An Optimal Algorithm for On-line Bipartite Matching

Richard M. Karp
University of California at Berkeley &
International Computer Science Institute

Umesh V. Vazirani
University of California at Berkeley

Vijay V. Vazirani
Cornell University

1. Introduction

There has been a great deal of interest recently in the relative power of on-line and off-line algorithms. An on-line algorithm receives a sequence of requests and must respond to each request as soon as it is receiveD. An off-line algorithm may wait until all requests have been received before determining its responses. One approach to evaluating an on-line algorithm is to compare its performance with that of the best possible off-line algorithm for the same problem. Thus, given a measure of "profit", the performance of an on-line algorithm can be measured by the worst-case ratio of its profit to that of the optimal off-line algorithm. This general approach has been applied in a number of contexts, including data structures [SlTa], bin packing [CoGaJo], graph coloring [GyLe] and the k-server problem [MaMcSl]. Here we apply it to bipartite matching and show that a simple randomized on-line algorithm achieves the best possible performance.

2. Problem Statement

Let $G(U,V,E)$ be a bipartite graph on $2n$ vertices such that G contains a perfect matching. Let B be an $n \times n$ matrix representing the structure of $G(U,V,E)$. The rows of B correspond to vertices in U (the boys) and the columns to vertices in V (the girls); each edge is represented by a 1 in the appropriate position. We consider the problem of constructing a large matching in $G(U,V,E)$ on-line. Assume that the girl vertices arrive in a preselected order, and that the edges incident to a vertex are revealed to us only when the vertex arrives. The task is to decide, as each girl vertex arrives, which boy vertex to match her to, so that the size of the matching obtained is maximized. Alternatively, we can view the matching as being constructed while the matrix is revealed column-by-column. As a convention we will assume that columns are revealed in the order $n, n-1, \cdots 1$.

The performance of a randomized algorithm A for this task is denoted by $p(A)$ and is defined to be:

$$\underset{G}{MIN} \quad \underset{\text{order of girl vertices}}{MIN} \quad E[\textit{size of matching achieved by } A]$$

where the expectation is taken over the internal coin flips of A.

Remark: A greedy algorithm which always matches a girl if possible (to an arbitrarily chosen boy among the eligible ones), achieves a maximal matching - and therefore a matching of size at least $\frac{n}{2}$. On the other hand an adversary can limit any deterministic algorithm to a matching of size $\frac{n}{2}$: for example, by letting the first $\frac{n}{2}$ columns contain all ones and the last $\frac{n}{2}$ columns contain ones only in those rows which are matched by the deterministic algorithm in the first $\frac{n}{2}$ steps.

Many of the results in the literature of on-line algorithms concern the performance of randomized on-line algorithms against an *adaptive on-line adversary* [BeBoKaTaWi]. In the context of the present problem, adaptive-

ness means that the adversary is permitted to specify the matrix column-by-column, and to take into account, in specifying any given column, the decisions that the randomized algorithm has made in response to the arrivals of earlier columns. The fact that the adversary is on-line means that the adversary must construct his own perfect matching column-by-column, choosing the row to be matched in each column at the same time as he specifies the column. An adaptive on-line adversary can limit any randomized on-line algorithm to a matching of expected size $n/2+O(\log n)$ by choosing the matrix, and his own perfect matching, as follows: for $i=0$ to $n/2$, there is a 1 in position $j, n-i$ if and only if row j does not lie in the matching constructed so far by the algorithm, and also does not lie in the matching constructed so far by the adversary; for his perfect matching, the adversary chooses a 1 in column $n-i$ at random. For $i=n/2+1$ to n, there is a 1 in position $j, n-i$ if and only if row j does not lie in the matching constructed so far by the adversary; in this case also, the adversary chooses for his perfect matching a random 1 in column $n-i$. To show that no randomized algorithm can achieve more than $n/2+O(\log n)$ on the average against this adversary, we argue as follows. First, any non-greedy randomized algorithm can be replaced by a greedy one that performs at least as well on the average. Secondly, for any greedy algorithm A, let T(A) be the set of rows that are matched in columns $n, n-1, \ldots, n/2+1$ by both A and the adversary. Then the expected cardinality of $T(A)$ is $O(\log n)$, and the size of the matching produced by algorithm A does not exceed $n/2+|T(A)|$.

The Ranking Algorithm:

We shall analyze the performance of the following randomized on-line matching algorithm, which we shall refer to as the RANKING algorithm:

Initialization: Pick a random permutation of the boy vertices - thereby assigning to each boy a random priority or ranking.

Matching Phase: As each girl arrives, match her to the eligible boy (if any) of highest rank.

Remark: At first sight it might appear more natural to analyze the algorithm RANDOM, which picks a boy at random from among the eligible boys each time a girl arrives. However, RANDOM performs nearly as poorly as a deterministic greedy algorithm;

it achieves a matching of expected size only $\frac{n}{2}+O(\log n)$ on the following matrix: $B_{ij}=1$ if $i=j$ or if $\frac{n}{2}\le j\le n$ and $1\le i<n/2$, and 0 otherwise. RANDOM performs poorly in this example because it concentrates too much effort on the dense upper half of the matrix for the first $\frac{n}{2}$ moves, thereby missing out on the crucial edges in the sparse lower half of the matrix. RANKING has an implicit self-correcting mechanism that tends to favor those currently eligible boys who have been eligible least often in the past. It is this feature of RANKING that allows it to perform well even on graphs where local density considerations are misleading.

2. Analysis of the Ranking Algorithm

The Duality Principle:

After the initialization phase of RANKING, there is an ordering on both the boy and girl vertices (the preselected ordering on girls and the randomly chosen ordering on the boys). At this point there is a symmetry between the boy vertices and the girl vertices: the performance of RANKING remains unchanged if we interchange the roles of the boys and girls by letting the boys arrive according to their ranking and picking the highest ranked eligible girl.

Lemma 1: For any fixed orderings of the boy and girl vertices, the matching picked during the matching phase of RANKING remains unchanged if the roles of the boy and girl vertices are interchanged.

Proof: The proof is by induction on the number of boys and girls. Let b be the highest ranked boy, and g, the highest ranked girl that b has an edge to. Now, if the matching is found from the boys' side, b will be matched to g in the first step. Also, if the matching is found from the girls' side, the first time that b is eligible to be matched is when g arrives; clearly, they are matched at that time. The lemma follows by removing b and g from the graph, and applying the induction hypothesis to the remaining graph.

Henceforth we shall regard the columns as ordered from 1 to n with column n having highest rank and column 1 lowest, and the rows as arriving in random order. As each row arrives it is matched to the highest ranking available eligible column. Viewing rows as

arriving in random order gives us a new notion of *time* which is crucial to our analysis of the algorithm. Let $\sigma(1) \cdots \sigma(n)$ be an ordering of the rows. By *time* t we mean the instant of the t^{th} row arrival, i.e. $\sigma(t)$.

We next give a technical lemma that will be useful at several points. Consider a variant of RANKING which, as each boy arrives, either matches him to the highest ranking eligible girl, or else refuses to match him at all, even though one or more eligible girls may be available. The rule that determines whether this algorithm refuses may be quite arbitrary.

Lemma 2: For any fixed ordering of the boys and ranking of the girls, the set of girls matched by RANKING is a superset of the set matched by any refusal algorithm.

Proof: By induction on t. By the induction hypothesis, the set of girls eligible to be matched at time $t+1$ by the refusal algorithm forms a superset of those eligible to RANKING. Now, since both algorithms use the same ranking on the girls, if the refusal algorithm chooses to match a girl who is also eligible for RANKING, then RANKING must match her too. Thus, in all cases, the set of girls matched by RANKING remains a superset of the set matched by the refusal algorithm.

Next, we prove that we can assume w.l.o.g. that the adjacency matrix B of the graph is upper-triangular.

Lemma 3: The expected size of the matching produced by RANKING is minimum for some upper-triangular matrix.

Proof: Let B be any matrix. Renumber the rows of B so that a perfect matching sits on the main diagonal (i.e. $B_{ii} = 1$ for $1 \le i \le n$). This renumbering has no effect on the performance of RANKING, since the rows arrive in a random order. Let B' be the matrix obtained when all entries of B below the main diagonal are replaced by 0 (i.e. $B'_{ij} = B_{ij}$ if $i \le j$ and 0 if $i > j$). Now RANKING on B' may be viewed as a refusal algorithm on B. Thus, by lemma 2, the expected size of matching obtained by ranking on B' is at most as large as on B. □

Remark: We conjecture that in fact the expected size of matching achieved by RANKING is minimized by the complete upper-triangular matrix. However, we do not know how to prove this directly. We shall show a performance guarantee for RANKING that is matched to within low order terms by its performance on the complete upper-triangular matrix, thus proving indirectly that this is the worst case (to within lower order terms). Proving the conjecture will yield the stronger result that RANKING has the best performance guarantee.

Henceforth we will assume that B is upper-triangular, with diagonal entries 1, corresponding to the unique perfect matching in the graph. Consider the symmetric difference of this perfect matching with the maximal matching M produced by RANKING. If $|M| = n/2$, each connected component of the symmetric difference is an augmenting path is of length 3, and no diagonal entries are picked. In this case, for each i, either row i or column i is matched, but not both. On the other hand, whenever many diagonal elements are chosen or many long augmenting paths occur, there will correspondingly be a large number of indices i such that row i and column i are both matched. The idea behind our proof is that , under a random ordering of the rows, RANKING is likely to yield a large number of such indices, and hence a large matching. This last implication is made precise in the following lemma.

Lemma 4: Let B be an $n \times n$ upper triangular matrix with diagonal entries 1. Let M be any matching in the associated graph such that for each i either row i or column i is matched, and let $D = \{i$: row i and column i are both matched in $M\}$. Then $|M| = \frac{n+|D|}{2}$

Proof: For each i, either row i or column i is matched, i.e. covered by some edge in M. $|D|$ = number of i such that both row i and column i are covered. Now, the number of vertices covered by M is $n+|D|$ and the number of edges in M is $\frac{n+|D|}{2}$.

Corollary: $E[|M|] = n/2 + 1/2\, E[|D|]$.

We will lower-bound $E[|M|]$ by lower-bounding $E[|D|] = \sum_{i=1}^{n} Pr$ [column i and row i both get matched], where the probability is over random row arrivals.

For the purpose of analyzing the performance of RANKING, it is useful to consider a modification - the algorithm EARLY - which refuses to match row i if it arrives after column i has already been matched. Notice that on the complete upper-triangular matrix algorithm EARLY is identical to RANKING.

Lemma 5: For every ordering of the rows, RANKING produces at least as large a matching as that produced by algorithm EARLY.

Proof: This follows from Lemma 2, since EARLY is a refusal algorithm.

We will lower-bound $E[|D|]$ for algorithm EARLY. Algorithm EARLY has the property that row i gets matched if and only if column i is not already matched when row i arrives. In particular, if in some ordering column i gets matched at time t and row i arrived at time $\leq t$ then row i must also get matched (because, in particular, column i was available for row i). Index i enters the set D in precisely this way.

Definition: Let σ be a permutation of the rows, and let $\sigma^{(i)}$ be the sequence obtained by deleting i from its original position in σ and moving it to the last position. If EARLY does not match column i under σ, then define $W(\sigma,i)=0$. Otherwise, define $W(\sigma,i)$ to be the time at which column i gets matched under the permutation $\sigma^{(i)}$; if column i remains unmatched under $\sigma^{(i)}$ then define $W(\sigma,i)=n$. Now, define

$$w_t^i = Pr[W(\sigma,i)=t] \text{ for } 0 \leq t \leq n.$$

where σ is a random permutation of the rows. Clearly, Pr [column i gets matched] $= \sum_{t=1}^{n} w_t^i$. The next lemma shows what fraction of this probability corresponds to the favorable event that column i and row i both get matched.

Lemma 6: Let $W(\sigma,i)=t$ and $t<n$. Obtain permutation σ' from $\sigma^{(i)}$ by moving row i into the j^{th} position. Then, under σ', EARLY will match row i as well as column i by time $t+1$, if $j \leq t$, and will not match row i at all if $j>t$.

Proof: If $j>t$ then column i will get matched under σ' at time t, before row i arrives, so row i will not get matched. If $j \leq t$ then the i^{th} column is eligible when row i arrives; therefore EARLY matches row i. Running EARLY on $\sigma^{(i)}$ for t steps can be regarded as a refusal algorithm on σ' run for $t+1$ steps. So by lemma 2, the columns matched under σ' by time $t+1$ form a super-set of the columns matched under $\sigma^{(i)}$ by time t; hence column i gets matched under σ' by time $t+1$.

Lemma 7: Pr[row i and column i both get matched] $= \sum_{t=1}^{n} \frac{t}{n} w_t^i$.

Proof: Firstly, notice that if $W(\sigma,i)=n$, then row i must have arrived at or before the time when column i got matched in σ, and hence row i must also have gotten matched. Consider any time t, $1 \leq t < n$, and consider the orderings such that $W(\sigma,i)=t$. Say that two such orderings σ and π are equivalent if $\sigma^{(i)}=\pi^{(i)}$. Clearly, each equivalence class has n orderings, and row i falls in one of the first t positions in t of these. The proof follows by Lemma 6. □

Let $w_t = \sum_{i=1}^{n} w_t^i$. By Lemma 6, $E[|D|] = \sum_{t=1}^{n} \frac{t}{n} w_t$. We will now lower-bound $E[|D|]$ by lower-bounding the right hand side. We first present an easy bound establishing that the expected size of the matching is at least $(2-\sqrt{2})n$.

Lemma 8: If column i gets matched at time t under σ then under $\sigma^{(i)}$ column i either remains unmatched, or gets matched at some time $\geq t-1$.

Proof: The algorithm under $\sigma^{(i)}$ for the first $t-1$ steps can be regarded as a refusal algorithm for our algorithm run on σ for t steps. Now the lemma follows by applying lemma 2.

Definition: Let $m_t = Pr$ [some column is matched at time t].

Corollary: $\sum_{s \leq t} w_s \leq \sum_{s \leq t+1} m_s$.

Lemma 9: EARLY produces a matching of size at least $(2-\sqrt{2})n$ on an $n \times n$ upper-triangular matrix.

Proof: Let αn be the size of matching produced. Then, by Lemmas 3 and 6,

$$\alpha n \geq \frac{n}{2} + \frac{1}{2n} \sum_{t=1}^{n} t w_t.$$

Since $m_t \leq 1$ and $\sum_{t=1}^{n} w_t = \alpha n$, we see by the corollary to lemma 7 that $\sum_{t=1}^{n} t w_t$ is minimized by setting $m_1 = m_2 = \cdots = m_{\alpha n} = 1$, $w_1 = m_1 + m_2$ and $w_t = m_{t+1}$, $t>1$. Substituting the resulting bound into the above inequality yields $\alpha \geq 2-\sqrt{2}$. □

In Lemma 9, we have made the pessimistic assumption that $m_t = 1$ for $1 \leq t \leq \alpha n$, which would mean that the first αn rows to arrive all get matched. This is, of course, not the case, since, even early in the process, a row may arrive after its column is already matched. Thus the m_t's, and hence also the w_t's, are spread out in time. Lemma 10 makes this observation more precise.

Lemma 10: For all t, $m_t = 1 - \frac{1}{n} \sum_{s<t} w_s$.

Proof: Let $m_t^i = Pr$[row i occurs at time t and gets

matched].

Then clearly $m_t = \sum_{i=1}^{n} m_t^i$. Now, Pr [row i occurs at *time* t and does not get matched] $= \frac{1}{n} \sum_{s<t} w_s^i$. i.e. pick a permutation σ such that $W(\sigma,i) < t$, and move row i into the t^{th} place. Therefore, $m_t^i = \frac{1}{n} - \frac{1}{n} \sum_{s<t} w_s^i$. The lemma follows by summing over i. □

Let αn be the expected size of matching produced by EARLY. We need to lower-bound $\sum_{t=1}^{n} t w_t$ subject to:

(i). $\sum_{t=1}^{n} w_t = \alpha n$

(ii). $m_t = 1 - \frac{1}{n} \sum_{s<t} w_s$, and

(iii). $\sum_{s \le t} w_s \le \sum_{s \le t+1} m_s$.

The solution is much simpler if condition (iii) is replaced by condition (iii)' below:

(iii)' $\sum_{s \le t} w_s \le \sum_{s \le t} m_s$

Also, we will drop condition (ii) for $t=n$ (this does not affect the validity of our bound). Lemma 11 establishes that replacing (iii) by condition (iii)' does not change the desired lower-bound by much. Lemma 12 asserts that, subject to (i), (ii) and (iii)', $\sum_{t=1}^{n} t w_t$ is minimized by picking the w_i's greedily, i.e. by making each w_i, $i=1,2, \cdots$ in turn as large as possible.

Lemma 11: Let $\bar{w} = (w_1, w_2, \cdots w_n)$ be any solution to conditions (i), (ii), and (iii). Then there is a solution $\bar{x} = (x_1, x_2, \cdots x_n)$ to conditions (i), (ii) and (iii)' such that the L_1 norm of $(\bar{w} - \bar{x})$ is at most 2.

Proof: By conditions (ii) and (iii) we have

(iv). $\sum_{s \le t} w_s \le t+1 - \frac{1}{n} \sum_{s \le t} (t+1-s) w_s$.

$\bar{x}$ is obtained from $\bar{w}$ by moving one unit from the lowest possible indices to w_n. Pick k such that $\sum_{i=1}^{k} w_i \le 1$ and $\sum_{i=1}^{k+1} w_i > 1$. Set $x_i = 0$ for $1 \le i \le k$, $x_{k+1} = w_{k+1} - (1 - \sum_{i=1}^{k} w_i)$ and $x_n = w_{n+1}$. The remaining indices of $\bar{x}$ are the same as those of $\bar{w}$. Clearly if $\bar{w}$ satisfies (iv), then $\bar{x}$ satisfies condition (v) stated below, and the L_1 norm of $(\bar{w} - \bar{x})$ is at most 2.

Lemma 12: Subject to conditions (i), (ii) and (iii)' $\sum_{t=1}^{n} t w_t$ is minimized by picking w_i's greedily.

Proof: By conditions (ii) and (iii)' we have:

(v) $\sum_{s \le t} w_s \le t - \frac{1}{n} \sum_{s<t} (t-s) w_s$

Suppose for contradiction that the w_i's that minimize $\sum_{t=1}^{n} t w_t$ are not picked greedily according to conditions (i) and (v). Let t be the last time such that w_t is not as large as possible. Let the deficiency in w_t be ε. Increase w_t by ε, decrease w_{t+1} by $\varepsilon(1+1/n)$, and increase w_n by ε/n. The new w_i's satisfy (i) and (v), and have a smaller $\sum_{t=1}^{n} t w_t$. Contradiction. □

Remark: The greedy solution resulting from condition (v) is $w_t = (1-\frac{1}{n})^{t-1}$.

Theorem 1: The performance of algorithm EARLY is $n(1-\frac{1}{e}) + o(n)$

Proof: By Lemma 10 and 11, it is sufficient to pick w_i's greedily subject to conditions (i), (ii) and (iii)'. This yields

$$w_t = (1-\frac{1}{n})^{t-1}, \text{for } t=1,2,\ldots T$$

where T is such that $\sum_{t=1}^{T} w_t = \alpha n$. Substituting for w_t and solving for T yields $T \le -n \ln(1-\alpha)$.

Let $(1-\frac{1}{n}) = \theta$. Then, $\theta^T = 1-\alpha$. Now,

$$\sum_{t=1}^{T} t\, w_t = \sum_{t=1}^{T} t\,\theta^{t-1} = \frac{(1-(\theta^T) - T\theta^T(1-\theta))}{(1-\theta)^2} \ge$$

$$n^2(\alpha + (1-\alpha)\ln(1-\alpha))$$

Substituting this into our lower-bound of $n/2 + E[|D|]$ on the size of the matching yields:

$$\alpha n \ge \frac{n}{2} + \frac{1}{2n} \sum_{t=1}^{T} t w_t$$

$$\ge \frac{n}{2} + \frac{n}{2}(\alpha + (1-\alpha)\ln(1-\alpha))$$

This gives $(\alpha - 1) \ge (1-\alpha)\ln(1-\alpha)$

Thus $\alpha \ge (1-\frac{1}{e})$. □

Remark: A simple consequence of our proof is that if RANKING is applied to a $n \times n$ matrix B for which the size of the maximum matching is $m < n$, then the expected size of the matching produced by RANKING is at least $(1-\frac{1}{e})m + o(m)$.

3. Bounding the Performance of Any On-Line Algorithm

In this section we will show that RANKING is optimal, up to lower order terms.

Theorem 2: The performance of any on-line bipartite matching algorithm is $\leq n(1-\frac{1}{e})+o(n)$.

Let T be the $n \times n$ complete upper-triangular matrix. As before, we assume that the columns of T arrive in the order $n, n-1, ..., 1$. By the k^{th} column arrival we mean the arrival of column number $n-k+1$. Consider the algorithm RANDOM, which matches each column to a randomly chosen eligible row.

Definition: Let T be the $n \times n$ complete upper-triangular matrix. With every permutation π on $\{1, \cdots n\}$ associate a problem instance (T,π), where the adjacency matrix is obtained by permuting the rows of T under π, and the columns arrive in the order $n, n-1, \cdots 1$. Let P denote the uniform probability distribution over these $n!$ instances.

Lemma 13: Let A be a deterministic on-line algorithm that is 'greedy' in the sense that it never leaves a column unmatched if there is an eligible row. Then, the expected size of matching produced by A when given an instance (T,π) from P is the same as the expected size of matching produced by RANDOM on T.

Proof: The lemma follows from the two claims listed below, which may be proved by a straightforward induction on time:
1. For algorithm A on (T,π), as well as for RANDOM on T, if there are k eligible rows at time t, then they are equally likely to be any set of k rows from among the first $n-t+1$ rows of T.
2. For each k, the probability that there are k eligible rows at time t is the same for RANDOM run on T as it is for A run on (T,π). □

Lemma 14: The performance of any on-line matching algorithm is upper bounded by the expected size of matching produced by the algorithm RANDOM on the complete upper-triangular matrix.

Proof: Let $E[R(T,\pi)]$ denote the expected size of matching produced by the given randomized on-line algorithm, and let $E[A(P)]$ denote the expected size of matching produced by a deterministic algorithm A when given an input from distribution P. By Yao's lemma [Ya],

$$\min_{\pi} \{E[R(T,\pi)]\} \leq \max_{A} \{E[A(P)]\}.$$

where the maximum is over all deterministic algorithms. *W.l.o.g.* the best deterministic algorithm is greedy (by simulating A, and matching the current column to the row matched by A, if the row is available, and to an arbitrary eligible row otherwise). The proof follows from Lemma 13. □

Lemma 16: The expected size of matching produced by algorithm RANDOM on T is $n(1-\frac{1}{e})+o(n)$.

Proof: The proof rests on the following crucial observation made in Lemma 13: given that there are l rows still eligible at the k^{th} arrival column, they are equally likely to be any set of l rows from among the first $n-k+1$ rows of T.

Let $x(t)$ and $y(t)$ be random variables representing the number of columns remaining and the number of rows still eligible at time t. Let $\Delta x = x(t+1)-x(t)$ and $\Delta y = y(t+1)-y(t)$. Then $\Delta x = -1$ and Δy is -2 if the diagonal entry in the $t+1^{st}$ column was eligible but was not matched, and -1 otherwise. Using the fact that the set of eligible rows is randomly chosen from among the first $n-t$:

$$E[\Delta y] = -1 - \frac{y(t)}{x(t)} \cdot \frac{y(t)-1}{y(t)} = -1 - \frac{y(t)-1}{x(t)}.$$

$$\textit{Therefore } \frac{E[\Delta y]}{E[\Delta x]} = 1 + \frac{y(t)-1}{x(t)}.$$

Kurtz's theorem [Ku] says that with probability tending to 1 as n tends to infinity, the solutions of the above stochastic difference equation are closely approximated by the solution of the differential equation:

$$\frac{dy}{dx} = 1 + \frac{y-1}{x}.$$

Solving this differential equation with the initial condition $x=y=n$, we get

$$y = 1 + x\left(\frac{n-1}{n} - \ln\frac{x}{n}\right)$$

So, when only one row is eligible, the number of columns remaining is $\frac{n}{e}+o(n)$. Therefore, the expected size of matching produced is $n(1-\frac{1}{e})+o(n)$.

Remark: 1) There is an interesting alternative description of the behavior of algorithm RANDOM on T. In this description, the algorithm begins by specifying a random permutation $\sigma=(\sigma(1), \sigma(2), ..., \sigma(n))$ of $\{1, 2, ..., n\}$. Then, as each column $n-i$ arrives, RANDOM matches that column with row τ, where τ is the first element of σ which has not previously been matched and is less than or

equal to $n-i$. It is easy to see that this is a faithful description of RANDOM, and as a consequence, the following two random variables have the same distribution: (i) the size of the matching produced by RANDOM on T; (ii) the length of the longest subsequence of a random permutation such that, for all k, the k^{th} element of the subsequence is greater than or equal to k. Thus, as a byproduct of Lemma 16 we obtain the interesting combinatorial result that the expectation of this latter random variable is $n(1-\frac{1}{e})+o(n)$.

2). It is easy to show that the expected size of matching produced by RANDOM and RANKING is the same on T. So, proving the conjecture that T is the worst matrix for RANKING together with Lemmas 13 and 14 will show that RANKING is the best possible on-line bipartite matching algorithm.

4. Open Questions:

1. Is the complete upper-triangular matrix the worst-case input for RANKING?

2. Is RANKING an optimal on-line matching algorithm in the non-bipartite case?

Acknowledgements:

We would like to acknowledge helpful discussions with Rajeev Motwani.

References:

[BeBoKaTaWi] S. Ben-David, A. Borodin, R. Karp, G. Tardos, A. Wigderson, *'On the Power of Randomization in On-Line Algorithms'*, STOC 1990.

[CoGaJo] E. G. Coffman, M. R. Garey, D. S. Johnson, *'Dynamic Bin Packing'*, SIAM J. comput., vol 12, 1983, pp. 227-258.

[Gy,Le] A. Gyarfas, J. Lehel, *'Online and First Fit Colorings of Graphs'*, J. Graph theory, Vol. 12, No. 2, pp. 217-227, 1988.

[Ku] T. G. Kurtz, *'Solutions of Ordinary Differential Equations as Limits of Pure Jump Markov Processes'*, Journal of Applied Probability, vol. 7, 1970, pp. 49-58.

[MaMcSl] M. Manasse, L.A. McGeoch, D. Sleator, *'Competitive Algorithms for Online Problems'*, STOC 1988, pp.322-333.

[Sl,Ta] D. Sleator, R.E. Tarjan, *'Amortized Efficiency of List Update and Paging Rules'*, Comm. ACM, vol. 28, 1985, pp. 202-208.

[Ya] A. C. Yao, *'Probabilistic Computations: Towards a Unified Measure of Complexity'*, FOCS 1977, pp. 222-227.