Online Matching with Stochastic Rewards

Aranyak Mehta* Debmalya Panigrahi†

Abstract

The online matching problem has received significant attention in recent years because of its connections to allocation problems in Internet advertising, crowd-sourcing, etc. In these real-world applications, the typical goal is not to maximize the number of allocations; rather it is to maximize the number of "successful" allocations, where success of an allocation is governed by a stochastic process which follows the allocation. To address such applications, we propose and study the online matching problem with stochastic rewards (called the ONLINE STOCHASTIC MATCHING problem) in this paper. Our problem also has close connections to the existing literature on stochastic packing problems; in fact, our work initiates the study of *online* stochastic packing problems.

We give a deterministic algorithm for the ONLINE STOCHASTIC MATCHING problem whose competitive ratio converges to (approximately) 0.567 for uniform and vanishing probabilities. We also give a randomized algorithm which outperforms the deterministic algorithm for higher probabilities. Finally, we complement our algorithms by giving an upper bound on the competitive ratio of any algorithm for this problem. This result shows that the best achievable competitive ratio for the ONLINE STOCHASTIC MATCHING problem is provably worse than that for the (non-stochastic) online matching problem.

*Google Research, Mountain View, CA 94043. Email: aranyak@google.com.

†Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA 02139. Email: debmalya@mit.edu. Part of this work was done when the author was an intern at Google Research, Mountain View, CA 94043.

1 Introduction

The *online matching* problem has gained considerable attention over the last few years, particularly because of its connections to **Internet Advertising**. In this problem (introduced in a celebrated paper of Karp, Vazirani, and Vazirani [20]), the input comprises a bipartite graph $G = (U \cup V, E)$, where the vertices in U (advertisers) are given offline, and a new vertex $v \in V$ (ad slot) and the set of edges incident on it are revealed in each online step. The algorithm can either match v to one of its available (i.e. currently unmatched) neighbors in U or not match v at all, with the overall goal of maximizing the number of matched pairs. This problem applies to other online settings as well, e.g. in matching tasks to users in **crowd-sourcing** (see e.g. [19, 18]). However, in many of these applications, the real objective is not the number of matched edges, rather it is the number of "successful matches". For example, the dominant revenue model in Internet advertising is that of *pay-per-click*, i.e. the advertiser pays only if the user clicks the ad. While the ad system has an estimate (called the *click-through-rate*) of the probability that an ad will be clicked if shown in the current ad slot, the actual clicking of the ad is governed by a stochastic process that comes after the ad has been shown to the user. Similarly, a crowd-sourcing system has an estimate of the probability that a user will successfully complete a job, but the actual completion is governed by a stochastic process that comes after the task has been allocated. This motivates us to study the online matching problem with stochastic rewards (we call it the ONLINE STOCHASTIC MATCHING problem and formally define it below). While a rich body of literature exists on multiple variants of the online matching problem, we are not aware of any previous attempt to study this problem in the presence of stochastic rewards.

From a modeling perspective, our problem has close connections to the class of *stochastic packing* problems that has received significant attention in recent years. This area of research was initiated by Dean, Goemans, and Vondrák [9] who defined the stochastic knapsack problem and studied it from an approximation algorithms perspective. Since then, a series of papers have considered a variety of optimization problems in this framework, including the knapsack problem [14, 4], variants of the matching problem on graphs and hypergraphs [8], budgeted learning and multi-armed bandit problems (see e.g. [13, 14]), etc. This setting can be abstractly defined as follows: At the outset, the algorithm is presented with a set of options and probability distributions for the costs and rewards associated with each option. At each step, the algorithm must choose one of the options, after which the costs and rewards of the chosen option are drawn from their respective distributions. The overall goal of the algorithm is to maximize the rewards while obeying the packing constraints on the costs. Clearly, the ONLINE STOCHASTIC MATCHING problem fits this general framework. However, previous work in this class of problems has focused on offline problems, i.e. the distribution of the input is given at the outset to the algorithm. Note that in the ONLINE STOCHASTIC MATCHING problem, not only are the rewards stochastic, but the distribution of the input is also not known in advance, rather it arrives online. Therefore, the ONLINE STOCHASTIC MATCHING problem belongs to a broader class of *online stochastic packing* problems and we hope our work will lead to further investigation in this domain.

We are now ready to formally define the ONLINE STOCHASTIC MATCHING problem.

The ONLINE STOCHASTIC MATCHING **Problem.** In the online matching problem with stochastic rewards, every edge (u, v) also has an associated probability of success p_{uv}. When a new vertex $v \in V$ arrives online, the algorithm either chooses not to assign it at all, or assigns it to one of its neighbors (these are the available options). If v is assigned to $u \in U$ using the edge (u, v), an independent random coin is tossed and the edge (u, v) becomes a successful assignment with probability p_{uv}. In this case, we say that u was successful and remove it from the set of *available* vertices. On the other hand, if the assignment (u, v) is not successful, u remains available and a neighbor of u that arrives later in the online order can be assigned to u (but v cannot

be assigned in the future). The overall objective is to maximize the *expected* number of successful vertices $u \in U$, or equivalently, the expected number of successful assignments.

Optimal solution and Competitive Ratio. To quantify algorithmic performance for the ONLINE STOCHASTIC MATCHING problem, we need to define an optimum (henceforth called OPT) that we can compare against. Our goal is to study the effect of both "online" input and "stochastic" rewards on the bipartite matching problem. This motivates us to introduce an offline, non-stochastic version of the ONLINE STOCHASTIC MATCHING problem where the graph is known beforehand, and the reward on edge (u,v) is (deterministically) p_{uv} (i.e. equal to its expectation); this corresponds to the well-studied BUDGETED ALLOCATION problem [24, 6, 1].

The BUDGETED ALLOCATION problem. Let $G = (U \cup V, E)$ be a bipartite graph where edge (u,v) has weight p_{uv}. For every vertex $v \in V$, we assign it to one its neighbors $u \in U$ using edge (u,v). The *load* L_u on a vertex $u \in U$ is defined as the sum of weights of assigned edges incident on u. The objective is to maximize $\sum_{u \in U} \min(L_u, 1)$.

For technical reasons, we allow fractional solutions, i.e. vertex $v \in V$ can be assigned to its neighbors $u \in U$ with fractions x_{uv} subject to $\sum_{u \in U} x_{uv} \leq 1$; then $L_u = \sum_{u \in U} x_{uv} p_{uv}$. For any instance of the ONLINE STOCHASTIC MATCHING problem, we define OPT to be the optimum fractional solution for the corresponding BUDGETED ALLOCATION problem. The next lemma claims that the value of OPT is greater than or equal to both its value for the previous definition and also the expected number of successes obtained by *any* ONLINE STOCHASTIC MATCHING algorithm.

Lemma 1. *For any instance of the* ONLINE STOCHASTIC MATCHING *problem, the expected number of successes produced by any (offline or online) algorithm is less than or equal to the fractional optimum for the corresponding instance of the* BUDGETED ALLOCATION *problem.*

Proof. Suppose the ONLINE STOCHASTIC MATCHING algorithm assigns vertex $v \in V$ to a neighbor u with probability q_{uv}. Then, the probability of success of u is at most $\min(q_{uv} p_{uv}, 1)$. A fractional solution to the corresponding instance of the BUDGETED ALLOCATION problem where q_{uv} fraction of v is assigned to u has an objective of $\min(q_{uv} p_{uv}, 1)$ at u, thereby proving the lemma. □

The *competitive ratio* of an ONLINE STOCHASTIC MATCHING algorithm is now defined as the worst-case ratio (over all input instances) between the expected number of successful vertices $u \in U$ in the algorithmic solution and OPT . Note that the choice of OPT and competitive ratio is consistent with the standard definition in the non-stochastic case, which is a special case of our problem (when $p_{uv} = 1,\ \forall (u,v) \in E$).

Choice of Optimum Solution. Our definition of OPT captures both the online and the stochastic aspect of the problem, since it knows the entire graph, and is not subject to stochastic rewards. The online aspect maintains consistency with the classic notion of competitive ratio. We briefly discuss other options for the optimum benchmark. Since the edge probabilities define an input distribution over graphs, one option is to define OPT as the expectation over the size of maximum matchings in these graphs. However, consider an instance where $V = \{v\}$, $|U| = n$, and $p_{uv} = 1/n$ for all $u \in U$. Any algorithm assigns v to some vertex $u \in U$ and achieves expected success of $1/n$ whereas OPT = $1 - (1 - 1/n)^n \geq 1 - 1/e$. The primary shortcoming of this definition is that the optimal solution is informed about the success/failure of every edge while the algorithm, even after it terminates, is provided such information only for edges it used for assignment. This deficiency can be overcome by modifying OPT to represent the expected number of successes of the best assignment. (Recall that given an assignment, the probability of success of a vertex $u \in U$ is the probability that at least one assignment made to u is successful.) However, we have now made

OPT too weak — unlike an adaptive online algorithm, it does not know the outcome of previous assignments before deciding on a new assignment (see more discussion on adaptive and non-adaptive online algorithms later in this section). For example, consider a complete bipartite graph where $U = \{u_1, u_2\}$, $|V| = 2n$, and $p_{u_i v} = 1/n$ for each $v \in V, i = 1, 2$. An algorithm that assigns the current vertex $v \in V$ to any of its available neighbors achieves $2(1-2/e^2)$ successes in expectation whereas OPT = $2(1-1/e)$ which is less. One way to introduce adaptivity is by modifying this to now represent the maximum expected number of successes where the optimal algorithm makes assignments for vertices in V in the online order, and is provided the outcome of its previous assignments in addition to the entire graph. This is another reasonable definition (note that the expected number of successes for any online algorithm is now at most OPT) but it captures only the online aspect of the problem. If one further removes the offline knowledge, then it degenerates to the best instance-wise online policy, which is inconsistent with the standard notion of competitive ratio, when $p_{uv} = 1$.

1.1 Our Contributions

In this paper we consider the special case of equal probabilities, i.e. $p_{uv} = p$ for all $(u, v) \in E$. Without loss of generality, we restrict to algorithms that always assign a vertex $v \in V$ if there is at least one available neighbor; we call such an algorithm *opportunistic*. We show that every opportunistic algorithm has a competitive ratio of at least 1/2. It can be shown that the GREEDY algorithm, which assigns the arriving vertex to an arbitrary available neighbor does no better than 1/2. This is true even for a simple random algorithm which chooses uniformly at random between available neighbors. Our first result is to provide a deterministic algorithm which improves over this baseline of 1/2.

Theorem 1. *There is a deterministic algorithm (which we call the* STOCHASTIC BALANCE *algorithm) for the* ONLINE STOCHASTIC MATCHING *problem with equal probabilities (i.e. $p_{uv} = p$ for all $(u, v) \in E$) that achieves a competitive ratio of $\eta(p)$, where $\eta(p) = \frac{1}{2}\left(1 + (1-p)^{2/p}\right)$. If $p \to 0$, $\eta(p) = 0.5(1 + e^{-2}) \simeq 0.567$.*

STOCHASTIC BALANCE assigns the next vertex to the available neighbor with the least number of failed assignments in the past. Next, we show that our analysis is nearly tight.

Theorem 2. *There is a family of instances of the* ONLINE STOCHASTIC MATCHING *problem for which* STOCHASTIC BALANCE *achieves a factor no better than* 0.588.

We also show the following result for the (randomized) RANKING algorithm (which was originally proposed for online matching [20]).

Theorem 3. *There is a randomized algorithm called* RANKING *, for the* ONLINE STOCHASTIC MATCHING *problem with equal probabilities (i.e. $p_{uv} = p$ for all $(u, v) \in E$) that achieves a competitive ratio of $\kappa(p) = (1-1/e) - (1-2/e)(1-p)^{1/p}$. For $p = 1$, $\kappa(p) = 1 - 1/e \simeq 0.632$, and for $p \to 0$, $\kappa(p) = 1 - 2/e + 2/e^2 \simeq 0.534$.*

One can see that the ratio for STOCHASTIC BALANCE deteriorates as p increases, while that for RANKING improves. The former is better for $p \leq 0.26$. The next natural question is whether the stochastic aspect of the ONLINE STOCHASTIC MATCHING problem hurts the competitive ratio at all: in particular, can we design algorithms for the ONLINE STOCHASTIC MATCHING problem that have a competitive ratio of $1 - 1/e$ (recall that this is the optimal ratio for the non-stochastic case [20], and our OPT reduces to theOPT used there). We refute this possibility in the following theorem.

Theorem 4. *No (randomized) algorithm for the* ONLINE STOCHASTIC MATCHING *problem has a competitive ratio of more than* $(1-11/(18e)-5/(6e^2)-5/(6e^3)) < 0.621 < 1-1/e$*, even in the case of equal and vanishing probabilities.*

Adaptive and Non-adaptive Algorithms. As has been observed in the literature, stochastic optimization problems admit two distinct classes of algorithms: *adaptive* and *non-adaptive* algorithms. While an adaptive algorithm for the ONLINE STOCHASTIC MATCHING problem is allowed access to the outcome (i.e. success/failure) of previous assignments, a non-adaptive algorithm is not provided this information. Since a non-adaptive algorithm does not know the set of available vertices in U, it may assign vertices in V to neighbors in U that are currently unavailable, i.e., already successful. But such an assigment does not count toward any more successes. It has been previously observed that for many stochastic packing problems, the expected approximation ratio of the best adaptive algorithm is provably better than that of the best non-adaptive algorithm. It turns out that this is indeed the case for the ONLINE STOCHASTIC MATCHING problem. We show an upper bound on the performance of non-adaptive algorithms for the ONLINE STOCHASTIC MATCHING problem.

Theorem 5. *No (deterministic or randomized) non-adaptive algorithm for the* ONLINE STOCHASTIC MATCHING *problem can achieve a competitive ratio greater than 1/2.*

It follows that the **Adaptivity Gap** (see [8]) for this problem is at least $\max\{\eta(p),\kappa(p)\}/0.5$, which is 1.13 for $p \to 0$, and the gap is at most $0.621/0.5 = 1.242$.

1.2 Our Techniques

We will now outline the gist of our techniques towards understanding the structure of the problem, and in particular for Theorem 1.

An alternate view. Our main technical insight comprises of an alternative view of the underlying probability space that is more amenable to anaytical tools. First, we make a conceptual transition by viewing the probability space of the problem from the perspective of the vertices in U. Fix any algorithm and focus attention on a vertex $u \in U$. As the algorithm proceeds, u gets allocated vertices $v_1^u, v_2^u, \ldots$ in V, until the first time that the edge (u, v_τ^u) becomes successful. We can visualize this as u having a sequence of coins, each with probability of heads equal to p, and u tossing all its coins in advance. This determines a threshold τ for u, which is the index of the first heads in the sequence. Then u succeeds if it gets allocated τ vertices, and fails if it is allocated less than τ vertices. Therefore, each vertex $u \in U$ chooses a threshold θ_u independently from the probability distribution $\mathbb{P}[\theta_u = pt] = p(1-p)^{t-1}$ over positive integers t at the outset. When the algorithm assigns $v \in V$ to a neighbor u, the assignment is successful iff the load on u reaches θ_u after this assignment. Clearly, this stochastic process is exactly identical to that of the ONLINE STOCHASTIC MATCHING problem.

Now, our problem resembles the ADWORDS problem [23] which is exactly identical to the BUDGETED ALLOCATION problem except that the vertices in V arrive online and each vertex in U might have a distinct *budget*. A key difference however is in the objective of the two problems: while the ONLINE STOCHASTIC MATCHING problem aims to maximize the expected number of vertices $u \in U$ that have a load equal to their threshold Θ_u (i.e. are successful), the ADWORDS problem aims to maximize the total load on the vertices in U. Our second key observation equates these two apparently distinct objectives in the next lemma. For any vertex $u \in U$, let L_u be the expected load on u and g_u be the probability that u is successful.

Lemma 2. *For any adaptive algorithm for the* ONLINE STOCHASTIC MATCHING *problem, the expected load on a vertex $u \in U$ is equal to its probability of success, i.e. $g_u = L_u$.*

Proof. Let X_{uv} and Y_{uv} be random variables defined as follows: $X_{uv} = 1$ if vertex $v \in V$ is assigned to vertex $u \in U$, else $X_{uv} = 0$; $Y_{uv} = 1$ if $X_{uv} = 1$ and the edge (u,v) produces a success, else $Y_{uv} = 0$. Clearly, $\mathbb{E}[Y_{uv}] = p_{uv}\mathbb{E}[X_{uv}]$. Note that the different X's and Y's are correlated because the algorithm is adaptive. But also, due to adaptvity, u will not be given two successes, and so $g_u = \mathbb{E}[\sum_v Y_{uv}]$. Now the lemma follows by linearity of expectation over all $v \in V$. □

Remark. *Observe that for non-adaptive algorithms for the* ONLINE STOCHASTIC MATCHING *problem, $g(u) = 1 - e^{-L(u)} = L(u) - L(u)^2/2 + \ldots$; in fact, it is precisely because of the trailing terms in this expression that non-adaptive algorithms have provably worse performance than adaptive algorithms.*

In light of the above lemma, the objective of the ONLINE STOCHASTIC MATCHING problem is exactly identical to that of the ADWORDS problem; therefore, it would be tempting to declare that an instance of the ONLINE STOCHASTIC MATCHING problem is identical to a probability distribution over instances of the ADWORDS problem. However, this is not accurate because of two reasons. First, while the budget of every vertex in U is known in advance in the ADWORDS problem (and is used by ADWORDS algorithms), only the probability distribution of the thresholds is known in the ONLINE STOCHASTIC MATCHING problem; the actual threshold Θ_u is revealed only if the load on u reaches Θ_u. Second, whereas the optimum in the ADWORDS problem is defined as the maximum allocable load subject to budget constraints, OPT for the ONLINE STOCHASTIC MATCHING problem is defined as the maximum allocable load in the expected instance rather than the expected maximum allocable load. This subtle difference is, in fact, quite significant — the ratio of expected optimum of the ADWORDS instances to OPT could be as small as $1 - 1/e$.

In spite of these differences, we show that the insight gained from the alternative view of the ONLINE STOCHASTIC MATCHING problem via the ADWORDS problem is quite useful. In particular, we propose the following algorithm for the ONLINE STOCHASTIC MATCHING problem:

STOCHASTIC BALANCE : *Assign the arriving $v \in V$ to its available neighbor with the least load, i.e., the least number of failed attempts (breaking ties arbitrarily).*

While the above algorithm is inspired by the ADWORDS algorithm (more precisely, by the BALANCE algorithm for b-Matching [16], since this is the case when probabilities are equal), its analysis, and similarly the analysis for RANKING , is much more complicated since the input is stochastic and we are comparing ourselves against a stiffer OPT . We outline our proof technique for Theorem 1 below.

Proof Techniques. The overall proof technique is to encode the adversary strategy as a primal-dual LP pair (called a factor-revealing LP [15]) and use weak duality to derive bounds on the competitive ratio of the algorithm. We summarize the key properties of STOCHASTIC BALANCE below; these appear as constraints in the LP.

Let $g(t)$ (resp. $f(t)$) be the expected number of vertices which succeed (resp. fail) with a load of t (i.e. after t/p assignments). We first note that for any vertex in U, there is a relationship between its contributions to the $g(t)$ and $f(t)$. This relationship can be seen by using the alternate view of the probability space: Keeping all other θ values fixed, vary the value of θ_u and use the monotonicity of the algorithm to claim that if $\theta_u \leq L_u$, then u succeeds with a load of θ_u; otherwise, it fails and has a load of L_u. Noting that the expected number of successful vertices plus failed vertices is simply $n = |U|$, we get our first constraint: $\int_0^\infty e^t f(t)dt = n$.

This constraint by itself has a bad solution, by setting $f(0) = n$, saying that all vertices fail with load 0, clearly an impossibility. For the next constraint we start with the identity that the total load obtained by

the algorithm is the load from vertices in V which were allocated in OPT to a $u \in U$ which failed during the algorithm, plus the load from the vertices in V allocated in OPT to a u which succeeded. Now observing that if a vertex u fails then all its vertices in V are allocated, and using the relationship between expected load and number of successes (Lemma 2), we see that the number of successes is the number of failures plus the load from the vertices in V allocated in OPT to a u which succeeded. Thus we need to lower bound the latter quantity. For this, we again appeal to the alternate view of the probability space: Keeping rest of the θ vales fixed, consider the lowest threshold θ_u^* for a vertex u so that it fails. If the threshold dereases by some amount δ from this value, then u succeeds and at most δ amount of vertices in V from the OPT for u could be left unallocated. This gives a second constraint on the performance of the algorithm.

We note that both these constraints are new, in the sense that they exist only because the budgets are stocahstic. Indeed, they do not hold in the non-stochastic case. For example, the ADWORDS algorithm may deterministically end up not assigning any of the vertices in V assigned by OPT to a vertex u which succeeds, i.e. finishes its budget.

Finally, we use the form of STOCHASTIC BALANCE itself: this ensures that for any vertex $u \in U$ that has a load of L_u at the end of the algorithm, one of the following must hold: either u was successful, or each neighbor $v \in V$ of u must have been assigned to vertex in U that had a load of at most L_u when v arrived. We use this property of the STOCHASTIC BALANCE algorithm to obtain our third constraint. The three constraints are sufficient to prove a good bound on the algorithm.

Note. For simplicity, we will assume that $p = 1/s$ for some integer s. (Violation of this assumption leads to rounding errors depending on the value of p.) Under this assumption, the constraints of the BUDGETED ALLOCATION problem represent a matching polytope; therefore, wlog, we will assume that there is an optimal integer solution to any instance of the BUDGETED ALLOCATION problem.

Open problem for unequal probabilities. We may try to extend our techniques for the general case of arbitrary p_{uv}. The case of large p_{uv} is well known to be hard even if we are trying to maximize the expected load, and is open. For the unequal but vanishingly small probabilities case, one may guess at an algorithm inspired by the scaled bids algorithms in [23] or [5]. In particular, we believe that the following algorithm should perform well: assign the arriving $v \in V$ to the neighbor u which maximizes $p_{uv}e^{-L_u}$. We can obtain a global "generalized balance" equation for this, but the difficulty is in obtaining a bound on the load from the OPT allocation of successful vertices. Without this extra constraint, just as in the case for Theorem 1, we can not obtain a bound better than 1/2. We leave this as an interesting open question.

1.3 Related Work

There is a growing literature on non-stochastic online matching (e.g., [20, 16, 23, 5]), pointers to which are distributed throughout the paper. We point out two closely related problems: First, the Online Bipartite Matching problem, which is a special case of our problem, where all the probabilities are 0 or 1, and for which RANKING has $1 - 1/e$ ratio [20]. Second, the ADWORDS problem which is the Online version of BUDGETED ALLOCATION , which has deterministic algorithms [16, 23, 5] achieving ratios of $1 - 1/e$.

A set of previous results that also goes by *online stochastic matching* makes distributional assumptions on the input graph in online matching (see e.g. [10, 2, 22, 21, 17]). In these problems it is the structure of the input graph that is stochastic, whereas the rewards are deterministic (i.e. all edge probabilities are 1). In fact, these problems are *easier* that online matching, and often yield competitive ratios greater than $1 - 1/e$. Our problem, on the other hand, is a *strict generalization* of online matching, and our competitive ratios are therefore less than $1 - 1/e$.

Another related line of work is that of Chen *et al* [7] and Bansal *et al* [3]. They consider an offline matching problem on general random graphs, with query budgets. We observe that the offline version of the ONLINE STOCHASTIC MATCHING problem is indeed a special case of this problem. The only online problem considered in this line of work is by Bansal *et al* [3], who study a hybrid of online stochastic arrivals (which is weaker than the classical online model) and stochastic rewards (similar to our problem), but with multiple trials, and achieve a competitive ratio of of about 0.13.

Finally, as noted earlier, the (offline) stochastic packing framework was introduced via the stochastic knapsack problem by Dean *et al* [9]. Subsequently, various optimization problems have been considered in this framework (e.g. [8, 4, 14, 12]), some of which have been mentioned earlier. A related line of work is that of Multi-Armed Bandits and Budgeted Learning (see e.g. [13, 14]). In this problem, there is a set of arms, each with a known Markov chain of states. Pulling an arm yields a random payoff as well as a probabilistic transition in the chain, and the goal is to maximize the expected payoff.

Roadmap. In the next section, we analyze some general properties of adaptive algorithms for ONLINE STOCHASTIC MATCHING . In Section 3, we analyze the STOCHASTIC BALANCE algorithm (Theorem 1). In Section 3.1 we show an upper bound for STOCHASTIC BALANCE (Theorem 2). In Section 4 we analyze the RANKING algorithm (Theorem 3). We provide an unconditional upper bound in Section 5 (Theorem 4). We finally end with an upper bound on non-adaptive algorithms in Section 6 (Theorem 5).

2 Properties of Adaptive Algorithms

In this section, we will discuss some general properties of adaptive algorithms that will be used later in analyzing our algorithm.

Definition 1. *The* load *on a vertex $u \in U$, denoted by L_u, is defined as the sum of probabilities associated with the assigned edges incident on u (this includes all the assignments which did not succeed and, in case u was successful, the one assignment which succeeded). Let $f_u(x)$ (resp., $g_u(x)$) denote the probability that vertex $u \in U$ failed (resp., is successful) at the end of the algorithm and has a load of x. Further, let $f(x) = \sum_{u \in U} f_u(x)$ (resp., $g(x) = \sum_{u \in U} g_u(x)$) be the expected number of failed (resp., successful) vertices in U with a load of x.*

Recall that Lemma 2 asserts that the expected load on a vertex $u \in U$ equals $\sum_x g_u(x)$. Further, recall the alternative view of the ONLINE STOCHASTIC MATCHING problem as an ADWORDS problem where the budget Θ_u of every vertex $u \in U$ is drawn i.i.d. from the distribution

$$\mathbb{P}[\Theta_u = pt] = p(1-p)^{t-1}.$$

Definition 2. *Let Θ be the vector of Θ_u for all $u \in U$, and let Θ_{-u} denote the entire vector Θ except Θ_u. For any fixed $(n-1)$-dimensional vector θ, let*

$$p_u(\theta) := \mathbb{P}[\Theta_{-u} = \theta]$$

Further, let $L_u(\theta)$ be the load on vertex u when $\Theta_{-u} = \theta$ and $\Theta_u = \infty$; correspondingly, let

$$q_u(x) := \sum_{\theta : L_u(\theta) = x} p_u(\theta)$$

Lemma 3. *For any adaptive algorithm and for every vertex $u \in U$,*

$$g_u(x) = p(1-p)^{x/p-1} \sum_{y \geq x} f_u(y)(1-p)^{-y/p}.$$

Proof. By definition, vertex u fails with a load of x if and only if $\Theta_u > L_u = x$. Increasing Θ_u to ∞ does not change any assignment, and therefore, the load on u remains x. Thus,

$$f_u(x) = Pr[(\Theta_u > x) \wedge (L_u(\Theta_{-u}) = x)] = Pr[\Theta_u > x] \cdot Pr[L_u(\Theta_{-u}) = x] = (1-p)^{x/p} q_u(x). \quad (1)$$

Similarly, vertex u succeeds with a load of x if and only if $L_u = \Theta_u = x$. Increasing Θ_u to ∞ cannot decrease the load on u. Therefore,

$$g_u(x) = Pr[(\Theta_u = x) \wedge (L_u(\Theta_{-u}) \geq x)] = Pr[\Theta_u = x] \cdot Pr[L_u(\Theta_{-u}) \geq x] = p(1-p)^{x/p-1} \sum_{y \geq x} q_u(y). \quad (2)$$

The lemma follows from Eqns. 1 and 2. □

The next lemma follows from the above lemma using the fact that $\sum_{u \in U} \sum_x (f_u(x) + g_u(x)) = n$.

Lemma 4. *For any adaptive algorithm, $\sum_x f(x)(1-p)^{-x/p} = n$.*

Opportunistic Algorithms. Recall that an adaptive algorithm is said to be *opportunistic* if it always assigns a vertex $v \in V$ provided it has a currently unsuccessful neighbor in U. Let OPT be a fixed offline optimal solution that achieves an objective value of L_u^* on vertex $u \in U$. Let $opt(v)$ denote the vertex in $u \in U$ that v is assigned to by OPT (if v is not assigned by OPT , then $opt(v)$ is undefined). Similarly, let $opt(u) = \{v \in V : opt(v) = u\}$. Finally, let $\mathbf{E}$ denote the total expected load due to vertices $v \in V$ for which $opt(v)$ is successful. We also use $g_u = \sum_x g_u(x)$ and $f_u = \sum_x f_u(x)$.

Lemma 5. *For any opportunistic algorithm for the* ONLINE STOCHASTIC MATCHING *problem, the expected number of successes $\sum_{u \in U} g_u = \sum_{u \in U} L_u^* f_u + \mathbf{E}$.*

Proof. We define random variables X_u, Y_u, and Z_u as follows:

- $X_u = 1$ iff vertex u is successful; $X_u = 0$ otherwise.
- Y_u is the total load due to vertices in $opt(u)$.
- $Z_u = Y_u$ iff $X_u = 1$; $Z_u = 0$ otherwise.

In any execution of an opportunistic algorithm, either u is successful or all the vertices in $opt(u)$ are assigned, i.e. $Y_u = L_u^*$ whenever $X_u = 0$. On the other hand, $Y_u = Z_u$ when $X_u = 1$. Therefore, the total load due to vertices in $opt(v)$ is

$$\mathbb{E}[Y_u] = L_u^* f_u + \mathbb{E}[Z_u].$$

The lemma now follows from Lemma 2. □

The following corollary is an immediate consequence of the above lemma.

Corollary 1. *Any opportunistic algorithm for the* ONLINE STOCHASTIC MATCHING *problem has a competitive ratio of at least 1/2.*

Proof. From the above lemma,

$$\sum_{u\in U} g_u \geq \sum_{u\in U} L^*_u f_u = \sum_{u\in U} L^*_u(1-g_u) = \sum_{u\in U} L^*_u - \sum_{u\in U} L^*_u g_u.$$

Therefore,

$$\sum_{u\in U} g_u \geq \sum_{u\in U} \frac{L^*_u}{\max_u(1+L^*_u)} \geq \frac{L^*_u}{2},$$

since $L^*_u \leq 1$ for every $u \in U$. □

3 The STOCHASTIC BALANCE Algorithm

Now, we describe the STOCHASTIC BALANCE algorithm and prove Theorem 1. The algorithm is simple:

STOCHASTIC BALANCE : Assign the new vertex $v \in V$ to its currently unsuccessful neighbor in U that has the least load.

We now prove a generic property that is satisfied by the STOCHASTIC BALANCE algorithm.

Lemma 6. *Consider the* STOCHASTIC BALANCE *algorithm. If the value of Θ_u for some vertex $u \in U$ is reduced by kp for any integer k, then*

- *Every vertex $v \in V$ that was previously unassigned remains unassigned.*
- *The decrease in overall load on all vertices $u \in U$ is at most kp.*

Proof. We will assume $k = 1$; this is wlog by repeated invocation. We will show that the load on any vertex $u' \in U$ at any stage of the algorithm is at least as much as the load on u' at the same stage previously, except if $u' = u$ and u has already succeeded. This follows by induction on the vertices in V. Clearly, the property holds at the outset. At any intermediate stage, consider an arriving vertex $v \in V$. If v was unassigned earlier, then the property holds trivially. Therefore, suppose v was assigned to a neighbor u' with load $L_{u'}$ earlier where either $u' \neq u$ or $u' = u$ but u has not succeeded yet. Then, by the inductive hypothesis, either the load on u' is at least $L_{u'} + p$ and the property holds trivially, or the load on u' is exactly $L_{u'}$ and the load on every other available neighbor of v is at least $L_{u'}$ leading to v being assigned to u'. If $u' = u$ and u has already succeeded, then the property holds by definition.

The above property implies that the set of successful vertices at any stage of the algorithm contains all vertices that has succeeded by the same stage earlier. Therefore, a previously unassigned vertex continues to be unassigned now since all its neighbors have already succeeded. Further, the decrease in overall load can only be due to a decrease of p in the load on u. □

The next lemma uses Lemmas 5 and 6 to derive a bound on the function $f(x)$.

Lemma 7. *For the* STOCHASTIC BALANCE *algorithm,*

$$\sum_{u\in U}\left((1-p)^{-L^*_u/p}\sum_{x\geq L^*_u} f_u(x) + \sum_{x<L^*_u}(1+L^*_u-x)(1-p)^{-x/p} f_u(x)\right) \leq n.$$

Proof. In this proof, we will use the notation that we used in the proof of Lemma 3. Recall that we fixed the value of Θ_{-u} to a vector θ, and denoted the load on vertex u is $\Theta_u = \infty$ by $L_u(\theta)$. As observed earlier, if $\Theta_u > L_u(\theta)$, then all vertices in $opt(u)$ are assigned by an opportunistic algorithm. Now, consider the case $\Theta_u \leq L_u(\theta)$. There are two possibilities. If $L_u(\theta) \geq 1$ and $0 \leq L_u(\theta) - \Theta_u \leq 1$, then by Lemma 6, the volume of assigned vertices in $opt(u)$ (i.e. their contribution to $\mathbf{E}$) is at least $L_u^* - (L_u(\theta) - \Theta_u)$. On the other hand, if $L_u(\theta) < 1$, then the above statement holds for the range $0 \leq \Theta_u \leq L_u(\theta)$. Since

$$q_u(x) = \sum_{\theta : L_u(\theta) = x} p_u(\theta),$$

we have

$$\begin{aligned}
\mathbf{E} &\geq \sum_{u \in U} \left(\sum_{x \geq L_u^*} q_u(x) \sum_{y=(x-L_u^*)/p+1}^{x/p} p(1-p)^{y-1}(L_u^* - (x-py)) + \sum_{x < L_u^*} q_u(x) \sum_{y=1}^{x/p} p(1-p)^{y-1}(L_u^* - (x-py)) \right) \\
&= \sum_{u \in U} \left(\sum_{x \geq L_u^*} q_u(x)(1-p)^{\frac{x-L_u^*}{p}} \left(1 - (1+L_u^*)(1-p)^{1/p}\right) + \sum_{x < L_u^*} q_u(x) \left(1 + L_u^* - x - (1+L_u^*)(1-p)^{x/p}\right) \right) \\
&= \sum_{u \in U} \left(\sum_{x \geq L_u^*} q_u(x)(1-p)^{(x-L_u^*)/p} + \sum_{x < L_u^*} q_u(x)(1+L_u^*-x) - (1+L_u^*) \sum_x q_u(x)(1-p)^{x/p} \right) \\
&= \sum_{u \in U} \left((1-p)^{-L_u^*/p} \sum_{x \geq L_u^*} f_u(x) + \sum_{x < L_u^*} (1+L_u^*-x)(1-p)^{-x/p} f_u(x) \right) - \sum_x f(x) - \sum_{u \in U} L_u^* f_u
\end{aligned}$$

Using Lemma 5 and re-arranging terms, we have

$$n = \sum_x (g(x) + f(x)) \geq \sum_{u \in U} \left((1-p)^{-L_u^*/p} \sum_{x \geq L_u^*} f_u(x) + \sum_{x < L_u^*} (1+L_u^*-x)(1-p)^{-x/p} f_u(x) \right).$$

□

Next, we prove another key property of the STOCHASTIC BALANCE algorithm.

Lemma 8. *For the* STOCHASTIC BALANCE *algorithm,*

$$\sum_{y \leq x} \sum_{u \in U} (1+L_u^*) f_u(y) + (1-p)^{x/p} \sum_{y > x} (1-p)^{-y/p} \left(\sum_{u \in U} f_u(y) \right) \leq n,$$

Proof. Consider a vertex $u \in U$. If u failed and had a load of x at the termination of the algorithm, then every vertex $v \in opt(u)$ must have been assigned by the STOCHASTIC BALANCE algorithm to a neighbor in U that had a load of at most x when v arrived online. Therefore,

$$\sum_{y \leq x} \sum_{u \in U} f_u(y) L_u^* \leq \sum_{y \leq x} y(f(y) + g(y)) + x \sum_{y > x} (f(y) + g(y)).$$

From Lemma 3, we can substitute for $g(y)$ in terms of the $f()$:

$$\begin{aligned}
\sum_{y\le x}\sum_{u\in U} f_u(y)L^*_u &\le \sum_{y\le x} y(f(y)+p(1-p)^{y/p-1}\sum_{z\ge y} f(z)(1-p)^{-z/p}) + x\sum_{y>x}(f(y)+p(1-p)^{y/p-1}\sum_{z\ge y} f(z)(1-p)^{-z/p}) \\
&= \sum_{y\le x} f(y)\left(y+(1-p)^{-y/p}\sum_{z\le y} zp(1-p)^{z/p-1}\right) + \\
&\quad + \sum_{y>x} f(y)\left((1-p)^{-y/p}\sum_{z\le x} zp(1-p)^{z/p-1} + x\left(1+(1-p)^{-y/p}\sum_{x<z\le y} p(1-p)^{z/p-1}\right)\right) \\
&= \sum_{y\le x} f(y)\left((1-p)^{-y/p}-1\right) + \sum_{y>x} f(y)(1-p)^{-y/p}\left(1-(1-p)^{x/p}\right).
\end{aligned}$$

Re-arranging the terms, we have

$$\sum_{y\le x}\sum_{u\in U}(1+L^*_u)f_u(y) + (1-p)^{x/p}\sum_{y>x}(1-p)^{-y/p}\left(\sum_{u\in U} f_u(y)\right) \le \sum_y f(y)(1-p)^{-y/p} = n,$$

where the last equality follows from Lemma 4. □

We use the above properties of the STOCHASTIC BALANCE algorithm to derive its competitive ratio using what is called a *factor-revealing LP*. Recall that $\sum_x (f(x)+g(x)) = n$, and that the expected number of successful assignments is $\sum_x g(x)$. Therefore, we visualize the adversary strategy as that of maximizing $\sum_x f(x)$ subject to the constraints imposed by Lemmas 4, 7 and 8. This pair of primal and dual programs are described in Fig. 1. For readability, in Fig. 2 we show the same programs in the continuous domain for the case when $p \to 0$ and all $L*_u$ are 1.

Equivalently, the adversary strategy can be thought of as minimizing the dual LP. The best case scenario for the dual LP is if $L^*_u = \ell$ for some ℓ for each vertex $u \in U$. Then, the optimal dual solution is

$$\begin{aligned}
\alpha &= -L(1-p)^{(1+\ell)/p} \\
\beta(x) &= \begin{cases} (1-p)^{(\ell-x)/p} - (1-p)^{(1+1/\ell)(\ell-x)/p} & \text{if } x \le \ell \\ 0 & \text{if } x > \ell. \end{cases} \\
\gamma &= (1-p)^{\ell/p}.
\end{aligned}$$

This gives a dual objective of $\frac{n\left(1-\ell^2(1-p)^{(1+\ell)/p}\right)}{1+\ell}$. This implies that the expected number of successful vertices in U in the solution produced by the algorithm is at least

$$n - \frac{n\left(1-\ell(1-p)^{(1+\ell)/p}\right)}{1+\ell} \quad = \quad n\ell\left(\frac{1+\ell(1-p)^{(1+\ell)/p}}{1+\ell}\right),$$

which yields a competitive ratio of

$$\frac{1+\ell(1-p)^{(1+\ell)/p}}{1+\ell} \quad \ge \quad \eta(p) \quad \text{for all } \ell \in [0,1],$$

where $\eta(p) = \frac{1+(1-p)^{2/p}}{2}$. This completes the proof of Theorem 1.

Maximize $\sum_x f(x)$ subject to

$$\begin{aligned}
\sum_y \sum_{u\in U} f_u(y)(1-p)^{-y/p} &= n \\
\sum_{y\le x}\sum_{u\in U}(1+L_u^*)f_u(y) + (1-p)^{x/p}\sum_{y>x}(1-p)^{-y/p}\left(\sum_{u\in U} f_u(y)\right) &\le n \quad \forall\, x>0 \\
\sum_{u\in U}\left((1-p)^{-L_u^*/p}\sum_{x\ge L_u^*} f_u(x) + \sum_{x<L_u^*}(1+L_u^*-x)(1-p)^{-x/p} f_u(x)\right) &\le n \\
f_u(y) &\ge 0 \quad \forall\, y>0,\ \forall\, u\in U
\end{aligned}$$

(a) The primal LP

Minimize $n(\alpha+\sum_x \beta(x)+\gamma)$ subject to

$$\begin{aligned}
\alpha(1-p)^{-y/p} + (1+L_u^*)\sum_{x\ge y}\beta(x) + (1-p)^{-y/p}\sum_{x<y}(1-p)^{x/p}\beta(x) + (1+L_u^*-y)(1-p)^{-y/p}\gamma &\ge 1 \\
&\quad \forall\, y<L_u^*,\ \forall\, u\in U \\
\alpha(1-p)^{-y/p} + (1+L_u^*)\sum_{x\ge y}\beta(x) + (1-p)^{-y/p}\sum_{x<y}(1-p)^{x/p}\beta(x) + (1-p)^{-L_u^*/p}\gamma &\ge 1 \\
&\quad \forall\, y\ge L_u^*,\ \forall\, u\in U \\
\gamma\ge 0, \quad \beta(y)\ge 0 \quad \forall\, y>0 &
\end{aligned}$$

(b) The dual LP

Figure 1: The factor-revealing LP and its dual for the STOCHASTIC BALANCE algorithm.

Maximize $\int_{x=0}^{\infty} f(x)dx$ subject to

$$\begin{aligned}
\int_{y=0}^{\infty} e^y f(y)dy &= 1 \\
\int_{y\le x} 2f(y)dy \;+\; e^{-x}\int_{y>x} e^y f(y)dy &\le 1 \quad \forall\, x>0 \\
\int_{x<1}(2-x)e^x f(x)dx \;+\; e\int_{x\ge 1} f(x)dx &\le 1 \\
f(y) &\ge 0 \quad \forall\, y>0
\end{aligned}$$

(a) The primal LP

Minimize $n(\alpha \;+\; \int_x \beta(x)dx \;+\; \gamma)$ subject to

$$\begin{aligned}
\alpha e^y \;+\; 2\int_{x\ge y}\beta(x)dx \;+\; e^y\int_{x<y} e^{-x}\beta(x)dx \;+\; (2-y)e^y\gamma &\ge 1 \quad \forall\, y<1 \\
e^y\alpha \;+\; 2\int_{x\ge y}\beta(x)dx \;+\; e^y\int_{x<y} e^{-x}\beta(x)dx \;+\; e\gamma &\ge 1 \quad \forall\, y\ge 1 \\
\gamma\ge 0, \quad \beta(y)\ge 0 \quad \forall\, y>0
\end{aligned}$$

(b) The dual LP

Figure 2: The factor-revealing LP and its dual for the STOCHASTIC BALANCE algorithm for the case when $p\to 0$ and all $L_u^*=1$.

3.1 Upper Bound on the Competitive Ratio of the STOCHASTIC BALANCE Algorithm

In this section we prove Theorem 2 by showing that our analysis for STOCHASTIC BALANCE is nearly tight, by describing a family of graphs on which it performs no better than 0.588. The graph is based on the expanded "z-graph" which is often used to construct difficult examples for online matching. For our purposes, we need to keep the two parts of the graph of different sizes.

The graph is $G(U_1\cup U_2, V_1\cup V_2, E)$, where $U_1=\{U_1^1,\ldots,U_1^{\alpha n}\}$, $U_2=\{U_2^1,\ldots,U_2^n\}$, and V_1 (resp. V_2) consists of αn (resp. n) batches of vertices, each with $1/p$ vertices each. The ith batch of vertices in V_1, caled V_1^i all have edges to U_1^i, as well as to all vertices in U_2. The ith batch of vertices in V_2, caled V_2^i all have edges to U_2^i. Thus we have a perfect matching between U and V, plus a bipartite clique between U_2 and V_1. The value of α will be determined later. All edges have a probability of p, which we will take to be vanishingly small.

The optimal allocation in the corresponding BUDGETED ALLOCATION problem is to allocate all vertices in V_i^j to the vertex U_i^j (for all existing (i,j)). This gives $OPT=(\alpha+1)n$.

We analyze the performance of STOCHASTIC BALANCE via an iterative calculation. Let step j be the step in which the vertices in V_1^j arrive (since these are identical, we do not differentiate between their arrival times). We first note that, by symmetry, all available vertices in U_2 will have the same load at every time $j\in[1,\alpha n]$. Let us define the following variables:

$$r_j = \text{the number of available vertices in } U_2 \text{ at step } j$$

$$L_j = \text{the load on the available vertices in } U_2 \text{ at step } j$$

Consider step $j+1$ when vertices in V_1^{j+1} arrive. Each has the following neighbors: U_1^{j+1} with a load of zero, and all the r_j available vertices in U_2, each with a load of L_j. STOCHASTIC BALANCE will first allocate up to L_j amount of load (i.e., L_j/p vertices) to U_1^{j+1}, until the first success, and then spread the rest over the available neighbors in U_2 (never allocating to a vertex which succeeds). The probability that U_1^{j+1} succeeds is $1-e^{-L_j}$. To calculate the expected number of vertices in U_2 which succeed, we first calculate the probability that U_1^{j+1} succeeds at the x/pth vertex in V_1^{j+1} (for $x \leq L_j$). This is $p(1-p)^{x/p-1}$. Then, the expected number of successes in the remaining $(1-x)/p$ vertices is $(1-x)$. So the expected number of successes from U_2 in the $j+1$th step (moving to continuous variables) is:

$$\int_{x=0}^{L_j} e^{-x}(1-x)dx \;+\; e^{-L_j}(1-L_j) \;=\; e^{-L_j}$$

Thus

$$r_{j+1} = r_j - e^{-L_j} \tag{3}$$

By a similar calculation, the increase in the load on the available vertices in U_2 is:

$$\int_{x=0}^{L_j} \frac{e^{-x}(1-x)}{r_j}dx \;+\; \frac{e^{-L_j}(1-L_j)}{r_j} \;=\; \frac{e^{-L_j}}{r_j}$$

Thus

$$L_{j+1} = L_j + \frac{e^{-L_j}}{r_j} \tag{4}$$

Once all the vertices in V_1 have arrived, we have an expected $r_{\alpha n}$ available vertices in U_2. Each of these gets its own exclusive batch of vertices from V_2, giving each one a probability of success of $1-1/e$. Also recall that U_1^j had a success probability of nearly $1-e^{-L_j}$. Thus the total expected number of successful vertices in $U_1 \cup U_2$ is

$$\sum_{j=1}^{\alpha n}(1-e^{-L_j}) \;+\; (n-r_{\alpha n}) + r_{\alpha n}(1-1/e)$$

We do not know how to analytically solve the recurrence system (3) and (4). We do so programatically, and find the value of α which minimizes the final competitve ratio. This gives $\alpha = 0.42$, and a competitive ratio of 0.588, proving Theorem 2.

4 A Randomized Algorithm

In this section, we describe a randomized algorithm for the ONLINE STOCHASTIC MATCHING problem. Our algorithm is simple — we fix a random permutation σ of the vertices in U, and for each arriving vertex $v \in V$, we match it to its highest unmatched neighbor boy in the permutation σ. Observe that this was the original algorithm proposed in [20] for the online matching problem — following their nomenclature, we call it the RANKING algorithm.

As earlier, let $|U| = n$ and let OPT be a fixed optimal offline solution that has an objective value of n. Further, let $opt(v)$ denote the neighbor in U that OPT matches $v \in V$ to; correspondingly, let $opt(u)$ denote the set of neighbors in V that are mapped to a vertex $u \in U$. Our proof will follow the structure of, and use some lemmas from, the proof of the RANKING algorithm presented in [11] for the online matching problem. In addition we will need to use the structure of the probability space defined by the stochastic process. For

this purpose, we will use the view of the probability space from the perspective of the vertices $u \in U$ as earlier. Recall that Θ_u is a random variable that denotes the load on vertex u when it is successful.

The following definitions (which were introduced in [11] but have been modified for our purpose) are crucial.

Definition 3. Permutation Groups. *Let Ω be the set of all permutations of U. For a permutation $\sigma \in \Omega$, $\sigma(s)$ denotes the vertex in U at position s, and $\sigma^{-1}(u)$, the position of vertex u. For a fixed vertex $u \in U$, we partition the set of all permutations Ω into $(n-1)!$ disjoint* groups *of n permutations each, such that in each group, the relative positions of all vertices in $U \setminus \{u\}$ are fixed. Let Ω_u denote one such group. Let $\sigma_t \in \Omega_u$ be the permutation which has vertex u at position t.*

Definition 4. Good and Bad matches. *Consider a run of* RANKING *with a fixed threshold vector θ, and a fixed permutation $\sigma \in \Omega$. A* matched edge (u,v) *is said to be a* bad match *if $opt(v)$ is at a position below u, i.e. $\sigma^{-1}(opt(v)) > \sigma(u)$. Otherwise, we call it a* good match*, i.e. when $\sigma^{-1}(opt(v)) \leq \sigma(u)$. For $s \in [n], b \in [n]$, we define*

- *$bad^{\theta}_{\sigma}(s,b)$ as the load on the vertex in U at position s due to bad matches with vertices in $opt(b)$.*
- *$good^{\theta}_{\sigma}(s,b)$ as the load on the vertex in U at position s due to good matches with vertices in $opt(b)$.*
- *$match^{\theta}_{\sigma}(s) = \sum_b \left(good^{\theta}_{\sigma}(s,b) + bad^{\theta}_{\sigma}(s,b)\right)$ as the total load on the vertex in U at position t.*

We also define the above variables averaged over the randomness in the stochastic matches (i.e. over θ) and the randomization of the algorithm (i.e. σ):

$$\begin{aligned} bad(s) &= \mathbb{E}_{\theta}\left[\mathbb{E}_{\sigma}\left[\sum_b [bad^{\theta}_{\sigma}(s,b)]\right]\right]. \\ good(s) &= \mathbb{E}_{\theta}\left[\mathbb{E}_{\sigma}\left[\sum_b [good^{\theta}_{\sigma}(s,b)]\right]\right]. \\ match(s) &= \mathbb{E}_{\theta}\left[\mathbb{E}_{\sigma}\left[match^{\theta}_{\sigma}(s)\right]\right]. \end{aligned}$$

Observe that for any θ and σ, and for any vertex $u \in U$, we have $match^{\theta}_{\sigma}(s) \leq \theta_u$; further, vertex u is successful iff $match^{\theta}_{\sigma}(s) = \theta_u$. The next lemma is a generalization of Lemma 2.2 in [11].

Lemma 9. *Fix a vertex $u \in U$, a permutation group Ω_u, and a threshold vector θ. Then, for all $t \in [n]$,*

$$\min(1,\theta_u) - match^{\theta}_{\sigma_t}(t) \leq \sum_{\sigma \in \Omega_u} \sum_{s<t} \frac{bad^{\theta}_{\sigma}(s,u)}{n-s}$$

Proof. We consider two cases:

- If vertex u was successful in σ_t, then $match^{\theta}_{\sigma_t}(t) = \theta_u$, and the lemma holds trivially.
- Suppose u was not successful in σ_t. Then, every vertex $v \in opt(u)$ must be matched to some neighbor in V (call it $alg(v)$) in some position $s \leq t$ in σ_t since u at position t was unmatched when v arrived online. Since u has an overall load of $match^{\theta}_{\sigma_t}(t)$, there is a set of vertices $opt'(u) \subseteq opt(u)$ of load at least $1 - match^{\theta}_{\sigma_t}(t)$ that are matched to neighbors in positions *strictly* above t. Let $v \in opt'(u)$ be matched to a vertex $alg(v) \in U$ at position $s < t$. We make the following observations:

– For any $r > s$, consider the run on σ_r and θ (recall that this means moving u to position r). Vertex v continues to be matched to $alg(v)$ which is at position s in each of these runs, and this match is a bad match (since u is at a position $r > s$).

– For any $r \leq s$, consider the run on σ_r and θ. Vertex v is either unmatched or in a good match, since it cannot be matched above r.

From the above observations, v contributes to $bad^{\theta}_{\sigma_r}(s)$ only when $r > s$, i.e. in $n-s$ permutations in Ω_u. Since $opt'(u)$ has a total load of at least $1 - match^{\theta}_{\sigma_t}(t)$, we conclude that

$$1 - match^{\theta}_{\sigma_t}(t) \leq \sum_{\sigma \in \Omega_u} \sum_{s<t} \frac{bad^{\theta}_{\sigma}(s,u)}{n-s}.$$

□

The next lemma is a straightforward generalization of Lemma 2.3 in [11].

Lemma 10. *Fix a vertex $u \in U$, a permutation group Ω_u, and a threshold vector θ. Then, for all $t \in [n]$,*

$$\sum_{\sigma \in \Omega_u} \sum_{s \leq t} bad^{\theta}_{\sigma}(s,u) \frac{s}{n-s} \leq \sum_{\sigma \in \Omega_u} \sum_{s \leq t+1} good^{\theta}_{\sigma}(s,u).$$

We now aggregate the inequalities in the two lemmas above over the random choice of $\theta \in \mathbb{Z}^n_+$ and $\sigma \in \Omega$. Lemma 9 aggregates to

$$\forall\, t \in [n]: \quad \mathbb{E}_{\theta}\left[\mathbb{E}_u\left[\min(1, \theta_u)\right]\right] - match(t) \leq \sum_{s<t} \frac{bad(s)}{n-s} \tag{5}$$

Recall that for every vertex $u \in U$, Θ_u is drawn i.i.d. from the distribution $\mathbb{P}[\Theta_u = pt] = p(1-p)^{t-1}$ over positive integers t. Therefore,

$$\mathbb{E}_{\theta}\left[\mathbb{E}_u\left[\min(1, \theta_u)\right]\right] \quad = \quad \sum_{t=1}^{1/p} (pt)\left(p(1-p)^{t-1}\right) + \sum_{t > 1/p} p(1-p)^{t-1} \quad = \quad 1 - (1-p)^{1/p}.$$

Thus Eqn. (5) becomes

$$\forall\, t: \quad \frac{1-(1-p)^{1/p}}{p} \; - \; match(t) \; \leq \; \sum_{s<t} \frac{bad(s)}{n-s} \tag{6}$$

On the other hand, Lemma 10 aggregates to

$$\forall\, t: \quad \sum_{s \leq t} bad(s) \frac{s}{n-s} \; \leq \; \sum_{s \leq t+1} good(s) \tag{7}$$

The final ingredient in our proof is the a global counting lemma that follows immediately from Lemma 5 in Section 2.

Lemma 11. *We have*

$$\sum_t match(t) \geq \frac{1}{2} + \frac{\sum_t good(t)}{2} \tag{8}$$

Minimize $\int_0^1 m(x)dx$ subject to

$$\begin{aligned}
\int_0^t m(x)dx - \int_0^t v(x)dx &\geq 0 \quad \text{for all } t \in [0,1] \\
m(t) + \int_0^t v(x)dx &\geq \rho(p) \quad := \quad 1-(1-p)^{1/p} \quad \text{for all } t \in [0,1] \\
\int_0^1 m(x)dx + \int_0^1 (1-x)v(x)dx &= 1 \\
m(x), v(x) &\geq 0 \quad \text{for all } x \geq 0
\end{aligned}$$

(a) The primal LP

Maximize $\rho(p)\int_0^1 a(t)dt + c$ subject to

$$\begin{aligned}
(1-t)c + \int_t^1 a(x)dx - \int_t^1 b(x)dx &\leq 0 \quad \text{for all } t \in [0,1] \\
c + a(t) + \int_t^1 b(x)dx &\leq 1 \quad \text{for all } t \in [0,1] \\
a(t), b(t) &\geq 0 \quad \text{for all } t > 0
\end{aligned}$$

(b) The dual LP

Figure 3: The factor revealing LP and its dual for the RANKING algorithm.

Remark 1. *The above lemma holds in the case of (non-stochastic) online matching as well. However, in that case this inequality is not explicitly required in the analysis that proves the optimal competitive ratio. However, in our problem, without the inequality, we can only prove a factor of* $(1-1/e)^2 \simeq 0.4$*, and adding the inequality improves the competitive ratio substantially.*

We first do a simple transformation of variables for convenience. First, by taking n to be large, we move to a continuous domain for positions in the permutation. Further, we define the transformations:

$$\begin{aligned}
\forall\, x \in [0,1]: \quad m(x) &= match(\lfloor nx \rfloor) \\
\forall\, x \in [0,1]: \quad v(x) &= \frac{bad(\lfloor nx \rfloor)}{1-x}.
\end{aligned}$$

Finally, recall (from Lemma 2) that the expected number of successful vertices in U is equal to the expected total load on vertices in U. Starting with inequalities (6), (7) and (8), we first substitute $good(t)$ by the equivalent $match(t) - bad(t)$, move to the continuous domain and variables $m(x)$ and $v(x)$, and finally define an objective function that minimize the expected total load. This yields the primal-dual factor-revealing LP pair given in Fig. 3. Finally, we can prove Theorem 3 whose statement we repeat here:

Theorem 3. *The competitive ratio of the* RANKING *algorithm is at least* $\kappa(p)$*, where*

$$\kappa(p) = 1 - \frac{1}{e} - (1-2/e)(1-p)^{1/p}.$$

For vanishing and equal probabilities (i.e. $p \to 0$*), the competitive ratio is*

$$\lim_{p\to 0} \kappa(p) = 1 - \frac{2}{e}\left(1 - \frac{1}{e}\right) \simeq 0.5349.$$

Proof. The following dual feasible solution puts the lower bound of $\kappa(p)$ on the objective function.

$$\begin{aligned} a(t) &= e^{t-1} - 1/e \\ b(t) &= e^{t-1} \\ c &= 1/e \end{aligned}$$

□

5 An Upper Bound less than $1 - 1/e$

We will now give an upper bound on the performance of any algorithm for the ONLINE STOCHASTIC MATCHING problem. Let G_k be a family of graphs where $U = \{u_1, u_2, \ldots, u_k\}$ and $V = V_1 \cup V_2 \cup \ldots \cup V_k$ with each V_i containing $1/p$ identical vertices that are connected to $u_i, u_{i+1}, \ldots, u_k$ via edges with probability $p \to 0$. The STOCHASTIC BALANCE algorithm for an input instance G_k assigns vertices in V in round-robin fashion among its available neighbors. The next lemma claims optimality of this algorithm for the input graph family G_k.

Lemma 12. *The* STOCHASTIC BALANCE *algorithm is optimal for input graph G_k (for any k).*

Proof. We will show a key symmetry property: on any graph G_k, there exists an optimal algorithm that equally distributes the expected load due to vertices in V_i (for any i) among its neighbors. Before proving the property, we show that it implies the optimality of the STOCHASTIC BALANCE algorithm. Let L_i be the expected load on vertices $u_i, u_{i+1}, \ldots, u_k$ after the arrival of vertices in $V_1 \cup V_2 \cup \ldots \cup V_i$. We will show that L_i is maximized by the STOCHASTIC BALANCE algorithm (for each i) among all algorithms satisfying the symmetry property. The lemma then follows from Lemma 2 since the final expected load on vertex u_i is L_i for each i. We prove this optimality property by induction on i using the fact that the STOCHASTIC BALANCE algorithm is opportunistic. For $i = 1$, the property follows immediately. Suppose the property is true for L_{i-1}; then we need to show that $L_i - L_{i-1}$ is maximized by the STOCHASTIC BALANCE algorithm, which is again an immediate corollary of the opportunistic property.

Now, we prove the symmetry property by induction on i. If the expected load on vertices in U are unequal after the assignment of vertices in V_1, then the adversary strategy would be to define the vertex with the least expected load as u_1. Observe that a modified algorithm that moves an arbitrarily small amount ε of expected load from any other vertex u_i to u_1 does not decrease the sum of expected load on the vertices in U (and therefore the expected number of successes by Lemma 2) since u_1 does not have any neighbors in the remaining input whereas u_i does. Repeating this operation ultimately leads to equal expected load on all neighbors of V_1.

By the inductive hypothesis, assume that the expected load on vertices $u_i, u_{i+1}, \ldots, u_k$ due to vertices in $V_1, V_2, \ldots, V_{i-1}$ are equal. Therefore, all the neighbors of V_i are identical at this point. This allows the adversary to again choose the vertex that has the minimum expected load due to vertices in V_i as u_i, and by the above argument, the expected number of successes does not decrease if we modify the algorithm to equalize all the expected loads. □

Finally, we can prove Theorem 4, whose statement we repeat here:

Theorem 4. *No (randomized) algorithm for the* ONLINE STOCHASTIC MATCHING *problem has a competitive ratio of more than* $(1-11/(18e)-5/(6e^2)-5/(6e^3)) < 0.621 < 1-1/e$*, even in the case of equal and vanishing probabilities.*

Proof. We calculate the competitive ratio of the STOCHASTIC BALANCE algorithm for input graph G_3. First, note that the expected number of successes for G_1 is $1-1/e$. Next, we consider G_2. There are three possible outcomes of the coin tosses corresponding to vertices in V_1 in the graph G_2.

1. **No success.** This happens with probability $1/e$; in this case, the expected number of successes overall is the same as that for G_1, i.e. $1-1/e$.
2. **One success.** There are two subcases:
   (a) u_1 **successful.** This happens with probability $1/2e$; in this case, the expected number of successes for V_2 is the same as that for G_1, i.e. $1-1/e$. Therefore, the expected number of successes overall is $2-1/e$.
   (b) u_2 **successful.** This happens with probability $1/2e$; in this case, the number of successes overall is 1.
3. **Two successes.** This happens with probability $1-2/e$; in this case, the number of successes overall is 2.

Overall, the expected number of successes for G_2 is $2(1-3/4e-3/4e^2)$.

For G_3, there are four possible outcomes of the coin tosses corresponding to vertices in V_1.

1. **No success.** This happens with probability $1/e$; in this case, the expected number of successes overall is the same as that for G_2, i.e. $2(1-3/4e-3/4e^2)$.
2. **One success.** There are three subcases:
   (a) u_1 **successful.** This happens with probability $1/3e$; in this case, the expected number of successes for V_2 and V_3 is the same as that for G_2, i.e. $2(1-3/4e-3/4e^2)$. Therefore, the expected number of successes overall is $3-3/2e-3/2e^2$.
   (b) u_2 **successful.** This happens with probability $1/3e$; in this case, the expected number of successes for V_2 and V_3 is $1-1/e^2$. Therefore, the expected number of successes overall is $2-1/e^2$.
   (c) u_3 **successful.** This happens with probability $1/3e$; in this case, the expected number of successes for V_2 and V_3 is the same as that for G_1, i.e. $1-1/e$. Therefore, the expected number of successes overall is $2-1/e$.
3. **Two successes.** There are three subcases:
   (a) u_1 **and** u_2 **successful.** This happens with probability $1/6e$; in this case, the expected number of successes for V_2 and V_3 is $1-1/e^2$. Therefore, the expected number of successes overall is $3-1/e^2$.
   (b) u_1 **and** u_3 **successful.** This happens with probability $1/6e$; in this case, the expected number of successes for V_2 and V_3 is the same as that for G_1, i.e. $1-1/e$. Therefore, the expected number of successes overall is $3-1/e$.

(c) u_2 **and** u_3 **successful.** This happens with probability $1/6e$; in this case, the number of successes overall is 2.

4. **Three successes.** This happens with probability $1-5/2e$; in this case, the number of successes overall is 3.

Combining the above observations, the expected number of successes overall is $3(1-11/(18e)-5/(6e^2)-5/(6e^3))$. □

A natural direction would be to consider graphs G_k with larger values of k but it turns out the bound is minimized for $k=3$. Considering alternative input graph families is another possible direction for improving the bound; however, the analysis of other graph families is significantly more complicated because it is challenging to define an optimal algorithm in such cases.

6 Upper Bound on Non-adaptive Algorithms

Recall that an algorithm for the ONLINE STOCHASTIC MATCHING problem is said to be non-adaptive if the assignment of the new vertex $v \in V$ is independent of the success/failure of previous assignments. In order to prove Theorem 5, we will focus on instances where the probabilities on all edges are equal and vanishing (i.e. $p_{uv} = p \to 0$ for all edges $(u,v) \in E$). We will give an input distribution for which the expected competitive ratio of any deterministic non-adaptive algorithm for ONLINE STOCHASTIC MATCHING is at most 1/2. By Yao's minmax principle, the lemma then follows. U contains n vertices that are permuted uniformly at random and called $u_1, u_2, \ldots, u_n$. On the other hand, V contains n/p vertices that are organized into n groups $V_1, V_2, \ldots, V_n$ of $1/p$ vertices each. Each vertex in V_i is a neighbor of $u_i, u_{i+1}, \ldots, u_n$. The n groups of vertices in V arrive online in numerical order; internal to a group, the vertices arrive in arbitrary order. Clearly, the optimal solution matches all vertices in V_i to u_i and has an objective of n for any permutation of the vertices in U.

The next lemma bounds the expected load on each vertex $u \in U$.

Lemma 13. *For any deterministic algorithm for the* ONLINE STOCHASTIC MATCHING *problem with the input drawn from the distribution described above, let L_j denote the expected load on the vertex denoted u_j. For any i, $\sum_{j=1}^{i} L_j \leq \sum_{j=0}^{i-1} \frac{i-j}{n-j}$.*

Proof. We use induction on i. For $i=1$, we note that the sum of loads on all the vertices in U after all the vertices in V_1 have arrived is at most 1. Since u_1 is chosen at random from U, $L_1 \leq 1/n$.

Now, suppose the lemma holds for i. Let L be the total expected load on vertices $u_1, u_2, \ldots, u_i$. Then, the sum of loads on vertices $u_{i+1}, \ldots, u_n$ after all the vertices in V_{i+1} have arrived is at most $i+1-L$. Since u_{i+1} is chosen uniformly at random among these vertices, the expected load on u_{i+1} is at most $\frac{i+1-L}{n-i}$. Therefore,

$$\sum_{j=1}^{i+1} L_j \leq L + \frac{i+1-L}{n-i} \leq \sum_{j=0}^{i-1} \frac{i-j}{n-j} + \frac{i}{n-i} + \frac{i - \sum_{j=0}^{i-1} \frac{i-j}{n-j}}{n-i} = \sum_{j=0}^{i-1} \frac{i-j}{n-j} + \frac{i}{n-i} + \sum_{j=0}^{i-1} \frac{1}{n-j} = \sum_{j=0}^{i} \frac{i+1-j}{n-j}. \quad \square$$

For non-adaptive algorithms, the probability of success of a vertex $u \in U$ that has an overall load of L_u is $1-e^{-L_u}$ for $p \to 0$. The concavity of the function $1-e^{-x}$ implies that the expected number of successful vertices in U is maximized when the load on each vertex u_i is deterministic, and Lemma 13 is tight for every i. It follows that the expected number of successful vertices in U is at most $\mathbf{S}_n = \sum_{i=1}^{n} \left(1 - e^{-\sum_{j=0}^{i-1} \frac{1}{n-j}}\right)$. Theorem 5 now follows from $\lim_{n\to\infty} \mathbf{S}_n = 1/2$.

References

[1] Yossi Azar, Benjamin E. Birnbaum, Anna R. Karlin, Claire Mathieu, and C. Thach Nguyen. Improved approximation algorithms for budgeted allocations. In *ICALP (1)*, pages 186–197, 2008.

[2] Bahman Bahmani and Michael Kapralov. Improved bounds for online stochastic matching. In *ESA (1)*, pages 170–181, 2010.

[3] Nikhil Bansal, Anupam Gupta, Jian Li, Julián Mestre, Viswanath Nagarajan, and Atri Rudra. When lp is the cure for your matching woes: Improved bounds for stochastic matchings - (extended abstract). In *ESA (2)*, pages 218–229, 2010.

[4] Anand Bhalgat, Ashish Goel, and Sanjeev Khanna. Improved approximation results for stochastic knapsack problems. In *SODA*, pages 1647–1665, 2011.

[5] Niv Buchbinder, Kamal Jain, and Joseph Naor. Online primal-dual algorithms for maximizing ad-auctions revenue. In *ESA*, pages 253–264, 2007.

[6] Deeparnab Chakrabarty and Gagan Goel. On the approximability of budgeted allocations and improved lower bounds for submodular welfare maximization and gap. In *FOCS*, pages 687–696, 2008.

[7] Ning Chen, Nicole Immorlica, Anna R. Karlin, Mohammad Mahdian, and Atri Rudra. Approximating matches made in heaven. In *ICALP (1)*, pages 266–278, 2009.

[8] Brian C. Dean, Michel X. Goemans, and Jan Vondrák. Adaptivity and approximation for stochastic packing problems. In *SODA*, pages 395–404, 2005.

[9] Brian C. Dean, Michel X. Goemans, and Jan Vondrák. Approximating the stochastic knapsack problem: The benefit of adaptivity. *Math. Oper. Res.*, 33(4):945–964, 2008.

[10] Jon Feldman, Aranyak Mehta, Vahab S. Mirrokni, and S. Muthukrishnan. Online stochastic matching: Beating 1-1/e. In *FOCS*, pages 117–126, 2009.

[11] Gagan Goel and Aranyak Mehta. Online budgeted matching in random input models with applications to adwords. In *SODA*, pages 982–991, 2008.

[12] Michel X. Goemans and Jan Vondrák. Stochastic covering and adaptivity. In *LATIN*, pages 532–543, 2006.

[13] Sudipto Guha and Kamesh Munagala. Approximation algorithms for budgeted learning problems. In *STOC*, pages 104–113, 2007.

[14] Anupam Gupta, Ravishankar Krishnaswamy, Marco Molinaro, and R. Ravi. Approximation algorithms for correlated knapsacks and non-martingale bandits. In *FOCS*, 2011.

[15] Kamal Jain, Mohammad Mahdian, Evangelos Markakis, Amin Saberi, and Vijay V. Vazirani. Greedy facility location algorithms analyzed using dual fitting with factor-revealing lp. *J. ACM*, 50(6):795–824, 2003.

[16] Bala Kalyanasundaram and Kirk Pruhs. An optimal deterministic algorithm for online b-matching. *Theor. Comput. Sci.*, 233(1-2):319–325, 2000.

[17] Chinmay Karande, Aranyak Mehta, and Pushkar Tripathi. Online bipartite matching with unknown distributions. In *STOC*, pages 587–596, 2011.

[18] David R. Karger, Sewoong Oh, and Devavrat Shah. Budget-optimal task allocation for reliable crowdsourcing systems. *CoRR*, abs/1110.3564, 2011.

[19] David R. Karger, Sewoong Oh, and Devavrat Shah. Iterative learning for reliable crowdsourcing systems. In *NIPS*, pages 1953–1961, 2011.

[20] Richard M. Karp, Umesh V. Vazirani, and Vijay V. Vazirani. An optimal algorithm for on-line bipartite matching. In *STOC*, pages 352–358, 1990.

[21] Mohammad Mahdian and Qiqi Yan. Online bipartite matching with random arrivals: an approach based on strongly factor-revealing lps. In *STOC*, pages 597–606, 2011.

[22] Vahideh H. Manshadi, Shayan Oveis Gharan, and Amin Saberi. Online stochastic matching: Online actions based on offline statistics. In *SODA*, pages 1285–1294, 2011.

[23] Aranyak Mehta, Amin Saberi, Umesh V. Vazirani, and Vijay V. Vazirani. Adwords and generalized online matching. *J. ACM*, 54(5), 2007.

[24] Aravind Srinivasan. Budgeted allocations in the full-information setting. In *APPROX-RANDOM*, pages 247–253, 2008.