*} b_{ij}/\phi(x_i) + |S^* \setminus S| - 2|S|}{\sum_{(i,j)\in S\cap S^*} b_{ij}/\phi(x_i) + |S \setminus S^*|} &\leq \frac{\sum_{(i,j)\in S\cap S^*} b_{ij}/\phi(x_i) + 2|S| + (B-W)/\phi(X) - 2|S|}{\sum_{(i,j)\in S\cap S^*} b_{ij}/\phi(x_i) + |S \setminus S^*|} \\
&= \frac{\sum_{(i,j)\in S\cap S^*} b_{ij}/\phi(x_i) + (B-W)/\phi(X)}{\sum_{(i,j)\in S\cap S^*} b_{ij}/\phi(x_i) + |S \setminus S^*|} \\
&\leq \frac{\sum_{(i,j)\in S\cap S^*} \frac{b_{ij}}{\phi(X)} + (B-W)/\phi(X)}{\sum_{(i,j)\in S\cap S^*} b_{ij}/\phi(x_i) + |S \setminus S^*|} \\
&= \frac{W/\phi(X) + (B-W)/\phi(X)}{\sum_{(i,j)\in S\cap S^*} b_{ij}/\phi(x_i) + |S \setminus S^*|} \\
&= \frac{B/\phi(X)}{\sum_{(i,j)\in S\cap S^*} b_{ij}/\phi(x_i) + |S \setminus S^*|} \\
&\leq \frac{B/\phi(X)}{\sum_{(i,j)\in S\cap S^*} b_{ij}/\phi(x_i) + \sum_{(i,j)\in S\setminus S^*} b_{ij}/\phi(x_i)} \\
&= \frac{B/\phi(X)}{\sum_{(i,j)\in S} b_{ij}/\phi(x_i)} \\
&= \frac{1}{\phi(X)\sum_{(i,j)\in S} \Delta(x_i)/\phi(x_i)},
\end{aligned}$$

where $\Delta(x_i) := x_{i+1} - x_i$. The first inequality is due to Equation (2), the second is due to monotonicity of ϕ and the third inequality is due to Equation (3). Since $(i,j) \in S$, OHA assigns task j to worker i and increases the fraction of budget consumed by b_{ij}/B, so $b_{ij}/B = \Delta(x_i)$.

We now estimate the sum with an integral. If $\Delta(x_i) \leq \epsilon$,

$$\sum_{(i,j)\in S} \Delta(x_i)\frac{1}{\phi(x_i)} \geq \int_0^{X-\epsilon} \frac{1}{\phi(x_i)}dx.$$

Letting $c = 1/(1+\ln(R))$, we have $\phi(x) = R$ if $x \leq c$. Similar to Zhou et al. [2008], we bound the integral as follows.

$$\begin{aligned}
\int_0^{X-\epsilon} \frac{1}{\phi(x_i)}dx &= \int_0^c \frac{1}{R}dx + \int_c^{X-\epsilon} \frac{1}{\phi(x_i)}dx \\
&= \frac{c}{R} + \frac{1}{Re}\cdot\frac{1}{1+\ln(R)}((Re)^{X-\epsilon} - (Re)^c) \\
&= \left(\frac{c}{R} - \frac{1}{Re}\cdot\frac{1}{1+\ln(R)}(Re)^c\right) + \frac{1}{Re}\cdot\frac{1}{\ln(R)+1}(Re)^{X-\epsilon} \\
&= \frac{1}{Re}\cdot\frac{1}{1+\ln(R)}(Re)^{X-\epsilon} \\
&= \frac{1}{\phi(X)}\cdot\frac{(Re)^{-\epsilon}}{1+\ln(R)}.
\end{aligned}$$

It is easy to show by algebraic manipulation that the first term in the 3rd line is equal to 0. Therefore,

$$\frac{\mathrm{OPT}(\sigma) - 2\cdot\mathrm{ALG}_{\mathrm{OHA}}(\sigma)}{\mathrm{ALG}_{\mathrm{OHA}}(\sigma)} \leq \frac{1}{\phi(X)\sum_{(i,j)\in S}\Delta(x_i)/\phi(x_i)} \leq (Re)^{\epsilon}(\ln(R)+1).$$

Thus, $\mathrm{OPT}(\sigma) \leq ((Re)^{\epsilon}(\ln(R)+3)\cdot\mathrm{ALG}_{\mathrm{OHA}}(\sigma)$, as desired. □

3.3 Lower Bound on the Competitive Ratio

In this section we show that if the large market assumption is the only assumption on the sequence of workers, then no algorithm (even randomized) can achieve a constant competitive ratio. We prove the result for the special case that all the tasks have the same deadline and the tasks are homogeneous.

The proof is based on the ideas from a lower bound for the competitive ratio in a variant of the online knapsack problem by Zhou et al. [2008]. In this variant, items arrive one by one online and we have a knapsack with some known capacity. Each item has a weight and a utility parameter and it is assumed that the utility to weight ratio for all the items are within a bounded range. Furthermore, it is assumed that the weight of each item is small compared to the capacity of the knapsack. We have to decide whether to pick an item or not upon arrival and the goal is to maximize the utility of the picked items while satisfying the knapsack capacity constraint. Since our setting when the tasks are homogeneous and have the same deadline is a special case of the online knapsack variant, the lower bound for that problem might not necessarily provide us with a lower bound. However, with a bit of care, we can achieve the same lower bound using a similar construction.

The main idea of the lower bound is to construct hard sequences of workers where a hard sequence can be described as follows. The sequence starts with workers with maximum bid, R, and then the following workers progressively have smaller and smaller bids compared to the preceding workers. Then at some random point until the end of the sequence only workers with bid R appear in the sequence. Intuitively, these instances are hard because no algorithm can foresee whether the bids will decrease (when it should wait) or increase (when it should spend its budget on the current workers).

Theorem 3. *For any (possibly randomized) online algorithm, there exists a set of sequences of worker arrivals satisfying the large market assumption such that the competitive ratio of the algorithm on the sequence is at least* $\Omega(\log(R))$.

Proof. We modify the proof of Theorem 2.2 by Zhou et al. [2008] to fit our problem formulation.

We use Yao's *minimax principle*: by constructing a distribution over a set of instances and showing that no deterministic algorithm can achieve a expected competitive ratio which is better than $\ln(R)+1$ on these instances, Yao's principle implies that there no *randomized* algorithm can beat this competitive ratio on all (deterministic) instances [Yao, 1977].

To construct the hard distribution, fix $\eta \in (0,1)$, let k be the smallest integer such that $(1-\eta)^k \leq 1/R$, and define $k+1$ instances indexed by I_0 to I_k as follows. I_0 contains B/R identical workers all with bids equal to R. For all $u > 0$, I_u is I_{u-1} followed by $B/(R(1-\eta)^u)$ workers all with bids equal to $R(1-\eta)^u$. Since these instances have different length we pad all the instances with enough workers with bids R so that all the instances have the same length.

We specify D by $k+1$ values $p_0, \ldots, p_k$ where p_u denotes the probability of occurrence of instance I_u. Let

$$p_0 = p_1 = \ldots = p_{k-1} := \frac{\eta}{(k+1)\eta+1} \quad \text{and} \quad p_k := \frac{1+\eta}{(k+1)\eta+1}.$$

On this distribution of inputs, any deterministic algorithm is fully specified by the fraction of budget spent on hiring workers with bid $R(1-\eta)^u$; call each fraction f_u. Since the optimal assignment for instance i is to only hire workers with bids $R(1-\eta)^u$, the *inverse* of the expected competitive ratio can be bounded as follows.

$$\sum_{u=0}^{k} p_u \frac{\sum_{v=0}^{u} f_v B/(R(1-\eta)^v)}{B/(R(1-\eta)^u)} = \sum_{u=0}^{k} p_u \sum_{v=0}^{u} f_v (1-\eta)^{u-v} = \sum_{v=0}^{k} f_v \sum_{u=v}^{k} p_u (1-\eta)^{u-v}, \tag{5}$$

where the last statement is by expanding the sums and reordering the terms.

The second sum in the RHS of Equation (5) is bounded by

$$\sum_{u=v}^{k} p_u (1-\eta)^{u-v} = \frac{2\eta(1-\eta)^{k-v} + (1-\eta)}{(k+1)\eta+1} \leq \frac{2\eta+(1-\eta)}{(k+1)\eta+1} = \frac{1+\eta}{(k+1)\eta+1}, \tag{6}$$

where the first equality is derived exactly similar to Zhou et al. [2008]. Replacing Equation (6) into the RHS of Equation (5),

$$\sum_{v=0}^{k} f_v \sum_{u=v}^{k} p_u (1-\eta)^{u-v} \leq \frac{1+\eta}{(k+1)\eta+1} \sum_{v=0}^{k} f_v \leq \frac{1+\eta}{(k+1)\eta+1},$$

since by definition $\sum_{v=0}^{k} f_v \leq 1$.

Now by definition of k, we know $(1-\eta)^k \leq 1/R$, which implies $k+1 \geq \ln(R)/\ln(1/(1-\eta))$. So,

$$\sum_{v=0}^{k} f_v \sum_{u=v}^{k} p_u (1-\eta)^{u-v} \leq \frac{1+\eta}{(k+1)\eta+1} \leq \frac{1+\eta}{\eta \ln(R)/\ln(1/(1-\eta))+1} = O\left(\frac{1}{\ln(R)}\right)$$

as $\eta \to 0$, since $\lim_{\eta \to 0} \eta/\ln(1/(1-\eta)) = 1$. Thus the competitive ratio is at least $\Omega(\ln R)$. □

If we consider the limit where worker's bids are very small compared to the budget, $\epsilon \to 0$ and Theorem 2 shows that Algorithm 3 has a competitive ratio approaching $O(\ln(R))$, the best possible by Theorem 3.

4 Random Permutation Setting

Now that we have considered the worst-case scenario of inputs, let us consider more "well-distributed" inputs. We consider the *random permutation model* [Devanur and Hayes, 2009]. In the random permutation model there is no assumption about the bids of the workers (they can still be chosen by an adversary) but we measure the competitive ratio a bit differently: we take all possible permutations of these workers, and take the average competitive ratio over all permutations. Intuitively, this assumption makes the task assignment easier because the premium workers get distributed evenly in the sequence.

As pointed out by Devanur and Hayes [2009], the random permutation model can be considered as drawing bids from an unknown distribution without replacement. Hence, the random permutation model is very similar to the model that assumes the bids of the workers are drawn $i.i.d.$ from an unknown distribution.

Under a distributional assumption, if there is just a single deadline, the offline optimal assignment is a random variable while the offline optimal assignment is invariant of the permutation. This makes the analysis of the competitive ratio cleaner in the random permutation model. Unfortunately, when there are multiple deadlines, the best offline assignment *depends* on the permutation. In this section we restrict our discussion to the case that all the tasks have the same deadline, but where workers may still give heterogeneous bids.

Throughout this section, we assume that the number of available workers, n, is large, and known to us. However, we modify the large market assumption of Section 3 to only consider inputs where the offline optimal algorithm assigns at least a constant fraction of workers, i.e., $\mathrm{OPT}(\sigma) = \Omega(n)$ for all inputs σ.

For the case that the workers are homogeneous, Singer and Mittal [2013] provide an algorithm with competitive ratio of 360. In this section, we extend their result to the case the tasks are heterogeneous. We also improve on their competitive ratio, though the setting is a bit different: as they were concerned with mechanism design properties, it was important for Singer and Mittal [2013] to carefully tune their payments to (i) incentivize workers to bid honestly and (ii) compensate all workers, even workers at the very beginning. Our algorithm does not satisfy these properties.

If we knew the sequence of the workers and their bids, we could run OA to compute a threshold price p and number of assignments Q where we know Q is at least a quarter of the optimal number of assignments by Theorem 1. However, since we are in the random permutation model, we can estimate this threshold with high probability by observing a subset of workers. This idea is summarized as the Random Permutation Algorithm (RPA) in Algorithm 4.

The algorithm takes an input α and return an estimate $\hat{p}$ for the threshold p such that this estimate is larger than p with high probability. In the proof, we show that although this overestimation might decrease the performance of the algorithm, it is necessary in our analysis.

Theorem 4. *Let $\alpha, \delta \in (0,1)$, and suppose the number of workers is at least*

$$n \geq \Omega\left(\frac{1}{\alpha} \log(\frac{1}{\delta})\right),$$

Algorithm 4 RPA

Input: Parameter $\alpha \in (0,1)$, set of tasks J with deadlines $t_j \in [0,T]$, budget B.
Online input: Worker bids $b_{ij} \in [1,R]$, arrival time $s_i \in [0,T]$.
Let C be the first half of workers, don't assign.
Let $\hat{p} := \mathrm{OA}(C, J, B/2)$.
On rest of input, run $\mathrm{FTP}((1+\alpha)\hat{p}, B/2)$.

and $\mathrm{OPT} \geq \Omega(n)$ *for every input. For any sequence of workers* σ*, let* $\mathrm{ALG}_{\mathrm{RPA}}(\sigma)$ *and* $\mathrm{OPT}(\sigma)$ *denote the number of tasks assigned by the* RPA *and the offline optimal algorithm for a sequence of workers* σ*, respectively. Then,*

$$\mathrm{OPT}(\sigma) \leq 8(1+\alpha)^2/(1-\alpha) \cdot \mathrm{ALG}_{\mathrm{RPA}}(\sigma),$$

with probability at least $1-\delta$*.*

Proof. If we knew the sequence of the workers and their bids, we could run OA to compute a price p and number of assignments Q. We know $Q \geq \mathrm{OPT}/4$ by Theorem 1. Let us refer to these Q workers hired by OA as *good workers*.

We first claim that if we use a price of $(1+\alpha)\cdot p$ instead of p, the number of assignments will decrease by a factor of $(1+\alpha)$. This is because we can still assign a task to workers who OA assigned a task to with a threshold of p, but now we also have access to workers with bids in $(p, (1+\alpha)\cdot p]$ which may cause us to exhaust the budget faster.

Second, for any α, we can estimate p from the first half of the worker. Call this estimate $\hat{p}$. We claim that at least $(1-\alpha)Q/2$ of the good workers are in the second half with high probability, and

$$p \leq (1+\alpha)\hat{p} \leq (1+\alpha)p/(1-\alpha). \tag{7}$$

This together with the first claim and the 4-approximation of OA will give us the competitive ratio claimed in the statement of the theorem.

We first show that there are enough good workers in the second half. Since we are in the random permutation model, then with high probability, we know the number of good workers in the first half of the workers is in $[Q(1-\alpha)/2, Q(1+\alpha)/2]$. Chvatal [1979] show that this probability is bounded by

$$1 - 2\sum_{i=Q(1+\alpha)/2}^{Q} \frac{\binom{i}{Q/2}\binom{n/2-i}{n-Q}}{\binom{n}{n/2}} = \begin{cases} 1 - O(e^{-\alpha Q^2/n}) & \text{if } Q \geq \frac{n}{2}, \\ 1 - O(e^{-\alpha(n-Q)^2 Q/n^2}) & \text{if } Q < \frac{n}{2} \end{cases} = 1 - O(e^{-\alpha n}).$$

To show Equation (7), we consider two cases.

1. The number of good workers in the first half is $[Q(1-\alpha)/2, Q/2)$. In this case $\hat{p} \geq p$ because we can obviously use the budget to assign tasks to all the good workers. And if there is leftover budget, we might be able to assign tasks to workers with bid greater than p. So,

$$\hat{p}\frac{Q}{2}(1-\alpha) \leq \frac{B}{2} = \frac{pQ}{2} \qquad \text{(last line of OA (Algorithm 2))}$$
$$\hat{p} \leq \frac{1}{(1-\alpha)}p,$$

   Combined with $\hat{p} \geq p$, the condition in Equation (7) holds in this case.

2. The number of good workers in the first half is $[Q/2, Q(1+\alpha)/2]$.

   In this case $\hat{p} \leq p$. Suppose by contradiction that $\hat{p} > p$. This means the workers hired by the OA with bids of at most $\hat{p}$ is at least $S = B/(2\hat{p})$. We know $S \geq Q/2$ because all good workers have bids of most p. So,

$$\frac{B}{2\hat{p}} = \frac{pQ}{2\hat{p}} \geq \frac{Q}{2}$$
$$p \geq \hat{p},$$

which is a contradiction. Thus,

$$\hat{p}\frac{Q}{2}(1+\alpha) \geq \frac{B}{2} = \frac{pQ}{2}$$
$$\hat{p} \geq \frac{1}{1+\alpha}p.$$

Combined with $\hat{p} \leq p$, the condition in Equation (7) also holds in this case, concluding the proof.

□

5 Related Work

Pricing and task assignments have been previously studied in the context of crowdsourcing markets. Singer and Mittal [2011, 2013] provide mechanisms for task assignment when tasks are homogeneous and the workers are arriving from a random permutation model. Our work in the random permutation section generalizes their work to heterogeneous tasks. Ho and Vaughan [2012] consider task assignment when the tasks are heterogeneous. However, the assume they are task types and the focus of their work is learning qualities of the workers from stochastic observations. Singla and Krause [2013] design price mechanism for crowdsourcing markets using tools from online learning. Goel et al. [2014] consider the heterogeneous task assignment in the offline setting and provide near optimal approximation algorithms. However, they focus on the mechanism design aspect of the problem—how to pay workers so that they report their bids truthfully. These pricing mechanisms have close connections to the stochastic online adwords problem [Devanur and Hayes, 2009] and the online primal-dual literature [Buchbinder et al., 2007].

Our work is inspired by variants of the online knapsack problem. In the adversarial setting with homogeneous tasks and a single deadline, our problem is an instance of the online knapsack problem studied by Zhou et al. [2008] (see references within for more information). However, it is not clear how to formulate our problem as a knapsack problem when tasks are heterogeneous or have deadlines.

Variants of *generalized online matching* and the *adwords* problem are also related to our problem ($e.g.$, see Mehta et al. [2005] and Mehta [2013] for an excellent survey). The adwords problem can be described as follows. There are some bidders and each bidder has a fixed budget. Queries arrive one at the time, bidders bid for the queries and we have to decide what bidder to assign to the query. If we assign a bidder to a query, the bidder pay us by the amount of her bid. Our goal is to maximize the revenue. While one might attempt to formalize our problem as an instance of the online adwords problem, our constraint on the total budget of the requester cannot be written as an adwords type budget constraint.

6 Future Work

It would be interesting to study the empirical performance of our algorithms on crowdsourcing markets. Unfortunately crowdsourcing markets such as Amazon Mechanical Turk do not allow workers to bid for the HITs, but ideas such as creating an intermediate platform ($e.g.$, see *Mechanical Perk* [Singer and Mittal, 2013]) could be used to get around this limitation.

Another extension would be to extend the results in the random permutation model for the case where the tasks have different deadlines. One difficulty is in defining the right notion for competitive ratio. Since the offline optimal will change by permuting the workers when the tasks have different deadlines, it is not clear whether the competitive ratio should be defined as an expected ratio or ratio of an expectation. Our bound for the adversarial model immediately gives an $O(R^{\epsilon}\ln(R))$ bound on the expected ratio over all permutations, but this requires the bids to be small: $R \leq \epsilon B$. Furthermore, the competitive ratio should be better under random permutation than under adversarial inputs, like the single deadline case. Sharpening the competitive ratio and removing the smallness condition are both fascinating open problems.

References

Niv Buchbinder, Kamal Jain, and Joseph Naor. Online primal-dual algorithms for maximizing ad-auctions revenue. In *Proceedings of the 15th Annual European Symposium on Algorithms*, pages 253–264, 2007.

Vaclav Chvatal. The tail of the hypergeometric distribution. *Discrete Mathematics*, 25(3):285 – 287, 1979.

Nikhil Devanur and Thomas Hayes. The adwords problem: online keyword matching with budgeted bidders under random permutations. In *Proceedings of 10th ACM Conference on Electronic Commerce*, pages 71–78. ACM, 2009.

Gagan Goel, Afshin Nikzad, and Adish Singla. Mechanism design for crowdsourcing markets with heterogeneous tasks. In *Proceedings of the Second AAAI Conference on Human Computation and Crowdsourcing*, 2014.

Chien-Ju Ho and Jennifer Vaughan. Online task assignment in crowdsourcing markets. In *Proceedings of the 26th AAAI Conference on Artificial Intelligence*. AAAI Press, 2012.

Aranyak Mehta. Online matching and ad allocation. *Foundations and Trends in Theoretical Computer Science*, 8(4): 265–368, 2013.

Aranyak Mehta, Amin Saberi, Umesh Vazirani, and Vijay Vazirani. Adwords and generalized on-line matching. In *46th Annual IEEE Symposium on Foundations of Computer Science*, pages 264–273, 2005.

Yaron Singer and Manas Mittal. Pricing tasks in online labor markets. In *Proceedings of the 3rd Human Computation Workshop*, 2011.

Yaron Singer and Manas Mittal. Pricing mechanisms for crowdsourcing markets. In *Proceedings of 22nd International World Wide Web Conference*, pages 1157–1166, 2013.

Adish Singla and Andreas Krause. Truthful incentives in crowdsourcing tasks using regret minimization mechanisms. In *Proceedings of the 22nd International World Wide Web Conference*, pages 1167–1178. International World Wide Web Conferences Steering Committee/ACM, 2013.

Andrew Chi-Chin Yao. Probabilistic computations: Toward a unified measure of complexity. In *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, pages 222–227. IEEE, 1977.

Yunhong Zhou, Deeparnab Chakrabarty, and Rajan Lukose. Budget constrained bidding in keyword auctions and online knapsack problems. In *Proceedings of Internet and Network Economics, 4th International Workshop*, volume 5385 of *Lecture Notes in Computer Science*, pages 566–576. Springer, 2008.