On-line Bipartite Matching Made Simple

Benjamin Birnbaum* Claire Mathieu†

Abstract

We examine the classic on-line bipartite matching problem studied by Karp, Vazirani, and Vazirani [8] and provide a simple proof of their result that the Ranking algorithm for this problem achieves a competitive ratio of $1 - 1/e$.

Introduction

Introduced in 1990 by Karp, Vazirani, and Vazirani [8], on-line bipartite matching was one of the first problems to receive the attention of competitive analysis. The input to the problem is a bipartite graph $G = (U, V, E)$, in which the vertices in U arrive in an on-line fashion and the edges incident to each vertex $u \in U$ are revealed when u arrives. When this happens, the algorithm may match u to a previously unmatched adjacent vertex in V, if there is one. Such a decision, once made, is irrevocable. The objective is to maximize the size of the resulting matching.

In their paper, Karp et al. first show that the competitiveness of the problem is trivial in the deterministic case. Any algorithm that always matches a vertex in U if a match is possible constructs a maximal matching, and therefore such an algorithm has a competitive ratio of $1/2$. On the other hand, given any deterministic algorithm, it is easy to construct an input (in which each vertex $u \in U$ has one or two neighbors in V) that forces that algorithm to find a matching of size no greater than half of the optimum.

The competitiveness of the problem in the randomized case is more interesting. The authors show that the naive randomized algorithm, which flips a coin each time a vertex $u \in U$ arrives and chooses a random match for u based on this coin flip, does no better than the deterministic algorithms. However, an equally simple, but subtly different, algorithm does significantly better, achieving a competitive ratio of $1 - 1/e \approx 0.63$. This algorithm, called Ranking, initially chooses a single random ranking on the vertices in V that is used to choose matches throughout the entire run of the algorithm, in a way described more precisely below.

Ranking Algorithm:
Initialization:
Choose a random permutation (ranking) σ of the vertices of V.
Online Matching:
Upon arrival of vertex u of U:
Let $N(u)$ be the set of neighbors of u that have not been matched yet.
If $N(u) \neq \emptyset$, match u to the vertex $v \in N(u)$ that minimizes $\sigma(v)$.

*Department of Computer Science, University of Washington. Email: birnbaum@cs.washington.edu.
†Computer Science Department, Brown University. Email: claire@cs.brown.edu.

The paper concludes by showing that Ranking's competitive ratio of $1 - 1/e$ is the best possible (asymptotically) for any randomized algorithm for the problem.

Since the publication of this initial paper, on-line bipartite matching has received considerable attention. For example, Azar, Naor, and Rom [3] use the idea of the Ranking algorithm to obtain an optimally-competitive randomized algorithm for an on-line assignment problem. As in on-line bipartite matching, the input to this problem is a bipartite graph $G = (U, V, E)$ in which the vertices in U arrive on-line. However, unlike the matching problem, every vertex in U must be assigned to a vertex in V, and the goal is to minimize the maximum load on a vertex in V. The authors provide the following algorithm for this problem: For $1 \leq j \leq |U|$, choose a separate random ranking σ_j of the vertices in V. When a vertex $u \in U$ arrives, let j be the lowest load of a vertex in V that is adjacent to u. Out of the vertices adjacent to u that have load j, choose the one that minimizes σ_{j+1}. They then use the result that Ranking achieves a competitive ratio of $1 - 1/e$ to show that their algorithm has a competitive ratio of $1 + \ln(|U|)$, a result which is optimal up to an additive constant of 1.

In a more recent paper, Azar and Richter [4] make use of the Ranking result in their analysis of a switch routing problem. In the "unit-version" of this problem, a network switch consists of several input FIFO queues. At each time step, some number of packets arrive on-line, each destined for one of the input queues. A packet is lost if the queue to which it is sent is full. After each time that packets arrive, the algorithm must choose one of the input queues to transmit one packet. The authors provide a randomized algorithm for this problem that can be shown to be equivalent to Ranking on a bipartite graph, thereby proving that their algorithm achieves a competitive ratio of $1 - 1/e$.

Azar and Chaiutin [2] reduce another switch routing problem to on-line bipartite matching in order to apply its results to their problem. In their model, the switch contains some number of output ports, each of which has a buffer for packets. At each time step, some number of packets arrive and some set of the output buffers transmit a packet. (Both events are controlled by the adversary). The algorithm must match each incoming packet to an output port that is not full, and the goal is to maximize the number of packets transmitted.

Other papers, though not using the result of Karp et al. directly, examine variants of the on-line bipartite matching problem. In the work of Agarwal and Puri [1], for example, the vertices on one side of the graph represent requests to send files to mobile nodes before some deadline, and the base station (algorithm) must decide on-line, for each mobile node and each time step, which file to send. In the special case where the price per unit of data transmitted is uniform and each channel to a mobile node has unit capacity, this reduces to on-line bipartite matching.

As another example, Blum, Sandholm, and Zinkevich [5] study a variant that arises in the context of on-line market clearing. In their model, two types of bids for a commodity (*buy* and *sell*) arrive on-line and are active for some amount of time. One of the objectives they study, that of maximizing the number of buy and sell bids matched, reduces to a graph matching problem whose main difference from on-line bipartite matching is that both sets of vertices arrive on-line.

More recently, Mehta et al. [11] illustrate a connection between on-line matching and sponsored search auctions, the way search engines like Google choose which advertisements to show for a search. In such an auction, companies, called *bidders*, bid on search terms, called *keywords*, in which they are interested. The search engine, which cannot exceed the declared budgets of the bidders, chooses which advertisements to display and how much to charge the bidders based on their bids. The authors of this paper show that the problem of maximizing the search engine's revenue in a

sponsored search auction can be modeled as a generalization of on-line bipartite matching. In this generalization, the vertices in U represent keywords and the vertices in V represent bidders. The keywords arrive on-line, and the bids from each bidder for a keyword $u \in U$ are revealed once u arrives. The algorithm then decides, irrevocably, which bidder to display for that keyword. At the end of the algorithm, the revenue of the search engine from each bidder is the minimum of its budget and the sum of its bids for the keywords for which it was chosen. It is not hard to see that this problem, called the Adwords problem, reduces to on-line bipartite matching when every bid is either 1 or 0 and the budget of every bidder is 1.

In their paper, Mehta et al. provide a deterministic algorithm achieving a competitive ratio of $1 - 1/e$ when the ratio of the maximum bid to the smallest budget approaches 0, an alternative proof of which is given by Buchbinder, Jain, and Naor [6]. Variants of this problem are also studied by Goel and Mehta [7] and Mahdian, Nazerzadeh, and Saberi [10]. Currently, the best competitive ratio for the Adwords problem with large bids (up to size of the budgets) is 1/2, achieved in a greedy deterministic algorithm from Lehmann, Lehmann, and Nisan [9] (for a generalization of the problem).

Beating 1/2 for the Adwords problem with large bids, or showing that it is impossible to do so, is an interesting, and so far unsolved problem. By the lower-bound for on-line bipartite matching, an algorithm for the Adwords problem with a competitive ratio better than 1/2 must be randomized. Therefore, to find such an algorithm, it is important to understand the proof of the result on the competitive ratio of Ranking, stated as Theorem 1 below. Towards this goal, we try in this paper to give an intuitive proof of this result.

Surprisingly, a mistake in a lemma of the original 1990 paper was discovered by Krohn and Varadarajan seventeen years after it first appeared. Very recently, Goel and Mehta [7] corrected this mistake and in the process provided a detailed proof of Theorem 1 that is simpler than the original proof. The proof presented here is a more concise alternative analysis.

Theorem 1 ([7, 8])**.** *The Ranking algorithm has competitive ratio at least* $1 - (1 - 1/(n + 1))^n$*, which is asymptotically* $1 - 1/e$*.*

A Simple Proof of the Theorem

Let Ranking(G, π, σ) denote the matching constructed on input G for arrival order π, when the ranking is σ. The following Lemma is an easy structural observation, and the argument is used in [7] and [8]. (See Figure 1 for an illustration.)

Lemma 2. *Let x be a vertex, $H = G \setminus \{x\}$, and π_H and σ_H be the orderings of U_H and V_H induced by π and σ respectively. If the matchings Ranking(H, π_H, σ_H) and Ranking(G, π, σ) are not identical, then they differ by a single alternating path starting at vertex x.*

Thus, for every σ, Ranking(G, π, σ) has at least as many edges as Ranking(H, π_H, σ_H). Applying this repeatedly to remove vertices which are not in the maximum matching, it follows that the competitive ratio is determined by graphs that have a perfect matching. Fix a graph G and an arrival order π. Henceforth we assume that G has a perfect matching $m^* : U \to V$, and let $n = |U|$. To simplify notation, we write Ranking(σ) to mean Ranking(G, π, σ).

Since the Ranking algorithm constructs a maximal matching, it has at least half as many edges as the maximum matching. The well-known one-line proof of this argues that for every edge $\{u, v\}$

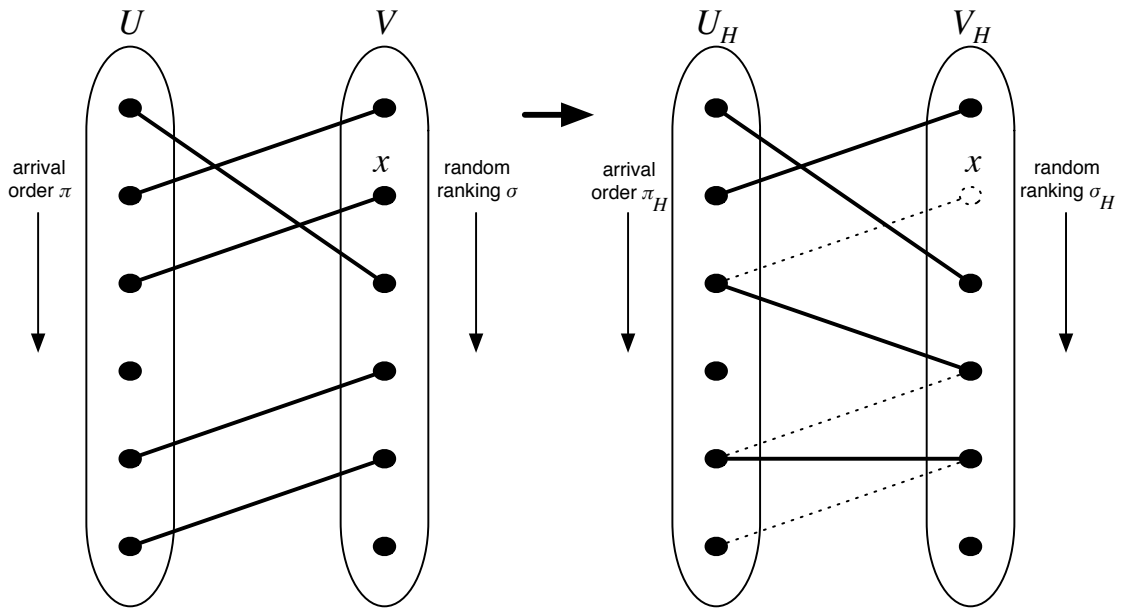

Figure 1: Illustration of Lemma 2. On the left is the original matching constructed by the algorithm, Ranking(G, π, σ), and on the right is the matching constructed by the algorithm after vertex x is removed, Ranking(H, π_H, σ_H) (with the edges from Ranking(G, π, σ) that are not in Ranking(H, π_H, σ_H) shown as dashed lines). The two matchings differ by at most a single alternating path.

of the maximum matching, if v is not matched by Ranking then u must be matched by Ranking. The following Lemma refines this structural observation slightly by focusing on the ranks of the matched vertices. The arguments of this Lemma, as well as the next, are used in [7] and [8].

Lemma 3. *Fix $u \in U$ and let $v = m^*(u)$. If v is not matched by Ranking(σ), then u is matched by Ranking(σ), to a vertex v' whose rank $\sigma(v')$ is less than $t = \sigma(v)$.*

Proof. If v is not matched by Ranking(σ), then, when u arrives, it has some eligible neighbors since v is one; so u gets matched to the eligible neighbor v' with minimum rank. □

With the help of Lemma 3, we can give an intuitive but incorrect "proof" of Lemma 5 below, from which the Theorem easily follows. Here is a slightly technical variant that yields a less intuitive but correct proof of Lemma 5.

Lemma 4. *Let $u \in U$ and $v = m^*(u)$. Let σ' be a permutation[1] and let σ_i be the permutation obtained from σ' by removing vertex v and putting it back in so that its rank is i. If v is not matched by Ranking(σ'), then, for every i, u is matched by Ranking(σ_i) to a vertex v_i whose rank $\sigma_i(v_i)$ is at most $t = \sigma'(v)$.*

Proof. Let $m =$ Ranking(σ') and $m_i =$ Ranking(σ_i). As in Lemma 2, matchings m and m_i, if not identical, differ along a single alternating path starting at vertex v with an edge of m_i. Moreover, it

[1]The reason for the change of notation from σ to σ' will become clear in the proof of Lemma 5.

is easy to see that the path is monotone: the vertices of V traversed have increasing rank in σ_i. So, m_i matches u to a vertex v_i whose rank is $\sigma_i(v_i) \leq \sigma_i(v')$, where $v' = m(u)$. (See Figure 2 for an illustration.) By definition of σ_i, $|\sigma_i(v') - \sigma'(v')| \leq 1$. By Lemma 3, $\sigma'(v') < t$. Thus $\sigma_i(v_i) < 1 + t$, and by integrality $\sigma_i(v_i) \leq t$. □

Lemma 5. *Let x_t denote the probability over σ that the vertex of V of rank t is matched by the algorithm. Then $1 - x_t \leq (1/n) \sum_{1 \leq s \leq t} x_s$.*

Before doing the proof, let us see how, equipped with this Lemma, it is now easy to complete the proof of Theorem 1. Since G has a perfect matching, the competitive ratio is the infimum of $(1/n) \sum_{1 \leq s \leq n} x_s$. We rewrite Lemma 5 as $S_t(1 + 1/n) \geq 1 + S_{t-1}$, where $S_t = \sum_{1 \leq s \leq t} x_s$. It is easy to see that the infimum occurs when all inequalities are tight equalities. This yields $S_t = \sum_{s=1}^{t} (1 - 1/(n+1))^s$ for all t. Hence, the competitive ratio is at least $(1/n) \sum_{s=1}^{n} (1 - 1/(n+1))^s = 1 - (1 - 1/(n+1))^n$, which approaches $1 - 1/e$ as n approaches infinity.

We now conclude the proof of Theorem 1 with a proof of Lemma 5.

Proof of Lemma 5. We first give an intuitive but incorrect "proof" based on Lemma 3. Let v denote the vertex of V whose rank is t. The probability that v is not matched by the algorithm is $1 - x_t$. Let u be such that $v = m^*(u)$. Let $R_{t-1} \subset U$ denote the set of vertices of U matched by the algorithm to the vertices of V who have rank less than or equal to $t - 1$. The expected cardinality of R_{t-1} is $\sum_{1 \leq s \leq t-1} x_s$. By Lemma 3, if v is not matched by the algorithm then u is an

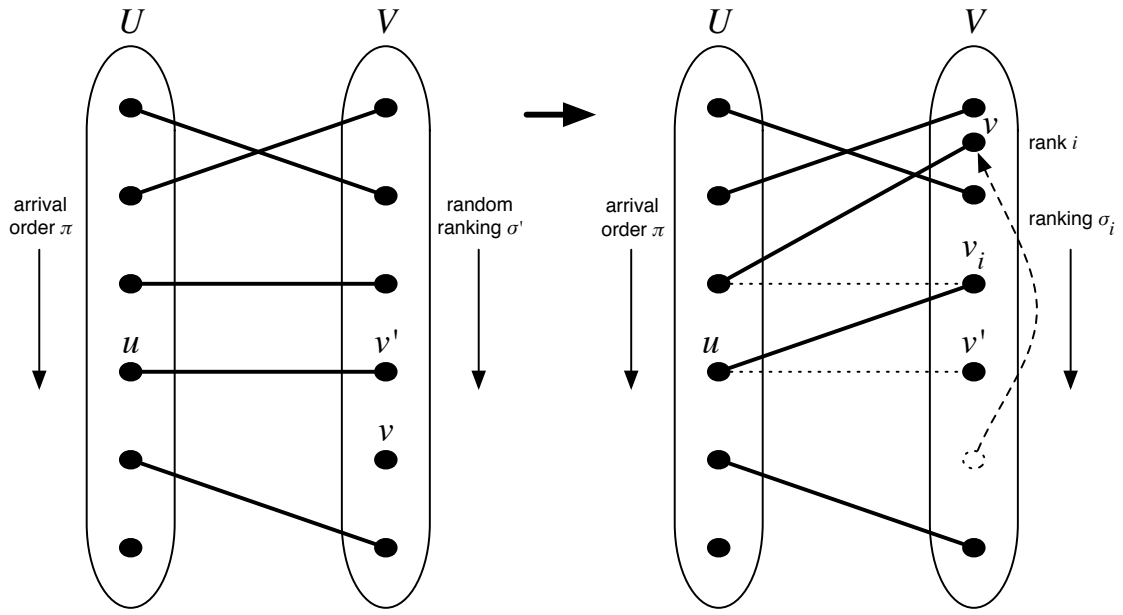

Figure 2: Illustration of Lemma 4. On the left is the matching m and on the right is the matching m_i (with the edges from m that are not in m_i shown as dashed lines.) The two matchings can differ by at most one alternating path starting at v, and this alternating path is monotone. Hence, in m_i, vertex u must be matched to a vertex whose rank is no greater than the rank of v' in m_i.

element of R_{t-1}. If only u and R_{t-1} were independent, we would write that, conditional on R_{t-1}, the event that $u \in R_{t-1}$ would have probability $|R_{t-1}|/n$ for a random $u \in U$, and this would imply $1 - x_t \leq (1/n) \sum_{1 \leq s \leq t-1} x_s$. Unfortunately, u and R_{t-1} are not independent.

Here is a corrected argument based on Lemma 4. Given a random permutation σ, define a new random permutation σ' obtained by choosing a vertex v of V uniformly at random, taking it out of σ, and moving it back in so that its rank is t. Consider the matching Ranking(σ'). Let u be such that $v = m^*(u)$. By Lemma 4 (applied for i defined appropriately so that $\sigma_i = \sigma$), if v is not matched by Ranking(σ') (as before, this event has probability $1 - x_t$), then u is matched by Ranking(σ) to a vertex $\tilde{v}$ such that $\sigma(\tilde{v}) \leq t$, or in other words, $u \in R_t$. Now, u is independent of σ, hence independent of R_t. So, conditional on σ, the event that $u \in R_t$ has probability $|R_t|/n$. Taking expectations over σ, this equals $(1/n) \sum_{1 \leq s \leq t} x_s$, hence the proof. □

Acknowledgements

The authors wish to thank Anna Karlin, under whose direction Ben Birnbaum started this project, Yossi Azar, for insightful discussions, and Elisa Celis and Alex Jaffe, for comments on an early version of this paper.

References

[1] M. Agarwal and A. Puri. Base station scheduling of requests with fixed deadlines. In *INFOCOM 2002: twenty-first annual joint conference of the IEEE Communications Societies*, pages 487–496. IEEE, 2002.

[2] Yossi Azar and Yoel Chaiutin. Optimal node routing. In *Proceedings of STACS '06 (Lecture Notes in Computer Science 3884*, pages 596–607. Springer, 2006.

[3] Yossi Azar, Joseph (Seffi) Naor, and Raphael Rom. The competitiveness of on-line assignments. In *SODA '92: Proceedings of the third annual ACM-SIAM symposium on discrete algorithms*, pages 203–210, Philadelphia, PA, USA, 1992. Society for Industrial and Applied Mathematics.

[4] Yossi Azar and Yossi Richter. Management of multi-queue switches in QoS networks. *Algorithmica*, 43(1-2):81–96, 2005.

[5] Avrim Blum, Tuomas Sandholm, and Martin Zinkevich. Online algorithms for market clearing. *J. ACM*, 53(5):845–879, 2006.

[6] Niv Buchbinder, Kamal Jain, and Joseph (Seffi) Naor. Online primal-dual algorithms for maximizing ad-auctions revenue. In *Proceedings of ESA '07 (Lecture Notes in Computer Science 4698)*, pages 253–264. Springer, 2007.

[7] Gagan Goel and Aranyak Mehta. Online budgeted matching in random input models with applications to adwords. In *SODA '08: Proceedings of the nineteenth annual ACM-SIAM symposium on discrete algorithms*, pages 982–991, Philadelphia, PA, USA, 2008. Society for Industrial and Applied Mathematics.

[8] R. M. Karp, U. V. Vazirani, and V. V. Vazirani. An optimal algorithm for on-line bipartite matching. In *STOC '90: Proceedings of the twenty-second annual ACM symposium on Theory of computing*, pages 352–358, New York, NY, USA, 1990. ACM Press.

[9] Benny Lehmann, Daniel Lehmann, and Noam Nisan. Combinatorial auctions with decreasing marginal utilities. In *EC '01: Proceedings of the 3rd ACM conference on electronic commerce*, pages 18–28, New York, NY, USA, 2001. ACM.

[10] Mohammad Mahdian, Hamid Nazerzadeh, and Amin Saberi. Allocating online advertisement space with unreliable estimates. In *EC '07: Proceedings of the 8th ACM conference on electronic commerce*, pages 288–294, New York, NY, USA, 2007. ACM.

[11] Aranyak Mehta, Amin Saberi, Umesh Vazirani, and Vijay Vazirani. Adwords and generalized online matching. *J. ACM*, 54(5):22, 2007.