A randomized algorithm for the on-line weighted bipartite matching problem *

Béla Csaba† András Pluhár‡

Preliminary Version

Abstract

We study the on-line minimum weighted bipartite matching problem in arbitrary metric spaces. Here n, not necessary disjoint, points of a metric space M are given, and to be matched on-line with n points of M revealed one by one. The cost of a matching is the sum of the distances of the matched points, and the goal is to find or approximate its minimum. The competitive ratio of the deterministic problem is known to be $\Theta(n)$, see [7, 11]. It was conjectured in [8] that a randomized algorithm may perform better against an oblivious adversary, namely with an expected competitive ratio $\Theta(\log n)$. We prove a little weaker result by showing a $o(\log^3 n)$ upper bound on the expected competitive ratio. As an application the same upper bound holds for the notoriously hard fire station problem, where M is the real line, see [6, 12].

1 Introduction

Finding a minimum weight matching in a weighted graph G is a well studied problem in graph theory. Much less is known about its on-line version; here we briefly introduce the set-up and the most important results. For more thorough references see [7, 8, 11, 12].

Let G be an arbitrary weighted graph, and two players, A and B, we consider the following *on-line matching game* on G: First, A picks the multiset $S = \{s_1, \ldots, s_n\}$ of $V(G)$, these are the *servers*. Then, one by one, A discloses the *requests*, that is again a multiset $R = \{r_1, \ldots, r_n\}$ of $V(G)$. When an element of R is requested, B has to match it with some unmatched element from S, and B wishes to minimize the cost of the resulted matching.

It is clear that usually B cannot reach the offline minimum, and the competitive ratio, that is the online cost/offline optimum is infinite if one has no further assumption on G, see Kalyanasundaram and Pruhs, and Khuller et al in [7, 11]. It was assumed in both papers that the weights are nonnegative, and satisfy the triangle inequality, so one may refer to the graph G as a metric space $\mathcal{M} = (X, d)$ with underlying set X and distance function d, while the multisets S and R are repeated points

*The authors were partially supported by OTKA grants T034475 and T049398.

†Bolyai Institute, University of Szeged, email: bcsaba@math.u-szeged.hu

‡Department of Computer Science, University of Szeged, email: pluhar@inf.u-szeged.hu

of $\mathcal{M}$. Then the best competitive ratio is exactly $2k-1$. This is achieved for $K_{1,k}$, the so-called star metric space, where the weights are all ones.

The randomized setup for the above on-line game is the following: first, A has to construct S and R in advance and disclose S. Then A gives the points of R, one by one, but this time he has no right to make any changes in the requests, no matter how B is playing. That is, not only R but the ordering in which the points of it are requested are determined in advance. In this setup B has the advantage of using randomness when deciding which point of S to be matched with the newly requested point. Let $\mathrm{opt}(\rho)$ be the total weight of the optimum matching for a sequence of requests ρ. We say that B's randomized strategy is c–competitive if for every request sequence ρ

$$E[B(\rho)] \leq c\mathrm{opt}(\rho),$$

where $E[B(\rho)]$ denotes the expected total weight of the matching B finds for ρ. Finding good randomized algorithms for the on-line minimum matching problem was first addressed by Kalyanasundaram and Pruhs in [8]. They stated that optimal competitive ratio for a star metric space is $2H_k - 1$, and conjectured an $O(\log n)$ upper bound on the best competitive ratio for arbitray metric spaces. Here and later n stands for the number of servers (or requests).

Our goal is to show the following theorem.

Theorem 1 *There is a randomized on-line weighted matching algorithm for arbitrary metric spaces which is $O(\log^3 n/\log\log n)$–competitive.*

The strategy of the proof is the following. First we show that it is enough to consider the case when the metric space $\mathcal{M}$ is a finite space, indeed X is the set of servers. This will cost only a constant factor of at most 3. Then we develop a randomized weighted greedy matching algorithm, shortly **RWGM** that has competitive ratio $O(\log n)$ if $\mathcal{M}$ is of special structure. Namely, the points of $\mathcal{M}$ are the leaves of a *hierarchically well separated tree*, or *HST*. Here the distance $d(x,y)$ is defined by adding up the weights on the edges of the unique paths connecting x and y, and the edge weights are growing exponentially by the levels of the tree. In our case the smallest weights are of size $O(\log n)$. In order to use this special case, we recall earlier results on probabilistically approximating arbitrary metric spaces by such trees next. This approximation involves a $O(\log^2 n/\log\log n)$ factor in the competitive ratio, so finally we arrive to an algorithm of competitive ratio $O(\log^3 n/\log\log n)$.

2 Discretizing the game

Assume, that we have an on-line matching algorithm MA, that is c-competitive in the possibly infinite metric space $\mathcal{M}$ in case $R \subset S$ (multiplicities allowed). In this subsection we will show, that with a small loss in the competitive factor MA can easily be extended to an on-line matching algorithm MAI which works for arbitrary $S, R \subset \mathcal{M}$. The extension of the algorithm is based on a transformation of R which we call *discretization*.

Given S assume that the elements of R appear after each other. For $r_i \in R$ we assign a new point $g(r_i) \in S$. We determine $g(r_i)$ in a greedy fashion: if $d(s_0, r_i) =$

$\min_{s\in S} d(s, r_i)$, then $g(r_i) = s_0$ (breaking ties arbitrarily). Clearly, we can find $g(r_i)$ on-line. For $s \in S$ denote rm_s the number of requests which are assigned to s by g. The new multiset of requests is called R', in which every $s \in S$ appears rm_s times. R' is the discretized version of R.

As above, assume that MA is a c-competitive on-line algorithm in case $R \subset S$. Clearly, after the discretization we arrive to an R' such that $R' \subset S$. We give another on-line algorithm MAI in the following way: we play another, *auxiliary* on-line matching game on $\mathcal{M}$ by MA, and use MA's decisions to determine which server MAI would choose to serve a request. Suppose that a request $r \in R$ appears. We determine $g(r)$, and serve this request by MA. If MA chooses $s \in S$ to serve $g(r)$, then MAI will serve r by s.

Lemma 2 *If MA is c-competitive, then MAI is $(2c + 1)$-competitive for arbitrary $S, R \subset \mathcal{M}$.*

Proof: We start with some more notation. For a matching algorithm A denote $A(r_i)$ the distance of r_i and s if A serves this request by s. Denote OM the optimal cost matching between S and R, and let $opt = cost(OM)$. OM induces a matching OM' (not necessarily of minimum cost) between S and R' in the obvious way: if $(r_i, s_j) \in OM$, then $(g(r_i), s_j) \in OM'$. Finally, let us denote by opt' the total cost of the minimum matching between S and R'.

We have a trivial lower bound on the optimum: $\sum_{i=1}^{n} d(r_i, g(r_i)) \le opt$. By the triangle inequality $MAI(r_i) \le MA(g(r_i)) + d(g(r_i), r_i)$, hence, $MAI(r_i) \le MA(r_i) + opt(r_i)$. Again by the triangle inequality: $cost(OM'(r_i)) \le cost(OM(r_i)) + d(g(r_i), r_i)$. Since $cost(OM'(r_i)) \ge opt'(r_i)$, we have that $opt'(r_i) \le cost(OM(r_i)) + d(g(r_i), r_i)$. Summing up for every r_i we get the inequality $opt' \le 2opt$.

MA is a c-competitive on-line algorithm by assumption, i. e., $\sum_{i=1}^{n} MA(r_i) \le copt'$. We know, that $MAI(r_i) \le MA(r_i) + opt(r_i)$, therefore, $\sum_{i=1}^{n} MAI(r_i) \le copt' + opt \le (2c + 1)opt$. □

Remark. Lemma 2 gives an alternative proof of the theorem of Kalyanasundaram and Pruhs [7], that the competitive ratio of the greedy algorithm is at most $2^n - 1$. Indeed, let MA and MAI be the greedy algorithm for $n - 1$ and n element input, respectively, and use induction.

3 The algorithm RWGM

Our algorithm, the randomized weighted greedy matching algorithm, or **RWGM** algorithm is first developed for special metric spaces. Assume that the metric space $\mathcal{M} = (X, d)$ is defined by a weighted tree T. The set of the leaves of T is $L \subset X$, and the distance $d(x, y)$ for the leaves x, y is the sum of the weights on the (unique) path connecting x and y.

Let $\lambda \ge 1$ be a real number.

Definition 1 *A λ–hierarchically well separated tree (λ–HST) is a rooted weighted tree with the following properties:*

- *the edge weight from any node to each of its children is the same,*
- *the edge weights along any path from the root to a leaf are decreasing by a factor of at least λ.*

We define the **RWGM** algorithm first, then show it in steps that it is $O(\log n)$–competitive on a metric space determined by a $O(\log n)$–HST.

3.1 RWGM - a randomized weighted matching algorithm for hierarchically well separated trees

Let us consider a $\log n$-HST, denote it by $T = T(V, E, r)$, where V is the vertex set, E is the edge set of T, and r is the root. When playing the matching game only leaves of T will be matched to leaves of T. We denote the set of leaves by L. We will need the notion of a subtree: given $v \in V$, the vertex $u \in V$ belongs to the subtree T_v if the only path from r to u contains v. Clearly, $T = T_r$, and if $w \in L$, then T_w contains only the leaf w. We have the relation "$\leq$" among the subtrees containing a certain leaf w: $T_u \leq T_{u'}$ if $|T_u| \leq |T_{u'}|$, and $w \in T_u$, $w \in T_{u'}$.

In order to get an easier formulation of **RWGM**, we assume that if u is a non-leaf vertex of a $\log n$-HST, then all of its children are non-leaves or all are leaves. This can be achieved by inserting "dummy" vertices in the tree. We can also assume that the edge weights on a level are equal. (See also in [5].)

During the course of satisfaction of the requests, certain vertices will be painted green, and leaves may have multiplicities. The colors and multiplicities of the vertices may change in time. We try to follow the greedy algorithm, and break ties by random selection by *levels*.

Formal description of **RWGM**

In the beginning the adversary A picks leaves of T with multiplicity, corresponding to the servers $S = s_1, \ldots, s_n$. (That is if a leaf x is provided m times as a server then x has multiplicity m.)

We color a vertex u of T green if T_u contains a leaf with positive multiplicity, and will call such subtrees green subtrees.

Then A will give us the requests of R one-by-one, denote them by $r_1, \ldots, r_n$.

Set $i = 1$.

- Step 1. The new request is r_i. B looks for the smallest subtree T_u which contains r_i, and u is green.
- Step 2. Pick a leaf of T_u among the leaves of positive multiplicity by the algorithm **Pick-a-leaf** with input u. This leaf is x, let s_i (perhaps after reordering) be an unused server corresponding to it, and s_i is the server which is matched to r_i. Decrease the multiplicity of x by one.
- Step 3. For every green $w \in V$ check whether T_w contains a leaf with positive multiplicity. If not, erase w's color.

- Step 4. If $i \leq n-1$, then set $i = i+1$, then go to Step 1.
- Step 5. If $i = n$, then STOP.

Algorithm **Pick-a-leaf**(u)

- Step 1. If the children of u are leaves, then pick randomly, uniformly a leaf among those of positive multiplicity. This is the leaf we have chosen. STOP.
- Step 2. If the children of u are not leaves, then denote $u_1, u_2, \ldots, u_t$ the green children of u. Pick one randomly, uniformly among them, say, it is u_i. Apply **Pick-a-leaf**(u_i).

Theorem 3 *The algorithm* **RWGM** *is $O(\log n)$–competitive on a metric space determined by a $O(\log n)$–HST.*

3.2 Proof of Theorem 3

We prove Theorem 3 in steps. First we consider the case of uniform metric space where the multiplicities are all ones, but the sizes of S and R may not be equal. Then we discuss the case of before with arbitrary multiplicities. Finally we prove the general statement for HST's; here the previous cases provide a basis for induction arguments.

3.2.1 Uniform case

Assume that U is the uniform metric space on u points. Let $S = \{s_1, \ldots, s_q\}$ and $R = \{r_1, \ldots, r_t\}$. We also assume that the points of R are requested in increasing order, first r_1, then r_2, etc., and finally r_t.

Definition 2 *We say that $s_i \in S$ has a pair if $s_i = r_j$ for some $r_j \in R$. Similarly, $r_j \in R$ has a pair if $s_i = r_j$ for some $s_i \in S$.*

We will give an ordering of the points of S using the ordering on R. Firstly if there exist r_j and r_l such that s_i is the pair of r_j and s_k is the pair of r_l where $j < l$, then $s_i < s_k$. If s_i has a pair and s_k has no pair, then $s_i < s_k$. Finally, we fix an arbitrary ordering among those points of S which has no pair in R. Notice, that we can extend the orderings of S and R into an ordering "$<$" of $S \cup R$. It is done by such that if r_i is the pair of s_j then $r_i < s_j$, and for $r_k > r_i$ we have $s_j < r_k$.

Given $r_i \in R$ we associate a weight $w(r_i)$ with it. Let us assume that r_i has no pair, then

$$v_i = |\{s_j : s_j > r_i\}| - |\{r_k : r_k < r_i \text{ and } r_k \text{ has no pair}\}|.$$

If r_i has a pair, then let $v_i = 0$. Then we define $w(r_i) = H_{v_i}$ (we let $H_f = 0$ if $f \leq 0$).

We need the following useful lemma.

Lemma 4 *For $n \geq 1$, $H_n = 1 + \frac{H_{n-1}+\ldots+H_1}{n}$.*

Proof: Trivial computation. □

Lemma 5 *Let $\delta = |R - S|$. Then in the case above the expected cost of* **RWGM** *is at most $H_q + H_{q-1} + \ldots + H_{q-\delta+1}$.*

Proof: We proceed by induction on q. Notice that we may assume that r_1 has no pair, otherwise we can immediately apply the induction hypotheses. Now r_1 is matched to some randomly chosen $s_j \in S$. One checks that the weights of the elements of $R \setminus \{r_1\}$ are invariant if s_j had no pair. If s_j had the pair r_i then the expected new weight of r_i is at most $(H_{q-1} + \ldots + H_1)/q$. Now by induction one can see that for the resulting smaller subproblem the random algorithm has expected cost $H_{q-1} + \ldots + H_{q-\delta+1}$. Matching of r_1 to s_j has costed one, hence, the expected cost of the algorithm is at most

$$1 + \frac{H_{q-1} + \ldots + H_1}{q} + H_{q-1} + \ldots + H_{q-\delta+1} = H_q + H_{q-1} + \ldots + H_{q-\delta+1},$$

by Lemma 4. □

3.2.2 The case of multiplicities

We want to handle the case when both the servers and the requests have various multiplicities. Note, that a server with zero multiplicity simply means that there is no server at that point. If $U = x_1, \ldots, x_u$, then let $ms(x_i)$ and $mr(x_j)$ are the multiplicities of servers and requests in point x_i and x_j, respectively. Let $diff(x_i) = \max\{0, mr(x_i) - ms(x_i)\}$, $\delta = \sum_{i=1}^{u} diff(x_i)$.

Lemma 6 *The expected cost of RWGM is at most $H_q + H_{q-1} + \ldots + H_{q-\delta+1}$.*

Proof: Fix a maximum matching between servers at requests which belong to the same point. Pretend the equal servers, requests to be different if they are not matched, and apply Lemma 5. □

Remark. Note the following: if $q \geq t$, then RWGM is a $O(\log n)$-competitive algorithm for uniform metric spaces, even if multiplicities are allowed.

3.2.3 General HST trees

We proceed by induction on the height of the HST tree. First of all we need a more technical form of the hypotheses and some definitions.

Definition 3 *Given $s \in S$ and $r \in R$, which are matched in some matching M, consider the path connecting them in the HST tree. Call the point at the highest level of this path the turning point of s and r, shortly $t_M(s, r)$. For a point u of the tree let $\tau_M(u)$ be the number of (s, r) matched pairs in M for which u is a turning point.*

Given a point u, $h(u)$ will denote the height of u. Observe that $\tau_M(u)$ is the same for any optimal matching M, hence we suppress subscript M. Note, that $\tau(u)$ is obvious to compute. Moreover, one can express the optimal cost:

$$\mathrm{opt} = 2\sum_u \tau(u) \sum_{i=1}^{h(u)} (\log n)^i.$$

For trees of height less than d our induction hypotheses is the following inequality:

$$\mathrm{E}[RWGM] \leq 4 \sum_{u:h(u)>1} \tau(u) \log n \sum_{i=1}^{h(u)} (\log n)^i + 2 \sum_{u:h(u)=1} \tau(u) \log^2 n. \quad (1)$$

For trees of height one the statement follows from Lemma 6 and its remark.

Consider a tree T of height d. We make a new tree T' and a new instance S' and R'. T' comes from T by pruning the leaves, and for a $u \in T$, $h(u) = 1$ we associate the server and request multiplicities that of the sum of the server and request multiplicities of its descendants in T.

One can cut the optimal cost for S, R and T in two parts. The first part is the optimal cost of S', R' and T', which we call opt'. The second part is the cost incurring on $T \setminus T'$, this is opt^*. Here we have to take care of cases when the number of requests are greater than the number of servers in a subtree T_u ($h(u) = 1$). Then we consider the partial optimal matching using that servers. Let us call the cost of this partial matching, opt^*_u the optimal for this case. Analogously, we can define $\tau'(u)$ if $h(u) > 1$ and $\tau^*(u)$ for $h(u) = 1$.

Clearly, $\mathrm{opt}^* = \sum_{u:h(u)=1} \mathrm{opt}^*_u$, and one concludes that $\mathrm{opt} = \mathrm{opt}' + \mathrm{opt}^*$. On the same way, $\mathrm{opt}^* = \sum_{u:h(u)=1} 2\tau^*_u \log n$, and $\mathrm{opt}' = 2\sum_{u:h(u)>1} \sum_{i=2}^{h(u)} (\log n)^i \tau'_u$.

One checks that (1) and the above expressions for the optimal cost imply Theorem 3. Hence if we prove (1), we are ready.

Unfortunately, the on-line cost on T is not the sum of the on-line costs of the parts if we handle the parts separately, but they are closely related.

For this reason we have to take care of the costs occurring in $T \setminus T'$ when two points are matched which do not belong to the same height-one subtree. This extra cost comes from two sources. Consider a subtree T_u of height one in which the number of requests, t exceeds the number of servers, q in the beginning. In this case even the optimal matching algorithm have to find $t - q$ servers outside of T_u. Let us denote by $RWGM_u$ the cost of RWGM inside T_u.

Lemma 7 $2(\mathrm{opt}^*_u + (t-q)\log n) \geq \mathrm{E}[RWGM_u] + (t-q)\log n$.

Proof: Throughout the proof we will assume, that $\mathrm{opt}^*_u = 0$. There are two cases to consider. When $t - q \geq q$, then $RWGM_u \leq q$ trivially. Then the statement translates to the true inequality $2(t-q)\log n \geq q + (t-q)\log n$. If $t - q < q$, then $\mathrm{E}[RWGM_u] \leq (t-q)\log q$ by Lemma 6. Now the statement turns to the inequality $2(t-q)\log n \geq (t-q)\log q + (t-q)\log n$. □

The other kind of extra cost is caused when a request r from outside is matched to a server s of T_u. We use the notation of Lemma 6.

Lemma 8 $\mathrm{E}[RWGM_u] \leq \sum_{i=0}^{\delta} H_{q-i}$.

Proof: The proof follows the same line of arguments as Lemma 6, the details are omitted. □

In order to finish the proof we put together the pieces. $\mathrm{E}[RWGM]$ is less than the expected on-line cost on T' and on $T \setminus T'$ and the extra costs controlled by Lemmas 7 and 8. Using the induction hypotheses it develops to

$$\mathrm{E}[RWGM] \leq 4 \sum_{u:h(u)>2} \tau(u) \log n \sum_{i=2}^{h(u)} (\log n)^i +$$

$$2 \sum_{u:h(u)=2} \tau(u)(\log^2 n + \log n) + 2 \sum_{u:h(u)=2} \tau(u)(\log^2 n + \log n) +$$

$$2 \sum_{u:h(u)=1} \tau(u) \log n =$$

$$4 \sum_{u:h(u)>1} \tau(u) \log n \sum_{i=1}^{h(u)} (\log n)^i + 2 \sum_{u:h(u)=1} \tau(u) \log^2 n.$$

Here the first and second terms are the cost spent on T' by the induction hypotheses. The third and fourth are the cost on $T \setminus T'$ plus the extra costs, since Lemmas 7 and 8 relate the cost on $T \setminus T'$ plus extra cost to the optimal cost on $T \setminus T'$, and the latter one is expressed in terms of function τ. □

4 Approximating by hierarchically well separated trees

The idea, the first results and applications of hierarchically well separated trees are due to Bartal, see in [2, 3]. It generalized the earlier works of Karp [10] and Alon et al [1] in which they approximated the distances in certain graphs by using randomly selected spanning trees.

Definition 4 *A metric space $M = (X, d_M)$ dominates a metric space $N = (X, d_N)$ if for every $x, y \in X$ we have $d_N(x, y) \leq d_M(x, y)$.*

Definition 5 *A set of metric spaces S over X α–probabilistically approximates a metric space M over X, if every metric space in S dominates M, and there exists a probability distribution over metric spaces $N \in S$ such that for every $x, y \in X$ we have $E[d_N(x, y)] \leq \alpha d_M(x, y)$.*

The proof of Theorem 1 is based on the following theorem.

Theorem 9 *[5, 4] Every weighted graph on n vertices can be α–probabilistically approximated by a set of λ–HSTs, where $\alpha = O(\lambda \log n / \log \lambda)$.*

As it was noted by Bartal [2] having an approximation of a metric space $\mathcal{M}$ by HST trees and a good algorithm for such trees always results in good randomized algorithm in that space. So, what we do is the following. First, preprocessing: given the set of servers S, these points span a sub-metric space $\mathcal{M}_S \subset \mathcal{M}$. Clearly, $|\mathcal{M}_S| \leq n$, since S is a multiset of n elements. We approximate $\mathcal{M}_S$ by a set of $O(\log n)$-HSTs. By Fakcharoenphol et al. [5] there is a probability distribution $\mathcal{P}$ on these trees such that the expected distortion is $O(\log^2 n/\log\log n)$. Choose one tree at random according to $\mathcal{P}$. This finishes the preprocessing. Whenever a request $r \in R$ appears, we determine $g(r)$ (see Section 2), and use RWGM with this new request $g(r)$. We proved in Section 3, that RWGM is a $O(\log n)$-competitive algorithm in this case. Applying Lemma 2 and Theorem 9, we get that RWGM is $O(\log^3 n/\log\log n)$ competitive for $\mathcal{M}$. This proves Theorem 1. □

Acknowledgment. We thank Endre Szemerédi and Kirk Pruhs for the fruitful discussions.

References

[1] N. Alon, R. M. Karp, D. Peleg, D. West, A graph–theoretic game and its application to the k-server problem, *SIAM J. Comput.* **24** (1) (1995) 78–100.

[2] Y. Bartal, Probabilistic approximations of metric spaces and its algorithmic applications, in: *IEEE Symposium on Foundations of Computer Science,* 1996, pp. 184–193.

[3] Y. Bartal, On approximating arbitrary metrics by tree metrics, in: *STOC,* 1998.

[4] Y. Bartal, M. Charikar and R. Raz, Approximating min-sum k-clustering in metric spaces, Thirty-Third Annual ACM Symposium on Theory of Computing, pages 11?20, 2001.

[5] J. Fakcharoenphol, S. Rao and K. Talwar, A tight bound on approximating arbitrary metrics by tree metrics, *J. Comput. System Sci.* **69** (2004), no. 3, 485–497.

[6] B. Fuchs, W. Hochstättler, and W. Kern. Online matching on a line, In Hajo Broersma, Ulrich Faigle, Johann Hurink, Stefan Pickl, and Gerhard Woeginger, editors, *Electronic Notes in Discrete Mathematics,* volume **13.** Elsevier, 2003.

[7] B. Kalyanasundaram and K. Pruhs, Online weighted matching, *Journal of Algorithms,* **14(3)** (1993) 478–488.

[8] B. Kalyanasundaram, K. Pruhs, On-line network optimization problems, in *Online algorithms: The State of the Art*, eds. A. Fiat and G. Woeginger, pages 268–280 Lecture Notes in Comput. Sci., 1442, *Springer, Berlin,* (1998)

[9] B. Kalyanasundaram, K. Pruhs, The online transportation problem, *SIAM J. Discrete Math.* **13** (2000), no. 3, 370–383.

[10] R. Karp, A 2k-competitive algorithm for the circle. *Manuscript*, August 1989.

[11] S. Khuller, S. G. Mitchell, V. V. Vazirani, On-line algorithms for weighted bipartite matching and stable marriages, *Theoret. Comput. Sci.* **127** (1994), no. 2, 255–267.

[12] E. Koutsoupias, A. Nanavati, The online matching problem on a line, *Approximation and online algorithms,* 179–191, Lecture Notes in Comput. Sci., 2909, *Springer, Berlin,* 2004.

[13] Y. T. Tsai, C. Y. Tang, Y. Y. Chen, Average performance of a greedy algorithm for the on-line minimum matching problem on Euclidean space, *Inform. Process. Lett.* **51** (1994), no. 6, 275–282.