Assigning Tasks to Workers based on Historical Data: Online Task Assignment with Two-sided Arrivals

John P. Dickerson
University of Maryland, College Park
john@cs.umd.edu

Karthik Abinav Sankararaman
University of Maryland, College Park
kabinav@cs.umd.edu

Aravind Srinivasan
University of Maryland, College Park
srin@cs.umd.edu

Pan Xu
University of Maryland, College Park
panxu@cs.umd.edu

ABSTRACT

Efficient allocation of tasks to workers is a central problem in crowdsourcing. In this paper, we consider a special setting inspired from spatial crowdsourcing platforms where both workers and tasks arrive dynamically. Additionally, we assume all tasks are heterogeneous and each worker-task assignment brings a distinct reward. The natural challenge lies in how to incorporate the uncertainty in the arrivals from both workers and tasks into our online allocation policy such that the total expected rewards are maximized. To attack this challenge, we assume the arrival patterns of worker "types" and task "types" are not erratic and can be predicted from historical data. To be more specific, we consider a finite time horizon T and assume in each time-step, a single worker and task are sampled (i.e., "arrive") from two respective distributions independently, and this sampling process repeats identically and independently for the entire T online time-steps.

Our model, called *Online Task Assignment with Two-Sided Arrival* (OTA-TSA), is a significant generalization of the classical online task assignment where the set of tasks is assumed to be available offline. For the general version of OTA-TSA, we present an optimal non-adaptive algorithm which achieves an online competitive ratio of 0.295. For the special case of OTA-TSA where the reward is a function of just the worker type, we present an improved algorithm (which is adaptive) and achieves a competitive ratio of at least 0.343. On the hardness side, along with showing that the ratio obtained by our non-adaptive algorithm is the best possible among all non-adaptive algorithms, we further show that no (adaptive) algorithm can achieve a ratio better than 0.581 (unconditionally), even for the special case of OTA-TSA with homogenous tasks (i.e., all rewards are same). At the heart of our analysis lies a new technical tool (which is a refined notion of the birth-death process), called the two-stage birth-death process, which may be of independent interest. Finally, we perform numerical experiments on two real-world datasets obtained from crowdsourcing platforms to complement our theoretical results.

Proc. of the 17th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2018), M. Dastani, G. Sukthankar, E. Andre, S. Koenig (eds.), July 2018, Stockholm, Sweden

https://doi.org/doi

CCS CONCEPTS

• **Theory of computation** → **Scheduling algorithms**; • **Mathematics of computing** → *Combinatorial optimization*; *Probabilistic algorithms*; *Discrete optimization*;

KEYWORDS

Crowdsourcing model; Online Matching; Randomized Algorithms

1 INTRODUCTION

Assigning workers to tasks is a central challenge in various crowdsourcing platforms. For example, in mobile crowd-sensing [29, 30], a central platform allocates mobile users to complex data collection and analysis tasks; in joint crowdsourcing [5, 16], workers answer small questions with varying difficulties; and in spatial crowdsourcing [26, 27], workers and tasks are matched in the context of a metric space.

More recently, a special class of the worker-task assignment , called the *online task assignment* (OTA), has attracted lots of attention. The basic setting is as follows: the set of tasks are known beforehand while the set of workers is revealed sequentially in an online manner; once a worker arrives, they have to be instantaneously and irrevocably assigned to a task. Each assignment gives a known profit (uniform or non-uniform) and the goal is to design an allocation policy such to maximize the (expected) total profit, while satisfying various practical constraints such as the total budget for payments for workers, deadlines of tasks, etc (*e.g.,* Assadi et al. [3]). There are three common arrival assumptions for the online workers: adversarial order (AO, the arrival sequence is unknown and can be arbitrarily fixed by an adversary), random arrival order (RAO, the arrival sequence is sampled from the set of all permutations over the workers) and known independent and identical distribution (KIID, a worker is sampled, with replacement, from a known distribution each time). Ho and Vaughan [14] considered OTA under RAO where they assume the profit for each assignment has to be learnt. Assadi et al. [3] studied a budgeted version of OTA under AO and RAO; in the budgeted version we have a global total budget and each assignment incurs a cost, which is the amount we need to pay the worker (this is equal to the bid they submitted for the task after arrival). The budgeted version of OTA and its generalizations have been vastly studied in the context of truthful mechanism design, where the goal is to elicit truthful bids from the online workers (*e.g.,* see [10, 11, 22–24, 31, 32]). In particular, Singer and Mittal [23], Singla and Krause [24] considered the KIID setting

while Zhao et al. [32] and Subramanian et al. [25] considered the RAO setting.

In OTA, the main limiting assumption is that tasks are static (known in advance). This fails to capture various applications where the tasks are not all available at once and come in an online manner similar to the workers. This is a common scenario in spatial crowdsourcing platforms. Hassan and Curry [13] considered a practical worker-task assignment under a converse setting to the OTA, where the spatial tasks come dynamically while the workers are static. The worker has to travel to the specific location of the task to finish it. [27] studied a generalized setting where both workers and tasks come online which was motivated from a spatial crowdsourcing platform on university campus, where anyone on campus can both post micro-tasks, (*e.g.,* buying drinks or collecting a package), and perform tasks as a worker. They assumed that the arrivals is sampled from the distribution over all permutations of both workers and tasks together and is unknown to the algorithm. They tested their algorithms on two real-world crowdsourcing datasets, namely gMission [7] and EverySender.

Inspired by the above work (Hassan and Curry [13], Tong et al. [27]), we propose the online task assignment with two-sided arrival (OTA-TSA) where both workers and tasks come online but under the arrival setting of KIID. We first briefly review the setting in the OTA under KIID — a known bipartite graph $G = (U, V, E)$ is given as input (this graph is also called the compatibility graph throughout this paper), where U and V represent the respective set of worker-types and task-types; we have a finite time horizon of T in which vertices in U are revealed step-by-step in each time-step (while all vertices in V are already given). In every time-step a worker of a particular type is sampled from a known distribution over U and the samples are independent across all the T rounds. We generalize the KIID setting from one-sided arrival to two-sided arrival in the following natural way — in each round (for a total of T rounds) a worker of type u is sampled from a known distribution over U, while simultaneously a task of type v is sampled from another known distribution over V independently. We now motivate the key assumptions in OTA-TSA.

Known independent and identical arrival distributions (KIID). In many crowdsourcing platforms, one collects meta-data about the tasks and workers. This data is used to predict both the performance and the arrival times of various workers and tasks in future (*e.g.,* [8], [21]). Hence allowing the underlying compatibility graph G and the arrival sequence of tasks and workers to be arranged by an adversary seems strong. We can exploit the rich historical data to predict both the compatibility graph and the arrival distributions of workers and tasks. This motivates us to consider the KIID model, which is assumed by many previous work (*e.g.,* [23, 24]).

The number of types: workers versus tasks. The majority of previous work in the OTA assumes that the number of task-types is far lesser than that of worker-types. This assumption is true in crowdsourcing platforms such as Amazon Mechanical Turk where individuals or organizations have a certain number of offline tasks and try to "crowdsource" the workers from Internet. In many spatial crowdsourcing platforms where both tasks and workers come online, as studied in this paper, the opposite is true: the number of task-types is far larger than that of worker-types. Hassan and Curry [13] run experiments on a real-world dataset of a location-based social network, called Gowalla, where the number of task-types is nearly 50 times that of worker-types. Moreover, in the two datasets considered by Tong et al. [27], namely gMission [7] and EverySender, the task-types are more than worker-types. We use the same in our experiments in Section 9.

Retention in the system: workers versus tasks. In the OTA-TSA model, we assume that (a) once a task arrives, it has to be instantaneously and irrevocably assigned to one of the workers who has arrived so far or reject the task; (b) once a worker arrives, they will stay in the system until being assigned. The assumption (2) here differs significantly from OTA since the motivating application is vastly different. OTA is primarily motivated by applications in crowdsourcing platforms such as Amazon Mechanical Turk where the tasks are offline while the number of available workers are plenty. In this context, once a worker comes into the system they expect to be allocated a task immediately; they have very little incentive to stay since they need to compete with a large pool of other workers for a limited set of tasks. However OTA-TSA is inspired from applications in spatial crowdsourcing platforms where worker-types are outnumbered by task-types. Any time a worker arrives but doesn't get a task assigned, they still have an incentive to stay since eventually they would be assured of an assignment.

In OTA-TSA, we consider a similar objective as that of OTA— every assignment $e = (u, v)$ fetches a non-uniform profit w_e and our goal is to design an allocation policy such that the expected total profit is maximized.

1.1 Our Contributions

Before detailing our contributions, we first review briefly some basic terminologies used in online algorithms.

Adaptive versus non-adaptive algorithms. Suppose we have a finite time horizon T. An algorithm ALG is called *non-adaptive* if the strategy for assigning a task v, when v comes for the k^{th} time for any $k = 1, 2, \ldots, T$ is specified before the start of the algorithm. In other words, the strategy does not depend on the *realization* of the arrival process thus far. If not, ALG is called *adaptive*.

Competitive ratio. Let $\text{ALG}(\mathcal{I}, \mathcal{P}_u, \mathcal{Q}_v)$ denote the expected value obtained by an algorithm ALG on an input $\mathcal{I}$ with arrival distributions being $\mathcal{P}_u$ and $\mathcal{Q}_v$ respectively for workers and tasks. Let $\text{OPT}(\mathcal{I})$ denote the expected *offline optimal*, which refers to the optimal value when we are allowed to make decisions after observing the entire sequence of online workers and tasks. The competitive ratio for a maximization program as studied in this paper, is defined as usual [4], $\inf_{\mathcal{I}, \mathcal{P}_u, \mathcal{Q}_v} \frac{\mathbb{E}[\text{ALG}(\mathcal{I}, \mathcal{P}_u, \mathcal{Q}_v)]}{\mathbb{E}[\text{OPT}(\mathcal{I})]}$. Thus when we say ALG achieves a ratio at least $\alpha \in (0, 1)$, it means that for any instance of the problem, the expected profit obtained by ALG is at least α fraction of the offline optimal.

Our contributions. First we propose a novel theoretical model, called Online Task Assignment with Two-Sided Arrival (OTA-TSA), where both workers and tasks arrive in an online manner. We consider the arrival setting of KIID and assume that the distributions can be learned from historical data.

Second we present a non-adaptive algorithm (NADAP) for the OTA-TSA, which is *optimal* among all possible non-adaptive algorithms (see Section 6). We show that NADAP achieves a ratio of almost 0.3, which is larger than that of 1/4 achieved by an adaptive algorithm shown in Tong et al. [27] for the same problem but under the arrival setting of RAO. This is a theoretical evidence showing the advantage of using historical data to predict the arrival distributions. Our main approach is to construct and solve an appropriate linear program (abbreviated as LP) and use that LP solution to guide the online actions.

Third we propose two adaptive algorithms for two special cases (see Section 7). The first one is a warmup algorithm, which is greedy (GREEDY), for the simple unweighted case. We show that it is as good as the best non-adaptive algorithm NADAP even when using a loose analysis. In fact there might exist a tighter analysis for GREEDY which shows that its performance is much better than NADAP. Our experimental analysis in Section 9 confirms this intuition. The second is an adaptive (ADAP) algorithm for the OTA-TSA when all the edges incident to each worker—representing the set of all acceptable tasks for that worker—have the same weight. We show that ADAP achieves an improved ratio of nearly 0.34. To accomplish this, we construct and solve a stronger LP than the one used for the non-adaptive algorithm and combine this with other ideas previously used for other online matching problems.

Fourth we show an unconditional hardness result for the OTA-TSA: no adaptive algorithm can achieve a ratio better than 0.58 even for the unweighted case (Section 8). Note that Brubach et al. [6] gave an adaptive algorithm, which yields a ratio of 0.729 for the classical online matching on an unweighted bipartite graph under KIID but with only one-sided arrival. This formally corroborates our intuition that the complexity significantly increases from one-sided arrival to the two-sided arrival.

Finally, we run numerical experiments on two real-world crowdsourcing datasets, namely gMission [7] and EverySender [27]. In particular, we find that despite having provable guarantees, we are able to obtain much better performance by using GREEDY algorithm. Our experimental analysis also generalizes this model, where we assume that at each time-step a batch of workers and batch of tasks arrive. We discuss intuitions and scenarios on when greedy is the right choice and when the LP based algorithms are the better option.

While building the theory in this paper, we construct a novel technical tool, called *two-stage birth-death process*, to attack the challenges arising in the competitive ratio analysis and derivation of hardness results. This technical tool might be of independent interest to prove competitive ratios in other settings.

2 OTHER RELATED WORK

We now briefly overview related research in the classical online matching; for a more in-depth review, we direct the readers to a recent review article by Mehta [19].

Modern online matching research is primarily motivated by Internet advertising applications. In this model, we are given a bipartite graph $G = (U, V, E)$ where U and V represent respectively the *offline* advertisers and *online* keywords (impressions). Each time, once a vertex $v \in V$ arrives, we have to make an instant and irrevocable decision: either reject v or assign v to an unmatched neighbor $u \in U$ and obtain a profit w_e for the match $e = (u, v)$. The central question is to design an online allocation policy such that the expected profit is maximized under different arrival assumptions such as AO, RAO, and KIID (see *e.g.,* [1, 2, 6, 9, 12, 15, 17, 18, 20]).

Departing from the traditional online matching, Wang and Wong [28] introduced a theoretical model of online matching (and online vertex cover) on a general graph G admitting the online arrival from all vertices. Their setting is as follows: each time a single vertex comes (in an *adversarial* order) and all its incident edges to previously arrived vertices are revealed. We are required to maintain a fractional matching (or vertex cover) on the revealed subgraph so far at all times and the goal is to maximize the size of the final matching (or minimize the size of the final vertex cover).

3 PROBLEM STATEMENT

Before we describe our OTA-TSA model, we define the following terminology. We group a set of similar tasks and call them "task types". Similarly, we group similar workers and call them "worker types". For example, in the context of spatial crowdsourcing, all workers present at a particular location belong to a single worker type.

Our model is as follows: suppose we have a bipartite graph (known to the algorithm) $G = (U, V, E)$ where U and V represent the set of worker-types and task-types respectively and E represents the set of worker-task pairs that are "compatible", *i.e.,* $(u, v) \in E$ iff any worker of type u can work on tasks of type v. We have a finite time horizon T (known beforehand) and for each time $t \in [T]$, a worker u from U and a task v from V is sampled (we also say u or v *arrives* or *comes* interchangeably) independently from *known* probability distributions $\mathcal{P}_u = \{p_u\}$ and $\mathcal{Q}_v = \{q_v\}$ respectively (*i.e.,* $\sum_u p_u = 1$ and $\sum_v q_v = 1$). The sampling process is independent across the different time-steps.

At each time $t \in [T]$, we first observe the online arrivals from U and V (in that order). Let u and v be the respective arrivals. We then need to make an *instantaneous and irrevocable decision* to either reject v or assign v to one of its available compatible workers in U. For each $u \in U$, once it arrives, it will stay in the system until being assigned to some task.[1] As discussed in introduction, we have that $|U| \ll |V|$. In our model, we additionally assume that each u has an integral arrival rate, *i.e.,* $T * p_u$ is an integral for every u, and thus w.l.o.g. we can assume this integer to be 1 (by splitting each u into $T * p_u$ copies). Hence, we assume that $|U| = T$ and $p_u = 1/T$ for all u.[2]

With each assignment $f = (u, v)$ we associate a non-negative profit w_f. Let $r_v = T * q_v$ (referred to as the arrival rate of v) be the expected number of arrivals of v during the T rounds. We assume this rate to be any number between $[0, 1]$ (upper bounding it by 1 is again w.l.o.g. via simple scaling). Our goal is to design an online assignment policy such that the total expected profits of all assignments made is maximized. Throughout this paper, we use edge $f = (u, v)$ and the assignment of v to u interchangeably.

[1] Here w.l.o.g. we assume that each worker has the capacity to perform only one task. In case, some worker type u can perform multiple tasks, we can split u into multiple copies. This forms the *matching constraint* for a worker.

[2] The assumption of integral arrival rate is a standard assumption in the classical online bipartite matching under known distributions, see *e.g.,* [9].

Additionally, when we say at time $t \in [T]$, we mean we are *at the beginning of time t* either before or after observing the arrivals from U and/or V (clarified in the context) but before the algorithm has made an online action.

4 TWO STAGE BIRTH-DEATH PROCESS

We propose a new stochastic process, called *two-stage birth-death process* (TS-BDP), and use it as a main tool to analyze our algorithms and derive hardness results. This technical tool seems more general and could be of independent interest. The process (described on random variables $\{X_t, Y_t\}$ and parameterized by values p and q) is described as follows. Consider a stochastic process with time horizon T such that, (1) the process starts at $t = 1$ with $X_1 = 0$; (2) at every round t, first there is a birth event *followed by an independent death event*. For the birth event, we have $Y_t = X_t + 1$ with probability p/T and $Y_t = X_t$ with probability $1 - p/T$. For the death event, it has a left boundary point of 0; *i.e.,* if $Y_t = 0$, then $X_{t+1} = Y_t$, else when $Y_t \geq 1$, we have $X_{t+1} = Y_t - 1$ with probability q/T and $X_{t+1} = Y_t$ with probability $1 - q/T$. We refer to p and q as the birth and death rate of TS-BDP respectively. TS-BDP differs from the classical birth-death process (BDP) in that, BDP is described as a process where in every round, birth and death occur each with a respective probability. On the other hand, in TS-BDP the two events occur *independently* in a *sequential* manner (the birth event is followed by the death event). Thus, TS-BDP is a special case of BDP. (TS-BDP is a BDP with time-horizon $2T$ where every odd step is a birth event and even step is a death event).

Definition 4.1. A two-stage birth-death process parameterized by (T, p, q) (time horizon, birth rate, death rate) refers to a sequence of random variables $\{X_t, Y_t | t \in [T]\} \cup \{X_{T+1}\}$ which satisfies (1) $X_1 = 0$ with probability 1; (2) For every $t \in [T]$, $Y_t = X_t + 1$ with probability p/T and $Y_t = X_t$ otherwise; (3) For every $t \in [T]$, if $Y_t = 0$, then $X_{t+1} = Y_t$ with probability 1; if $Y_t \geq 1$, then $X_{t+1} = Y_t - 1$ with probability q/T and $X_{t+1} = Y_t$ otherwise.

In this paper, we are particularly interested in the case when $p = 1$, $q \geq 0$ is a constant for a sufficiently large T ($T \to \infty$). We denote this specialization with TS-BDP$(1, q)$ (or TS-BDP(q) when the context is clear). For every $t \in [T]$, let $\Delta(t, T) := Y_t - X_{t+1}$ and $\Delta(T) := \sum_{t \in [T]} \Delta(t, T)$, which can be interpreted as the total number of death events in which Y_t is decreased by 1. Let $\kappa(q) := \lim_{T \to \infty} \mathbb{E}[\Delta(T)]$. We now state some useful lemmas which we use later.

Lemma 4.2. *(1)* $\kappa(0) = 0$, *(2)* $\kappa'(0) = 1/e$, *where* $\kappa'(0)$ *is the first derivative of* $\kappa(q)$ *at* $q = 0$.

Lemma 4.3. *(1)* $0.295 \leq \kappa(1) \leq 0.302$ *and (2)* $\kappa\left(1 + \frac{1}{e(e-1)}\right) \geq 0.343$.

Lemma 4.4. $\kappa(q)$ *is non-decreasing and concave over* $q \in [0, \infty]$.

Parts of the proof for the above lemmas require rigorous as well as computer-based analysis. To perform these numerical computations we use Mathematica 10. All numerical results are precise up to the third decimal place.

5 LINEAR PROGRAMS (LP)

As is common in this line of work, our algorithms use optimal solutions to linear programs (LP) constructed on the offline graph as a guide to the online algorithm. Additionally, this *benchmark* LP is used to upper bound the expected value of the optimal solution on a particular (offline) instance. Hence, to compute a lower bound on the competitive ratio, it suffices to compute the ratio of the value obtained by the algorithm to the optimal solution of this benchmark LP. We now describe the benchmark LP we use for our non-adaptive algorithm. Later, we show that this can further be strengthened based on some observations, which is used in our adaptive algorithm.

We associate a variable with every edge f in the graph. For each edge f, x_f denotes the expected number of matches in any offline optimal matching. For each u (resp. v), let E_u (resp E_v) be the set of its neighboring edges. Consider the following LP:

$$\text{maximize } \sum_{f \in E} w_f x_f \tag{1}$$

$$\text{subject to } \sum_{f \in E_v} x_f \leq r_v \quad \forall v \in V \tag{2}$$

$$\sum_{f \in E_u} x_f \leq 1 \quad \forall u \in U \tag{3}$$

$$x_f \geq 0 \quad \forall f \in E \tag{4}$$

The constraints (2) represent the fact that the expected total number of matches incident to a task v is no more than the expected number of arrivals of v. The same reasoning applies to constraints (3) but for workers. The constraint (4) represents the fact that the expected number of matches is non-negative. The objective function computes the expected reward obtained in the optimal offline solution. Thus we claim that for any offline optimal, $\{x_f\}$ should be feasible to the above LP. This suggests that LP-(1) is a valid benchmark LP (*i.e.,* the optimal value is an upper bound on the offline optimal). Throughout the paper we assume that $\{x^*_f\}$ is an optimal solution to this LP (or the stronger LP we define later, as appropriate). We now formally state the following lemma 5.1 showing the correctness of the benchmark LP.

Lemma 5.1. *The optimal value to* LP-(1) *is a valid upper bound for the offline optimal.*

6 NADAP: AN OPTIMAL NON-ADAPTIVE ALGORITHM

In this section, we present a non-adaptive algorithm, denoted by NADAP, which is *optimal* among all possible non-adaptive algorithms. Algorithm 1 describes our algorithm formally.

Algorithm 1: An optimal non-adaptive algorithm (NADAP)

1 Let v_t be a task arriving at time $t \in [T]$.
2 Sample an edge $f = (u, v_t) \in E_{v_t}$ with probability x^*_f / r_{v_t}. If worker u is available, then assign v_t to u; otherwise, skip v_t.

Constraint $\sum_{f \in E_v} x^*_f / r_v \leq 1$ in LP (2) justifies line 2 in NADAP.

Theorem 6.1. *The non-adaptive algorithm* NADAP *achieves a competitive ratio of* $\kappa(1) \geq 0.295$ *for the* OTA-TSA.

Proof. Consider a given u. Let X_t and Y_t be the number of copies of u before and after the arrival process from U at time t, respectively. From the assumption that u arrives with probability $1/T$ in each round, we have $Y_t = X_t + 1$ with probability $1/T$ and $Y_t = X_t$ with probability $1-1/T$. From NADAP, we have that if $Y_t \geq 1$, then it decreases by 1 with probability $\frac{x_u^*}{T} \doteq \sum_{f \in E_u} \frac{x_f^*}{r_v} \frac{r_v}{T} \leq \frac{1}{T}$ and it remains unchanged with the remaining probability (here $x_u^* = \sum_{f \in E_u} x_f^*$). From the definition of TS-BDP in 4.1, we have that $\{X_t, Y_t\}$ is a TS-BDP$(1, x_u^*)$ with time horizon of T.

Let A_f be the (random) number of matches for $f = (u, v)$ in NADAP over the T online rounds. Thus, we have:

$$\begin{aligned}
\mathbb{E}[A_f] &= \sum_{t \in [T]} \frac{r_v}{T} \frac{x_f^*}{r_v} \Pr[Y_t \geq 1] = \sum_{t \in [T]} \frac{x_f^*}{T} \Pr[Y_t \geq 1] \\
&= \frac{x_f^*}{x_u^*} \sum_{t \in [T]} \frac{x_u^*}{T} \Pr[Y_t \geq 1] \\
&= \frac{x_f^*}{x_u^*} \sum_{t \in [T]} \mathbb{E}[\Delta_t] \text{ (by the definitions of } \Delta_t) \\
&= x_f^* \frac{\kappa(x_u^*)}{x_u^*} \text{ (taking } T \to \infty \text{ and by the definition of } \kappa)
\end{aligned}$$

From Lemma 4.4, we have that κ is non-decreasing and concave over $[0, 1]$. Thus $\frac{\kappa(x)-\kappa(0)}{x-0}$ should be non-increasing over $x \in [0, 1]$. We also have that $\kappa(0) = 0$. Therefore,

$$\frac{\kappa(x_u^*)}{x_u^*} = \frac{\kappa(x_u^*)-\kappa(0)}{x_u^*-0} \geq \frac{\kappa(1)-\kappa(0)}{1-0} = \kappa(1)$$

Thus, we have that $\mathbb{E}[A_f] \geq x_f^* * \kappa(1)$. Since LP (1) is a valid upper bound on the optimal offline solution, by linearity of expectation, we have that NADAP achieves a competitive ratio of $\kappa(1)$. □

Hardness of non-adaptive algorithms. We will now show that any algorithm that is non-adaptive, cannot achieve a ratio better than $\kappa(1)$. In particular, we prove the following lemma.

Lemma 6.2. *No non-adaptive algorithm can achieve a competitive ratio better than $\kappa(1)$ even for the unweighted* OTA-TSA.

7 TWO ADAPTIVE ALGORITHMS

7.1 Warmup: Greedy for the unweighted case

Consider a simple special case of OTA-TSA where all assignments have uniform weights and all tasks have an integral arrival rate. In other words, we assume $|U| = |V| = T$, $p_u = p_v = 1/T$ for all $u \in U, v \in V$, and $w_f = 1$ for all $f \in E$. We formally state our greedy algorithm in Algorithm 2.

Algorithm 2: Greedy Algorithm (GREEDY)

1 Let v_t be a task arriving at time $t \in [T]$.
2 Choose an edge $f = (u, v_t)$ such that f has the largest weight among all available assignments to v_t at time t and assign v_t to u (break ties arbitrarily). Skip v_t if none is available.

Notice that for the unweighted case, GREEDY will choose an arbitrary available worker u whenever a task v arrives. We show that for the unweight case, GREEDY has a performance at least as good as that of the optimal non-adaptive algorithm NADAP.

Theorem 7.1. GREEDY *achieves a competitive ratio of at least $\kappa(1) \geq 0.295$ for the unweighted* OTA-TSA.

Proof. Consider an input graph $G = (U, V, E)$ and suppose we use LP-(1) as the benchmark. Since G is unweighted, we observe that the optimal value to LP-(1) is exactly equal to the size of a largest matching, say $\mathcal{M}$, on G. Let G' be the graph consisting of a perfect matching induced by $\mathcal{M}$. Note that the performance of GREEDY on G is no worse than G'. This can be seen as follows. Recall that during the online process, both u and v will join the system stochastically; each time when a v comes, GREEDY will match it to an arbitrary available neighbor u at that time, in which case we say u is shot down by v. The final performance of GREEDY is exactly the expected number of u which gets shot down. Consider a given arrival sequence from U and V, say S_u and S_v. Since the set of neighbors of v on G includes that of v on G' as a subset, v will always have more choice to shoot on G than G'. This implies that for any given S_u and S_v, the number of u shot down on G will be at least as much as that on G'.

Now we analyze the performance of GREEDY on G'. For a given $f \in \mathcal{M}$, we have that the expected number of matches of f is equal to $\kappa(1)$ (from the definition of κ). Thus we can claim that GREEDY has a performance of $\kappa(1) * |\mathcal{M}|$ on G'. Therefore the ratio of GREEDY is at least $\frac{\kappa(1)*|\mathcal{M}|}{|\mathcal{M}|} = \kappa(1)$. □

7.2 Adaptive algorithm for the node-weighted case

In this section, we consider a *relaxed* version of the problem where for any $u \in U$, all edges in E_u have the same weight $w_u \geq 0$. We denote this relaxed problem as OTA-TSA with left-hand side (LHS) vertex weighted. For this relaxation, one can *strengthen* the benchmark LP (1) by making the following observation; the probability that an edge can be matched is at most the probability that both the worker and the task is present at least once in the arrival sequence. This boils down to computing the expected value of the minimum of two i.i.d. Poisson random variables with mean upper bounded by 1. We later show that this expected value is at most $(1 - 1/e)r_v$ and hence adding this stronger constraint, we obtain the following strong LP (5). As a side note, this constraint is also valid for the general version of edge-weighted OTA-TSA, but the simpler LP suffices for an optimal non-adaptive algorithm.

$$\begin{aligned}
&\text{maximize } \sum_{u \in U} w_u \sum_{f \in E_u} x_f && && (5)\\
&\text{subject to } \sum_{f \in E_v} x_f \leq r_v && \forall v \in V && (6)\\
&\qquad \sum_{f \in E_u} x_f \leq 1 && \forall u \in U && (7)\\
&\qquad 0 \leq x_f \leq \left(1 - \tfrac{1}{e}\right) r_v && \forall f \in E && (8)
\end{aligned}$$

Lemma 7.2. *The optimal value to* LP (5) *is an upper bound on the offline optimal for the* OTA-TSA *with LHS vertex weighted.*

Our adaptive algorithm is inspired from an idea used in [17]. Let $\{x_f^*\}$ be an optimal solution to LP (5). At a particular time-step, when a task v arrives, we generate a random *ordered* list $\mathcal{L}$ of two choices from E_v such that it satisfies properties **(P1)** and **(P2)**.

(P1): $\Pr[\mathcal{L}(1) = f] = \frac{x_f^*}{r_v}$ for each $f \in E_v$.

(P2): $\Pr[\mathcal{L}(2) = f \wedge \mathcal{L}(1) \neq f] \geq \frac{x_f^*}{r_v} \frac{1}{e-1}$ for each $f \in E_v$.

Here $\mathcal{L}(1)$ and $\mathcal{L}(2)$ denotes the first and second choice on this list $\mathcal{L}$, respectively.

Later in this section, we will describe how to efficiently generate a random list satisfying the above two properties. Property **(P2)** relies critically on the stronger constraint (8) added into LP (5). On constructing a random list $\mathcal{L}$ at time t, ADAP will make the online decision as follows: try the first choice $\mathcal{L}(1)$ if it is available; then go to the second choice $\mathcal{L}(2)$; skip v_t if neither of the two choices are available. Thus compared to NADAP, ADAP offers each edge f a second chance to be tried. Property **(P1)** ensures that the marginal distribution is maintained for the first choice; Property **(P2)** gives a lower bound that each f can be tried as a second choice — this is the exact source for the improvement on the final ratio over the previous NADAP. Algorithm 3 formally describes ADAP.

Algorithm 3: An adaptive algorithm (ADAP)

1 Let v_t be a task arriving at time $t \in [T]$.
2 Generate a random list $\mathcal{L}$ satisfying properties **(P1)** and **(P2)**.
3 If first choice $\mathcal{L}(1)$ is available, assign v_t to $\mathcal{L}(1)$; else if second choice $\mathcal{L}(2)$ is available, assign v_t to $\mathcal{L}(2)$; otherwise skip v_t.

Theorem 7.3. *The adaptive algorithm* ADAP *achieves a competitive ratio of at least* $\kappa\left(1+\frac{1}{e(e-1)}\right) \geq 0.343$ *for the* OTA-TSA *with LHS vertex weighted.*

Proof. Consider a worker u. Let X_t and Y_t be the number of copies of u at time t before and after observing the arrival from U. Notice that from the assumption that u arrives with probability $1/T$ in each round, we have $Y_t = X_t + 1$ with probability $1/T$ and $Y_t = X_t$ with probability $1 - 1/T$.

Consider the case when $Y_t \geq 1$ and one compatible task v of u arrives at t. Let $\mathcal{L}$ be the random list that is generated for v at t. From ADAP, we have that Y_t decreases by 1 iff either (1) the assignment $f = (u, v)$ is made as a first choice ($\mathcal{L}(1) = f$) or (2) the assignment $f = (u, v)$ is made as a second choice ($\mathcal{L}(2) = f$) and the first choice $\mathcal{L}(1)$ is unavailable. Thus, we have:

$$\begin{aligned}
&\Pr[X_{t+1} = Y_t - 1 | v \text{ comes at } t\,] \\
&= \Pr[\mathcal{L}(1) = f] \\
&\quad + \Pr[\mathcal{L}(2) = f \wedge \mathcal{L}(1) \neq f] \Pr[\mathcal{L}(1) \text{ is not available}] \\
&\geq \frac{x^*_f}{r_v} + \frac{x^*_f}{r_v}\frac{1}{e-1} \Pr[\mathcal{L}(1) \text{ is not available}] \\
&\geq \frac{x^*_f}{r_v} + \frac{x^*_f}{r_v}\frac{1}{e-1}\frac{1}{e} = \frac{x^*_f}{r_v}\left(1+\frac{1}{e-1}\frac{1}{e}\right)
\end{aligned}$$

The inequality on the second line directly follows from properties **(P1)** and **(P2)**. The inequality on the third line is due to the fact that for each given $\mathcal{L}(1) = (u', v)$, the probability that it is unavailable is at least $(1 - 1/T)^t \geq 1/e$ (this refers to the probability that u' never comes in the first t time-steps). Thus, after considering all possible neighbors of u, we have

$$\begin{aligned}
\Pr[X_{t+1} = Y_t - 1] &\geq \sum_{f=(u,v)\in E_u} \frac{r_v}{T}\frac{x^*_f}{r_v}\left(1+\frac{1}{e-1}\frac{1}{e}\right) \\
&= \frac{\sum_{f\in E_u} x^*_f}{T}\left(1+\frac{1}{e-1}\frac{1}{e}\right)
\end{aligned}$$

Note that, $x^*_u = \sum_{f\in E_u} x^*_f \leq 1$ due to the constraint on each u in LP (5). We have that $\{X_t, Y_t\}$ is a TS-BDP with death rate of $x^*_u * q \doteq x^*_u\left(1+\frac{1}{e-1}\frac{1}{e}\right)$. From the definition of the function κ, we have that $\kappa(x^*_u * q)$ is equal to the expected number of matches for worker u. Note that x^*_u is the expected number of matches for u from the benchmark LP (5). Thus the resultant ratio is,

$$\begin{aligned}
\frac{\kappa(x^*_u * q)}{x^*_u} &= q * \frac{\kappa(x^*_u * q) - \kappa(0)}{x^*_u q - 0} \\
&\geq q * \frac{\kappa(q)}{q} = \kappa(q) = \kappa\left(1+\frac{1}{e-1}\frac{1}{e}\right)
\end{aligned}$$

The inequality above is due to the fact that κ is a concave function over $[0, \infty]$ and $x^*_u \leq 1$. □

Generating $\mathcal{L}$ satisfying properties (P1) and (P2). We can generate a random list $\mathcal{L}$ satisfying properties **(P1)** and **(P2)** as follows ([17] first use this idea). For every $e \in E_v$, let $y_e = x^*_e/r_v$; we have that $\sum_{e\in E_v} y_e \leq 1$. Add a dummy edge $e' = (u', v)$ with $y_{e'} = 1 - \sum_{e\in E_v} y_e$ (the edge $e' = (u', v)$ means we do nothing when v comes). Create two unit intervals, $\mathcal{I}_1$ and $\mathcal{I}_2$ as follows: (1) Sort $\{y_e | e \in E_v\} \cup \{y_{e'}\}$ in an increasing order; let $y_{e_1} \leq y_{e_2} \leq \ldots \leq y_{e_n}$ be this order; (2) Let $\mathcal{S}_i$ be a segment of length y_{e_i} with a label of e_i for each $i \in [n]$. Let $\mathcal{I}_1$ be the unit interval formed by $\{\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3, \ldots, \mathcal{S}_n\}$ and let $\mathcal{I}_2$ be the unit interval formed by $\{\mathcal{S}_n, \mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_{n-1}\}$.

The random list $\mathcal{L}$ is obtained from $(\mathcal{I}_1, \mathcal{I}_2)$ is as follows. Choose a value $x \in [0, 1]$ uniformly at random. Let $\mathcal{I}_1(x)$ and $\mathcal{I}_2(x)$ be the respective label of the segment where x falls on, in the intervals $\mathcal{I}_1$ and $\mathcal{I}_2$. Set $\mathcal{L}(1) = \mathcal{I}_1(x)$ and $\mathcal{L}(2) = \mathcal{I}_2(x)$.

Lemma 7.4. *The random list $\mathcal{L}$ generated by the procedure described above satisfies properties **(P1)** and **(P2)**.*

Proof. To verify property **(P1)**, notice that x is takes a value in $[0, 1]$ uniformly at random. Thus for each given $f \in E_v$, x falls in the segment labelled by f in $\mathcal{I}_1$ with probability $y_f = x^*_f/r_v$.

To verify property **(P2)**, we use Observation 4.1 from [17]. From this Observation, we have that $\Pr[\mathcal{L}(1) = \mathcal{L}(2) = f] = 0$, for every $f \in E_v$ with $y_f \leq 1/2$. Hence we have, $\Pr[\mathcal{L}(2) = f \wedge \mathcal{L}(1) \neq f] = y_f$. Consider the harder case when $y_f > 1/2$. The event that $\mathcal{L}(2) = f \wedge \mathcal{L}(1) \neq f$ occurs only when x falls in the segment labelled by f in $\mathcal{I}_2$ and x does not fall in the segment labelled by f in $\mathcal{I}_1$. Thus,

$$\begin{aligned}
\Pr[\mathcal{L}(2) = f \wedge \mathcal{L}(1) \neq f] &= y_f - (2y_f - 1) = y_f\left(\frac{1}{y_f} - 1\right) \\
&\geq y_f\left(\frac{1}{1-1/e} - 1\right) = \frac{y_f}{e-1}
\end{aligned}$$

The last inequality is because $y_f = x^*_f/r_v \leq 1 - 1/e$ for every $f \in E_v$ (this follows from the constraint (8)). □

8 HARDNESS RESULTS

We will now prove a hardness result, which also holds for the special case when there are no weights. (*i.e.,* unweighted) This hardness result does not depend on the choice of the benchmark LP and hence is unconditional. This hardness result is obtained due to the inherent nature of the online process and can be viewed as the

online-offline stochastic gap. In particular, we have the following theorem

THEOREM 8.1. *No algorithm can achieve a competitive ratio better than* $\frac{\kappa'(0)}{1-1/e} = \frac{1}{e-1} \sim 0.581$, *even for the unweighted* OTA-TSA.

PROOF. Consider an unweight bipartite graph $G = (U, V, E)$ where $|U| = |V| = T$ and $|E| = T$ which consists of a perfect matching. Let the arrival rates for every u be 1 with $p_u = 1/T$ and let every v have an arrival rate of ϵ (where ϵ is very small) with $p_v = \epsilon/T$. Here we can arrange a dummy node v' such that $p_{v'} = 1 - \epsilon$ and v' has no any neighbor of u.

Consider a given $f = (u, v)$. Let OPT-A and OPT-B be the respective offline and online optimal algorithms. Let X_f be the number of matches of f in OPT-A after the T rounds. Let X_u and X_v be the respective number of arrivals of u and v in an offline instance. We have that $X_f = \min(X_u, X_v)$. Observe that $X_u \sim \text{Pois}(1)$ and $X_v \sim \text{Pois}(\epsilon)$. Thus, we have:

$$\begin{aligned}
&\mathbb{E}[\min(X_u, X_v)] \\
&= \textstyle\sum_{k=1}^{\infty} \Pr[X_u \geq k] * \Pr[X_v \geq k] \\
&= \textstyle\sum_{k=1}^{\infty} \Pr[\text{Pois}(1) \geq k] * \Pr[\text{Pois}(\epsilon) \geq k] \\
&= \left(1 - \tfrac{1}{e}\right)\left(1 - e^{-\epsilon}\right) + \left(1 - \tfrac{2}{e}\right)\left(1 - e^{-\epsilon} - \epsilon e^{-\epsilon}\right) + \dots \\
&= \left(1 - \tfrac{1}{e}\right)\epsilon + o(\epsilon)
\end{aligned}$$

Hence we have $\mathbb{E}[X_f] = \left(1 - \frac{1}{e}\right)\epsilon + o(\epsilon)$. Let Y_f be the number of matches of f in OPT-B. Similar to the proof in Theorem 6.1, we can verify that $\mathbb{E}[Y_f] = \kappa(\epsilon)$.Thus the online ratio on the above instance should be

$$\frac{\mathbb{E}[Y_f]}{\mathbb{E}[X_f]} = \frac{\kappa(\epsilon)}{\left(1-\frac{1}{e}\right)\epsilon + o(\epsilon)}$$

Taking $\epsilon \to 0$, we have that the above value is

$$\lim_{\epsilon\to 0} \frac{\kappa(\epsilon)}{\left(1-\frac{1}{e}\right)\epsilon + o(\epsilon)} = \lim_{\epsilon\to 0} \frac{\kappa(\epsilon)}{\epsilon} \frac{1}{1-1/e} = \frac{\kappa'(0)}{1-1/e} \qquad \square$$

From Lemma 4.2, $\kappa'(0) = 1/e$ and thus we get our claim.

9 EXPERIMENTS

In this section, we describe the experimental results in this paper. We consider two datasets from popular crowdsourcing platforms, namely gMission [7] and EverySender [27]. We test our adaptive and non-adaptive algorithms on these two datasets. Additionally, we also consider a generalized version of our model and run experiments to show that these algorithms are robust enough for practical scenarios which might slightly vary from the actual model.

Dataset and preprocessing. Both the datasets have the following information. With every worker there is an associated location (x, y) where the worker is present, range of the worker which denotes the distance up to which they can perform a task, and a success probability which denotes the chance that this worker will complete any task. With every task there is an associated location (x, y) and a payoff value for completing the task. We group the workers (likewise for tasks) into a "type" if they share the same location in the sense that the first two decimal points in the x and y coordinates are the same. For example, workers at $(0.345, 3.546)$ and $(0.342, 3.549)$ are grouped as the same "type". To construct the compatibility graph between the tasks and workers (*i.e.,* the potential tasks a worker can perform), we consider every pair of task and worker type and add an edge between a task and worker type if the Euclidean distance between them is within the range of the worker type. To construct the edge weight, we multiply the payoff of the corresponding task type with the success probability of the corresponding worker type. In the (LHS) vertex-weighted version of the problem, we use the success probability as the edge-weight for all the edges incident to this worker. Recall that in our model all worker types have an uniform arrival probability $1/|U|$. We generate the task arrival probabilities by choosing a random vector $\{p_v\}$ such that each p_v is uniformly distributed over $[0, 1]$ conditioning on $\sum_v p_v = 1$. We achieve this by running the file `randfixedsum.m` due to Roger Stafford.[3] Finally to simulate large batch sizes for workers and tasks, we derive a sparse version of EverySender, called EverySenderSample, where each worker and task is chosen with probability 0.25. Table 1 gives basic statistics of the dataset, which corroborates some of our assumptions as discussed in the introduction.

Dataset	#worker types	#task types	#edges
gMission	532	712	39758
EverySender	817	3994	340051
EverySenderSample	204	999	21247

Table 1: Properties of our datasets

Heuristics. Alongside our main algorithms NADAP and ADAP, we adapt certain heuristics previously used for such problems (*e.g.,* [26]) and compare and contrast them with our algorithms under various *practical* scenarios. In particular, we consider the following three heuristics– GREEDY, LP-SCALED and UR-ALG. Both GREEDY and UR-ALG are agnostic to the underlying LP. The heuristic GREEDY matches the incoming task to the available worker where the weight of the assignment is the largest (breaking ties arbitrarily) while UR-ALG chooses one of the available workers uniformly at random. The heuristic LP-SCALED uses the optimal solution $\vec{x}$ to LP (1) as a guide to its online actions. When a task arrives, let $w_1, w_2, \dots, w_k$ denote the set of compatible workers who are available. Let $x_{w_1}, x_{w_2}, \dots, x_{w_k}$ denote the corresponding LP optimal values. We choose the worker w_i with probability $x_{w_i} / \sum_{j=1}^{k} x_{w_j}$.

Methodology. We parametrize the model with a parameter η which denotes the number of workers sampled (a.k.a. batch size of workers) in each time-step. Let Δ denote the ratio of total number of task *arrivals* to that of worker *arrivals*. For each given integral η, in each time-step we sample η workers and $\eta * \Delta$ tasks (by repeating the sampling process i.i.d., η times for workers and $\eta * \Delta$ for tasks). We set $\Delta = 2$ in gMission and $\Delta = 5$ in EverySender and EverySenderSample datasets. The values of Δ are chosen based on the ratios in the real arrival sequence for a snapshot when the dataset was curated. Our experiments are as follows with each experiment consisting of taking an average over 20 independent runs.

[3] https://www.mathworks.com/matlabcentral/fileexchange/9700-random-vectors-with-fixed-sum/content/randfixedsum.m

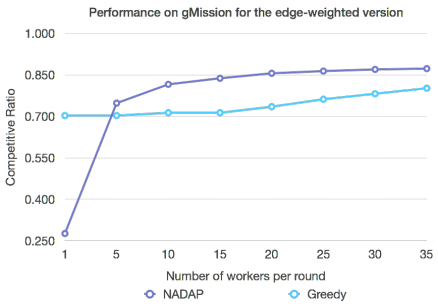

Figure 1: Edge-Weighted case on gMission dataset

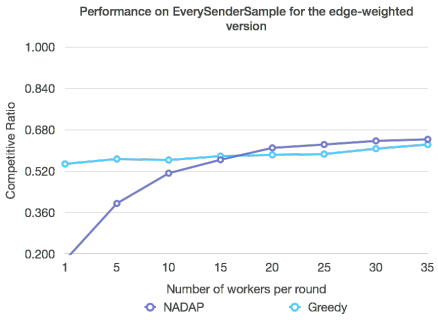

Figure 2: Edge-Weighted case on EverySender dataset

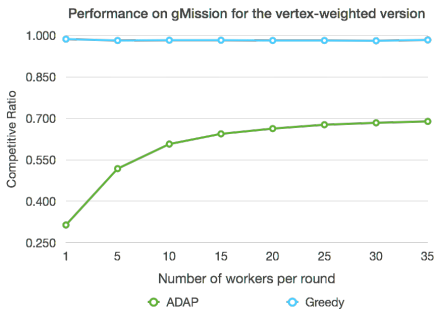

Figure 3: Vertex-Weighted case on gMission dataset

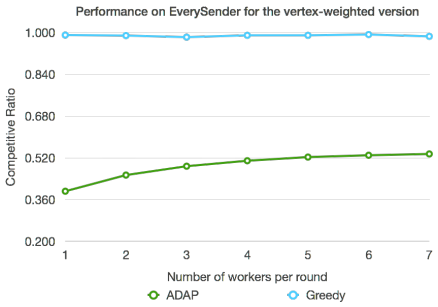

Figure 4: Vertex-Weighted case on EverySender dataset

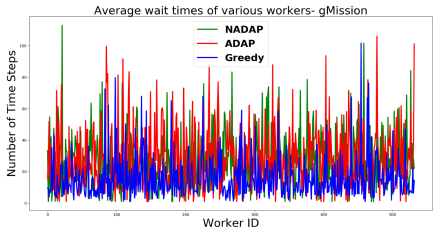

Figure 5: Average waiting times for workers, gMission dataset

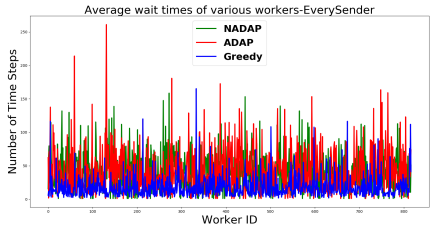

Figure 6: Average waiting times for workers, EverySender dataset

First we consider the case where the batch size of workers is 1 and the batch size of tasks is 1. For this case, we run GREEDY, ADAP and NADAP on the datasets gMission and EverySender with the node-weighted assumption and compute the average waiting time for all worker types. In particular, for each run of the algorithm and each worker in the system we measure the time until which this worker stays in the system before getting matched to a task. We then compute the average waiting time for every worker type across all the runs (counting it multiple times in a single run if a worker type arrives twice). Next we run our main experiments as follows. For the edge-weighted case, we test NADAP against the three heuristics GREEDY, LP-SCALED and UR-ALG, on the datasets of gMission and EverySenderSample over the choices of $\eta \in \{1, 5, 10, 15, 20, 25, 30, 35\}$. For the LHS vertex-weighted case, we test ADAP against the three heuristics, on the datasets gMission and EverySender over the choices of $\eta \in \{1, 2, 3, 4, 5, 6, 7\}$.

Results and discussion. For brevity, we only show the results of NADAP, ADAP and GREEDY in the plots. The performance of LP-SCALED and UR-ALG followed a similar pattern as GREEDY with LP-SCALED performing slightly better on average and UR-ALG performing slightly worse on average. The following are several interesting observations. From Figures 1 and 2, we see that NADAP performs better once the size of batch arrivals in each time increases. This can be explained as follows. When the batch-arrival size is small, each arriving task has a limited number of workers to choose from, since the number of workers who have arrived and are compatible is small. In this case, the advantage of greedily matching an available worker outweighs the potential loss from a mismatch. However, when batch-arrival size increases and each arriving task has more options to choose from, the guidance from the LP becomes effective, since it takes the future arrivals into consideration (in expectation). For the vertex-weighted case we have that GREEDY is *near optimal*. From Figures 3 and 4 we can see that the ratio obtained by GREEDY is almost close to 1 in all cases. On the other hand the performance of ADAP slowly increases as the batch size increases. Our experiments show that GREEDY is the best algorithm when there are no edge-weights or the weights are only on the workers. Finally Figures 5 and 6 show the average waiting time for each worker in the two datasets, in the run of the three algorithms. Since, GREEDY makes a choice whenever a compatible worker is available, it has the least waiting times. Similarly since ADAP makes strictly more assignments than NADAP, the workers in ADAP have the next least waiting time and in many cases much lesser than NADAP. Note however that the difference in the average of averages for GREEDY and NADAP is around 1.5-2.5% with respect to T in both the datasets and hence, is not considerably large.

ACKNOWLEDGMENTS

Aravind Srinivasan's research was supported in part by NSF Awards CNS-1010789, CCF-1422569 and CCF-1749864, and by research awards from Adobe, Inc. The research of Karthik Sankararaman and Pan Xu was supported in part by NSF Awards CNS 1010789 and CCF 1422569.

REFERENCES

[1] Elliot Anshelevich, Meenal Chhabra, Sanmay Das, and Matthew Gerrior. 2013. On the Social Welfare of Mechanisms for Repeated Batch Matching. In *AAAI-13*.

[2] Itai Ashlagi, Patrick Jaillet, and Vahideh H. Manshadi. 2013. Kidney Exchange in Dynamic Sparse Heterogenous Pools. In *EC-13*. 25–26.

[3] Sepehr Assadi, Justin Hsu, and Shahin Jabbari. 2015. Online Assignment of Heterogeneous Tasks in Crowdsourcing Markets. In *Third AAAI Conference on Human Computation and Crowdsourcing*.

[4] Allan Borodin and Ran El-Yaniv. 2005. *Online computation and competitive analysis*. Cambridge University Press.

[5] Jonathan Bragg, Andrey Kolobov, Mausam Mausam, and Daniel S Weld. 2014. Parallel task routing for crowdsourcing. In *Second AAAI Conference on Human Computation and Crowdsourcing*.

[6] Brian Brubach, Karthik Abinav Sankararaman, Aravind Srinivasan, and Pan Xu. 2016. New Algorithms, Better Bounds, and a Novel Model for Online Stochastic Matching. In *ESA-16*.

[7] Zhao Chen, Rui Fu, Ziyuan Zhao, Zheng Liu, Leihao Xia, Lei Chen, Peng Cheng, Caleb Chen Cao, Yongxin Tong, and Chen Jason Zhang. 2014. gmission: A general spatial crowdsourcing platform. *Proceedings of the VLDB Endowment* 7, 13 (2014).

[8] Djellel Eddine Difallah, Gianluca Demartini, and Philippe Cudré-Mauroux. 2013. Pick-a-crowd: tell me what you like, and i'll tell you what to do. In *WWW-13*.

[9] Jon Feldman, Aranyak Mehta, Vahab Mirrokni, and S Muthukrishnan. 2009. Online stochastic matching: Beating 1-1/e. In *FOCS-09*.

[10] Gagan Goel, Afshin Nikzad, and Adish Singla. 2013. Matching workers expertise with tasks: Incentives in heterogeneous crowdsourcing markets. In *NIPS Workshop on Crowdsourcing*.

[11] Gagan Goel, Afshin Nikzad, and Adish Singla. 2014. Allocating tasks to workers with matching constraints: truthful mechanisms for crowdsourcing markets. In *WWW-14*.

[12] Bernhard Haeupler, Vahab S Mirrokni, and Morteza Zadimoghaddam. 2011. Online stochastic weighted matching: Improved approximation algorithms. In *WINE-11*.

[13] U. U. Hassan and E. Curry. 2014. A Multi-armed Bandit Approach to Online Spatial Task Assignment. In *11th IEEE International Conference on Ubiquitous Intelligence and Computing (UIC-14)*. 212–219.

[14] Chien-Ju Ho and Jennifer Wortman Vaughan. 2012. Online Task Assignment in Crowdsourcing Markets.. In *AAAI-12*.

[15] Patrick Jaillet and Xin Lu. 2013. Online stochastic matching: New algorithms with better bounds. *Mathematics of Operations Research* 39, 3 (2013).

[16] Andrey Kolobov and Daniel S Weld. 2013. Joint crowdsourcing of multiple tasks. In *HCOMP-13*.

[17] Vahideh H Manshadi, Shayan Oveis Gharan, and Amin Saberi. 2012. Online stochastic matching: Online actions based on offline statistics. *Mathematics of Operations Research* 37, 4 (2012).

[18] Nicholas Mattei, Abdallah Saffidine, and Toby Walsh. 2017. Mechanisms for online organ matching. In *IJCAI-17*.

[19] Aranyak Mehta. 2012. Online matching and ad allocation. *Theoretical Computer Science* 8, 4 (2012).

[20] Reshef Meir, Yiling Chen, and Michal Feldman. 2013. Efficient parking allocation as online bipartite matching with posted prices. In *AAMAS-13*.

[21] Jeffrey M Rzeszotarski and Aniket Kittur. 2011. Instrumenting the crowd: using implicit behavioral measures to predict task performance. In *UIST-11*.

[22] Yaron Singer and Manas Mittal. 2011. Pricing Tasks in Online Labor Markets.. In *Human Computation*.

[23] Yaron Singer and Manas Mittal. 2013. Pricing mechanisms for crowdsourcing markets. In *WWW-13*.

[24] Adish Singla and Andreas Krause. 2013. Truthful incentives in crowdsourcing tasks using regret minimization mechanisms. In *WWW-13*.

[25] Ashwin Subramanian, G Sai Kanth, Sharayu Moharir, and Rahul Vaze. 2015. Online incentive mechanism design for smartphone crowd-sourcing. In *WiOpt-15*.

[26] Yongxin Tong, Jieying She, Bolin Ding, Lei Chen, Tianyu Wo, and Ke Xu. 2016. Online minimum matching in real-time spatial data: experiments and analysis. *Proceedings of the VLDB Endowment* 9, 12 (2016).

[27] Yongxin Tong, Jieying She, Bolin Ding, Libin Wang, and Lei Chen. 2016. Online mobile micro-task allocation in spatial crowdsourcing. In *ICDE-16*.

[28] Yajun Wang and Sam Chiu-wai Wong. 2015. Two-sided online bipartite matching and vertex cover: Beating the greedy algorithm. In *ICALP-15*.

[29] Mingjun Xiao, Jie Wu, Liusheng Huang, Ruhong Cheng, and Yunsheng Wang. 2017. Online task assignment for crowdsensing in predictable mobile social networks. *IEEE Transactions on Mobile Computing* 16, 8 (2017).

[30] Dejun Yang, Guoliang Xue, Xi Fang, and Jian Tang. 2012. Crowdsourcing to smartphones: Incentive mechanism design for mobile phone sensing. In *Proceedings of the 18th annual international conference on Mobile computing and networking*.

[31] Qi Zhang, Yutian Wen, Xiaohua Tian, Xiaoying Gan, and Xinbing Wang. 2015. Incentivize crowd labeling under budget constraint. In *INFOCOM-15*.

[32] Dong Zhao, Xiang-Yang Li, and Huadong Ma. 2014. How to crowdsource tasks truthfully without sacrificing utility: Online incentive mechanisms with budget constraint. In *INFOCOM-14*.