

Online Bipartite Matching with Unknown Distributions

Chinmay Karande
Google Research
Mountain View, CA
chinmayk@google.com

Aranyak Mehta
Google Research
Mountain View, CA
aranyak@google.com

Pushkar Tripathi^{*}
Georgia Inst. of Technology
Atlanta, GA
pushkar.tripathi@gatech.edu

ABSTRACT

We consider the online bipartite matching problem in the unknown distribution input model. We show that the RANKING algorithm of [KVV90] achieves a competitive ratio of at least 0.653. This is the first analysis to show an algorithm which breaks the natural $1 - 1/e$ ‘barrier’ in the unknown distribution model (our analysis in fact works in the stricter, random order model) and answers an open question in [GM08]. We also describe a family of graphs on which RANKING does no better than 0.727 in the random order model. Finally, we show that for graphs which have $k > 1$ disjoint perfect matchings, RANKING achieves a competitive ratio of at least $1 - \sqrt{\frac{1}{k} - \frac{1}{k^2} + \frac{1}{n}}$ – in particular RANKING achieves a factor of $1 - o(1)$ for graphs with $\omega(1)$ disjoint perfect matchings.

Categories and Subject Descriptors

G.2.1 [Discrete Mathematics]: Combinatorics—*Permutations and Combinations*; G.2.2 [Discrete Mathematics]: Graph Theory—*Graph Algorithms*

General Terms

Algorithms, Theory

Keywords

Online Algorithms, Bipartite Matching

1. INTRODUCTION

Online bipartite matching is a central problem in algorithms and has been recently found to be an important and useful model for allocation of ad space to advertisers. In its basic form, the problem involves a bipartite graph $G(L, R, E)$,

^{*}Work done partially during the author’s internship at Google Research, Mountain View

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

STOC’11, June 6–8, 2011, San Jose, California, USA.
Copyright 2011 ACM 978-1-4503-0691-1/11/06 ...\$10.00.

with one side L (ads, jobs, or items to sell, in different motivating examples) known beforehand to the algorithm, and vertices from the other side R (ad-slots, job-candidates, or buyers) arriving one by one online. When a vertex $r \in R$ arrives, its incident edges are revealed, and the algorithm can match it to some currently unmatched neighbor in L . The objective is to maximize the size of the matching obtained at the end.

As with previous literature, we measure the performance of an online algorithm by the expected competitive ratio (also referred to simply as the ‘factor’), where the expectation is over randomness in the algorithm as well as the input. This being a maximization problem, we use the convention that a lower bound is a positive result, while an upper bound is a negative result.

1.1 Input Models and Previous Results

This basic problem can be studied in different input models of how much information the algorithm has about the vertices in R .

Adversarial: The strictest model is the *adversarial order* model which means that the algorithm knows L but has no information about R (and therefore E) – this is the standard model for online algorithms. In this model, a simple Greedy algorithm achieves a competitive ratio of $1/2$ since it produces a maximal matching, and this is optimal among deterministic algorithms. A simple randomized algorithm which matches the arriving vertex to a random unmatched neighbor does no better. In a beautiful result [KVV90], Karp, Vazirani and Vazirani described an optimal randomized algorithm, which achieves an expected competitive ratio of $1 - 1/e \simeq 0.632$ (see also, [GM08, BM08] for different proofs for the same result). This algorithm, called RANKING, uses *correlated* randomness to match arriving vertices: pick a random permutation of L , and match each arriving vertex $r \in R$ to the neighbor with the highest rank in the permutation.

Known Distribution: In Feldman et al. [FMMM09], the authors introduced a distributional input model, which we shall call the *known distribution input model*. In this model, the algorithm knows beforehand, a base graph $\hat{G}(L, \hat{R}, \hat{E})$, and a distribution \mathcal{D} on \hat{R} . The arriving vertices are sampled i.i.d. (without replacement) from \hat{R} according to \mathcal{D} . Each arriving vertex has the same incident edges as its copy in \hat{G} . Thus this is a weaker model, since the algorithm is guaranteed i.i.d. samples, and also knows the underlying distribu-

Model	Adversarial Input	Known Distribution	Unknown Distribution
Lower Bounds (algorithms)	$1 - \frac{1}{e}$ [KVV90]	0.67 [FM09] 0.699 [BK10] 0.702 [MOGS11]	$1 - \frac{1}{e}$ [KVV90] 0.653 [This paper]
Upper Bounds (hardness)	$1 - \frac{1}{e}$ [KVV90]	0.998 [FM09] 0.902 [BK10] 0.823 [MOGS11]	$5/6$ [GM08] 0.823 [MOGS11]

Table 1: Summary of Results for Online Bipartite Matching in Various Models

tion. Clearly, RANKING achieves $1 - 1/e$ in this model as well, since it does so in the adversarial model (for any sequence of inputs, without any prior information). Feldman et al. [FM09] provided an algorithm which achieves a factor strictly greater than $1 - 1/e$, of $\frac{1-2/e^2}{4/3-2/3e} \simeq 0.670$. They also showed that the optimal factor for any online algorithm in this setting was bounded away from 1. Their upper and lower bounds were subsequently improved by Bahmani and Kapralov [BK10] to 0.902 and 0.699 respectively in the same setting. They also presented a randomized algorithm that achieves a competitive ratio of $1 - O(1/\sqrt{d})$ for d -regular graphs. Subsequently, [MOGS11] provided an improved algorithm with a factor of 0.702 and proved an upper bound of 0.823.

Unknown Distribution / Random Order: The model we study in this paper lies in between these two models in terms of how much information about the arriving vertices the algorithm has beforehand. In this model, which we call the *unknown distribution model*, the arriving vertices are guaranteed to be picked from a distribution on some base graph, but the algorithm has no knowledge about the base graph and the distribution. Closely related to this model, but stricter than this model, is the *random order arrival model*, in which there is a base graph $G(L, R, E)$, the algorithm only knows L , and the arriving vertices are guaranteed to be in a random permutation of R . An algorithm which achieves a competitive ratio of α in the random order model also has a factor of α in the unknown distribution model (See Section 2.2 for a proof).

The only previous result in this model follows, in fact, from the structure of the RANKING algorithm. The analysis for RANKING in the adversarial model itself implies that a simple GREEDY algorithm (with consistent tie-breaking) achieves a competitive ratio of $1 - 1/e$ in the random order model. Clearly, RANKING itself also achieves at least $1 - 1/e$, since it does so on any input sequence and without any prior information. Furthermore, it was proved in [GM08] that no online algorithm can achieve competitive ratio better than $5/6$. However, unlike in the case for the known distribution model result (in which [FM09] broke through the $1 - 1/e$ ‘barrier’), and despite considerable effort, there was no algorithm known to achieve more than $1 - 1/e$ factor in the unknown distribution model or in the random order model. We summarize the above discussion in Table 1.

1.2 Our Results

In this paper we analyze the competitive ratio of the RANKING algorithm in the random order arrival model. We prove the following three results:

(a) RANKING achieves a factor of at least 0.653 in the ran-

dom order model (and hence in the unknown distribution model)¹.

(b) There are graphs for which RANKING achieves a ratio of no more than 0.727 in the random order model.

(c) RANKING achieves a factor of at least $1 - \sqrt{\frac{1}{k} - \frac{1}{k^2} + \frac{1}{n}}$ for graphs which have at least $k > 1$ disjoint perfect (or near-perfect) matchings.

Remark 1 No algorithm crossing the $1 - 1/e$ barrier in the unknown distribution or the random order input models was previously known. This answers an open question in [GM08].

Remark 2 Since the random order model is stricter than the unknown distribution model, the positive results (a and c) hold there as well.

Remark 3 The third result (c) above states that if the graph has certain redundancy with respect to optimal matchings then RANKING is almost optimal! If the underlying distribution is such that with high probability any instance drawn from the distribution has $\omega(1)$ disjoint matchings of size at least $n - o(n)$ then the RANKING algorithm achieves a factor of $1 - o(1)$. We note that this result is stronger than the result in [BK10] mentioned earlier, which provides an algorithm which achieves $1 - O(1/\sqrt{d})$ for d -regular graphs (noting that d regular graphs have d disjoint perfect matchings).

As an additional remark, result (c) resolves a puzzling mystery which is observed during simulations of RANKING on the graph with the upper triangular matrix as its adjacency matrix. If the input order is adversarial then the ratio is $1 - 1/e$ (this is the tight example for the algorithm in that model and can be proved analytically [KVV90]), but with random order input, the ratio is observed to go to 1 as n increases. The mystery is resolved by observing that this graph indeed has many $(\omega(1))$ almost-perfect matches.

Remark 4 We point out that in the unknown distribution model the support for the distribution may be exponentially large, whereas all previous work in the known distribution case assumes that the support is polynomially bounded. So, e.g., our model allows a distribution on a base graph which is of exponential size (\hat{R} is exponential in size), and an exponential sized distribution on this graph. The algorithm only uses the guarantee that the arriving sequence of n vertices is sampled i.i.d. from some distribution.

¹We can in fact prove a slightly stronger factor of 0.667 for this problem, but this requires us to use computational means to minimize a non-convex function over fixed number of variables. We omit that proof from this paper.

1.3 Techniques

The basic RANKING algorithm on $G(L, R, E)$ can be thought of as taking permutations ρ of L and π of R , processing the next vertex in R in order according to π , and greedily matching it to the neighbor with highest rank according to ρ . In the adversarial input order model, the analysis is done for a fixed adversarial π and a uniform distribution over ρ . In the random order input, we need to analyze the performance over uniform distributions over both ρ and π .

The most optimistic approach one may hope for would be a simple extension of the $1 - 1/e$ proof for the adversarial order (we pick the one due to Birnbaum and Matheiu [BM08]). As we will describe briefly in Section 3.2, the core of this proof is a mapping, which maps events in the probability space in which a vertex does not get matched, to n different events in which some other vertices do get matched, and in higher ranks according to ρ . These n events are obtained by moving a vertex in L to all n positions in the permutation. In our problem R is also in a random permutation, so the simplest approach would be to have a 1 to $2n$ map (or maybe a 1 to n^2 map) by moving one vertex in L and one in R to each of n positions. In this way one could hope to achieve a ratio larger than $1 - 1/e$. However, any such map can be seen to be doing double counting, and this method does not seem to work. Thus, a different and possibly more detailed method is required.

Gains above 1/2: We start with a classification of matches made by our algorithm into *B-events*, *A-events* and *P-events*, as follows: Fix an optimal offline matching, OPT, which matches $u \in L$ to $u^* \in R$. If in a particular event (ρ, π) , the algorithm matches $u \in L$ to $v^* \in R$, then we call this match (a) a B-event for u , if u^* is ranked higher than v^* in π , (b) an A-event for u if u^* is ranked lower than v^* in π , and (c) a P-event (perfect match) for u if $u^* = v^*$, i.e., u was actually matched to its OPT partner. The counterparts of these events for u^* in R - \hat{B} , \hat{A} and \hat{P} -events are defined symmetrically. We will define these events more rigorously in Section 2.3.

The first observation is that the gain of the algorithm above 1/2 is the number of vertex pairs (v, v^*) (according to OPT) which both got matched in the algorithm. This is equal to the number of B-events on the right, plus the number of B-events on the left, plus the number of P-events (perfect match). In the worst case examples for the analysis of RANKING in adversarial input, the number of B-events and P-events approaches 0 asymptotically. Thus the gain above 1/2 in the adversarial order comes from only \hat{B} -events on the right. The next core observation is that in the random order input model, the number of B-events on the left and on the right can be proved to be equal without loss of generality. In effect, this gives twice as much gain over 1/2 than in the adversarial order (which would be $2(1 - 1/e - 1/2)$) giving a factor of $3/2 - 2/e \simeq 0.764$. In fact we can prove that if there are no perfect matches then the number of B-events on both sides is so substantial that the ratio actually goes to 1! Thus we get a ratio close to 1 when the graph has a structure which enforces very few perfect matches. One such class of graphs is the one referred to in the third result (c) in Section 1.2.

Now, intuitively, perfect matches should not hurt the algorithm: After all if a vertex v was matched to its OPT neighbor v^* , then that is a good turn of events, so why can

we not simply excise this event from the probability space? It turns out that this can not be done, and in fact the factor stays no more than 0.727 in the presence of perfect matches. The core reason is that due to the two sided randomness, any below matches on the left also imply below matches on the right, which can be counted separately, while perfect matches can only be counted once. The rest of the proof involves trading off the below matches and perfect matches, to prove the factor of 0.653 in general.

1.4 Other Related Models and Results

Besides the results mentioned in Section 1.1, there has been considerable interest in online matching and allocation problems over the last few years and several variants and generalizations of the bipartite matching problem have been studied. The case of weighted edges was considered by Korrula and Pal in [KP09] and they design an 8-competitive algorithm for this problem. The variant where there are weights on the incoming vertices was studied by Aggarwal et al. in [AGKM11]. The authors present tight $1 - 1/e$ factor algorithms for this problem.

The online bipartite matching problem has also been studied in the context of the Adwords model. In this model each vertex in L has a budget and each edge has a bid. All bids are very small with respect to the budgets. A match deducts the bid amount from the corresponding budget. The Adwords setting is incomparable to graph matching: It is easier in the sense of having large budgets (hence mistakes can be rectified in the future), but harder because of different bid values.

The adversarial arrival model for the Adwords problem was first analyzed by Mehta et al. in [MSVV05]. They obtained a tight $1 - 1/e$ competitive algorithm for this model. Buchbinder et al. [BJN07] gave a primal dual algorithm for the same problem. Understanding the Adwords allocation problem in distributional models is important, and was an open question in [MSVV05, MNS07].

In [DH09], Devanur and Hayes studied the adwords problem in the unknown distribution model similar to the one presented in this paper and obtained a $1 - o(1)$ competitive algorithm for this problem. However, this method does not work for bipartite matching which can be considered as the adwords problem with every budget equal to 1 and every bid equal to 1 or 0.

Simultaneously and independent of our work, Mahdian and Yan [MY11] also analyzed the performance of RANKING in the random order model, and also breached the barrier of $1 - 1/e$ by showing that RANKING achieves a factor of at least 0.696. Their techniques are different from ours. They find a family of strongly factor revealing LPs for this problem and solve large LPs in this family computationally to provide a good lower bound on the factor.

2. PRELIMINARIES

2.1 Problem Statement

In this model there is a fixed bipartite graph $\hat{G}(L, \hat{R}, \hat{E})$; one side (L) of the graph corresponds to a fixed set of vertices and the other side (\hat{R}) represents the set of all possible vertices that may arrive online. There is also an (unknown) distribution \mathcal{D} over the vertices of \hat{R} . At each time step, a vertex in \hat{R} is sampled independently from \mathcal{D} (with re-

placement), and it needs to be matched to an unmatched neighboring vertex in L upon its arrival. Thus the sample space consists of the different sequences of vertices obtained by drawing n times from the fixed but unknown distribution. We call this the unknown distribution model (UD Model). The goal is to maximize the expected size of the matching.

Let \mathcal{A} be an (possibly randomized) online algorithm. For any realization graph G based on n random samples from \mathcal{D} , let $ALG(G)$ be the expected size of the matching produced by \mathcal{A} . Let $OPT(G)$ be the size of the largest matching in G . The competitive ratio for \mathcal{A} is defined to be $E_G \left[\frac{ALG(G)}{OPT(G)} \right]$, where the expectation is over the realization graphs.

2.2 From Unknown Distribution to Random Order Arrival Model

In this section we define a different, stricter model for online bipartite matching called the Random Order Arrival Model (ROA Model), and establish its relationship with the Unknown Distribution Model (UD Model). As before, in this model, there is a fixed bipartite graph $G(L, R, E)$ with vertex set $L \cup R$. The vertices of L are known in advance, while the vertices of R arrive in random order. Whenever a vertex arrives its incident edges are revealed, and it can be matched off to one of its available (unmatched) neighbors in R . The goal is to maximize the expected size of the matching. The ROA Model can be thought of as sampling from the uniform distribution on R without replacement. In the next lemma we show that the ROA Model is in fact stricter than the UD Model.

LEMMA 1. *Any online algorithm \mathcal{A} that achieves a competitive ratio of α in the ROA Model also achieves a competitive ratio of at least α in the UD Model. Lower bounds for the performance of online algorithms in the UD Model carry over to the ROA Model.*

PROOF. Consider a problem instance in the UD Model, with n arriving vertices. Divide the sample space for this instance into classes, each of size $factorial(n)$ such that for each class, the multi-set of arriving vertices (from \widehat{R}) is the same for every sequence in that class. Each class consists of all the permutations of some set, and furthermore, the probability of occurrence of each sequence in a class is the same. Thus each class can be thought of as an instance of a random order arrival input. Since \mathcal{A} has a competitive ratio of α in the ROA Model, it performs at least as well for all samples in a particular class. Taking expectation over the different classes we get the first part of the lemma. The second part can be proved by a similar argument. \square

Hence we will now focus our attention on designing an online algorithm for the Random Order Arrival Model.

2.3 Definitions in the ROA Model

Throughout the paper for convenience we will assume that both bi-partitions L and R , have equal size n , and the underlying graph has a perfect matching which we will refer to as OPT (if there are multiple perfect matchings then we choose one arbitrarily). It is an easy exercise to verify that this assumption is without loss of generality. We will use $u, v \in [n]$ to denote the vertices of R (the streaming side) and $s, t \in [n]$, to index the the vertices of L . For any $u \in R$, we will use u^* to denote its match in OPT we will refer to

u^* (resp. u) as the *partner* of u (resp. u^*) with respect to OPT .

We also use the notion of time: time t will denote the event when the t^{th} vertex of R is revealed to the algorithm. Define Ω_R (resp. Ω_L) to be the set of all permutations of R (resp. L). For any permutation ρ of the vertices, we will use $\rho(t)$ to denote the vertex at the t^{th} position in ρ and $\rho^{-1}(u)$ to denote the position of the vertex u in ρ . For vertices u, v we say u is above v in ρ if $\rho^{-1}(u)$ is less than $\rho^{-1}(v)$. We can similarly define the notion of a vertex u being below another vertex v . For any permutation $\rho \in \Omega_L$ let $\rho[u \rightsquigarrow s]$ denote the permutation obtained by moving u to position s keeping the order of the other vertices unchanged. We will use $\pi \in \Omega_R$ to represent the order in which the vertices of R arrive.

The RANKING Algorithm: In this paper we will analyze RANKING which was first proposed in [KVV90]. The algorithm takes an ordering of R as input and produces a matching as its output. The algorithm is described below.

RANKING

1. Choose a random permutation ρ from Ω_L uniformly at random.
2. Apply permutation ρ to L - thereby assigning each vertex a priority or rank.
3. For each arriving vertex from R match it to the vertex (if any) of L , with the highest rank.

By a slight abuse of notation we will use $Ranking(\rho, \pi)$ to denote an invocation of the above algorithm where ρ is the permutation chosen in the first step and π denotes the arrival order of the vertices in R . For any $t \in [n]$ define x_t to be the probability that the vertex $\rho(t) \in L$ at position t get matched in RANKING, where the probability is taken over the random choices of ρ and π .

Events, B, \widehat{B}, P -events: For each $\rho \in \Omega_L$, $\pi \in \Omega_R$ and every vertex $v \in L$ (or $v^* \in R$), we define an *event* to be the tuple (ρ, v, π) (or (ρ, π, v^*)). Thus an event is simply the specification of what the two permutations are and which vertex we are talking about. Note that an event specifies the position of the vertex and its optimal match.

An event (ρ, v, π) is called a B -event if v is matched in $Ranking(\rho, \pi)$, and furthermore it is matched to some vertex $u^* \neq v^*$ such that $\pi^{-1}(u^*) > \pi^{-1}(v^*)$ (i.e., v^* is ranked higher than u^* in π). Informally, a B -event is an event in which v is matched 'below' v^* . Note that this also means that v^* is itself matched.

Symmetrically, an event (ρ, π, v^*) is called a \widehat{B} -event if v^* is matched in $Ranking(\rho, \pi)$, and furthermore it is matched to some vertex $u \neq v$ such that $\rho^{-1}(u) > \rho^{-1}(v)$ (i.e., v is ranked higher than u in ρ).

An event (ρ, v, π) is called a P -event if v is matched to v^* in $Ranking(\rho, \pi)$. An event (ρ, π, v^*) is called a \widehat{P} -event if v^* is matched to v in $Ranking(\rho, \pi)$.

We define B (resp. \widehat{B} , resp. P) to be the total probability of B -events (resp. \widehat{B} -events, resp. P -events).

2.4 Symmetry of RANKING in the ROA Model

In this section we will present two important lemma's that distinguish the behavior of RANKING in ROA Model from the adversarial model considered in [KVV90]. The first lemma

states that the roles of the streaming and the static bipartitions may be switched without altering the outcome of the algorithm. The second lemma asserts that for the purpose of analysis, without loss of generality we may assume that the given graph is symmetric.

LEMMA 2. *For a given graph $G(L, R, E)$ and for a fixed $(\rho, \pi) \in \Omega_L \times \Omega_R$, the output of $\text{Ranking}(\rho, \pi)$ on G is same as the output of $\text{Ranking}(\pi, \rho)$ on $G'(R, L, E)$ (where G' is obtained from G by switching the two partitions).*

PROOF. Let M be the output for $\text{Ranking}(\rho, \pi)$ for the graph G . Let us simulate $\text{Ranking}(\pi, \rho)$ over G' . We will prove the above claim by induction on the number of vertices in the set L from G' which have arrived at any moment. Suppose the part of $\text{Ranking}(\pi, \rho)$ constructed over $\rho(1), \dots, \rho(t-1)$ in our simulation is consistent with M . Let $v = \rho(t) \in L$ be the t 'th vertex to arrive. If possible let v be unmatched in M and suppose it gets matched to $u^* \in R$ in our simulation. By our hypothesis, u^* must have been matched below position t in M . This yields a contradiction since, u^* could have matched a higher vertex in M namely v . Similarly, we can argue the case when v is unmatched in our simulation, but is matched in M .

The only other possibility is that v matches different vertices in M and in our simulation. Suppose v is matched to w^* in M and u^* in our simulation. If $\pi^{-1}(w^*) < \pi^{-1}(u^*)$, then w^* is also matched, to say z , in our simulation. If $\rho^{-1}(z) < t$, then this contradicts the induction hypothesis. On the other hand if $\rho^{-1}(z) > t$, then v should have matched w^* and not u^* . A similar argument works for the case when $\pi^{-1}(u^*) > \pi^{-1}(w^*)$. \square

LEMMA 3. *Without loss of generality, we may assume that the worst example for RANKING in the ROA Model is symmetric.*

PROOF. Let $W(L, R, E)$ be a worst example for RANKING in the ROA Model. Let $W_1(L_1, R_1, E_1)$ and $W_2(L_2, R_2, E_2)$ be two copies of W . Consider the graph $W'(L', R', E')$ where $L' = L_1 \cup L_2$, $R' = R_1 \cup R_2$ and $E' = E_1 \cup E_2$. Note that W' is a symmetric graph. Since the two copies are disjoint the competitive ratio for RANKING on W' is just the average competitive ratio for the two components. By Lemma 2, both W_1 and W_2 attain the same competitive ratio. Hence there exists a symmetric graph W' for which RANKING attains its worst competitive ratio. \square

Lemma 3 yields the following corollary.

COROLLARY 4. *Without loss of generality, we may assume that in the worst example for RANKING in the ROA Model, $B = \hat{B}$.*

3. ANALYSIS OF THE COMPETITIVE RATIO OF RANKING

In the Section 1.3, we mentioned the components missing from previously known analyses of RANKING, *viz.* the B-events and P-events. If the order of arrival is arbitrary, they are irrelevant, because RANKING produces almost no B-events and P-events when faced with the tight example of the complete upper triangular matrix. In Section 3, we prove that these components gain significance in the random arrival model, and help push the approximation factor beyond $(1 - \frac{1}{e})$.

In Section 3.1, we consider the special case when the aggregate probability of a P-event is small and prove a bound on the competitive ratio of RANKING as a function of this probability. This implies the near optimal result in the case with many disjoint matchings. In Section 3.2, we prove a factor of 0.653 in general.

First we prove a simple but important counting lemma which expresses the gains of the algorithm above 1/2 in terms of the B , \hat{B} and P -events. In fact the lemma holds for RANKING based on any distribution of ρ and π (even for fixed ρ, π).

LEMMA 5.

$$ALG \geq \frac{n}{2} + \frac{B}{2} + \frac{\hat{B}}{2} + \frac{P}{2}$$

PROOF. We count the total number of vertices matched and divide by two to get the size of the matching. We have fixed an OPT; for every pair v, v^* in the OPT matching, we know that at least one of them is matched in ALG (in every ρ, π). The greedy property of RANKING implies that (ρ, π, v) and (ρ, π, v^*) cannot simultaneously be B-event and \hat{B} -event respectively. If both v and v^* are matched, we pick the vertex which is not a B-match (resp. \hat{B} -match). If v and v^* are matched to each other in a P-event, then we pick v (to break ties). This gives us n matched vertices. Now, each B-match and \hat{B} -match has not been already counted in these n vertices. So also, each P-event corresponds to two matched vertices, but we counted only the left vertex. Thus we can add $B + \hat{B} + P$ to n to get a lower bound on the total number of matched vertices. This proves the lemma. \square

From the statement of Lemma 5, it is clear that we need to prove that in random permutation model (with symmetry implied by Corollary 4), B-events, \hat{B} -events and P-events make a sizeable combined contribution. We achieve this in the following lemma:

LEMMA 6.

$$\sum_t \frac{(t-1)}{n} x_t \leq \hat{B} + P - \frac{P^2}{2n}$$

where x_t is the probability that the vertex at rank t in L is matched in RANKING.

This lemma is the technical core of our main result. Before moving on to the proof, we first introduce some notation which will be useful.

Notice that the events that contribute to the x_t variables on the LHS are simply the set of all occurrences of matched vertices, whereas the RHS consists of events where $v^* \in R$ is matched to $u \in L$ such that $\rho^{-1}(u) \geq \rho^{-1}(v)$. Our proof involves designing a many-to-many map between the former set of events to the latter.

In order to do that, we need to classify the relevant events into two types with distinct properties. We need the following definitions: If ρ is any permutation on L , then ρ_{-v} is the permutation on $L - \{v\}$ consistent with ρ . As defined in Section 2.3, $\rho[v \rightsquigarrow s]$ is the permutation obtained by inserting v into ρ_{-v} at position s . A *column* is a collection of n events defined by a permutation ρ_{-v} (on L) and π (on R), by inserting v into ρ_{-v} at all n positions.

This notion of *columns* of events has been used in related literature without being defined explicitly. For example, it

forms the basis of analysis of RANKING in the arbitrary arrival model, as found in [KVV90, GM08, BM08].

We will use the above property, but in our analysis, we will also look at events in columns where v is always matched, but v^* may be unmatched. From this point of view, we need to classify the columns into two different types.

DEFINITION 1. A column (ρ_{-v}, π) is said to be a T1 (type 1) column if in the configuration $(\rho[v \rightsquigarrow n], \pi)$,

- v is unmatched OR
- $(\rho[v \rightsquigarrow n], v, \pi)$ is a B-event OR
- $(\rho[v \rightsquigarrow n], v, \pi)$ is an A-event and $(\rho[v \rightsquigarrow n], \pi, v^*)$ is also an A-event

Otherwise, the column is said to be a T2 (type 2) column, i.e. if in the configuration $(\rho[v \rightsquigarrow n], \pi)$,

- v is matched to v^* (P-event) OR
- $(\rho[v \rightsquigarrow n], v, \pi)$ is an A-event and v^* is unmatched

We will use the following properties, which are simple consequences of the above definitions and the RANKING algorithm:

- v^* is always matched in a configuration where v is in a T1 column.
- If v is in a T2 column, then v^* can match no higher than v .

We can now prove Lemma 6.

PROOF. (of Lemma 6) Let \mathcal{X} be the set of all events (ρ, u, π) , where $u = \rho(t)$ is matched to $v^* \in R$. Similarly, $\hat{\mathcal{B}}, \mathcal{P}$ and $\hat{\mathcal{P}}$ are the sets of all \hat{B} -events, P-events and \hat{P} -events respectively. Partition \mathcal{X} into two sets: \mathcal{X}_1 such that (ρ_{-v}, π) is a T1 column for v and \mathcal{X}_2 such that (ρ_{-v}, π) is a T2 column for v .

We now define our many-to-many maps. For $(\rho, u, \pi) \in \mathcal{X}_1$, consider the situation of v^* , in the configuration $(\rho[v \rightsquigarrow s], \pi)$ for $s < t$. In other words, we move v to all positions above u . We claim that $(\rho[v \rightsquigarrow s], \pi, v^*)$ is either a \hat{B} -event or a \hat{P} -event. The fact that v^* remains matched in $(\rho[v \rightsquigarrow s], \pi)$ follows from the fact the configuration is a T1 column for v , and Lemma 9. Let w be the vertex to which v^* is matched in $(\rho[v \rightsquigarrow s], \pi)$. To prove $\rho[v \rightsquigarrow s](w) \geq s$, observe that if we remove v from ρ , v^* may be unmatched, or matched no earlier than its original match u which was at position t . Now, adding v back to ρ_{-v} at any position s above that of u , can improve the position of the match of v^* to no higher than s . This implies that $(\rho[v \rightsquigarrow s], \pi, v^*)$ is either a \hat{B} -event (if $\rho[v \rightsquigarrow s](w) > s$) or a \hat{P} -event (if $\rho[v \rightsquigarrow s](w) = s$).

Hence, we can now map each event $(\rho, u, \pi) \in \mathcal{X}_1$ to $t-1$ different events $(\rho[v \rightsquigarrow s], \pi, v^*)$, all which are either \hat{B} -events or \hat{P} -events. For each $(\rho, u, \pi) \in \mathcal{X}_1$, we define:

$$f(\rho, u, \pi) = \{ (\rho[v \rightsquigarrow s], \pi, v^*) \mid 1 \leq s < t \} \subseteq \hat{\mathcal{B}} \cup \hat{\mathcal{P}}$$

Let

$$M = \bigsqcup_{(\rho, u, \pi) \in \mathcal{X}_1} f(\rho, u, \pi)$$

where the \bigsqcup symbol represents multiset union. Clearly, although each $f(\rho, u, \pi)$ is a simple set, the same \hat{B} -event (or \hat{P} -event) may be part of two different such sets. We will now quantify this over-counting. First, let us partition M into $M_b = M \cap \hat{\mathcal{B}}$ and $M_p = M \cap \hat{\mathcal{P}}$.

Any event $(\rho, \pi, v^*) \in M_b$ appears in the maps of at most n distinct events from \mathcal{X}_1 , those obtained by moving v to all positions from 1 to n , keeping the rest of the configuration constant. Therefore,

$$|M_b| \leq n|\hat{\mathcal{B}}_1| \quad (1)$$

where $\hat{\mathcal{B}}_1$ is the simple set obtained by discarding duplicates from M_b .

For each \hat{P} -event (ρ, π, v^*) in M_p , we can form a set N_p of the corresponding P-events: (ρ, v, π) . Let \mathcal{P}_1 be the simple set obtained by discarding duplicates from N_p . Our goal is to count the number of times each event $(\rho, v, \pi) \in \mathcal{P}_1$ appears in the multiset N_p . We claim that this number is at most $n - U(\rho, v, \pi)$ where U is defined for a P-event (ρ, v, π) as $U(\rho, v, \pi)$ is the number of P-events $(\rho[v \rightsquigarrow s], v, \pi)$ in column (ρ_{-v}, π) such that $s < \rho^{-1}(v)$. This follows from the fact that if (ρ, v, π) and $(\rho[v \rightsquigarrow s], v, \pi)$ are both P-events and $\rho^{-1}(v) < s$ then $(\rho[v \rightsquigarrow s], v, \pi)$ does not appear in $f(\rho, v, \pi)$. Now for notational convenience, we define:

$$P(\rho_{-v}, \pi) = \text{Total number of P-events in column } (\rho_{-v}, \pi)$$

Therefore, we can now bound the size of M_p as:

$$\begin{aligned} |M_p| &= |N_p| \\ &\leq \sum_{(\rho, v, \pi) \in \mathcal{P}_1} n - U(\rho, v, \pi) \\ &= n|\mathcal{P}_1| - \sum_{\text{T1}(\rho_{-v}, \pi)} \left(\sum_{(\rho, v, \pi) \in (\rho_{-v}, \pi)} U(\rho, v, \pi) \right) \\ &= \left(n \sum_{\text{T1}(\rho_{-v}, \pi)} P(\rho_{-v}, \pi) \right) - \\ &\quad \left(\sum_{\text{T1}(\rho_{-v}, \pi)} \frac{[P(\rho_{-v}, \pi)]^2}{2} \right) \end{aligned} \quad (2)$$

First let y_t be the probability that an event (ρ, v, π) such that $\rho^{-1}(v) = t$ is in \mathcal{X}_1 . Using the fact that such an event has $|f(\rho, v, \pi)| = t-1$, and using equations (1) and (2),

$$\begin{aligned} \sum_t (t-1)y_t &= \frac{|\hat{M}_1|}{|\Omega_L \times \Omega_R|} = \frac{|M_b| + |M_p|}{|\Omega_L \times \Omega_R|} \\ &\leq \frac{n|\hat{\mathcal{B}}_1|}{|\Omega_L \times \Omega_R|} + \\ &\quad \frac{\sum_{\text{T1}(\rho_{-v}, \pi)} \left(nP(\rho_{-v}, \pi) - \frac{[P(\rho_{-v}, \pi)]^2}{2} \right)}{|\Omega_L \times \Omega_R|} \end{aligned} \quad (3)$$

Now, we will deal with events in \mathcal{X}_2 . Since these are events in T2 columns, (ρ, π, v^*) must already be a \hat{B} -event or a \hat{P} -event. Let $\hat{\mathcal{B}}_2$ be the set of \hat{B} -events (ρ, π, v^*) such that v^* is matched to u in the corresponding $(\rho, u, \pi) \in \mathcal{X}_2$. By this one-to-one correspondence, $|\hat{\mathcal{B}}_2| = |\mathcal{X}_2 - \mathcal{P}|$. Let $\mathcal{P}_2 = \mathcal{X}_2 \cap \mathcal{P}$. Therefore, $|\mathcal{X}_2| = |\hat{\mathcal{B}}_2| + |\mathcal{P}_2|$.

Since the probability that $(\rho, u, \pi) \in \mathcal{X}_2$ is $x_t - y_t$, we have:

$$\sum_t (x_t - y_t) = \frac{|\hat{\mathcal{B}}_2|}{|\Omega_L \times \Omega_R|} + \frac{|\mathcal{P}_2|}{|\Omega_L \times \Omega_R|} \quad (4)$$

Let \mathcal{Q} be the multiset formed by including each event $(\rho, v, \pi) \in \mathcal{P}_2$ a total of $\rho^{-1}(v)$ times. Then we can multiply the t 'th term on the LHS of equation (4) by $(t-1)$ and compensate for the over-counting by (a) raising the coefficient of $|\hat{\mathcal{B}}_2|$ to n and (b) by using $|\mathcal{Q}|$ instead of $|\mathcal{P}_2|$

$$\sum_t (t-1)(x_t - y_t) \leq \frac{n|\hat{\mathcal{B}}_2|}{|\Omega_L \times \Omega_R|} + \frac{|\mathcal{Q}|}{|\Omega_L \times \Omega_R|} \quad (5)$$

Now we will bound $|\mathcal{Q}|$. Every P -event $(\rho, v, \pi) \in \mathcal{P}_2$ where appears in \mathcal{Q} a total of $\rho^{-1}(v)$ times. Now $\rho^{-1}(v) = n - (n - \rho^{-1}(v))$. But for every P -event in a T2 column, there are $n - \rho^{-1}(v)$ P -events $(\rho[v \rightsquigarrow s], v, \pi)$ at positions $s > \rho^{-1}(v)$ in the same column. Therefore, we can say that each event in $(\rho, v, \pi) \in \mathcal{P}_2$ appears $n - W(\rho, v, \pi)$ times in \mathcal{Q} , where W is defined for $(\rho, v, \pi) \in \mathcal{P}_2$ as $W(\rho, v, \pi)$ is the number of P -events $(\rho[v \rightsquigarrow s], v, \pi)$ in column (ρ_{-v}, π) such that $s > \rho^{-1}(v)$.

And finally, we will borrow the notation $P(\rho_{-v}, \pi)$ defined earlier to mean the total number of P -events in the column.

Following arguments analogous to the proof of equation (3), we can now rewrite equation (5) as:

$$\begin{aligned} \sum_t (t-1)(x_t - y_t) &= \frac{n|\hat{\mathcal{B}}_2|}{|\Omega_L \times \Omega_R|} + \\ &\frac{\sum_{\text{T2 } (\rho_{-v}, \pi)} \left(nP(\rho_{-v}, \pi) - \frac{[P(\rho_{-v}, \pi)]^2}{2} \right)}{|\Omega_L \times \Omega_R|} \end{aligned} \quad (6)$$

Next, we observe that $\hat{\mathcal{B}}_1$ and $\hat{\mathcal{B}}_2$ are necessarily disjoint, since the events in $\hat{\mathcal{B}}_1$ have v in a T1 column and events in $\hat{\mathcal{B}}_2$ have v in a T2 column. Adding equations (3) and (6),

$$\begin{aligned} \sum_t (t-1)x_t &\leq \frac{n(|\hat{\mathcal{B}}_1| + |\hat{\mathcal{B}}_2|)}{|\Omega_L \times \Omega_R|} + \\ &\frac{\sum_{\text{Column } (\rho_{-v}, \pi)} \left(nP(\rho_{-v}, \pi) - \frac{[P(\rho_{-v}, \pi)]^2}{2} \right)}{|\Omega_L \times \Omega_R|} \end{aligned}$$

Now, the sum of $P(\rho_{-v}, \pi) - \frac{[P(\rho_{-v}, \pi)]^2}{2}$ over all columns is maximized when $P(\rho_{-v}, \pi)$ is equal over all columns. This follows from the fact that the sum of squares of k numbers with fixed sum is minimized when they are all equal. Therefore, equalizing the value of $P(\rho_{-v}, \pi)$ over all columns, we arrive at:

$$\begin{aligned} \sum_t (t-1)x_t &\leq \frac{n|\hat{\mathcal{B}}|}{|\Omega_L \times \Omega_R|} + \frac{n|\mathcal{P}|}{|\Omega_L \times \Omega_R|} - \frac{|\mathcal{P}|^2}{2|\Omega_L \times \Omega_R|^2} \\ &= n\hat{B} + nP - \frac{P^2}{2} \end{aligned}$$

giving the lemma. \square

3.1 The Case with Few Perfect Matches

In this section, we will consider the case that RANKING produces very few perfect matches. In fact, there exists a class of graphs for which this property holds: Graphs with many disjoint perfect matchings. If the graph has k disjoint perfect matchings, then there exists a perfect matching so that RANKING produces at most $\frac{1}{k}$ perfect matches.

THEOREM 7. *Defining p as the aggregate probability of getting a perfect match (according to some fixed optimal matching), RANKING achieves a competitive ratio of $1 - \sqrt{p - p^2 + \frac{1}{n}}$ in the random order input model.*

PROOF. From Lemma 5, and Corollary 4 we have

$$ALG = \sum_t x_t \geq \frac{n}{2} + \hat{B} + \frac{P}{2} \quad (7)$$

Define $a = \frac{ALG}{n}$, which is the final competitive ratio and $p = \frac{P}{n}$, the aggregate probability of a perfect match. From Lemma 6 we have:

$$\sum_t \frac{t-1}{n} x_t \leq \hat{B} + P - \frac{P^2}{2n}$$

Now we minimize $\sum_t \frac{t-1}{n} x_t$ by simply top aligning all the $\sum_t x_t = ALG$, to give

$$\sum_t \frac{t-1}{n} x_t \geq \frac{ALG(ALG-1)}{2n}$$

So we have $\hat{B} \geq \frac{ALG(ALG-1)}{2n} - P + \frac{P^2}{2n}$. Substituting in Equation (7) we get:

$$\begin{aligned} ALG &\geq \frac{n}{2} + \frac{ALG(ALG-1)}{2n} - \frac{P}{2} + \frac{P^2}{2n} \\ \Rightarrow 2a &\geq 1 + a^2 - a/n - p + p^2 \\ \Rightarrow (1-a)^2 &\leq p - p^2 + a/n \\ \Rightarrow a &\geq 1 - \sqrt{p - p^2 + \frac{1}{n}} \end{aligned}$$

\square

COROLLARY 8. *If a graph has $k > 1$ disjoint perfect matchings, then RANKING achieves a competitive ratio of at least $1 - \sqrt{\frac{1}{k} - \frac{1}{k^2} + \frac{1}{n}}$*

PROOF. There exists a perfect matching \mathcal{M} such that RANKING makes at most n/k perfect matches according to \mathcal{M} . \square

3.2 The General Case

If the aggregate probability of perfect matches p equals $1/2$, then the previous result does not say much (a ratio of $1/2$). In this section we prove a more interesting bound on $\sum_t \frac{t-1}{n} x_t$ which will help us prove a bound which beats $1 - 1/e$ in general.

The following lemma gives us a sense of the distribution of x_t and would be helpful in proving the main result in this section

LEMMA 9.

$$\forall t: 1 - x_t \leq \frac{\sum_{s \leq t} x_s}{n}$$

PROOF. (Sketch) For fixed ρ and π consider the execution $\text{Ranking}(\rho, \pi)$. Suppose $v = \pi(t)$ does not get matched in this simulation. Then clearly v^* must have been matched, otherwise v and v^* would have matched each other. In particular, observe that v^* should have matched to a vertex, say u , arriving before v .

Let $\Pi = \{\pi[v \rightsquigarrow s] \mid \forall s \in [n]\}$. We claim that, for all $\tilde{\pi} \in \Pi$, v would be matched no lower than position t in $\text{Ranking}(\rho, \tilde{\pi})$. This is because if v is moved to a position below $\pi^{-1}(u)$ then v^* is already matched by the time v arrives. On the other hand, moving v to a higher position than $\pi(u)^{-1}$ only increases the options available to v^* , thus can only increase its likelihood of getting matched.

Let $\mathcal{I}_\pi^\rho(t)$ be the indicator variable for the event that $\pi(t)$ is missed in $\text{Ranking}(\rho, \pi)$. Thus from the above arguments we can define a map that takes a *miss event* at t to n matches belonging to the set of matches above position t i.e.

$$n \left[\sum_{\rho, \pi} 1 - \mathcal{I}_\pi^\rho(t) \right] \leq \sum_{s \leq t} \sum_{\rho, \pi} \mathcal{I}_\pi^\rho(s)$$

Normalizing the above equation proves the lemma. \square

We now move on to the main result of this section - a bound on $\sum_t \frac{t-1}{n} x_t$.

LEMMA 10.

$$\sum_t \frac{(t-1)x_t}{n} \geq 0.26n - o(1)$$

PROOF. From Lemma 9, we have: $\forall t: 1 - x_t \leq \frac{\sum_{s < t} x_s}{n}$.

Defining $LB_t := \frac{1 - \sum_{s < t} x_s}{(1 + \frac{1}{n})}$ and rearranging, we have

$$\forall t: x_t \geq LB_t$$

Thus, we look for the optimal solution to the following linear program:

$$\left\{ \min \sum_t \frac{tx_t}{n} \text{ s.t. } \forall t, x_t \geq LB_t \right\}$$

In the program, we have ignored the lower order term $\sum_t x_t/n$. We now define an operation which takes one feasible solution to another: If there is a t such that $x_t < 1$ and $x_t > LB_t$, and if there is an $s < t$ s.t. $x_s < 1$, then we reduce x_t and increase x_s by the same amount, equal to $\min\{x_t - LB_t, 1 - x_s\}$. This operation keeps the solution feasible:

- for $r < s$, there is no change in either x_r or in LB_r .
- for $s \leq r < t$, x_r goes up and LB_r stays the same.
- x_t drops to a value at least the original LB_t and LB_t goes down.
- for $r > t$, there is no change in either x_r or LB_r .

Furthermore, it is clear that the objective function value goes down.

Given a feasible solution $\{x_i\}$, we repeatedly apply this operation to obtain another feasible solution with lower value, until we can not apply it any more. This proves that the optimal solution has the following form: For some $t^* \in [1, n]$,

- for all $s < t, x_s = 1$

- $LB_t \leq x_t \leq 1$
- for all $s > t, x_s = LB_s$.

Thus we know the form of the solution, now we can minimize over t^* . Minimizing the resulting function gives $\sum_t \frac{tx_t}{n} \geq 0.26n$. \square

THEOREM 11. RANKING achieves a competitive ratio of at least 0.653 in the random order input model.

PROOF. This proof is a generalization of the proof of Theorem 7. From Lemma 5 and Corollary 4 we have

$$ALG \geq \frac{n}{2} + \hat{B} + \frac{P}{2} \quad (8)$$

Define $a = \frac{ALG}{n}$, which is the final competitive ratio, and define $p = \frac{P}{n}$, the probability of a perfect match. From Lemma 6 and Lemma 10 we obtain:

$$0.26n \leq \sum_t \frac{t-1}{n} x_t \leq \hat{B} + P - \frac{P^2}{2n}$$

Substituting in Equation (8) we get:

$$\begin{aligned} ALG &\geq \frac{n}{2} + 0.26n - P + \frac{P^2}{2n} + \frac{P}{2} \\ \Rightarrow a &\geq \frac{1}{2} + 0.26 - \frac{p}{2} + \frac{p^2}{2} \end{aligned} \quad (9)$$

Also, from 8 we have

$$a \geq \frac{1}{2} + \frac{p}{2} \quad (10)$$

Minimizing the maximum of the two bounds obtained in (9) and (10) we get $a \geq 0.653$ (for $p \simeq 0.3$). \square

4. UPPER BOUNDS FOR THE COMPETITIVE RATIO OF RANKING

We can prove that the competitive ratio of RANKING is at most 0.750 and 0.727 for the families of graphs depicted in Figure 1 and 2 respectively. We will prove the former result in Section 4.1. Due to space constraints, we will defer the proof of the latter - which follows similarly, but with more involved analysis - to the full version of this paper.

4.1 An Example where RANKING Achieves 0.75

Let G be the bipartite graph over n vertices whose adjacency matrix A is defined below. The graph is shown in figure 1.

$$A[i][j] = \begin{cases} 1 & \text{if } i = j \\ 1 & \text{if } i < n/2, j > n/2 \\ 0 & \text{otherwise} \end{cases}$$

The rows and columns of A represent the two partitions of vertices of G . For the purpose of the algorithm we will assume that vertices corresponding the columns of A arrive in random order, while those represented by the rows are shuffled according to a uniform random permutation. We will introduce a notion of time and assume that the vertices arrive according to a Poisson distribution with arrival rate one per unit time. This does not affect the result, it only simplifies the discussion.

For the analysis we will partition the vertices of G in to the following four sets. Let C_1 be the vertices corresponding

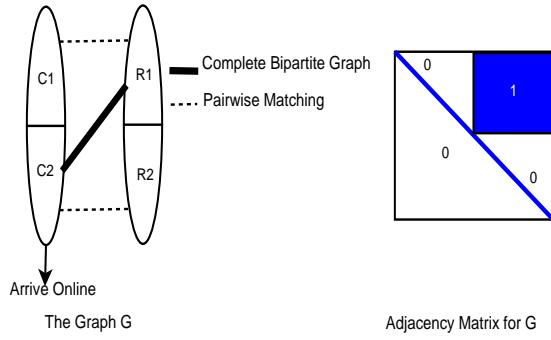


Figure 1: The 0.75 example

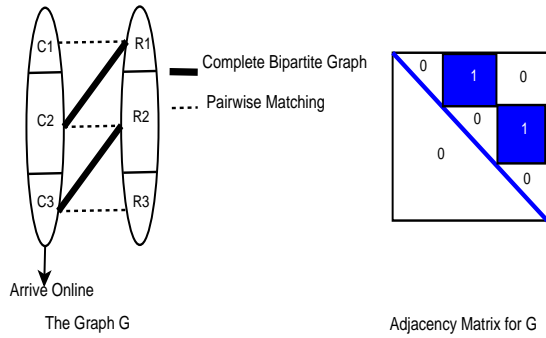


Figure 2: The 0.727 example

to columns 0 through $n/2$ and let C_2 be the vertices for rest of the columns. Similarly let R_1 be the vertices represented by rows 0 through $n/2$ and R_2 be the remaining vertices. At any time, t let $H(t)$ be the number of unmatched vertices in C_1 . This includes vertices in C_1 which are yet to arrive by time t and those vertices of C_1 that could not be matched upon arrival. Define $F(t)$ to be the number of unmatched vertices in R_1 at time t .

Note that all vertices in C_2 will surely get matched, but some vertices in C_1 may not get matched if their unique neighbor in R_1 gets matched to a vertex in C_2 . Thus the factor for the algorithm is $(|C_1| + |C_2| - E[H(n)]) / n = 1 - E[H(n)/n]$.

In Lemma 12 we represent the expected behavior of $H(t)$ in terms of $F(t)$. In Lemma 13 we find an explicit formula for the expected behavior of $F(t)$. Combining these two lemmas gives us a differential equation representing the limiting behavior of H .

For the rest of the analysis we will only consider the vertices that arrive in the interval $[0, n - n^{0.9}]$. This is done to make the analysis amenable to concentration bounds. Furthermore, ignoring the last $n^{0.9}$ vertices only affects the competitive ratio by a $o(1)$ factor.

LEMMA 12. For all $t \in [0, n - n^{0.9}]$,

$$E[H(t+1) - H(t)] \geq -\frac{F(t)}{n-t} - o(1/n)$$

PROOF. Let v_{t+1} be the vertex arriving at time $t+1$. H reduces by 1 at time $t+1$ if v_{t+1} belongs to C_1 and its unique neighbor in R_1 is still unmatched. Let $Q(t)$ be the

number of vertices in C_1 that are yet to arrive by time t . By Chernoff bounds $Q(t)$ is tightly concentrated around its expected value of $\frac{n-t}{2}$ i.e. $Q(t)$ is less than $\frac{(n-t)}{2} - 2\sqrt{n \log n}$ with probability at most $1/n^2$. Of the $Q(t)$ vertices in C_1 that are yet to arrive, $F(t)$ are such that their unique neighbor in R_1 is still unmatched. Hence,

$$\begin{aligned} E[H(t+1) - H(t)] &= \\ &= -\Pr[v_{t+1} \in C_1, N(v_{t+1}) \text{ is still unmatched}] \\ &= -\frac{1}{2} \Pr[N(v_{t+1}) \text{ is still unmatched} | v_{t+1} \in C_1] \\ &= -\frac{1}{2} \frac{F(t)}{Q(t)} \\ &\geq -\frac{1}{2} \frac{F(t)}{(n-t)/2 - 2\sqrt{n \log n}} \\ &\quad + \Pr\left[Q(t) < \frac{n-t}{2} - 2\sqrt{n \log n}\right] \\ &\geq -\frac{F(t)}{n-t} - o(1/n) \end{aligned}$$

□

LEMMA 13. For all $t \in [0, n - n^{0.9}]$,

$$E[F(t+1) - F(t)] \leq -2\frac{F(t)}{n-t} + o(1/n)$$

PROOF. Let v_{t+1} be the vertex arriving at time $t+1$. If v_{t+1} belongs to C_1 then F reduces by 1 iff H falls by 1 at time $t+1$. By the proof of Lemma 12 this happens with probability at most $\frac{F(t)}{n-t} + o(1/n)$. If v_{t+1} belongs to C_2 then F reduces by 1 only if v_{t+1} is matched to a vertex in R_1 . Next we analyze the probability of this event.

Consider $u^* \in R_2$ and let u be its unique neighbor in C_2 . The probability that u is yet to arrive by time t is $1 - t/n$. Recall that the RANKING algorithm randomly permutes the vertices in $R_1 \cup R_2$ and each arriving vertex chooses the highest unmatched vertex in its neighborhood. If u is to match a vertex in R_1 , u^* must lie below the highest unmatched vertex in R_1 . This happens with probability at least $2F(t)/n$. Combining the above two statements, we get $\Pr[v_{t+1} \text{ does not match } v_{t+1}^* | v_{t+1} \in C_2] \geq 2F(t)/(n-t)$. Thus we have,

$$\begin{aligned} E[F(t+1) - F(t)] &= \Pr[v_{t+1} \text{ matches } v_{t+1}^* | v_{t+1} \in C_1] \\ &\quad - \Pr[v_{t+1} \text{ doesn't match } v_{t+1}^* | v_{t+1} \in C_2] \\ &\leq -\frac{F(t)}{n-t} - \Pr[v_{t+1} \text{ doesn't match } v_{t+1}^* | v_{t+1} \in C_2] \\ &\leq -\frac{F(t)}{n-t} - \frac{1}{2} \frac{2F(t)}{n-t} + o(1/n) \\ &= -\frac{2F(t)}{n-t} + o(1/n) \end{aligned}$$

□

The statement of Lemma 13 becomes clearer if we scale the time by a factor of n , i.e we let t be the time when exactly nt vertices have arrived and let $F(t)$ and $H(t)$ be the fraction of unmatched vertices in R_1 and C_1 respectively.

Lemma 13 can now be stated as,

$$\frac{E[F(t+1/n) - F(t)]}{1/n} = -\frac{2F(t)}{1-t/n} + o(1/n) \quad (11)$$

We claim the random process described by the above equation is well approximated by the trajectory of the differential equation

$$\frac{df}{dt} \leq -\frac{2f}{n-t} + o(1/n) \quad (12)$$

where this equation has been obtained from equation (11) by replacing the right-hand side with the appropriate limiting value as n tends to infinity, dy/dt . This follows easily from known techniques, such as, for example, Kurtz's theorem (Refer [Kur70]).

To clarify the connection we state a version of Kurtz's theorem.

THEOREM 14 (KURTZ'S THEOREM [KUR70]). *Suppose we are given a finite set of vectors $\{e_1, e_2 \dots e_k\}$ in R^d . We consider an initial process $\vec{x}(t)$ with generator*

$$Lf(\vec{x}) = \sum_{i=1}^k \lambda_i(\vec{x})(f(\vec{x} + \vec{e}_i) - f(\vec{x}))$$

and a scaled process $\vec{z}_n(t)$ with generator

$$L_n f(\vec{x}) = \sum_{i=1}^k n \lambda_i(\vec{x})(f(\vec{x} + \frac{\vec{e}_i}{n}) - f(\vec{x}))$$

The limiting operator L_∞ satisfies

$$L_\infty f(\vec{x}) = \sum_{i=1}^k n \lambda_i(\vec{x}) \langle \nabla f(\vec{x}), \vec{e}_i \rangle$$

and corresponds to the deterministic solution \vec{z}_∞ of the equation

$$\frac{d}{dt} \vec{z}_\infty(t) = \sum_{i=1}^k \lambda_i(\vec{z}_\infty(t)) \vec{e}_i \quad (13)$$

Let $\lambda_i(\vec{x}) : R^d \rightarrow R^+$ be uniformly bounded and Lipschitz continuous, and let \vec{z}_∞ be the unique solution of equation (13) with $\vec{z}_\infty(0) = \vec{x}(0)$. For each finite T there exist a positive constant α_1 and a function α_2 with

$$\lim_{\epsilon \downarrow 0} \frac{\alpha_2(\epsilon)}{\epsilon^2} \in (0, \infty) \text{ and } \lim_{\epsilon \uparrow \infty} \frac{\alpha_2(\epsilon)}{\epsilon} = \infty$$

such that, for all $n \geq 1$ and $\epsilon > 0$,

$$\Pr \left[\sup_{0 \leq t \leq T} |\vec{z}_n(t) - \vec{z}_\infty(t)| \geq \epsilon \right] \leq \alpha_1 e^{-n \alpha_2(\epsilon)}$$

Moreover, k_1 and k_2 can be chosen independently of \vec{x} .

By applying Kurtz's theorem, we see that as n goes to infinity, the limiting process for equation (11) is given by the differential equation (12).

Solving equation (12) we get $F(t) \leq \frac{(n-t)^2}{2n} - o(1)$. By a similar argument and substituting in equation (14) the limiting process representing the evolution of H is given by the following differential equation,

$$\frac{dh}{dt} \geq -\frac{n-t}{2n} + o(1/n) \quad (14)$$

Solving equation (14), we get $H(n) \geq n/4 - o(1)$. Thus the RANKING algorithm achieves a competitive ratio of at most $0.75 + o(1)$ for the above graph.

5. REFERENCES

- [AGKM11] Gagan Aggarwal, Gagan Goel, Chinmay Karande, and Aranyak Mehta. Online vertex-weighted bipartite matching and single-bid budgeted allocations. *SODA*, 2011.
- [BJN07] N. Buchbinder, K. Jain, and J.S. Naor. Online Primal-Dual Algorithms for Maximizing Ad-Auctions Revenue. In *Algorithms-ESA 2007 15th Annual European Symposium, Eilat, Israel, October 8-10, 2007: Proceedings*, page 253. Springer, 2007.
- [BK10] Bahman Bahmani and Michael Kapralov. Improved bounds for online stochastic matching. *ESA*, 2010.
- [BM08] B. Birnbaum and C. Mathieu. On-line bipartite matching made simple. 2008.
- [DH09] N. Devanur and T Hayes. The adwords problem: Online keyword matching with budgeted bidders under random permutations. In *ACM Conference on Electronic Commerce*, 2009.
- [FMMM09] Jon Feldman, Aranyak Mehta, Vahab S. Mirrokni, and S. Muthukrishnan. Online stochastic matching: Beating $1-1/e$. In *FOCS*, pages 117–126, 2009.
- [GM08] Gagan Goel and Aranyak Mehta. Online budgeted matching in random input models with applications to adwords. In *SODA*, pages 982–991, 2008.
- [KP09] Nitish Korula and Martin Pál. Algorithms for secretary problems on graphs and hypergraphs. In *Proceedings of the 36th International Colloquium on Automata, Languages and Programming: Part II, ICALP '09*, pages 508–520, Berlin, Heidelberg, 2009. Springer-Verlag.
- [Kur70] Thomas G. Kurtz. Solutions of ordinary differential equations as limits of pure jump markovprocesses. *Journal of Applied Probability*, 7:49–58, 1970.
- [KVV90] R.M. Karp, U.V. Vazirani, and V.V. Vazirani. An optimal algorithm for online bipartite matching. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing*, 1990.
- [MNS07] Mohammad Mahdian, Hamid Nazerzadeh, and Amin Saberi. Allocating online advertisement space with unreliable estimates. In *ACM Conference on Electronic Commerce*, pages 288–294, 2007.
- [MOGS11] Vahideh H. Manshadi, Shayan Oveis-Gharan, and Amin Saberi. Online stochastic matching: Online actions based on offline statistics. In *SODA*, 2011.
- [MSVV05] Aranyak Mehta, Amin Saberi, Umesh Vazirani, and Vijay Vazirani. Adwords and generalized online matching. In *FOCS*, 2005.
- [MY11] Mohammad Mahdian and Qiqi Yan. Online bipartite matching with random arrivals: A strongly factor-revealing lp approach. In *STOC*, 2011.