On-line Algorithms for Weighted Bipartite Matching and Stable Marriages [*]

Samir Khuller [†]
Dept. of Computer Science and
Institute for Advanced Computer Studies
University of Maryland
College Park, MD 20742

Stephen G. Mitchell
Dept. of Electrical Engineering
Cornell University
Ithaca, NY 14853

Vijay V. Vazirani [‡]
Dept. of Computer Science & Engg.
Indian Institute of Technology
New Delhi 110016, India

Abstract

We give an on-line deterministic algorithm for the weighted bipartite matching problem that achieves a competitive ratio of $(2n-1)$ in any metric space (where n is the number of vertices). This algorithm is optimal – there is no on-line deterministic algorithm that achieves a competitive ratio better than $(2n-1)$ in all metric spaces.

We also study the stable marriage problem, where we are interested in the number of unstable pairs produced. We show that the simple "first come, first served" deterministic algorithm yields on the average $O(n \log n)$ unstable pairs, but in the worst case no deterministic or randomized on-line algorithm can do better than $\Omega(n^2)$ unstable pairs. This appears to be the first on-line problem for which provably one cannot do better with randomization; for most on-line problems studied in the past, randomization has helped in improving the performance.

Keywords: on-line algorithms, weighted matching, stable marriage problem.

1. Introduction

There has been a great deal of interest recently in studying the relative power of on-line *vs.* off-line algorithms [ST, MMS, PY]. On-line algorithms attempt to model a real life situation,

[*]Supported by NSF grant DCR 85-52938 and PYI matching funds from AT&T Bell Labs and Sun Microsystems, Inc.

[†]Currently supported by NSF Research Initiation Award CCR-9307462. Part of this research was done while this author was supported by an IBM Graduate Fellowship at Cornell University.

[‡]Part of this research was done while this author was at Cornell University.

where the entire input is not available beforehand. The input is obtained incrementally, and the algorithm has to make (irrevocable) decisions to respond to the input. Usually, the question of interest is: how "good" a solution can we obtain, given the fact that each part of the solution is obtained without *a priori* knowledge about the entire input.

Typically, the on-line algorithm is compared to an optimal off-line algorithm that knows the entire request sequence in advance. The *competitiveness* of an on-line algorithm is the ratio of its performance to the performance of an optimal off-line algorithm.

An optimal randomized on-line algorithm for bipartite matching (without weights) was given in [KVV]. In this paper we study two related problems — on-line weighted bipartite matching, and on-line stable marriages.

The Weighted Bipartite Matching Problem

The weighted bipartite matching problem occupies a central place in combinatorial optimization, and has a variety of applications to transshipment problems. The problem is formally defined as follows: obtain a minimum weight perfect matching in an edge-weighted bipartite graph. (This problem is also known as the assignment problem.) We are able to give an optimal deterministic algorithm for on-line weighted bipartite matching, where the weights satisfy triangle inequality.

On-line model: We assume that the graph is a weighted complete bipartite graph; the woman vertices are all given in advance, and the men arrive one at a time (revealing weights on the edges to each of the women). The men have to be matched off immediately; once a man has been matched, we are not permitted to change his mate. The objective is to obtain a perfect matching of "small" total weight (compared to the minimum weight perfect matching). The competitiveness in our case is the ratio of the weight of the on-line matching to the weight of the minimum weight perfect matching.

Without the triangle inequality imposed on the weights any (deterministic or randomized) on-line algorithm can have a very bad competitiveness ratio, that exceeds any function of n.

Results: We give a deterministic algorithm that is $(2n-1)$-competitive if the edge weights are distances in some metric on the set of vertices. This algorithm is optimal — any deterministic algorithm for on-line weighted bipartite matching (with triangle inequality) must perform as badly as $(2n-1)$ in arbitrary metric spaces.

An interesting feature of our algorithm is that as each man arrives, it computes the best off-line maximum matching among the women and the currently arrived men to determine the woman vertex to match with the newly arrived man vertex. The $(2n-1)$ bound is established by using the notion of augmenting paths.

The Stable Marriage Problem

The other related problem that we study is the stable marriage problem. The stable marriage problem has been studied extensively for its rich structure [GI], and is useful in several situations, e.g., assigning interns to hospitals, students to colleges, *etc.* The stable marriage problem is defined as follows: There are n men and n women. Each person provides a preference list that ranks the members of the opposite sex. Given a pairing of men and women, we say that a man and a woman form an *unstable pair* if they are not married, but prefer each other to their current partners. A matching is *unstable* if there is at least one unstable pair, and *stable* otherwise.

On-line model: We study a natural on-line model in which we assume that all the women's preference lists are known in advance, and the men's preference lists are revealed one at a time. Each man has to be married off as soon as his preference list is revealed. This models the situation in which the hospitals rate the interns in advance, and the interns arrive one at a time with their ratings of the hospitals.

How can one measure how "good" the obtained solution is? One way (that we consider) is to actually count the number of unstable pairs in any given marriage. A marriage is considered particularly "bad", in case *many* men prefer *many* women who also prefer them to their current partners. These are all unstable pairs. Clearly, in a stable marriage the number of unstable pairs is zero, so we cannot measure the ratio of the on-line solution to the off-line solution. Our goal is to obtain an on-line algorithm with a "small" number of unstable pairs.

We would like to note that there are many possible ways of measuring how "good" an obtained on-line solution is. For example, we could measure the number of divorces that need to be done to make the marriage a stable marriage. This measure would make sense in a situation where we actually had the opportunity to modify a solution obtained by an on-line algorithm. For our purposes, we have chosen to simply count the actual number of unstable pairs as a measure of the "quality" of the on-line solution.

Results: Our results for the stable marriage problem are of a more pessimistic nature. We show that the simple "first come, first served" deterministic algorithm yields on the average $O(n \log n)$ unstable pairs, but in the worst case no deterministic algorithm can do better than $\Omega(n^2)$ unstable pairs. (An unstable pair is a man-woman pair that are not married to each other, but prefer each other to their current partners.) We also show that no randomized on-line algorithm can do better than $\Omega(n^2)$ unstable pairs. The proof for the randomized lower bound (see Theorem 3.6 in Section 3.2) makes use of Yao's lemma [Y]. This is the first on-line problem we know of where *provably* one cannot do better with randomization.

Related work
A $(2n-1)$ competitive algorithm for the weighted matching problem was independently obtained by [KP] as well.

2. On-line Weighted Bipartite Matching

We describe an on-line algorithm for minimum weight perfect matchings in bipartite graphs, where the edge weights obey the triangle inequality. We give an algorithm that produces a perfect matching with a weight at most $2n-1$ times the weight of the minimum weight perfect matching. We show that this is optimal for graphs obeying the triangle inequality[1].

Let $G = (R, B, E)$ be a complete bipartite graph, where R is a set of n red vertices and B is a set of n blue vertices. Suppose there is a metric d on $R \times B$ (satisfying the triangle inequality); for each red vertex r_i and each blue vertex b_j let the weight of edge (r_i, b_j) be $w_{ij} = d(r_i, b_j) > 0$. Initially, the blue vertices are known; as each red vertex r_i arrives (with its edge weights), the algorithm matches r_i to one of the available blue vertices. Once matched, a pair cannot later be separated. The algorithm tries to produce the smallest weight perfect matching possible.

[1]By this we mean that the weight of any edge in the graph is at most the weight of any path between the two endpoints.

If the weights do not obey the triangle inequality, no performance bound (for a deterministic algorithm) depending only on n is possible (see Fig. 1). To see this, consider the following small example. There are two blue vertices, b_1 and b_2. The first red vertex r_1 has edges to b_1 and b_2, each of weight 1. Let us assume that r_1 matches with b_2. Now r_2 comes with edges to b_1 and b_2 of weights W and 1, respectively. The only choice is to match r_2 with b_1. The weight of the matching produced is $W+1$, whereas the optimal matching is of weight 2 (matching b_i with r_i). The performance ratio can be made arbitrarily bad by taking W arbitrarily large.

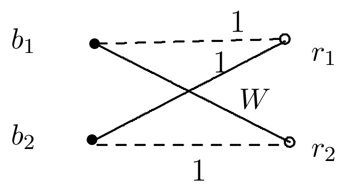

Figure 1: Example to show that no ratio possible without triangle inequality

Remark: The same holds when one is trying to design a randomized on-line algorithm. Using a theorem of Yao (Theorem 3.5, below), it is easy to construct a similar example showing that without the triangle inequality one cannot hope to get reasonable performance bounds for a randomized on-line algorithm.

The algorithm for graphs obeying the triangle inequality is quite simple, but not immediately intuitive. The more intuitive greedy method (which matches a new red vertex to the closest available blue vertex) does extremely badly in the worst case. To see this, consider the following example in the real line R^1 (see Fig. 2). The blue points (vertices) are placed as follows: b_i is placed at location 2^{i-1}, for $1 \leq i \leq n-1$; we place b_n at -0.1. We now locate r_1 at 0.5. Since the closest available blue vertex is b_1 (at 1), we match r_1 to b_1. Now, for $2 \leq i \leq n$ locate r_i at b_{i-1}. Since b_i is the closest available blue vertex, r_i is matched to b_i. Finally, we must match r_n to b_n. The weight of the optimal off-line matching is obtained by matching r_1 with b_n, and matching all the other red points r_i to b_{i-1}, at no cost. The weight of the optimal matching is

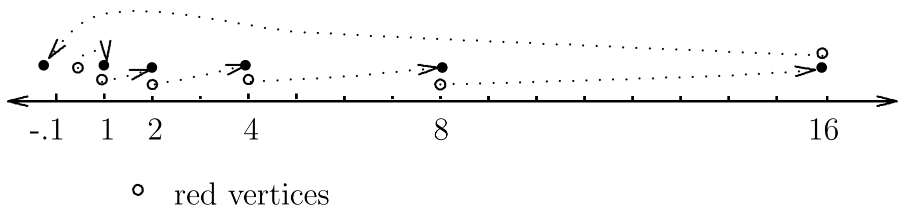

Figure 2: Example to show that greedy does poorly

constant, whereas the weight of the matching produced by the greedy method is at least 2^{n-1}, so the performance ratio of the greedy method is exponential in n.

2.1. The on-line algorithm

Let M_k be the on-line matching computed by the algorithm after the arrival of r_k. (Initially M_0 and N_0 are empty.)

Step 1. At the arrival of r_k, compute the off-line minimum weight maximum matching N_k matching $r_1, \ldots, r_k$ to the blue vertices. Without loss of generality, we can assume that $N_k \oplus N_{k-1}$ consists of a single odd length augmenting path from r_k to a blue vertex b_k. (We will explain why momentarily.)

Step 2. The vertex b_k will be free in M_{k-1}; match r_k to b_k to obtain M_k.

Lemma 2.1: *Without loss of generality, we can assume that $N_k \oplus N_{k-1}$ consists of a single odd length augmenting path from r_k to a blue vertex b_k.*

Proof: The symmetric difference $N_k \oplus N_{k-1}$ is the disjoint union of paths and even length cycles. If such a cycle exists, it cannot contain r_k (since r_k is unmatched in N_{k-1}). Therefore the total weight of the "even" edges in the cycle must equal the weight of the "odd" edges. (Otherwise we could improve either N_k or N_{k-1}.) So we can modify N_k so that $N_k \oplus N_{k-1}$ contains no cycles.

An even length path cannot begin and end with red vertices, because there is only one red vertex (r_k) which is matched in N_k, but not in N_{k-1}. An even length path which begins and ends with blue vertices cannot contain r_k. As above, the total weight of the "even" edges in the path must equal the weight of the "odd" edges, and we can modify N_k so that $N_k \oplus N_{k-1}$ contains no even length paths.

Finally, an odd length path in $N_k \oplus N_{k-1}$ must begin with r_k, and end with a blue vertex, b_k. □

Remark: Although the above guarantees that any off-line algorithm for minimum weight perfect matchings would suffice to compute N_k, we note that it is efficient to generate N_k from N_{k-1} by running a single phase of the primal-dual algorithm [PS], which works by finding the augmenting path from r_k to b_k (in the "equality subgraph"). In this case, $N_k \oplus N_{k-1}$ is already just a single path.

We show that b_k is unmatched in M_{k-1} by showing that the set of matched blue vertices in M_{k-1} is the same as in N_{k-1}. (By definition b_k is free in N_{k-1}).

Lemma 2.2: *For each k, the set of blue vertices matched in N_k is the same as the set of blue vertices matched in M_k (the on-line matching).*

Proof: Suppose by induction that M_{k-1} and N_{k-1} have the same set of matched blue vertices. Then N_k takes on only the additional blue vertex b_k, as does M_k. □

We now prove an upper bound on the weight of the matching produced by the algorithm.

Theorem 2.3: *The weight of M_n is $w(M_n) \leq (2n-1)w(N_n)$.*

Proof: Notice that $w(N_0) \leq w(N_1) \leq \ldots \leq w(N_n)$. Consider the edge e_k added at stage k. By the triangle inequality

$$\begin{aligned} w(e_k) &\leq w(\text{augmenting path from } r_k \text{ to } b_k\,) \\ &\leq w(N_k) + w(N_{k-1}) \\ &\leq 2w(N_n) \end{aligned}$$

Finally, $w(M_n) = \sum w(e_k) \leq (2n-1)w(N_n)$. (Since $w(e_1) \leq w(N_n)$.) □

Remark: Although this analysis may seem overly pessimistic, there is an example in R^1 to show that the algorithm's performance ratio is no better than n. The blue points $b_1, b_2, \ldots, b_n$ are placed at $-1, 2, -3, 4, -5, \ldots, \pm n$, and the red points $r_1, r_2, \ldots, r_n$ are placed, in order of appearance, at $0, -1, 2, -3, \ldots, \mp(n-1)$. If n is odd, the optimal matching pairs r_{2i} with b_{2i+1} and r_{2i+1} with b_{2i} (and r_1 with b_1). If n is even, the optimal matching pairs r_{2i} with b_{2i-1} and r_{2i-1} with b_{2i}. In each case the optimal matching has cost n. For each k, r_k is matched in M_k with b_k, at a cost of $2k-1$, so that $w(M_n) = \sum(2k-1) = n^2$, or a factor of n over the optimal $w(N_n) = n$.

2.2. The lower bound

Theorem 2.4: *There is no deterministic on-line algorithm that achieves a performance ratio better than* $(2n-1)$ *for all metric spaces.*

Proof: Consider a "star" graph with n leaf vertices, and one center. The n blue vertices are placed at the leaf vertices of the star. The first red point is placed at the center; the on-line algorithm matches it to some leaf b_1. From then on, r_k is placed at b_{k-1}. The weight of the on-line matching is $2n-1$, and the weight of the off-line matching is 1. (The weight of each edge on the star is 1.) □

Observation: Let us suppose that the points are chosen from some finite dimensional metric space (say R^d). Suppose the n blue vertices are placed at the grid points of a grid in R^d of side 1 unit, centered at the origin. The first red point is placed at the origin; the on-line algorithm matches it to some vertex b_1 of the cube. From then on, r_k is placed at b_{k-1}. The weight of the matching produced by any on-line algorithm is $\Omega(n)$, whereas the minimum weight perfect matching has weight $n^{\frac{1}{d}}$. (The matching is obtained by matching r_1 to b_n, and r_i to b_{i-1} for $2 \leq i \leq n$.) Hence the performance ratio of any on-line algorithm is $\Omega(n^{1-\frac{1}{d}})$.

Remark: In R^2 we obtain a lower bound of only $\Omega(\sqrt{n})$. The algorithm's performance ratio however is $O(n)$, and this gap is very large. For the special cases of graphs in R^1 and R^2 is it possible to close this gap? If we go to $d = \log n$ dimensions (hypercube), then we can obtain a lower bound of $\Omega(\frac{n}{\sqrt{\lceil \log n \rceil}})$.

3. On-line Stable Marriages

In this section we describe the stable marriage problem and its on-line version. An instance of the stable marriage problem consists of two disjoint sets of size n, the men and the women;

associated with each person is an ordered preference list that ranks all the members of the opposite sex. The object is to obtain a one-to-one correspondence (*i.e.*, a perfect matching, or *marriage*) between the men and the women.

We say that a man and a woman form an *unstable pair* for a marriage $\mathcal{M}$ if the man and woman are not married to each other, and each prefers the other to his/her spouse. A marriage is *unstable* if there is at least one unstable pair, and *stable* otherwise.

A stable marriage can always be found off-line; see [GI] for a simple algorithm and an excellent survey of research done on the stable marriage problem.

In the on-line version of the stable marriage problem, the women's preference lists are specified in advance (call this the *women's matrix*), and the men arrive in an arbitrary order. As each man arrives he provides his preference list and he is immediately assigned a mate from the pool of available women. It is not permitted to reassign any previously married woman.

An on-line algorithm may produce unstable marriages. We are interested in on-line algorithms that produce a small number of unstable pairs.

3.1. An On-Line Algorithm

The obvious deterministic on-line algorithm produces a marriage with only $O(n \log n)$ unstable pairs in the average case.

Algorithm: Assign each man his favorite available woman.

Label the men $1, 2, \ldots, n$ according to their order of arrival, and label the women arbitrarily. In the average case, each man's preference list is a uniformly chosen random permutation. The above algorithm is *independent* of the women's matrix. We give a specific women's matrix which yields the *worst case behavior*, and show that the average number of instabilities produced is $O(n \log n)$.

Lemma 3.1: *The algorithm behaves the worst when all the women prefer the men in order opposite to their arrival.*

Proof: The first man will not participate in any unstable pairs, since he marries his favorite woman. In general, the k^{th} man will form an unstable pair only with those of the $k-1$ previously married women whom he prefers over his wife, and who prefer him over their husbands. The expected number of instabilities involving man k is therefore maximized when the $k-1$ previously married women all prefer him over their husbands. □

Remark: Using this approach it is easy to see that every deterministic on-line algorithm will produce $\Omega(n^2)$ unstable pairs.

Theorem 3.2: *The natural "first come, first served" on-line marriage algorithm produces on average $O(n \log n)$ unstable pairs.*

Proof: Assume that the women's matrix is arranged as in the lemma. Let G_k be the set of women available to man k; $|G_k| = n - k + 1$. Suppose that man k marries his j^{th} choice; then man k is involved in $j-1$ instabilities. Since G_k is uniformly distributed in man k's preference list, we can show by a simple "balls and walls" symmetry argument (with G_k as the walls) that

$$E(j-1) = \frac{k-1}{n-k+2}.$$

To do so, imagine man k's preference list arranged in a line, with each of the $k-1$ unavailable women represented as a "ball", and each of the $n-k+1$ available women represented as a "wall". Then, with the walls uniformly distributed among the balls, we're interested in the expected position of the first wall, j, and the number $j-1$ of balls in the first "bin". If we imagine instead that the walls are placed first, and that the balls are added uniformly thereafter, intuition tells us that any ball is equally likely to fall into any of the $n-k+2$ bins. If intuition is not enough, reason that we have conditioned on the relative order of the walls, and condition further that a given ball lands in either the first or the second bin. So far we have conditioned that our ball and the first wall both appear before all the other walls. This is not enough to affect the relative order of our ball and the first wall (in a uniform distribution). Since it is equally likely that the ball lands in the first or second bin, these bins have equal expected sizes. In fact, all the bins have the same expected size, as asserted.

Having the expected number of instabilities involving man k, we calculate that the expected total number of instabilities is

$$\sum_{k=1}^{n} \frac{k-1}{(n-k)+2} = \sum_{i=1}^{n-1} \frac{i}{n-i+1} \leq \sum_{i=1}^{n-1} \frac{n}{n-i+1} = O(n \log n).$$

$\square$

Remark: It is clear that for the above women's matrix, the "first come, first served" algorithm is average-case optimal. If we allow the women's matrix to be random as well, the expected number of unstable pairs produced by our algorithm goes down only by half. However, for a random women's matrix our algorithm may not be average-case optimal.

A *stable* man or woman is one who is not involved in any unstable pairs. As might be expected, this algorithm produces lots of stable men, but very few stable women.

Theorem 3.3: *In the stable marriage produced by the on-line algorithm, the expected number of stable men is at least* $\frac{n+1}{2}$.

Proof: The probability that man k is stable is at least $\frac{n-k+1}{n}$. Let X_k be a random variable that is 1 if and only if man k is stable (and 0 otherwise). Then the expected number of stable men is

$$\begin{aligned} E(\text{stable men}) &= \sum_{k=1}^{n} E(X_k) \\ &\geq \sum_{k=1}^{n} \frac{n-k+1}{n} \\ &= \frac{n+1}{2}. \end{aligned}$$

$\square$

Theorem 3.4: *In the stable marriage produced by the on-line algorithm, if the women's matrix is fixed, the expected number of stable women lies between $\Omega(\log n)$ and n. If the women's matrix is chosen randomly (i.e., each woman's preference list is a random permutation of $b_1, \ldots, b_n$), the expected number of stable women is only $\Theta(\sqrt{n})$.*

Proof: If the women's matrix is fixed so that each woman prefers the men in their order of arrival (*i.e.*, they like b_1 the best and b_n the least), then the algorithm produces a stable marriage, with n stable women.

We now show that the most unstable women arise if the women prefer the men opposite to the men's order of arrival, in which case $\Theta(\log n)$ stable women are expected. Let $b_1, b_2, \ldots, b_n$ denote the men in order of arrival, and let $g_{\pi(1)}, g_{\pi(2)}, \ldots, g_{\pi(n)}$ denote their matches. Then, for any k, $g_{\pi(k)}$ is not unstable with $b_1, \ldots, b_{k-1}$ (since none of these men chose $g_{\pi(k)}$), and for $i > k$ $g_{\pi(k)}$ is unstable with b_i if and only if $g_{\pi(k)}$ appears before $g_{\pi(i)}$ in b_i's preference list and $g_{\pi(k)}$ prefers b_i to b_k. Clearly, then, $g_{\pi(k)}$ is most likely to be unstable if she prefers $b_{k+1}, \ldots, b_n$ over b_k, and the highest expected number of unstable women occurs when all the women prefer the men in the order $b_n, \ldots, b_1$.

Suppose that the women's matrix is so arranged; we will calculate the expected number of stable women produced by the algorithm. Let

$$X_k = \begin{cases} 1, & \text{if } g_{\pi(k)} \text{ is stable} \\ 0, & \text{otherwise.} \end{cases}$$

Then we are trying to calculate

$$\begin{aligned} E(\sum_{k=1}^{n} X_k) &= \sum_{k=1}^{n} \Pr[g_{\pi(k)} \text{ is stable}] \\ &= \sum_{k=1}^{n} \prod_{i=k+1}^{n} \Pr[g_{\pi(k)} \text{ is stable with } b_i]. \end{aligned}$$

We know from the proof of Theorem 3.2 ("balls and walls") that, for $i > k$, b_i will prefer $g_{\pi(k)}$ to $g_{\pi(i)}$ with probability $\frac{1}{n-(i-1)+1}$. Since we assume that $g_{\pi(k)}$ prefers b_i to b_k,

$$\Pr[g_{\pi(k)} \text{ is stable with } b_i] = 1 - \frac{1}{n-i+2}.$$

It turns out that in this case the product above telescopes, and that the sum reduces to

$$\sum_{k=1}^{n} \frac{1}{n-k+1} = \Theta(\log n).$$

If we take the women's matrix to be random, then

$$\Pr[g_{\pi(k)} \text{ prefers } b_i \text{ to } b_k] = \frac{1}{2},$$

so

$$\Pr[g_{\pi(k)} \text{ is stable with } b_i] \quad = \quad \frac{1}{2} \cdot 1 + \frac{1}{2} \cdot \left(1 - \frac{1}{n-i+2}\right).$$

In this case, the k^{th} term in the above sum is

$$\frac{2(n-k)+1}{2(n-k)+2} \cdots \frac{7}{8} \cdot \frac{5}{6} \cdot \frac{3}{4}.$$

Using the fact that $(2n-1)(2n-3)\cdots 7 \cdot 5 \cdot 3 = (2n)!/2^n n!$ and Stirling's approximation

$$n! \approx \left(\frac{n}{e}\right)^n \sqrt{2\pi n}[1 + O(\frac{1}{n})],$$

we approximate this term as $2/\sqrt{\pi(n-k+1)}$, and the entire sum as $\Theta(\sqrt{n})$. $\square$

3.2. A Worst Case Lower Bound

Our lower bound is based on the following theorem due to Yao [Y]:

Theorem 3.5: *Given a cost minimization problem, a randomized algorithm R, and an input distribution d, there is a deterministic algorithm A whose d-average case performance is better than R's worst case expected performance.*

The theorem says that a lower bound on the worst case performance of randomized algorithms may be found by giving an input distribution d for which every deterministic algorithm does badly in the d-average case.

Theorem 3.6: *No on-line randomized (or deterministic) algorithm can achieve better than $\Omega(n^2)$ unstable pairs in the worst case.*

Proof: By Theorem 3.5 it suffices to exhibit a probability distribution on the preference lists for which no deterministic algorithm can achieve less than $\Omega(n^2)$ instabilities. As before, we stipulate that the women all prefer the men in order opposite to their arrival. We proceed to describe the distribution on the men's matrix.

Let σ be a random permutation of the women. Then man k prefers the women in the order $\sigma(1), \ldots, \sigma(k)$, and the rest in a fixed arbitrary order (say, increasing numerically). The resulting men's matrix, where row k contains man k's preference list, is ordered by σ below the main diagonal and is ordered numerically above the diagonal.

We show that any deterministic algorithm produces on average $\Omega(n^2)$ unstable pairs on this input distribution. First we show that if β is the number of men $1, \ldots, \frac{n}{2}$ who marry among the women $\sigma(1), \ldots, \sigma(\frac{n}{2})$, then the expected value of β is at least $\frac{n}{16}$. This is clearly true if in fact at least $\frac{n}{16}$ of the men $1, \ldots, \frac{n}{4}$ marry women among $\sigma(1), \ldots, \sigma(\frac{n}{4})$. If this is not the case, then at least $\frac{3n}{16}$ of the men $1, \ldots, \frac{n}{4}$ marry women $\sigma(\frac{n}{4}+1), \ldots, \sigma(n)$. Notice that any man $1, \ldots, \frac{n}{4}$ who marries among women $\sigma(\frac{n}{4}+1), \ldots, \sigma(n)$ is equally likely to marry any of these women (*i.e.*, the conditional probability distribution in this case is uniform). Therefore we expect that at least $\frac{1}{3} \cdot \frac{3n}{16} = \frac{n}{16}$ of the men $1, \ldots, \frac{n}{4}$ will marry in the range $\sigma(\frac{n}{4}+1), \ldots, \sigma(\frac{n}{2})$. In either case we expect at least $E(\beta) \geq \frac{n}{16}$ of the men $1, \ldots, \frac{n}{2}$ to marry women $\sigma(1), \ldots, \sigma(\frac{n}{2})$.

Furthermore, at least β of the men $\frac{n}{2}+1, \ldots, n$ marry women $\sigma(\frac{n}{2}+1), \ldots, \sigma(n)$ (by the process of elimination). Each of these men is unstable with at least β women, so the total number of unstable pairs is at least

$$E(\beta^2) \geq [E(\beta)]^2 \geq \Omega(n^2).$$

$\square$

4. Open Problems

- Obtain a competitive on-line algorithm for the general graph weighted matching problem when the edge weights satisfy the triangle inequality. One nice model for this problem is as follows: the vertices arrive one at a time, and distances are revealed from the newly arrived vertex to all the vertices (even the ones that have not arrived yet). The algorithm is required to match off each vertex as it arrives. It is easy to show that in case distances are revealed only to the vertices that have already arrived, no on-line algorithm having a competitive ratio only as a function of n is possible.
- For the on-line weighted matching problem, is it possible to achieve better performance when the points are constrained to lie on the real line, or in the plane?
- Can we get better performance if we consider randomized on-line weighted matching algorithms? For example, how badly do randomized greedy algorithms do?

Acknowledgments: We would like to thank Amotz Bar-Noy, Omer Berkman and Baruch Schieber for useful discussions. We would also like to thank the referees for their comments. Finally, we would like to note that a slightly stronger bound for Theorem 3.3 has been obtained by Raymond Board (personal communication).

References

[GI] D. Gusfield and R. Irving, *The Stable Marriage Problem: Structure and Algorithms*, The MIT Press (1989).

[KP] B. Kalayanasundaram and K. Pruhs, “On-line Weighted Matching”, *Journal of Algorithms*, 14(3), (1993), pp 478–488.

[KVV] R. M. Karp, U. V. Vazirani and V. V. Vazirani, “An Optimal Algorithm for On-line Bipartite Matching”, *Proc. of STOC Conference*, (1990), pp. 352–358.

[MMS] M. Manasse, L. A. McGeoch and D. Sleator, “Competitive Algorithms for Server Problems”, *Journal of Algorithms* 11(2), (1990), pp. 208–230.

[PS] C. H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity*, Prentice Hall (1982).

[PY] C. H. Papadimitriou and M. Yannakakis, “Shortest Paths Without a Map”, *Theoretical Computer Science*, 84(1), (1991), pp. 127–150.

[ST] D. Sleator and R. E. Tarjan, “Amortized Efficiency of List Update and Paging Rules”, *Communications of the ACM*, 28, (1985), pp. 202–208.

[Y] A. C. Yao, “Probabilistic Computations: Towards a Unified Measure of Complexity”, *Proc. of FOCS Conference*, (1977), pp. 222–227.