Lower Bounds for Randomized k-Server and Motion-Planning Algorithms

Howard Karloff * Yuval Rabani † Yiftach Ravid †

Abstract

In this paper, we prove lower bounds on the competitive ratio of randomized algorithms for two on-line problems: the k-server problem, suggested by [MMS], and an on-line motion-planning problem due to [PY]. We prove, against an oblivious adversary,

1. an $\Omega(\log k)$ lower bound on the competitive ratio of any randomized on-line k-server algorithm in any sufficiently large metric space;
2. an $\Omega(\log \log k)$ lower bound on the competitive ratio of any randomized on-line k-server algorithm in any metric space with at least $k+1$ points; and
3. an $\Omega(\log \log n)$ lower bound on the competitive ratio of any on-line motion-planning algorithm for a scene with n obstacles.

Previously, no superconstant lower bound on the competitive ratio of randomized on-line algorithms was known for any of these problems.

1 Introduction

On-line algorithms handle sequences of events, each event being handled before future events are known. Among the on-line problems recently studied are paging (Sleator and Tarjan [ST]), on-line vertex coloring (Lovász, Saks and Trotter [LST]), metrical task systems (Borodin, Linial and Saks [BLS]),

*Department of Computer Science, University of Chicago, Chicago, IL 60637. This author was supported in part by NSF grant CCR 8807534.

†Computer Science Department, School of Mathematics, Tel-Aviv University, Tel-Aviv 69978, Israel.

the k-server problem (Manasse, McGeoch and Sleator [MMS]), layered graph traversal (Baeza-Yates, Culberson and Rawlins [BCR] and Papadimitriou and Yannakakis [PY]), and on-line motion-planning [PY].

Sleator and Tarjan [ST] suggested comparing on-line algorithms not to each other, but to an optimal off-line algorithm that knows the entire sequence in advance. This approach is called *competitive analysis*. We say a (randomized) on-line algorithm A is *c-competitive* if there is a constant a dependent on the initial configuration, but independent of the event sequence, so that for all event sequences σ, the (expected) cost incurred by A on σ is at most a plus c times the optimal cost to handle σ. The infimum of all c such that A is c-competitive is A's *competitive ratio*.

In the case of a randomized algorithm it is important to accurately define the power of the adversary (see Raghavan and Snir [RS] and Ben-David *et al.* [BBKTW]). The *adaptive off-line* adversary may adapt the sequence of requests it produces to the random choices made to date by the on-line algorithm, then pay for the entire sequence optimally. Ben-David et al. show that randomization against this adversary does not improve on-line performance compared with deterministic algorithms [BBKTW]. The *adaptive on-line* adversary adapts the request sequence to the on-line algorithm's random choices. However, it must serve each request before the on-line algorithm serves it, and therefore the cost of this adversary might be far from optimal. Ben-David et al. show that randomization against this adversary cannot help much. If there exists a c-competitive randomized algorithm against an adaptive on-line adversary, then there exists a c^2-competitive deterministic algorithm [BBKTW]. This paper deals with the *oblivious* adversary, the weakest of the three and the "traditional" adversary. In contrast to adaptive adversaries, the oblivious adversary must fix the sequence of requests in advance, then pay for it optimally. Randomization can be used by the on-line algorithm to "hide" its choices from the oblivious adversary.

We study the k-server and motion-planning problems. In the *k-server problem*, k servers move among the points of a metric space $\mathcal{M}$, serving requests. A *request* is a point of $\mathcal{M}$. To *serve* the request means to move a server to the request site. The algorithm pays a price equal to the distance moved. Requests are served *on-line*. This means that each request is served before future requests are known. Manasse *et al.* proved that in every metric space on at least $k+1$ points there is a lower bound of k on the competitive ratio of any deterministic on-line k-server algorithm [MMS]. This lower bound is applicable for randomization against adaptive adversaries as well, but not for randomization against an oblivious adversary. Manasse *et al.*

also conjectured that for every metric space and every k, there is a deterministic k-competitive on-line algorithm [MMS]. They proved their conjecture for $k = 2$ and for $k = n - 1$, where n is the cardinality of the (finite) metric space. This conjecture was proven for uniform metric spaces [ST] and for the infinite metric space which is isomorphic to the real line by Chrobak *et al.* [CKPV]. Chrobak and Larmore [CL] generalized the result to infinite metric spaces which are isomorphic to trees. Deterministic competitive algorithms for all metric spaces and all k were given by Fiat, Rabani and Ravid [FRR]. Grove [Ge] proved that the so-called Harmonic randomized server algorithm, suggested by Raghavan and Snir [RS], is competitive for all k.

A searcher in the on-line motion-planning model of Papadimitriou and Yannakakis [PY] is a point particle which starts at a point s in the Euclidean plane and moves to a known target. In our version of the problem, the target is a vertical line, and the searcher need only reach a point of his own choosing on it. However, the plane is peppered with stationary open rectangular obstacles which are disjoint from each other and from the source s and target t. Each rectangle has integral side lengths. The searcher can "see" only those obstacles which are connected by an obstacle-free line segment to the searcher's position. We compare the cost incurred by the searcher to the length of a shortest obstacle-free source-target path, which is the cost incurred by an optimal algorithm that sees the entire scene. Papadimitriou and Yannakakis gave an $\Omega(\sqrt{n})$ lower bound on the ratio between the cost of a deterministic on-line algorithm and the optimal cost in a scene containing n rectangles [PY].

The use of randomization against an oblivious adversary has indeed led to superior algorithms. Fiat *et al.* exhibited a randomized paging algorithm with a competitive ratio bounded by $2\mathcal{H}_k$, where $\mathcal{H}_k = 1 + \frac{1}{2} + \cdots + \frac{1}{k} \sim \ln k$ [FKLMSY] (a different algorithm was subsequently proven $\mathcal{H}_k$-competitive by McGeoch and Sleator [MS]). This is to be contrasted with the lower bound of k for a deterministic paging algorithm. Fiat *et al.* also proved a lower bound of $\mathcal{H}_k$ for the competitive ratio of randomized paging algorithms [FKLMSY]. They conjectured that $\mathcal{H}_k$ is an upper bound for all metric spaces and all k. This conjecture was disproved by Karlin *et al.* [KMMO], who exhibited a family of 3-point metric spaces and a lower bound approaching $\frac{e}{e-1} > 1.5 = \mathcal{H}_2$ for the competitive ratio of any randomized 2-server algorithm for those metric spaces.

Vishwanathan showed how to color 3-colorable graphs on-line with only $O(\sqrt{n \log n})$ colors [Vn], a great improvement over the

$O((n \log\log\log n)/\log\log n)$ bound of [LST]. An exponential improvement in the performance of algorithms that traverse certain layered graphs was exhibited by Fiat *et al.* [FFKRRV].

No superconstant lower bound was known for the competitive ratio of general metric spaces (with at least $k+1$ points). Indeed, several researchers conjectured that constant-competitive randomized k-server algorithms exist for certain infinite metric spaces. No superconstant lower bound for the on-line motion-planning problem was known either.

We prove two k-server lower bounds:

1. The competitive ratio of any randomized k-server algorithm for any sufficiently large metric space is $\Omega(\log k)$.

2. The competitive ratio of any randomized k-server algorithm for *any* metric space with at least $k+1$ points is $\Omega(\log\log k)$.

In light of the [FKLMSY] algorithm, the first result is tight to within a constant factor.

Blum, Raghavan and Schieber [BRS] used randomized k-server algorithms to construct randomized on-line motion-planning algorithms. Instead, we adapt randomized k-server lower bounds to prove a lower bound for the on-line motion planning problem:

3. The competitive ratio of any randomized motion-planning algorithm is $\Omega(\log\log n)$, where n is the number of obstacles in the scene.

The proof of the first result goes as follows. First, we prove that for a *superincreasing* metric space $\mathcal{M}(k)$, a metric space on $\{z_0, z_1, \ldots, z_k\}$ with the distance from z_i to z_{i+1} much greater than that from z_{i-1} to z_i, no sublogarithmic competitive ratio is possible. This part of the proof generalizes techniques of Karlin *et al.* [KMMO]. Next, we prove a "Ramsey-like" theorem for metric spaces: every sufficiently large metric space $\mathcal{M}$ contains a $(k+1)$-point subspace resembling either the superincreasing metric space or the uniform metric space. From the $\Omega(\log k)$ lower bounds for the superincreasing and uniform metric spaces we construct an $\Omega(\log k)$ lower bound for $\mathcal{M}$.

For $k+1$-point metric spaces, we will use the Ramsey-like theorem to generate a set $\mathcal{S}$ of $s+1$ points (where $s = \left\lceil \sqrt{\lg(k+1)} \right\rceil$) that resembles either the superincreasing or uniform metric space. The $k-s$ servers initially on the $k-s$ points outside of $\mathcal{S}$ can be fixed at their initial locations by

replacing a request to a point $z \in \mathcal{S}$ by a long sequence of requests to z and all the points outside of $\mathcal{S}$. Thus the on-line algorithm can be forced to use only s servers on the specified $s+1$ points. We can use the first result to get an $\Omega(\log s)$ bound, and this is $\Omega(\log \log k)$.

The idea for the motion-planning result is to construct an obstacle scene in which the distance an on- or off-line server moves to reach the target is approximately the cost a k-server algorithm incurs when serving a sequence of requests in a metric space that resembles $\mathcal{M}(k)$. The lower bound for $\mathcal{M}(k)$ will yield a lower bound on the competitive ratio of motion planning algorithms.

2 The Superincreasing Metric Space

Define $c_1 = 1$ and let

$$c_i = c_{i-1}\left[1 + \frac{1}{2e^{2c_{i-1}} - 1}\right]$$

for $i \geq 2$.

Definition. Let $k \geq 0$ and let $\mathcal{M}(k)$ be a metric space on $k+1$ points $z_0, z_1, z_2, \ldots, z_k$. Suppose that there are integers $d_1 < d_2 < d_3 < \cdots < d_k$ so that $d_1 = 1$ and that for $i < j$, $\text{dist}(z_i, z_j) = d_j$. Then the sequence $d_1, d_2, d_3, \ldots, d_k$ and the metric space $\mathcal{M}(k)$ are called *superincreasing* if $d_i \geq (4c_{i-1}e^{2c_{i-1}})d_{i-1}$ for every $i \geq 2$.

We will often use $\mathcal{M}(k)$ to mean any superincreasing metric space on $k+1$ points.

For metric space $\mathcal{M}(k)$, we prove that $c_k - 1$ is a lower bound on the competitive ratio of any on-line (randomized) k-server algorithm.

The following lemmas show that c_k is $\Theta(\log k)$.

Lemma 1 *$c_k > \frac{1}{2}\ln k$ for all k.*

Proof. The definition of c_i gives the following:

$$c_k - c_{k-1} = \frac{c_{k-1}}{2e^{2c_{k-1}} - 1},$$

or

$$(c_k - c_{k-1})(2e^{2c_{k-1}} - 1) = c_{k-1} \geq 1,$$

or

$$(c_k - c_{k-1})e^{2c_{k-1}} \geq \frac{1}{2}.$$

Now, if f is a nondecreasing continuous function and $x_1 < x_2 < \cdots < x_r$, then

$$\int_{x_1}^{x_r} f(x)dx \geq \sum_{i=2}^{r}(x_i - x_{i-1})f(x_{i-1}).$$

Thus

$$\int_{c_1}^{c_k} e^{2x}dx \geq \sum_{j=2}^{k}(c_j - c_{j-1})e^{2c_{j-1}} \geq \frac{k-1}{2}.$$

Therefore

$$\frac{e^{2c_k} - e^{2c_1}}{2} \geq \frac{k-1}{2}.$$

$$e^{2c_k} \geq e^2 + (k-1) > k.$$

$$2c_k > \ln k.$$

$$c_k > \frac{1}{2}\ln k. \blacksquare$$

Lemma 2 *For all k, $c_k \leq 1 + 1.5\ln k$.*

Proof. For all y, $1 + y \leq e^y$. Therefore $1 + y \leq 2e^{2y}$ for all $y \geq 0$.

$$1 + c_{k-1} \leq 2e^{2c_{k-1}}$$

$$c_{k-1} \leq 2e^{2c_{k-1}} - 1$$

Therefore

$$\frac{c_{k-1}}{2e^{2c_{k-1}} - 1} \leq 1$$

$$c_k - c_{k-1} = \frac{c_{k-1}}{2e^{2c_{k-1}} - 1} \leq 1$$

Therefore $c_k \leq k$ for all k.

$$(c_k - c_{k-1})2e^{2c_{k-1}} = c_{k-1} + (c_k - c_{k-1}) = c_k.$$

$$\begin{aligned}(c_k - c_{k-1})2e^{2c_k} &\leq (c_k - c_{k-1})2e^{2+2c_{k-1}} \\ &= e^2(c_k - c_{k-1})2e^{2c_{k-1}} \\ &= e^2 c_k \leq ke^2.\end{aligned}$$

Therefore

$$(c_k - c_{k-1})2e^{2c_k} \le ke^2.$$

$$(c_k - c_{k-1})e^{2c_k} \le \frac{ke^2}{2}.$$

If f is continuous and nondecreasing and $x_1 < x_2 < \cdots < x_r$, then

$$\int_{x_1}^{x_r} f(x)dx \le \sum_{i=2}^{r}(x_i - x_{i-1})f(x_i).$$

Hence

$$\int_{c_1}^{c_k} e^{2x}dx \le \sum_{i=2}^{k}(c_i - c_{i-1})e^{2c_i} \le \sum_{i=2}^{k}\frac{ke^2}{2} \le \frac{k^2e^2}{2}.$$

$$\frac{e^{2c_k} - e^{2c_1}}{2} \le \frac{k^2e^2}{2}.$$

$$e^{2c_k} \le e^2 + k^2e^2 = e^2(k^2+1).$$

Hence

$$c_k \le \frac{2 + 3\ln k}{2} = 1 + 1.5\ln k. \blacksquare$$

Let us generalize the k-server problem. The *multipoint k-server problem* is the problem of serving requests each of which consists of a *set* of at most k points. (A request to $\{z\}$ will be abbreviated as z.) To serve the request means to move one or more servers so that *all* the points in the requested set are covered. We will only study multipoint k-server problems on $(k+1)$-point metric spaces. This allows us to assume without loss of generality that the algorithm is *lazy*: it never moves more than one server to serve a request, and if the requested set is already covered, it does nothing.

In $\mathcal{M}(k)$, let τ_i be the set $\{z_1, z_2, z_3, \ldots, z_i\}$. Let γ_i be the set $\{z_0, z_1, z_2, \ldots, z_{i-1}\}$. Let $N_1, N_2, N_3, \ldots$ denote the integer sequence $N_1 = 2$ and $N_i = (2d_i)N_{i-1} + 2$ for $i \ge 2$.

Fix k and a lazy, randomized multipoint k-server algorithm A for $\mathcal{M}(k)$. If $1 \le i \le k$, say a request sequence σ is *i-convergent for A*, or simply *i-convergent*, if A covers τ_i with probability 1 just after σ is served.

Lemma 3 *For each $1 \le i \le k$, for each i-convergent request sequence σ for $\mathcal{M}(k)$, there is a request sequence Δ with the following properties:*

1. *Δ consists of a request sequence for $\mathcal{M}(i-1)$ preceded immediately by a request to z_0 and followed immediately by a request to τ_i. (Thus $\sigma\Delta$ is i-convergent.)*

2. *The length of* Δ *is at most* N_i.

3. *Suppose* A *serves request sequence* $\sigma\Delta$. *Let* $s_1, s_2, \ldots, s_i$ *be the servers that occupy* $\{z_1, z_2, \ldots, z_i\}$ *just after* σ *is served,* s_j *occupying* z_j. *Let* $t = P[z_0$ *is vacant just after* A *has served* $\sigma\Delta]$. *Define the* i-cost *of* Δ *to be the cost incurred by* $s_1, s_2, \ldots, s_i$ *during the time* A *is serving the* Δ *of* $\sigma\Delta$. *Let* $w \in \{1, 2, ..., 2d_i\}$ *be the optimal cost of serving* Δ *by an* i*-server algorithm whose servers start on* $\{z_1, \ldots, z_i\}$. *Then the expected* i*-cost of* Δ *is at least* $c_i w t$.

Intuitively, the reason why such a costly sequence Δ exists is that the on-line algorithm does not know in advance how many requests in $\mathcal{M}(i-1)$ will be given before the request to τ_i. Suppose that z_0 is vacant prior to the arrival of Δ. If only few requests in $\mathcal{M}(i-1)$ appear, it makes little sense to move a server on one of the distant points $\{z_i, \ldots, z_k\}$ and pay a high price. If there are many requests, however, a distant server must be moved, so as to avoid indefinitely having to shuffle $i-1$ servers among the i points of $\mathcal{M}(i-1)$.

To prove Lemma 3, we need a technical lemma, the proof of which appears in the appendix.

Lemma 4 *Let* $2 \leq i \leq k$. *Let* ℓ *be an integer,* $\ell \geq c_{i-1}$. *Let* $d \geq 4c_{i-1}e^{2c_{i-1}}\ell$ *be an integer. Let* $w_1, w_2, ..., w_Q \in \{1, 2, ..., 2\ell\}$ *where* Q *satisfies* $\sum_{i=1}^{Q-1} w_i < 2d \leq \sum_{i=1}^{Q} w_i$. *Let* $p_1, p_2, ..., p_{Q-1} \in [0, 1]$.

Then either

$$d + c_{i-1} \sum_{s=1}^{Q-1} p_s w_s \geq c_i(2d)$$

or there is an $h \in \{1, 2, ..., Q-1\}$ *such that*

$$d(1 - p_h) + c_{i-1} \sum_{s=1}^{h} p_s w_s \geq c_i \sum_{s=1}^{h} w_s.$$

Proof of Lemma 3. By induction on i.
Basis: $i = 1$. Let $\Delta = z_0 z_1$. The optimal cost w of serving Δ by a 1-server algorithm whose server starts on z_1 is 2. Algorithm A leaves z_0 vacant after serving $\sigma\Delta$ with probability t, so with probability at least t server s_1 must have served both requests, incurring an expected i-cost of at least $2t = c_i w t$.
Inductive Step: $i > 1$. Let σ be an i-convergent request sequence for $\mathcal{M}(k)$. It is also $(i-1)$-convergent. By induction, there is a request sequence Δ_1 (of

length at most N_{i-1}) consisting of a request sequence for $\mathcal{M}(i-2)$ preceded by a request to z_0 and followed by a request to τ_{i-1} such that the following holds. If A serves $\sigma\Delta_1$, the expected $(i-1)$-cost incurred while serving Δ_1 is at least $c_{i-1}w_1t_1$ (where $w_1 \in \{1, 2, ..., 2d_{i-1}\}$ is the optimal cost of serving Δ_1 with $i-1$ servers and t_1 is the probability that z_0 is vacant after $\sigma\Delta_1$ is served).

Since $\sigma\Delta_1$ is $(i-1)$-convergent, we can apply induction again. Thus there is a Δ_2 (of length at most N_{i-1}) consisting of a request sequence for $\mathcal{M}(i-2)$ preceded by a request to z_0 and followed by a request to τ_{i-1} such that the following holds. If A serves $\sigma\Delta_1\Delta_2$, the expected $(i-1)$-cost incurred while serving Δ_2 is at least $c_{i-1}w_2t_2$ (where $w_2 \in \{1, 2, ..., 2d_{i-1}\}$ is the optimal cost of serving Δ_2 with $i-1$ servers and t_2 is the probability that z_0 is vacant after $\sigma\Delta_1\Delta_2$ is served).

Since $\sigma\Delta_2$ is $(i-1)$-convergent, we can apply induction again. Thus there is a Δ_3 (of length at most N_{i-1}) consisting of a request sequence for $\mathcal{M}(i-2)$ preceded by a request to z_0 and followed by a request to τ_{i-1} such that the following holds. If A serves $\sigma\Delta_1\Delta_2\Delta_3$, the expected $(i-1)$-cost incurred while serving Δ_3 is at least $c_{i-1}w_3t_3$ (where $w_3 \in \{1, 2, ..., 2d_{i-1}\}$ is the optimal cost of serving Δ_3 with $i-1$ servers and t_3 is the probability that z_0 is vacant after $\sigma\Delta_1\Delta_2\Delta_3$ is served).

Repeat this process, getting $\Delta_1, \Delta_2, \ldots, \Delta_Q$ and $w_1, w_2, \ldots, w_Q$ such that $w_1 + w_2 + \cdots + w_Q \geq 2d_i$, but $w_1 + \cdots + w_{Q-1} < 2d_i$. Now change Δ_Q: replace it by $\Delta_Q\gamma_i$.

Let "time j" mean "just after $\sigma\Delta_1 \cdots \Delta_j$ has been served." Let $t_j = P[z_0$ is vacant at time $j]$. Let $u_j = P[z_i$ is vacant at time $j]$ and let $r_j = t_j + u_j$, the probability that either z_0 or z_i is vacant at time j. Because $r_j = P[z_{i+1}, z_{i+2}, \ldots, z_k$ are occupied at time $j]$ (even for $j = Q$), we have $r_1 \geq r_2 \geq \cdots \geq r_Q$. Let

$$q_j = \begin{cases} u_j/r_j & \text{if } r_j \neq 0 \\ 1 & \text{otherwise} \end{cases}.$$

Note that $r_jq_j = u_j$ always. The expected $(i-1)$-cost of phase $j < Q$ is at least

$$c_{i-1}w_jt_j = c_{i-1}w_j(r_j - u_j) = c_{i-1}w_jr_j(1 - q_j).$$

Clearly $t_Q = 0$. Therefore $r_Q = u_Q$, so $q_Q = 1$. (Of course the expected $(i-1)$-cost of phase Q is at least $0 = c_{i-1}w_Qr_Q(1 - q_Q)$.) $\Delta_1\Delta_2 \cdots \Delta_Q$ is a request sequence for $\mathcal{M}(i-1)$ preceded by a request to z_0.

Choose $1 \leq h \leq Q$. The expected cost incurred by s_i to serve $\Delta_1\Delta_2 \cdots \Delta_h\tau_i$ (after A has already served σ) is at least d_iu_h, since z_i is

occupied just after σ is served. Thus the expected i-cost, if we have $h \le Q$ phases, is at least

$$\begin{aligned} & d_i u_h + \textstyle\sum_{j=1}^{h} c_{i-1} w_j r_j (1 - q_j) \\ \ge\ & r_h (d_i q_h) + c_{i-1} r_h \textstyle\sum_{j=1}^{h} w_j (1 - q_j) \\ =\ & r_h (d_i q_h + c_{i-1} \textstyle\sum_{j=1}^{h} w_j (1 - q_j)) \end{aligned}$$

and $q_Q = 1$. When $h = Q$, this last quantity is

$$r_Q (d_i + c_{i-1} \sum_{j=1}^{Q-1} w_j (1 - q_j)).$$

By Lemma 4, with $\ell = d_{i-1}$, $d = d_i$ and $p_s = 1 - q_s$ for all s, either

$$d_i + c_{i-1} \sum_{j=1}^{Q-1} w_j (1 - q_j) \ge c_i (2 d_i)$$

or there is an $h \in \{1, 2, \ldots, Q-1\}$ such that

$$d_i q_h + c_{i-1} \sum_{j=1}^{h} w_j (1 - q_j) \ge c_i \sum_{j=1}^{h} w_j.$$

In the former case, the optimal cost incurred by an adversary having i servers in serving $\Delta_1 \Delta_2 \cdots \Delta_Q \tau_i$ after serving σ equals $2d_i$. A's expected i-cost to serve the same sequence after serving σ is at least $r_Q[c_i(2d_i)]$. Let

$$\Delta = \Delta_1 \Delta_2 \cdots \Delta_Q \tau_i.$$

If z_0 is vacant after $\sigma\Delta$ is served, then either z_0 or z_i is vacant after $\sigma \Delta_1 \Delta_2 \cdots \Delta_Q$ is served. Thus the probability t that z_0 is vacant after $\sigma\Delta$ is served is at most r_Q, and therefore Δ suffices, since Δ's length is at most $(2d_i)N_{i-1} + 2 = N_i$. ($Q \le 2d_i$ since $w_j \ge 1$ for all j.)

In the latter case, the optimal cost incurred by an i-server adversary in serving $\Delta_1 \Delta_2 \cdots \Delta_h \tau_i$ after serving σ is at most $\sum_{j=1}^{h} w_j$. A's expected i-cost to serve the same sequence after serving σ is at least $r_h c_i \sum_{j=1}^{h} w_j$. Let

$$\Delta = \Delta_1 \Delta_2 \cdots \Delta_h \tau_i.$$

If z_0 is vacant after $\sigma\Delta$ is served, then either z_0 or z_i is vacant after $\sigma \Delta_1 \Delta_2 \cdots \Delta_h$ is served. Thus the probability t that z_0 is vacant after $\sigma\Delta$ is served is at most r_h, and therefore Δ suffices. ■

We use the notation $\mathrm{OPT}(\sigma)$ to denote the optimal off-line multipoint cost to serve σ, and we use $A(\sigma)$ to denote the (random) cost of our on-line multipoint algorithm A to serve σ.

Theorem 5 *For all r, there is a multipoint request sequence σ_r of length at most rN_k and optimal cost at least r such that*

$$E[A(\sigma_r)] \geq c_k \cdot \mathrm{OPT}(\sigma_r).$$

Proof. Take $i = k$. Build request sequence $\sigma_r = \Delta_1\Delta_2\Delta_3 \cdots \Delta_r$ via repeated applications of Lemma 3, by constructing Δ_1, then Δ_2, then Δ_3, and so on. Each Δ_j has length at most N_k. The optimal cost of serving Δ_j with k servers is $w_j \geq 1$. Each time the lemma is applied, $t = 1$.

$$\mathrm{OPT}(\sigma_r) = w_1 + w_2 + \cdots + w_r.$$

(We have equality because each Δ_j ends with τ_k.) The expected value of $A(\sigma_r)$ is at least $c_k w_1 + c_k w_2 + \cdots + c_k w_r = c_k \cdot \mathrm{OPT}(\sigma_r)$. ■

Corollary 6 *There is no c-competitive multipoint k-server algorithm for $\mathcal{M}(k)$ if $c < c_k$.*

Definition. Let $\mathcal{M}$ be a metric space on the $k+1$ points $\{z_0, z_1, ..., z_k\}$. Let $S \underset{\neq}{\subset} \mathcal{M}$, $S \neq \emptyset$. Say an *S-request* is a sequence of requests to all $|S|$ points in S, one at a time, in increasing order by index.

Definition. Let $\mathcal{M}$ be a metric space on the $k+1$ points $\{z_0, z_1, ..., z_k\}$. We say a (single point) server algorithm A for $\mathcal{M}$ is *finitely converging* if it has the following property. Let α be a request sequence and let $S \underset{\neq}{\subset} \mathcal{M}$, $S \neq \emptyset$. If A serves a sequence consisting of α followed by enough S-requests, then at the end all the points in S are occupied with probability one.

Lemma 7 *Let $\mathcal{M}$ be a metric space on $k+1$ points. If there is a lazy, c-competitive, finitely converging algorithm for the single-point k-server problem on $\mathcal{M}$, then there is a lazy, c-competitive multipoint k-server algorithm for $\mathcal{M}$.*

Proof. Suppose A is a lazy, finitely converging c-competitive k-server algorithm for $\mathcal{M}$. We argue that there is a lazy, c-competitive multipoint k-server algorithm B for $\mathcal{M}$. Let σ be a multipoint request sequence for $\mathcal{M}$.

B simulates A on a single point request sequence σ' for $\mathcal{M}$. B constructs σ' on the fly by replacing a request to the set S in σ by a long sequence of S-requests. The number of these S-requests is chosen so large that A is known to cover S with probability one after serving the S-requests. Further, the number is chosen so large that even the adversary is known to cover S afterward.

To serve a request to S, B flips coins for A and "watches" A's behavior on the long string of S-requests. At the end, B moves to A's configuration by moving at most one server.

Let OPT and OPT$'$ denote the optimal costs of a multipoint and single point request sequence, respectively. $B(\sigma) \leq A(\sigma')$ and hence

$$E[B(\sigma)] \leq E[A(\sigma')] \leq c \cdot \mathrm{OPT}'(\sigma') + a$$

for a suitable a. But $\mathrm{OPT}(\sigma) = \mathrm{OPT}'(\sigma')$, and thus B is a lazy, c-competitive, multipoint k-server algorithm for $\mathcal{M}$. ■

Theorem 8 *There is no lazy, finitely converging c-competitive algorithm for the single point k-server problem on $\mathcal{M}(k)$ if $c < c_k$.*

Proof. Follows from Corollary 6 and Lemma 7. ■

Now we relate finitely converging and non-finitely converging algorithms.

Lemma 9 *Suppose that A is a lazy, c-competitive k-server algorithm for a $(k+1)$-point metric space M. Then there is a lazy, finitely converging $(c+1)$-competitive k-server algorithm A' for M.*

Proof sketch. Let the points of M be ordered $z_0, z_1, ..., z_k$. Without loss of generality, suppose that the minimum nonzero distance in M equals one and the maximum distance equals, say, D. Choose $a \geq 0$ such that $E[A(\sigma)] \leq c \cdot \mathrm{OPT}(\sigma) + a$ for all σ.

At all times, A' simulates either A or the deterministic, lazy, k-competitive algorithm BAL of [MMS], initially the former. A long sequence of S-requests will ensure that A fails to occupy all the points of S with probability approaching zero, for otherwise it could not be competitive. (Indeed, enough S-requests will ensure that any given competitive algorithm, on-line or off-line, occupies S with probability approaching one. The adversary himself can be forced to occupy S in this way.)

At all times, A' attempts to write the list of requests seen to date, including the current request, as τS^L, where

$$L = L(\tau) = 1 + \lceil k^2 D|\tau|^2(c|\tau| + 2cD + a)\rceil.$$

(Here, S represents an S-request.) If it fails, it simply flips coins and serves the request as A would have. If it succeeds—and in this case τ and S are unique—A' flips its coins to simulate A on the current request. If the coin flips dictate that A move to a configuration covering S, A' continues to mimic A. Otherwise, A' switches to BAL in a lazy way, never to return to A. If the next $|S|$ requests are an S-request, then L is so large that after those requests are served A' covers S. Thus A' covers S with probability one by the time it has served τS^{L+1}.

It is not hard to prove that L is so large that the probability that A' switches to BAL when the list of requests seen to date is of the form τS^L is at most $1/(k^2 D|\tau|^2)$. Therefore the probability that A' ever switches to BAL is at most $\sum_{l=1}^{\infty} 1/(k^2 D l^2) \le 2/(k^2 D)$. Now $A'(\sigma) = A(\sigma)$ if A' never switches to BAL, and $A'(\sigma) \le A(\sigma) + (D + \mathrm{BAL}(\sigma))$ if A' does switch.

$$\begin{aligned} D + \mathrm{BAL}(\sigma) &\le D + (k \cdot \mathrm{OPT}(\sigma) + \binom{k}{2} D) \\ &\le k \cdot \mathrm{OPT}(\sigma) + (k^2/2)D. \end{aligned}$$

Therefore

$$\begin{aligned} E[A'(\sigma)] &\le E[A(\sigma)] + (k \cdot \mathrm{OPT}(\sigma) + \frac{k^2}{2} D) \cdot P[A' \text{ switches to BAL}] \\ &\le E[A(\sigma)] + \frac{2}{kD} \cdot \mathrm{OPT}(\sigma) + 1 \\ &\le E[A(\sigma)] + (\mathrm{OPT}(\sigma) + 1) \\ &\le (c+1) \cdot \mathrm{OPT}(\sigma) + (a+1). \blacksquare \end{aligned}$$

Theorem 10 *There is no c-competitive algorithm A for the single point k-server problem on $\mathcal{M}(k)$ if $c < c_k - 1$.*

Proof. Lemma 9 proves that the existence of a c-competitive algorithm implies the existence of a lazy, finitely converging $(c+1)$-competitive algorithm. Theorem 8 completes the proof. $\blacksquare$

3 Ramsey Theory for Metric Spaces

This section deals with the structure of metric spaces. Theorem 12 shows that every metric space of cardinality n contains either a roughly uniform subset of at least $\frac{1}{2}\lg n$ points, or an approximately superincreasing sequence of $\Omega((\lg n)/\lg\lg n)$ points.

Lemma 11 *Let $\mathcal{M}$ be a metric space on n points, $w(e)$ denoting the length of edge e, and let $c > 1$ be a real. $\mathcal{M}$ contains either*

1. *a subset S of size at least $s = \lg n$ such that the distances within S differ by no more than a factor of c^2, or*
2. *a point P and a subset T of size at least $t = n/\lg n$ such that $P \notin T$ and*
$$\frac{\min_{x\in T} d(x,P)}{\max_{x,y\in T} d(x,y)} \geq \frac{c}{4} - \frac{1}{2}.$$

Proof. We may assume $n \geq 3$. Label edge e with $\lfloor \log_c w(e) \rfloor$. Let j be the largest label, and call those edges labeled j or $j-1$ *large* and the rest, if any, *small*. Build a graph G on the points of $\mathcal{M}$: $\{u,v\} \in E(G)$ if and only if $\{u,v\}$ is small in $\mathcal{M}$. The value of d, a nonnegative real, will be chosen later.

If every vertex in G has degree at most d, then G has an independent set S of size at least $n/(d+1)$. (The greedy independent set algorithm generates such an S.) If u and v are distinct points of S, $\{u,v\}$ is large. But the lengths of two large edges cannot differ by more than a factor of c^2.

Otherwise, some vertex v has degree exceeding d. Let $\{a,b\}$ be the longest edge in the metric space, labeled so that $d(a,v) \geq d(b,v)$; $d(a,v) \geq \frac{1}{2}d(a,b) \geq \frac{1}{2}c^j$. Let T be v together with its neighborhood in G. All edges between points in T are of length less than $2c^{j-1}$. If $x \in T$,

$$d(a,x) \geq d(a,v) - d(v,x) \geq \frac{1}{2}c^j - c^{j-1}.$$

Thus
$$\frac{\min_{x\in T} d(x,a)}{\max_{x,y\in T} d(x,y)} \geq \frac{\frac{1}{2}c^j - c^{j-1}}{2c^{j-1}} = \frac{c}{4} - \frac{1}{2}.$$

We have found either a set S of size at least $n/(d+1)$ whose interpoint distances differ by at most a factor of c^2, or a set T for which

$$\frac{\min_{x\in T} d(x,a)}{\max_{x,y\in T} d(x,y)} \geq \frac{c}{4} - \frac{1}{2}$$

with $|T| > 1 + d$. Taking $d = (n/\lg n) - 1$ and $P = a$, the proof is complete. ■

Theorem 12 *Let $\mathcal{M}$ be a metric space on n points and let $c > 1$. $\mathcal{M}$ contains either*

1. *a subset of size at least $\frac{1}{2}\lg n$ whose interpoint distances differ by at most a factor of c^2; or*
2. *a sequence of distinct vertices $P_1, P_2, \ldots, P_t$ for $t = \lceil \frac{1}{2}(\lg n)/\lg\lg n \rceil - 1$ such that*
$$\frac{\min_{j>i} d(P_j, P_i)}{\max_{j>i+1} d(P_j, P_{i+1})} \geq \frac{c}{4} - \frac{1}{2}$$
for $i = 1, 2, \ldots, t-2$.

Proof. Construct a sequence of metric spaces $\mathcal{M} = \mathcal{M}_1, \mathcal{M}_2, \mathcal{M}_3, \ldots, \mathcal{M}_r$ and a set of points $P_1, P_2, \ldots, P_{r-1}$ (P_i in $\mathcal{M}_i$), as follows. Apply Lemma 11 to $\mathcal{M}_i$. If case 1 is true (where $n = |\mathcal{M}|$, not $|\mathcal{M}_i|$), halt. If case 2 is true but T has fewer than $\sqrt{n}$ points, halt without constructing $\mathcal{M}_{i+1}$. Otherwise, let P_i be the point P of case 2, and let $\mathcal{M}_{i+1}$ be the metric space induced by T.

If case 1 is ever true, we have a set of at least $\lg\sqrt{n} = \frac{1}{2}\lg n$ points within the current metric space whose interpoint distances differ by at most c^2. Otherwise, since

$$|\mathcal{M}_{i+1}| \geq \frac{|\mathcal{M}_i|}{\lg|\mathcal{M}_i|} \geq \frac{|\mathcal{M}_i|}{\lg n},$$

the number r of metric spaces we construct satisfies $n/(\lg n)^r < \sqrt{n}$, i.e., $r > \frac{1}{2}(\lg n)/\lg\lg n$. The $r - 1$ P's satisfy Condition 2 above. ■

4 General Lower Bounds

Lemma 13 *Let $\mathcal{M}$ and $\mathcal{M}'$ be two metric spaces defined on the same set of points with distance functions d and d', respectively. Let $b \geq 1$. Suppose that for every two points x, y, $d'(x, y) \leq d(x, y) \leq b \cdot d'(x, y)$. Let c be a lower bound on the competitive ratio for $\mathcal{M}'$. Then c/b is a lower bound on the competitive ratio for $\mathcal{M}$.*

Proof. The cost of serving σ in $\mathcal{M}$ is bounded above and below by b times and one times the cost of serving it in $\mathcal{M}'$, respectively. ■

Lemma 14 *Let $b \geq 1$ and let $\mathcal{M}$ be a metric space where*

$$\frac{\max_{x,y \in \mathcal{M}} \operatorname{dist}(x,y)}{\min_{x \neq y \in \mathcal{M}} \operatorname{dist}(x,y)} \leq b.$$

Then the competitive ratio for any randomized on-line k-server algorithm for $\mathcal{M}$ is at least $\mathcal{H}_k/b$, where $\mathcal{H}_k = 1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{k}$ is the kth harmonic number.

Proof. Scale the distances in $\mathcal{M}$ so that the minimum nonzero distance is 1. Apply Lemma 13 and the lower bound of $\mathcal{H}_k$ for a uniform metric space [FKLMSY]. ■

Lemma 15 *Let $\mathcal{M}$ be a metric space defined on the $k+1$ points $\{x_0, \ldots, x_k\}$ by the distance function d, which satisfies*

1. *$d(x_0, x_1) = 1$, and*
2. *for every i, $1 < i \leq k$,*

$$\frac{\min_{j<i} d(x_j, x_i)}{\max_{j<i-1} d(x_j, x_{i-1})} \geq 8c_{i-1}e^{2c_{i-1}}.$$

Then $(c_k - 1)/4$ is a lower bound on the competitive ratio of any on-line randomized k-server algorithm serving requests in $\mathcal{M}$.

Proof. For all i, $1 \leq i \leq k$, define $a_i = \lceil \min_{0 \leq j < i} d(x_j, x_i) \rceil$. Conditions 1 and 2 imply that the sequence $a_1, a_2, \ldots$ is superincreasing. Let $\mathcal{M}'$ be the superincreasing metric space defined on $\{x_0, \ldots, x_k\}$ by setting $\operatorname{dist}(x_j, x_i) = a_i$ for $j < i$. It is not hard to prove that $\frac{1}{2}a_i \leq \operatorname{dist}(x_j, x_i) \leq a_1 + a_2 + \cdots + a_i \leq 2a_i$ for all $j < i$. Theorem 10 and Lemma 13 complete the proof. ■

Lemma 16 *Let $t > 2$ and let $P_0, P_1, \ldots, P_{t-1}$ be a sequence of points in a metric space such that $\operatorname{dist}(P_0, P_1) = 1$ and for every i, $1 < i \leq t-1$, $(\min_{j<i} \operatorname{dist}(P_j, P_i))/(\max_{j<i-1} \operatorname{dist}(P_j, P_{i-1})) \geq 2$. If $r > 1$, then there is a subset $\{Q_0, Q_1, \ldots, Q_{u-1}\}$ of size $u \geq t/(1 + \lg r)$ so that for every i, $1 < i \leq u$,*

$$\frac{\min_{j<i} \operatorname{dist}(Q_j, Q_i)}{\max_{j<i-1} \operatorname{dist}(Q_j, Q_{i-1})} \geq r.$$

Proof sketch. Take every $\lceil \lg r \rceil$th point of the P's. ■

Theorem 17 *If $n = |\mathcal{M}|$ is sufficiently large, then there is a lower bound of $\Omega(\log k)$ on the competitive ratio of any randomized on-line k-server algorithm for $\mathcal{M}$.*

Proof. Apply Theorem 12 to $\mathcal{M}$ with $c = 10$. If case 1 holds and $\frac{1}{2}\lg n \geq k+1$, we apply Lemma 14 to obtain a lower bound of $\mathcal{H}_k/100$. So suppose case 2 holds. Lemma 2 implies that $c_i \leq 1 + 1.5 \ln i$, so that $8c_i e^{2c_i} \leq 8(1 + 1.5 \ln i)e^2 i^3$. Define $r = 8(1 + 1.5 \ln k)e^2 k^3$. If

$$\frac{\lceil \frac{1}{2}(\lg n)/\lg\lg n \rceil - 1}{1 + \lg r} \geq k+1,$$

then Theorem 12, Lemma 15, and Lemma 16 give us a lower bound of $\frac{1}{4}(c_k - 1) \geq \frac{1}{8}\ln k - \frac{1}{4}$. It is clear that there is a polynomial $p(k)$ so that if $n \geq 2^{p(k)}$, then

$$\frac{\lceil \frac{1}{2}(\lg n)/\lg\lg n \rceil - 1}{1 + \lg r} \geq k+1. \blacksquare$$

Theorem 18 *For any metric space with at least $k+1$ points, there is a lower bound of $\Omega(\log\log k)$ on the competitive ratio of every randomized on-line k-server algorithm.*

Proof. Let $\mathcal{M}$ be a metric space on exactly $k+1$ points. (Ignore any others.) Let $s = \left\lceil \sqrt{\lg(k+1)} \right\rceil$. The technique of Theorem 17 can be used to construct an $(s+1)$-point metric space $\mathcal{S}$ within $\mathcal{M}$ whose interpoint distances either differ by at most a factor of 100, or which "grow" by at least a factor of s^4. (This holds for sufficiently large k.) In either case we have a lower bound of $f(s)$ on the competitive ratio of any randomized s-server algorithm for $\mathcal{S}$, where $f(s)$ is $\Omega(\log s)$ and hence $\Omega(\log\log k)$. However, the algorithm has k servers, not s.

Suppose there is a lazy, finitely converging c-competitive k-server algorithm for $\mathcal{M}$. Then by Lemma 7 we infer that there is a lazy, c-competitive multipoint k-server algorithm A for $\mathcal{M}$. Then there is a c-competitive s-server algorithm A' for $\mathcal{S}$: A' simply replaces a request to $z \in \mathcal{S}$ by a request to the set $\{z\} \cup (\mathcal{M} - \mathcal{S})$ and feeds the request to A. Neither A nor the adversary will ever move any server initially outside of $\mathcal{S}$. It follows easily that A' is a c-competitive s-server algorithm for $\mathcal{S}$. However, for $c < f(s)$, this cannot be. Now Lemma 9 tells us that no $(f(s)-1)$-competitive k-server algorithm for $\mathcal{M}$ can exist, finitely converging or not. ■

5 On-Line Motion-Planning

In this section we study the on-line motion-planning problem of Papadimitriou and Yannakakis, as described in the introduction. [PY] proved that no deterministic search strategy achieves a constant ratio. In this section we prove that even a randomized searcher cannot achieve a constant ratio. Rather, the ratio must grow with the number of obstacles.

Let $d_1, d_2, d_3, ..., d_k$ be a superincreasing sequence of integers. Set $d_0 = 0$. Define a new metric space $\mathcal{M}'(k)$ on $\{z_0, z_1, ..., z_k\}$ by identifying z_i with the point d_i on the real line: $\text{dist}(z_i, z_j) = |d_i - d_j|$.

Theorem 19 *For all multipoint k-server algorithms A for $\mathcal{M}'(k)$, for all r, there is a request sequence σ_r of optimal cost at least r and of length at most rN_k such that*

$$E[A(\sigma_r)] \geq \frac{c_k}{2} \cdot \text{OPT}(\sigma_r).$$

Proof. Let A be a multipoint k-server algorithm for $\mathcal{M}'(k)$. Let A' be the algorithm for $\mathcal{M}(k)$ which mimics the behavior of A on $\mathcal{M}'(k)$. Each distance in $\mathcal{M}'(k)$ is between one half and one times the corresponding distance in $\mathcal{M}(k)$. It follows that

$$E[A(\sigma)] \geq \frac{1}{2} E[A'(\sigma)]$$

for all σ. By Theorem 5, for each r there is a request sequence σ'_r for $\mathcal{M}(k)$ of optimal cost at least r and of length at most rN_k such that

$$E[A'(\sigma_r)] \geq c_k \cdot \text{OPT}(\sigma'_r).$$

The optimal cost of σ'_r in $\mathcal{M}(k)$ is at least its optimal cost in $\mathcal{M}'(k)$. We may set $\sigma_r = \sigma'_r$. ■

Let $N = N_k$. Fix a randomized searching algorithm. For scenes with at most $(k+2)N+2$ obstacles, we will prove a lower bound of $\Omega(\log k)$ on the performance ratio.

Choose k. Let $h = N+1$. We first build a collection of $(k+2)N+2$ open rectangles as follows. For each $i = 1, 2, \ldots, N$, place k rectangles of width one in the region $\{(x, y) | i-1 \leq x < i\}$ known as *column i*. The jth rectangle C_{ij} $(1 \leq j \leq k)$ runs from $y = d_{j-1}h$ to $y = d_j h$. These k rectangles cover the region $\{(x, y) | i - 1 < x < i, 0 < y < hd_k\}$. Now add a rectangle C_{i0} of width one and infinite height just below C_{i1}, and add a rectangle $C_{i,k+1}$ of

the same size as C_{i0} just above C_{ik}. Now add one infinite open rectangle L covering all of the plane to the left of these $(k+2)N$ rectangles. To their right add an infinite rectangle R covering everything to the right. Now we need to shift the rectangles slightly. Define $\epsilon_i = 2^{-i}$ and slide upward by ϵ_i all $k+2$ rectangles in column i, $1 \le i \le N$.

To summarize, the final positions of the rectangles are as follows:
For $1 \le j \le k$,

$$C_{ij} = \{(x,y) | i-1 < x < i, d_{j-1}h + \epsilon_i < y < d_j h + \epsilon_i\},$$

$$C_{i0} = \{(x,y) | i-1 < x < i, y < \epsilon_i\}, \text{ and}$$

$$C_{i,k+1} = \{(x,y) | i-1 < x < i, y > h d_k + \epsilon_i\}.$$

These rectangles together with L and R cover all but a set of measure 0 of the entire plane.

The input consists of a sequence $\sigma_1, \sigma_2, \ldots, \sigma_N$, chosen in advance, with $\sigma_i \subsetneq \{0, 1, \ldots, k\}$ and $\sigma_i \neq \emptyset$ for all i. In column i, $1 \le i \le N$, the adversary "fuses" rectangles C_{ij} and $C_{i,j+1}$ for all $j \in \sigma_i$. The searcher's origin s is $(0, \epsilon_1)$, and his target, the vertical line $x = N$.

As soon as the searcher first reaches column i, i.e., his x-coordinate first reaches $i-1$, the adversary tells him σ_i. The slight vertical displacement between columns $i-1$ and i prevents the searcher from learning anything about σ_i before he enters column i.

Against these possible inputs, we can convert any searching algorithm to an algorithm A of no greater cost with this property:

- As soon as the searcher's x-coordinate reaches $i-1$ (and he learns σ_i), he chooses a column-i rectangle C_{ij} with $0 \le j \le k$ and $j \notin \sigma(i)$. (The choice of $j \in \{0, 1, \ldots, k\}$ is random and probably not uniform.) Because $j \notin \sigma(i)$, rectangle C_{ij} has not been fused with its neighbor above. The searcher then moves vertically to the upper-left corner of C_{ij} and then one unit rightward.

Let $a_i \notin \sigma_i$ be the searcher's random choice of j in column i and let $a_0 = 0$. His cost of moving from the upper-right corner $(i-1, d_{a_{i-1}}h + \epsilon_{i-1})$ of $C_{i-1,a_{i-1}}$ to the upper-right corner $(i, d_{a_i}h + \epsilon_i)$ of C_{i,a_i} is at least $h|d_{a_i} -$

$d_{a_{i-1}}|$. His total cost is therefore at least

$$h\sum_{i=1}^{N}|d_{a_i}-d_{a_{i-1}}|.$$

The random choice of $a_i \notin \sigma_i$ depends only on $\sigma_1, \sigma_2, \ldots, \sigma_i$. This is precisely the situation of a randomized multipoint k-server algorithm which serves N multipoint requests in $\mathcal{M}'(k)$ and starts with its "hole" at d_0: d_{a_i} is the location of the algorithm's "hole" after serving the ith request σ_i. We will view A as a multipoint k-server algorithm for $\mathcal{M}'(k)$.

Given $\sigma_1, \ldots, \sigma_N$, set $b_0 = 0$ and let $b_1, b_2, \ldots, b_N$ be the optimal way to serve requests $\sigma_1, \ldots, \sigma_N$ in $\mathcal{M}'_k$. In other words, $b_1, b_2, \ldots, b_N$ minimizes $\sum_{i=1}^{N}|d_{b_i}-d_{b_{i-1}}|$ subject to the constraint $b_i \notin \sigma_i$ for all i. Then the length of the shortest obstacle-free $s-t$ path is at most

$$\sum_{i=1}^{N}[1+2^{-i}+h|d_{b_i}-d_{b_{i-1}}|]$$

$$< N+1+h\sum_{i=1}^{N}|d_{b_i}-d_{b_{i-1}}|.$$

By Theorem 19 applied to A with $r=1$, there is a request sequence σ of length at most N such that

$$E[A(\sigma)] \geq \frac{c_k}{2}\cdot \mathrm{OPT}(\sigma).$$

(Theorem 19 goes through even if A knows the length of the request sequence in advance.) For that sequence,

$$\frac{E[\sum_{i=1}^{N}|d_{a_i}-d_{a_{i-1}}|]}{\sum_{i=1}^{N}|d_{b_i}-d_{b_{i-1}}|} \geq \frac{c_k}{2}.$$

Thus

$$\begin{aligned}\frac{E[h\sum_{i=1}^{N}|d_{a_i}-d_{a_{i-1}}|]}{N+1+h(\sum_{i=1}^{N}|d_{b_i}-d_{b_{i-1}}|)} &= \frac{hE[\sum_{i=1}^{N}|d_{a_i}-d_{a_{i-1}}|]}{h(1+\sum_{i=1}^{N}|d_{b_i}-d_{b_{i-1}}|)}\\ &\geq \frac{E[\sum_{i=1}^{N}|d_{a_i}-d_{a_{i-1}}|]}{2\sum_{i=1}^{N}|d_{b_i}-d_{b_{i-1}}|}\\ &= \frac{c_k}{4}.\end{aligned}$$

With at most $2+(k+2)N$ rectangles, we have proven a lower bound of $c_k/4$, which is $\Omega(\log k)$. If desired, each of the rectangles with an infinite side length can be replaced by a rectangle with finite but very large dimensions.

We choose

$$d_k = \lceil 4c_{k-1}e^{2c_{k-1}} \rceil d_{k-1}$$

for all $k \geq 2$ and $d_1 = 1$. A simple calculation yields $N_k \leq 4^k(d_1 d_2 \cdots d_k)$, $d_k \leq k^{10} d_{k-1}$, and $N_k \leq 4^k (k!)^{10k}$. This means that we have a lower bound of $(\ln \ln n)/24$, n being the number of rectangles, for infinitely many n.

6 Acknowledgments

We are grateful to Prabhakar Raghavan for inspiring us to seek a general k-server lower bound, to Lyle McGeoch for his help in fixing a mistake, to Mario Szegedy for simplifying the proof of Lemma 11, and to Dean Foster and Amos Fiat for their helpful remarks.

7 Appendix

Let $\beta \geq 1$ be a real and let $\ell \geq \beta$ be a positive integer. Let $d \geq 4\beta e^{2\beta}\ell$ be an integer. Let $w_1, w_2, ..., w_Q \in \{1, 2, ..., 2\ell\}$ where Q satisfies $\sum_{i=1}^{Q-1} w_i < 2d \leq \sum_{i=1}^{Q} w_i$.

We prove a lower bound on the solution of the following linear program LP_1:

Find $p_1, p_2, ..., p_{Q-1}, \gamma$ so as to minimize γ subject to

$$d(1-p_t) + \beta \cdot \sum_{s=1}^{t} p_s w_s \leq \gamma \cdot \sum_{s=1}^{t} w_s, \quad \text{for } t = 1, \ldots, Q-1,$$

$$d + \beta \cdot \sum_{s=1}^{Q-1} p_s w_s \leq \gamma \cdot 2d,$$

and

$$0 \leq p_s \leq 1, \quad \text{for } s = 1, \ldots, Q-1.$$

We first prove that the solution of linear program LP_2:

Find $r_1, r_2, ..., r_{2d-2\ell}, \gamma$ so as to minimize γ subject to

$$d(1-r_j) + \beta \cdot \sum_{i=1}^{j} r_i \leq \gamma \cdot (j + 2\ell), \quad \text{for } j = 1, \ldots, 2d-2\ell,$$

$$d + \beta \cdot \sum_{i=1}^{2d-2\ell} r_i \leq \gamma \cdot 2d$$

is a lower bound on the solution of LP_1. Then we prove that the solution of linear program LP_3:

Find $t_1, t_2, ..., t_{2d-2\ell}, \gamma$ so as to minimize γ subject to

$$d(1 - t_j) + \beta \cdot \sum_{i=1}^{j} t_i = \gamma \cdot (j + 2\ell), \quad \text{for } j = 1, \ldots, 2d - 2\ell,$$

$$d + \beta \cdot \sum_{i=1}^{2d-2\ell} t_i \leq \gamma \cdot 2d,$$

is a lower bound on the solution of LP_2. LP_2 and LP_3 are identical, except that the first $2d - 2\ell$ inequalities in LP_2 are equalities in LP_3.

Lemma 20 *The solution of* LP_2 *is a lower bound on the solution of* LP_1.

Proof. Suppose $p_1, \ldots, p_{Q-1}, \gamma$ is a feasible solution to LP_1. Then the assignment

$$\begin{aligned}
r_1 &= p_1 \\
r_2 &= p_1 \\
&\vdots \\
r_{w_1} &= p_1 \\
r_{w_1+1} &= p_2 \\
r_{w_1+2} &= p_2 \\
&\vdots \\
r_{w_1+w_2} &= p_2 \\
&\vdots \\
r_{w_1+\cdots+w_{Q-2}+1} &= p_{Q-1} \\
r_{w_1+\cdots+w_{Q-2}+2} &= p_{Q-1} \\
&\vdots \\
r_{w_1+\cdots+w_{Q-1}} &= p_{Q-1}
\end{aligned}$$

and γ is a solution to LP_2. (Because $w_1 + w_2 + \cdots + w_{Q-1} \geq 2d - 2\ell$, we have constructed at least as many r's as we need, maybe more.) ■

Lemma 21 *The solution of* LP_3 *is a lower bound on the solution of* LP_2.

Proof. Suppose $r_1, r_2, \ldots, r_{2d-2\ell}, \gamma$ is a feasible solution to LP_2. Define $R_0 = 0$ and $R_j = r_1 + r_2 + \cdots + r_j$, $1 \le j \le 2d - 2\ell$. Then

$$d(1 - R_j + R_{j-1}) + \beta R_j \le \gamma(j + 2\ell)$$

for $1 \le j \le 2d - 2\ell$, and

$$d + \beta R_{2d-2\ell} \le \gamma \cdot 2d.$$

Therefore

$$d - R_j(d - \beta) + dR_{j-1} \le \gamma(j + 2\ell),$$

which is equivalent to

$$R_j \ge \frac{d}{d-\beta} R_{j-1} + \frac{d - \gamma(j+2\ell)}{d-\beta}.$$

Also

$$R_{2d-2\ell} \le \frac{d(2\gamma - 1)}{\beta}.$$

Now define $T_0 = 0$ and

$$T_j = \frac{d}{d-\beta} T_{j-1} + \frac{d - \gamma(j+2\ell)}{d-\beta}$$

if $1 \le j \le 2d - 2\ell$, so that

$$d(1 - T_j + T_{j-1}) + \beta T_j = \gamma(j + 2\ell)$$

for $1 \le j \le 2d - 2\ell$. An easy inductive proof shows that $T_j \le R_j$ for all j. Thus

$$T_{2d-2\ell} \le R_{2d-2\ell} \le \frac{d(2\gamma - 1)}{\beta}.$$

Define $t_j = T_j - T_{j-1}$ for $j = 1, 2, \ldots, 2d - 2\ell$.
For $1 \le j \le 2d - 2\ell$,

$$\begin{aligned} &d(1 - t_j) + \beta(t_1 + t_2 + \cdots + t_j) \\ &= d(1 - T_j + T_{j-1}) + \beta T_j \\ &= \gamma(j + 2\ell). \end{aligned}$$

Also

$$T_{2d-2\ell} = t_1 + t_2 + \cdots + t_{2d-2\ell} \le \frac{d(2\gamma - 1)}{\beta}.$$

This means that $t_1, t_2, ..., t_{2d-2\ell}, \gamma$ is a feasible solution to LP_3. ■

Lemma 22 *If $t_1, t_2, ..., t_{2d-2\ell}, \gamma$ is a solution to* LP_3*, then*

$$\gamma \geq \beta \left[1 + \frac{1}{2e^{2\beta} - 1}\right].$$

Proof. We have

$$d(1 - t_1) + \beta t_1 = \gamma(1 + 2\ell),$$

i.e.,

$$t_1 = \frac{d - \gamma(1 + 2\ell)}{d - \beta}.$$

For $j = 2, 3, \ldots, 2d - 2\ell$,

$$d(1 - t_j) + \beta \cdot \sum_{i=1}^{j} t_i = \gamma \cdot (j + 2\ell)$$

$$d(1 - t_{j-1}) + \beta \cdot \sum_{i=1}^{j-1} t_i = \gamma \cdot (j - 1 + 2\ell).$$

Subtracting,

$$d(t_{j-1} - t_j) + \beta t_j = \gamma,$$

or

$$t_j = \frac{d}{d - \beta} t_{j-1} - \frac{\gamma}{d - \beta},$$

$j = 2, 3, \ldots, 2d - 2\ell$. An easy proof verifies that

$$t_j = \frac{\gamma}{\beta} - \left(\frac{\gamma}{\beta} + 2\gamma\frac{\ell}{d} - 1\right)\left(\frac{d}{d - \beta}\right)^j$$

for $j = 1, 2, \ldots, 2d - 2\ell$. We now use this assignment in the last constraint of LP_3 to get

$$d + \gamma(2d - 2\ell) + \beta\left(1 - \frac{\gamma}{\beta} - 2\gamma\frac{\ell}{d}\right)\sum_{i=1}^{2d-2\ell}\left(\frac{d}{d - \beta}\right)^i \leq \gamma \cdot 2d.$$

But

$$\sum_{i=1}^{2d-2\ell}\left(\frac{d}{d - \beta}\right)^i = \frac{d}{\beta} \cdot \left[\left(\frac{d}{d - \beta}\right)^{2d-2\ell} - 1\right]$$

$$= \frac{d}{\beta}(D - 1)$$

where

$$D = \left(\frac{d}{d-\beta}\right)^{2d-2\ell}.$$

Therefore

$$d + \gamma(2d - 2\ell) + \beta\left(1 - \frac{\gamma}{\beta} - 2\ell\frac{\gamma}{d}\right)\left[\frac{d}{\beta}(D-1)\right] \leq \gamma(2d)$$

$$d + \gamma(2d - 2\ell) + d\left(1 - \frac{\gamma}{\beta} - 2\ell\frac{\gamma}{d}\right)(D-1) \leq \gamma(2d)$$

$$d(1 + D - 1) + \gamma(-2\ell - \left(\frac{d}{\beta} + 2\ell\right)(D-1)) \leq 0$$

$$dD \leq \gamma\left(2\ell + (D-1)\left(\frac{d}{\beta} + 2\ell\right)\right)$$

$$\gamma \geq \frac{Dd}{2\ell + (D-1)(\frac{d}{\beta} + 2\ell)} = \frac{Dd\beta}{2\ell\beta + (D-1)(d + 2\ell\beta)}$$

$$= \frac{Dd\beta}{2\ell\beta + Dd - d + D2\ell\beta - 2\ell\beta}$$

$$\gamma \geq \beta\left[\frac{Dd}{Dd - d + D(2\ell\beta)}\right]$$

$$= \beta\left[\frac{Dd - d + D(2\ell\beta) + [d - D(2\ell\beta)]}{Dd - d + D(2\ell\beta)}\right]$$

$$= \beta\left[1 + \frac{d - D(2\ell\beta)}{Dd - d + D(2\ell\beta)}\right]$$

$$= \beta\left[1 + \frac{1 - 2\beta\frac{\ell}{d}D}{D(1 + 2\beta\frac{\ell}{d}) - 1}\right]$$

Now

$$D = \left(\frac{d}{d-\beta}\right)^{2d-2\ell}$$

$$= \left(1 + \frac{\beta}{d-\beta}\right)^{2d-2\ell} \leq e^{\frac{\beta}{d-\beta}(2d-2\ell)}$$

Because $(2d - 2\ell)/(d - \beta) \leq 2$, $D \leq e^{2\beta}$. But

$$\gamma \geq \beta\left[1 + \frac{1 - 2\beta\frac{\ell}{d}D}{D(1 + 2\beta\frac{\ell}{d}) - 1}\right]$$

As D increases, the bracketed quantity decreases. Therefore

$$\gamma \geq \beta \left[1 + \frac{1 - 2\beta \frac{\ell}{d} e^{2\beta}}{e^{2\beta} - 1 + e^{2\beta}(2\beta \frac{\ell}{d})} \right].$$

Because $d \geq 4\beta e^{2\beta}\ell$, $2\beta \frac{\ell}{d} e^{2\beta} \leq 1/2$. Thus

$$\gamma \geq \beta \left[1 + \frac{1 - \frac{1}{2}}{e^{2\beta} - 1 + \frac{1}{2}} \right] = \beta \left[1 + \frac{1}{2e^{2\beta} - 1} \right]. \blacksquare$$

Lemma 23 *The optimal value of* LP_1 *is at least*

$$\beta \left[1 + \frac{1}{2e^{2\beta} - 1} \right].$$

Proof. The lemma follows from Lemmas 20, 21, and 22. $\blacksquare$

Proof of Lemma 4. If Lemma 4 is not true, then

$$d + c_{i-1} \sum_{s=1}^{Q-1} p_s w_s < c_i(2d)$$

and

$$d(1 - p_h) + c_{i-1} \sum_{s=1}^{h} p_s w_s < c_i \sum_{s=1}^{h} w_s$$

for all h, $1 \leq h \leq Q - 1$. In this case, there is a $C < c_i$ such that

$$d + c_{i-1} \sum_{s=1}^{Q-1} p_s w_s \leq C(2d)$$

and

$$d(1 - p_h) + c_{i-1} \sum_{s=1}^{h} p_s w_s \leq C \sum_{s=1}^{h} w_s$$

for all h, $1 \leq h \leq Q - 1$. This contradicts Lemma 23 if we set $\beta = c_{i-1}$ and $\gamma = C$, since then

$$\beta \left[1 + \frac{1}{2e^{2\beta} - 1} \right] = c_i. \blacksquare$$

References

[BBKTW] S. Ben-David, A. Borodin, R.M. Karp, G. Tardos, and A. Wigderson. On the Power of Randomization in Online Algorithms. In *Proc. of the 22nd Ann. ACM Symp. on Theory of Computing*, pages 379–386, May 1990.

[BCR] R.A. Baeza-Yates, J.C. Culberson, and G.J.E. Rawlins. Searching with Uncertainty. Technical report, University of Waterloo, October 1987.

[BLS] A. Borodin, N. Linial, and M. Saks. An Optimal On-Line Algorithm for Metrical Task Systems. In *Proc. of the 19th Ann. ACM Symp on Theory of Computing*, pages 373–382, May 1987.

[BRS] A. Blum, P. Raghavan, and B. Schieber. Navigating in Unfamiliar Geometric Terrain. In *Proc. of the 23rd Ann. ACM Symp. on Theory of Computing*, May 1991. Also submitted for publication.

[CKPV] M. Chrobak, H.J. Karloff, T. Payne, and S. Vishwanathan. New Results on Server Problems. *SIAM Journal on Discrete Mathematics* 4(2):172–181, 1991.

[CL] M. Chrobak and L. Larmore. An Optimal On-line Algorithm for the Server Problem on Trees. *SIAM Journal of Computing* 20:144–148, 1991.

[FFKRRV] A. Fiat, D.P. Foster, H.J. Karloff, Y. Rabani, Y. Ravid, and S. Vishwanathan. Competitive Algorithms for Layered Graph Traversal. In *Proc. of the 32nd Ann. Symp. on Foundations of Comp. Sci.*, pages 288–297, October 1991.

[FKLMSY] A. Fiat, R.M. Karp, M. Luby, L.A. McGeoch, D.D. Sleator, and N.E. Young. Competitive Paging Algorithms. *Journal of Algorithms* 12:685–699, 1991.

[FRR] A. Fiat, Y. Rabani, and Y. Ravid. Competitive k-Server Algorithms. In *Proc. of the 31st Ann. IEEE Symp. on Foundations of Computer Science*, pages 454–463, October 1990. Also to appear in *Journal of Computer and System Sciences*.

[Ge] E. Grove. The Harmonic k-Server Algorithm is Competitive. In *Proc. of the 23nd Ann. ACM Symp. on Theory of Computing*, pages 260–266, May 1991.

[KMMO] A.R. Karlin, M.S. Manasse, L.A. McGeoch, and S. Owicki. Competitive Randomized Algorithms for Non-Uniform Problems. In *Proc. of the 1st Ann. ACM-SIAM Symp. on Discrete Algorithms*, pages 301–309, January 1990. Also submitted for publication.

[LST] L. Lovász, M. Saks, and W.T. Trotter. An Online Graph Coloring Algorithm with Sublinear Performance Ratio. *Discrete Mathematics*, pages 319–325, 1989.

[MMS] M.S. Manasse, L.A. McGeoch, and D.D. Sleator. Competitive Algorithms for On-Line Problems. *Journal of Algorithms* 11:208–230, 1990.

[MS] L.A. McGeoch and D.D. Sleator. A Strongly Competitive Randomized Paging Algorithm. *Algorithmica* 6:816–825, 1991.

[PY] C.H. Papadimitriou and M. Yannakakis. Shortest Paths Without a Map. *Springer-Verlag Lecture Notes in Computer Science*, volume 372, pages 610–620, July 1989.

[RS] P. Raghavan and M. Snir. Memory versus Randomization in On-Line Algorithms. *Springer-Verlag Lecture Notes in Computer Science*, volume 372, pages 687–703, July 1989.

[ST] D.D. Sleator and R.E. Tarjan. Amortized Efficiency of List Update and Paging Rules. *Communication of the ACM*, 28(2) pages 202–208, 1985.

[Vn] S. Vishwanathan. Randomized Online Graph Coloring. In *Proc. of the 31st Ann. IEEE Symp. on Foundations of Computer Science*, October 1990. Also to appear in *Journal of Algorithms*.