On-line Algorithms for Weighted Matching and Stable Marriages*

Samir Khuller**
Stephen G. Mitchell
Vijay V. Vazirani

TR 90-1143
July 1990

Department of Computer Science
Cornell University
Ithaca, NY 14853-7501

*Supported by NSF Grant DCR-85-52938 and PYI matching funds from AT&T Bell Labs and Sun Microsystems, Inc.
**Supported by an IBM Graduate Fellowship.

On-line Algorithms for Weighted Matching and Stable Marriages *

Samir Khuller † *Stephen G. Mitchell* *Vijay V. Vazirani*

Computer Science Department
Cornell University
Ithaca, NY 14853

Abstract

We give an on-line deterministic algorithm for the bipartite weighted matching problem that achieves a competitive ratio of $O(n)$. In fact, this algorithm is almost optimal – the lower bound on the performance ratio of any deterministic online algorithm is $\Omega(n/\sqrt{\log n})$.

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.

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

†Supported by an IBM Graduate Fellowship.

1. Introduction

There has been a great deal of interest recently in studying the relative power of on-line *vs.* off-line algorithms [ST 85,CGJ 83,MMS 88]. An optimal randomized on-line algorithm for bipartite matching was given in [KVV 90]. In this paper we study two related problems — bipartite weighted matching (with triangle inequality), and on-line stable marriages.

We are able to give a near-optimal deterministic algorithm for on-line weighted bipartite matching, with triangle inequality. The on-line model is the similar to that in [KVV 90]. We assume that the graph is a weighted complete bipartite graph; the girl vertices are all given in advance, and the boys arrive one at a time (revealing weights on the edges to each of the girls). The boys have to be matched off immediately; once a boy 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). Without the triangle inequality imposed on the weights, any (deterministic or randomized) on-line algorithm can have a very bad competitiveness ratio, which will in general depend on the sizes of the weights themselves, rather than on n. (The competitiveness of an on-line algorithm is the ratio of its performance to the performance of the optimal solution; in our case it is the ratio of the weight of the on-line matching to the weight of the minimum weight perfect matching.) We give a deterministic algorithm that is $O(n)$-competitive if the edge weights obey the triangle inequality (equivalently, if the edge weights are distances in some metric on the set of vertices). In fact, this algorithm is almost optimal — any deterministic algorithm for on-line weighted bipartite matching with triangle inequality must perform as badly as $\Omega(n/\sqrt{\log n})$. An interesting feature of the algorithm is that it computes the optimal off-line matching among the already arrived boy vertices to determine which girl vertex to match to the newly arrived boy.

We get more pessimistic results for the stable marriage problem. We study a natural on-line model in which we assume that all the girls' preference lists are known in advance, and the boys' preference lists are revealed one at a time. Each boy has to be married off as soon as his preference list is revealed. We are interested in the number of unstable pairs an on-line algorithm produces. 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 algorithm can do better than $\Omega(n^2)$ unstable pairs. The stable marriage problem has been studied extensively for its rich structure [GI 89], and is useful in several situations, e.g., assigning interns to hospitals, students to colleges, *etc.*

2. On-line Weighted 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 $O(n)$ times the weight of the minimum weight perfect matching. We show that this is optimal up to a log factor for graphs obeying the triangle inequality.

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 $R \cup B$ (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 $w_{ij} = d(r_i, b_j)$. 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.

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.

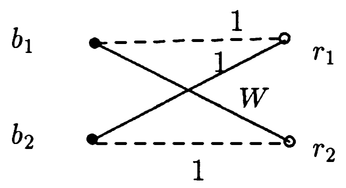

Figure 1: Example to show that no ratio possible without triangle inequality

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 $1 \le i \le n-1$; 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 $2 \le i \le n$ 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

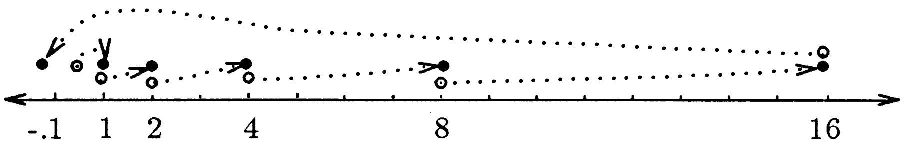

Figure 2: Example to show that greedy does poorly

matching all the other red points r_i to b_{i-1}, at no cost. The weight of the optimal matching is constant, whereas the weight of the matching produced by the greedy method is 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 perfect 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). 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. $\square$

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 82], 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. $\square$

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 $w(M_n) \leq 2nw(N_n)$.*

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, $w(M_n) = \sum w(e_k) \leq 2nw(N_n)$. $\square$

Hence the performance ratio of the algorithm is $O(n)$.

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 k is odd, the optimal matching pairs r_{2i} with b_{2i+1} and r_{2i+1} with b_{2i}; if k 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 k. 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* $\Omega(n/\sqrt{\log n})$.

Proof: Let $d = \lceil \log n \rceil$, and suppose the n blue vertices are placed at the corners of the cube in R^d of side 2 centered at the origin. The first red point is placed at the origin; the on-line algorithm matches it to some corner 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 $\sqrt{d} = \sqrt{\log n}$. (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 $\Omega(n/\sqrt{\log n})$.

□

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?

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 boys and the girls; 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 boys and the girls.

We say that a boy and a girl form an *unstable pair* for a marriage $\mathcal{M}$ if the boy and girl are not married, 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 89] 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 girls' preference lists are specified in advance (call this the *girls' matrix*), and the boys arrive in an arbitrary order. As each boy arrives he provides his preference list and he is immediately assigned a mate from the pool of available girls. It is not permitted to reassign any previously married girls.

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 boy his favorite available girl.

Label the boys $1, 2, \ldots, n$ according to their order of arrival, and label the girls arbitrarily. In the average case, each boy's preference list is a uniformly chosen random permutation. We give a specific girls' 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 girls prefer the boys in order opposite to their arrival.*

Proof: The first boy will not participate in any unstable pairs, since he marries his favorite girl. In general, the k^{th} boy will form an unstable pair only with those of the $k-1$ previously married girls whom he prefers over his wife, and who prefer him over their husbands. The expected number of instabilities involving boy k is therefore maximized when the $k-1$ previously married girls 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 girls' matrix is arranged as in the lemma. Let G_k be the set of girls available to boy k; $|G_k| = n - k + 1$. Suppose that boy k marries his j^{th} choice; then boy k is involved in $j - 1$ instabilities. Since G_k is uniformly distributed in boy k's preference list, by a simple "balls and walls" symmetry argument (with G_k as the walls)

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

Therefore the expected total number of instabilities is

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

□

Remark: It is clear that for the above girls' matrix, the "first come, first served" algorithm is average-case optimal. If we allow the girls' 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 girls' matrix our algorithm may not be average-case optimal.

A *stable* boy or girl is one who is not involved in any unstable pairs. As might be expected, this algorithm produces lots of stable boys, but very few stable girls.

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

Proof: The probability that boy 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 boy k is stable (and 0 otherwise). Then the expected number of stable boys is

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

□

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

Proof: If the girls' matrix is fixed so that each girl prefers the boys 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 girls.

We now show that the most unstable girls arise if the girls prefer the boys opposite to the boys' order of arrival, in which case $\Theta(\log n)$ stable girls are expected. Let $b_1, b_2, \ldots, b_n$ denote the boys 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 boys 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_k'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 girls occurs when all the girls prefer the boys in the order $b_n, \ldots, b_1$.

Suppose that the girls' matrix is so arranged; we will calculate the expected number of stable girls 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 girls' matrix to be random, then

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

so

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

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

$$n! \approx \left(\frac{n}{e}\right)^n \sqrt{2\pi n},$$

we approximate this term as $2/\sqrt{\pi n}$, and the entire sum as $\Theta(\sqrt{n})$. □

3.2. A Worst Case Lower Bound

Our lower bound is based on the following observation of Yao [Y 77]:

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: As before, we can argue that the worst behavior arises if the girls all prefer the boys in order opposite to their arrival. By Theorem 3.5 it suffices to exhibit a probability distribution on the boys' preference lists for which no deterministic algorithm can achieve less than $\Omega(n^2)$ instabilities. We proceed to describe such a distribution on the boys' matrix.

Let σ be a random permutation of the girls. Then boy k prefers the girls in the order $\sigma(1), \ldots, \sigma(k)$, and the rest in a fixed arbitrary order (say, increasing numerically). The resulting boys' matrix, where row k contains boy 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 at an expected number of $\beta \geq \frac{n}{16}$ of the boys $1, \ldots, \frac{n}{2}$ marry girls $\sigma(1), \ldots, \sigma(\frac{n}{2})$. This is clearly true if in fact at least $\frac{n}{16}$ of the boys $1, \ldots, \frac{n}{4}$ marry girls $\sigma(1), \ldots, \sigma(\frac{n}{4})$. If this is not the case, then at least $\frac{3n}{16}$ of the boys $1, \ldots, \frac{n}{4}$ marry girls $\sigma(\frac{n}{4}+1), \ldots, \sigma(n)$. Notice that any boy $1, \ldots, \frac{n}{4}$ who marries $\sigma(\frac{n}{4}+1), \ldots, \sigma(n)$ is equally likely to marry any of these girls (*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 boys $1, \ldots, \frac{n}{4}$ will marry in the range $\sigma(\frac{n}{4}+1), \ldots, \sigma(\frac{n}{2})$. In either case we expect at least $\beta \geq \frac{n}{16}$ of the boys $1, \ldots, \frac{n}{2}$ to marry girls $\sigma(1), \ldots, \sigma(\frac{n}{2})$.

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

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

□

Open Problems:

- 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?

References

[CGJ 83] E. G. Coffman, M.R. Garey and D. S. Johnson, "Dynamic Bin Packing", *SIAM Journal on Computing*, 12, (1983), pp. 227–258.

[GI 89] D. Gusfield and R. Irving, *The Stable Marriage Problem: Structure and Algorithms*, The MIT Press (1989).

[KVV 90] 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 88] M. Manasse, L. A. McGeoch and D. Sleator, "Competitive Algorithms for Online Problems", *Proc. of STOC*, (1988), pp. 322–333.

[PS 82] C. H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity*, Prentice Hall (1982).

[ST 85] D. Sleator and R. E. Tarjan, "Amortized Efficiency of List Update and Paging Rules", *Communications of the ACM*, 28, (1985), pp. 202–208.

[Y 77] A. C. Yao, "Probabilistic Computations: Towards a Unified Measure of Complexity", *Proc. of FOCS Conference*, (1977), pp. 222–227.