A Better Lower Bound for On-Line Bottleneck Matching

Ramana M. Idury* Alejandro A. Schäffer *

1 Introduction:

These notes concern the on-line bottleneck matching problem posed by Kalyanasundaram and Pruhs[Proc. 2nd Ann. ACM-SIAM Symp. Discr. Alg., pp. 234–240]. In this problem the initial input consists of a metric space and k servers placed in the metric space. Then a sequence of k requests is placed. After each request is placed, the on-line algorithm must match an unused server to the new request. The cost of a matching is the length of the heaviest edge (i.e., server-to-request distance) used. The competitiveness for a particular request sequence is the ratio of the on-line cost to the cost of the optimal off-line matching. The competitiveness of an algorithm is the limit as k grows without bound of the maximum ratio over all inputs with k servers and k requests.

Kalyanasundaram and Pruhs exhibited an algorithm with competitiveness $2k-1$ and claimed a lower bound of $k+1$. In these notes we improve the lower bound to $c(k)k$, where $c(k)$ is a slowly decreasing function of k whose value is 1.5 for $k=2$ and whose limit is $1/(\ln 2)$ or approximately 1.44. In our lower bound construction the metric space is the real line.

Throughout these notes we define $S = \{S_1, \ldots, S_k\}$ to be the sequence of servers in the left-to-right order on the real line and $\{s_1, \ldots, s_k\}$ to be their locations. Similarly we define $R = \{R_1, \ldots, R_k\}$ and $\{r_1, \ldots, r_k\}$ for the sequence of requests. In general R does not necessarily represent the chronological order of the requests.

2 A Lower Bound:

We need the following natural lemma before proving our lower bound.

Lemma 2.1 Let $S = \{S_1, \ldots, S_k\}$ and $R = \{R_1, \ldots, R_k\}$ be any set of servers and requests respectively in the left-to-right order on the real line. The matching

$$\{(S_1, R_1), (S_2, R_2), \ldots, (S_k, R_k)\}$$

*Department of Computer Science, Rice University, P. O. Box 1892, Houston, Texas 77251 U.S.A.

is an optimal off-line matching.

Proof: We can represent any bipartite perfect matching as a permutation of the set R; the i^{th} request in the permutation is matched to S_i. It suffices to prove that the identity permutation $\{R_1, \ldots, R_k\}$ represents an optimal off-line matching. Let $\Pi(R)$ be an optimal matching. We use the unique decomposition C_Π of the permutation $\Pi(R)$ into cycles to deduce that $\{(1), (2), \ldots, (k)\}$ is an optimal matching. Specifically, suppose that C_Π has $c < k$ cycles. We construct a different matching whose cost is no more than that of $\Pi(R)$ whose cyclic representation has at least $c + 1$ cycles .

In writing out parts of the cycle decomposition we consider only the subscripts as the symbol R is redundant. Let m be the smallest element in R that is not a fixed point of $\Pi(R)$ Let $(\cdots i\ m\ j\ \cdots)$ be the cycle containing m in C_Π; note that we may have $i = j$, but m is distinct from both i and j and less than them. This definition implies that $(S_i, R_m), (S_m, R_j) \in \Pi(R)$. There are two cases depending on the relative positions of R_m and S_m.

case 1: R_m is to the left of S_m. R_j must be to the right of R_m and S_i to the right of S_m giving us a linear arrangement (from left-to-right) of either $R_m R_j S_m S_i$ or $R_m S_m R_j S_i$ or $R_m S_m S_i R_j$. In all the three arrangements we can rearrange the matching and replace $\{(S_i, R_m), (S_m, R_j)\}$ by $\{(S_m, R_m), (S_i, R_j)\}$ without increasing the weight of the bottleneck edge. In each case the replacement increases the number of cycles by at least one.

case 2: S_m is to the left of R_m. This case is essentially similar to case 1.

We can inductively do the above replacement and eventually obtain the identity permutation that has k cycles. ∎

We use an *on-line adversary* and present a lower bound of $2 + \frac{1}{2^{\frac{1}{k-1}} - 1}$ for the on-line bottleneck matching problem with $k \geq 2$ servers. Initially the adversary provides the set of servers $S = \{S_1, \ldots, S_k\}$ with locations $\{s_1, \ldots, s_k\}$. For $1 \leq i < k$, the distance between S_i and S_{i+1}, or $s_{i+1} - s_i$, is set to $(1 + \epsilon)^i$ where $\epsilon = 2^{\frac{1}{k-1}} - 1$. Then the adversary specifies the request sequence $R = \{R_1, \ldots, R_k\}$ with locations $\{r_1, \ldots, r_k\}$ according to the following strategy:

for $i \leftarrow 1$ to $k - 1$
    if S_1 is already matched, say to R_j, then $r_i = s_{i+1} - (1 + \epsilon)^j + 1$
    else $r_i = s_i + 1$
if S_1 is already matched, say to R_j, then $r_k = s_1 - (1 + \epsilon)^j + 1$
else $r_k = s_k + 1$

Using the above adversary construction, we prove:

Theorem 2.2 The competitiveness of any deterministic on-line algorithm for bottleneck matching with $k \geq 2$ servers is at least $2 + \frac{1}{2^{\frac{1}{k-1}}-1}$.

Proof:

case 1: S_1 is matched to R_j for some $1 \leq j < k$. In this case the adversary places a request at most $(1+\epsilon)^j - 1$ to the left of every server. The cost of optimal off-line matching is $(1+\epsilon)^j - 1$ by Lemma 2.1. However, from the fact that S_1 *is matched to* R_j *for some* $1 \leq j < k$, the length of the edge (S_1, R_j) is $s_{j+1} - s_1 + (1+\epsilon)^j - 1$. Hence the cost incurred by any on-line algorithm is at least $s_{j+1} - s_1 + (1+\epsilon)^j - 1$, which is equal to $(1+\epsilon)^j - 1 + \sum_{i=1}^{j}(1+\epsilon)^i = \frac{(1+\epsilon)^j - 1}{\epsilon} \cdot (1+2\epsilon)$. The competitiveness is at least $\frac{1+2\epsilon}{\epsilon}$ which is equal to $2 + \frac{1}{2^{\frac{1}{k-1}}-1}$.

case 2: S_1 is matched to R_k. In this case, the adversary places a request exactly 1 unit to the right of every server. Therefore, the cost of optimal off-line matching is 1 by Lemma 2.1. Because of the edge (S_1, R_k) the cost incurred by any on-line algorithm and hence its competitiveness is at least $r_k - s_1 = s_k - s_1 + 1 = 1 + \sum_{i=1}^{k-1}(1+\epsilon)^i = 1 + \frac{1+\epsilon}{\epsilon}[(1+\epsilon)^{k-1} - 1] = 1 + \frac{1+\epsilon}{\epsilon}[(2^{\frac{1}{k-1}})^{k-1} - 1] = 1 + \frac{1+\epsilon}{\epsilon}[2-1] = \frac{1+2\epsilon}{\epsilon} = 2 + \frac{1}{2^{\frac{1}{k-1}}-1}$. ∎

Our lower bound is not a linear function of k but we show that it is bounded from below by $k/\ln 2$. Let us express our lower bound as $c(k)k$. For example, $2 + \frac{1}{2^{\frac{1}{k-1}}-1}$ evaluates to 3 for $k = 2$ implying that we can choose $c(2) = 1.5$. To obtain the asymptotic bound $c(k) = 1/\ln 2$, we proceed as follows. The choice of ϵ in the proof of Theorem 2.2 balances the competitiveness values in cases 1 and 2. Thus we set

$c(k)k := \frac{1+2\epsilon}{\epsilon} = 1 + \frac{1+\epsilon}{\epsilon}[(1+\epsilon)^{k-1} - 1]$
This implies $\epsilon = \frac{1}{c(k)k-2}$.
For large k, we set $\epsilon \approx \frac{1}{c(k)k}$ and obtain the underestimate
$c(k)k = c(k)k[(1 + \frac{1}{c(k)k})^k - 1]$ or
$(1 + \frac{1}{c(k)k})^k = 2$ or asymptotically
$e^{\frac{1}{c(k)k}} = 2$ or
$c(k) = \ln 2 \approx 1.44$.