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.

[BKT] P. Berman, H.J. Karloff, and G. Tardos. A Competitive Three-Server Algorithm. In *Proc. of the 1st Ann. ACM-SIAM Symp. on Discrete Algorithms*, pages 280–290, January 1990.

[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.

[CDRS] D. Coppersmith, P. Doyle, P. Raghavan, and M. Snir. Random Walks on Weighted Graphs and Applications to On-line Algorithms. In *Proc. of the 22nd Ann. ACM Symp. on Theory of Computing*, pages 369–378, May 1990.

[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.

[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.

[IR] S. Irani and R. Rubinfeld. A Competitive 2-Server Algorithm. *Information Processing Letters*, 39:85–91, 1991.

[Kar] R.M. Karp. A randomized $(n+1)k$ competitive algorithm on the graph. personal communication, 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.

[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.

is at most 1. Since the optimal cost is at least 0.25, the cost of serving the requests in phase p using our on-line strategy is at most $(12k^3 + 4k^2 + 4) \cdot \mathsf{cost}_{\mathrm{OPT}}(\tau_1 \ldots \tau_m r)$. $\square$

Theorem 6: *Our on-line strategy is $O(k^3)$-competitive.*

Proof: Let $\sigma = \phi_1, \ldots, \phi_t$ be a request sequence, where ϕ_i is the sequence of requests in the i-th phase of our strategy. From Lemma 5 it follows that the cost of serving ϕ_i, for $i = 1, \ldots, t-1$, using our strategy is at most $O(k^3) \cdot \mathsf{cost}_{\mathrm{OPT}}(\phi_i)$. From Observation 1 and Corollary 3 we get that the cost of serving ϕ_t using our on-line strategy is at most $4k^2 \cdot \mathsf{cost}_{\mathrm{OPT}}(\phi_t) + 3k^3$. It is easy to see that $\mathsf{cost}_{\mathrm{OPT}}(\sigma) \geq \sum_{j=1}^{t} \mathsf{cost}_{\mathrm{OPT}}(\phi_j)$. Hence, the cost of serving σ using our strategy is at most $O(k^3) \cdot \mathsf{cost}_{\mathrm{OPT}}(\sigma) + 3k^3$. $\square$

4. Generalizations

Our algorithm can be generalized to obtain c_k-competitive algorithms for other metric spaces. The properties needed from such a space $\mathcal{M}$ are as follows. There exist $n(k)$ sub-spaces $\mathcal{M}_1, \ldots, \mathcal{M}_{n(k)}$ whose union is the space $\mathcal{M}$, such that:

1. There exists a d_k-competitive algorithm A_i for each sub-space $\mathcal{M}_i$.
2. For each sub-space $\mathcal{M}_i$, and any two points x and y in $\mathcal{M}_i$, the ratio of the distance between x and y in $\mathcal{M}_i$ to their distance in $\mathcal{M}$ is bounded by some function of k.
3. Let σ be any request sequence that requests points not in $\mathcal{M}_i$, for $i = 1, \ldots, n(k)$. Let b be a lower bound on $\mathsf{cost}_{\mathrm{OPT}}(\sigma)$. The ratio of the cost of serving a single request and of moving from a configuration of A_i to a configuration of A_j to the lower bound b is bounded by some function of k.

A concrete example is a metric space defined by a 1-tree (a connected graph with only one cycle). In this case the sub-spaces are the trees given by cutting the cycle. The resulting algorithm is $O(k^3)$-competitive.

Acknowledgements: We thank the referee for point us to the generalizations of our algorithm.

To upper bound Φ note that the initial servers' positions are arbitrary. Therefore, the maximal initial value of Φ may be $k(k-1)/2 + k^2 \leq 1.5k^2$. Following the arguments of [CKPV], we conclude that the cost of serving τ using DC_i is at most $k \cdot \mathsf{cost}_{\mathrm{LOPT}}(\tau) + 1.5k^2$. $\square$

Corollary 3: *The cost of serving τ using DC_i starting from* any *initial servers' position is at most $4k^2 \cdot \mathsf{cost}_{\mathrm{OPT}}(\tau) + 1.5k^2$.*

Consider a phase p of our strategy. Suppose that this phase consists of m sub-phases. Let τ_i be the sequence of requests in the i-th sub-phase. Recall that all the phases, except possibly the last, end with an additional request, denoted r, that exhausts all the double-cover strategies.

Observation 1:

$$\mathsf{cost}_{\mathrm{OPT}}(\tau_1 \dots \tau_m) \geq \sum_{j=1}^{m} \mathsf{cost}_{\mathrm{OPT}}(\tau_j).$$

Proof: Let I_1 be the initial position for which $\mathsf{cost}_{\mathrm{OPT}}(\tau_1 \dots \tau_m) = \mathsf{cost}_{\mathrm{OPT}}(\tau_1 \dots \tau_m, I_1)$. Then,

$$\mathsf{cost}_{\mathrm{OPT}}(\tau_1 \dots \tau_m) = \sum_{j=1}^{t} \mathsf{cost}_{\mathrm{OPT}}(\tau_j, I_j),$$

where I_j denotes the servers' position after serving the prefix $\tau_1 \dots \tau_{j-1}$ using a strategy achieving $\mathsf{cost}_{\mathrm{OPT}}(\tau_1 \dots \tau_m, I_1)$. Clearly, $\mathsf{cost}_{\mathrm{OPT}}(\tau_j, I_j) \geq \mathsf{cost}_{\mathrm{OPT}}(\tau_j)$, for all $1 \leq j \leq t$. The observation follows. $\square$

Suppose that phase p ends with a request r that exhausts all the double-cover strategies.

Lemma 4: *The optimal cost of serving the sequence of requests in p is at least* 0.25.

Proof: The sequence of requests $\tau_1 \dots \tau_m r$ consists of at least one request on each arc C_{2i}, for $1 \leq i \leq 2k$. Since only k mobile servers are available, at least half of the odd-indexed arcs has to be crossed by the servers in order to serve all the requests, incurring a cost of 0.25. $\square$

Lemma 5: *The cost of serving the requests in phase p using our on-line strategy is at most $(12k^3 + 4k^2 + 4)$ times the optimal cost.*

Proof: From Observation 1 and Corollary 3 we get that the cost of serving $\tau_1 \dots \tau_m$ using our on-line strategy is at most $4k^2 \cdot \mathsf{cost}_{\mathrm{OPT}}(\tau_1 \dots \tau_m) + 1.5mk^2$. Since the number of sub-phases is at most $2k$, this is bounded by $4k^2 \cdot \mathsf{cost}_{\mathrm{OPT}}(\tau_1 \dots \tau_m) + 3k^3$. The cost of serving r

of serving σ using A and starting from the initial position I. Define $\mathsf{cost}_{\mathrm{OPT}}(\sigma)$ to be the minimum over all initial positions I of $\mathsf{cost}_{\mathrm{OPT}}(\sigma, I)$.

Consider a sub-phase in which DC_i is used to serve the requests. Let τ be the sequence of requests in this sub-phase. Recall that all the requests in τ are on S_i. Our goal is to prove that the cost of serving τ using DC_i is at most $4k^2$ times the cost of serving τ by an optimal strategy for the circle C, plus an $O(k^2)$ additive factor. This is done in two stages. First, we prove that the cost of serving τ by an optimal strategy for the line S_i is at most $4k$ times the cost of serving τ by an optimal strategy for the circle C. Then, we prove that the cost of serving τ by DC_i is at most k times the cost of serving it by an optimal strategy for S_i, plus an $O(k^2)$ additive factor.

Lemma 1: *Let $LOPT$ be an optimal strategy for the line S_i that starts with the initial position I for which $\mathsf{cost}_{\mathrm{OPT}}(\tau) = \mathsf{cost}_{\mathrm{OPT}}(\tau, I)$. Then, the cost of serving τ using $LOPT$, denoted $\mathsf{cost}_{\mathrm{LOPT}}(\tau)$, is at most $4k \cdot \mathsf{cost}_{\mathrm{OPT}}(\tau)$.*

Proof: Let OPT be the strategy for C that achieves $\mathsf{cost}_{\mathrm{OPT}}(\tau, I)$. Without loss of generality we may assume that OPT is *lazy*; i.e., it moves a server only to serve a request. Consider the strategy A for the line S_i that simulates OPT by serving each request moving the same server as OPT. Note that whenever OPT moves a server across C_{2i} the cost of OPT and the cost of A might differ. However, in this case OPT pays at least $\frac{1}{4k}$ and A pays at most 1. Therefore, $\mathsf{cost}_{\mathrm{LOPT}}(\tau) \leq \mathsf{cost}_A(\tau) \leq 4k \cdot \mathsf{cost}_{\mathrm{OPT}}(\tau)$. □

Lemma 2: *The cost of serving τ using DC_i starting from* any *initial servers' position is at most $k \cdot \mathsf{cost}_{\mathrm{LOPT}}(\tau) + 1.5k^2$.*

Proof: Using the potential function defined in [CKPV], we compare the cost incurred by $LOPT$ and the cost incurred by DC_i. At any given time, let $x_1, \ldots, x_k$ denote the positions of the servers moved using the strategy $LOPT$ ordered clockwise starting from $\frac{i\pi}{k}$. Let $s_1, \ldots, s_k$ denote the positions of the servers moved using the strategy DC_i ordered in the same order. By [CKPV], the function

$$\Phi(x_1, \ldots, x_k, s_1, \ldots, s_k) = \sum_{1 \leq i < j \leq k} \mathrm{dist}(s_i, s_j) + k \cdot \sum_{i=1}^{k} \mathrm{dist}(x_i, s_i)$$

exhibits the following properties:

1. Whenever $LOPT$ moves a server, Φ is increased by at most k times the work of $LOPT$.
2. Whenever DC_i moves, Φ is decreased by at least the work of DC_i.

The line S_i defines a metric space $\mathcal{M}_i$ by setting the distance between two points on S_i as the length of the portion of S_i connecting these points.

Notice that $C_1, \ldots, C_{4k}$ partition the circle C into $4k$ arcs of equal size.

Our on-line strategy for the circle works in phases. For $i = 1, \ldots, 2k$, let DC_i to be the double-cover strategy for the line segment S_i. Each phase begins with all the double-cover strategies $DC_1, \ldots, DC_{2k}$ marked as *non-exhausted*, and ends when all of them are *exhausted*. When a request r within a phase has to be served, the following two steps are taken by our on-line strategy.

Step 1: If request r is for a point on C_{2i}, $1 \leq i \leq 2k$, then mark strategy DC_i as *exhausted*.

Step 2: Find the lowest indexed double-cover strategy that has not yet been exhausted. If there is no such strategy then serve r using an arbitrary server, terminate the phase, and start a new phase by marking all the double-cover strategies as *non-exhausted*. Otherwise (i.e., there exists such a strategy), serve r using the server determined by this lowest indexed non-exhausted double-cover strategy.

Consider a phase of the strategy. The phase can be divided into sub-phases, where in each sub-phase a single double-cover strategy DC_i is used to serve the requests. Notice that in each such sub-phase all the requests are on S_i. For all phases, except possibly the last, we have an additional last request that exhausts the last double-cover strategy used.

In the next section we prove that the cost of serving the requests in each sub-phase using the double-cover strategy is at most $4k^2$ times the cost of serving them using an optimal strategy for the circle, plus an $O(k^2)$ additive factor. We also prove that the optimal cost of serving all the requests in each phase, except possibly the last, is at least 0.25. Since there are at most $2k$ sub-phases in a phase and the cost of serving the last request in a phase is at most one, we conclude that our strategy is $O(k^3)$-competitive.

3. Competitive analysis

In this section we prove that the competitive ratio of our strategy is $O(k^3)$.

Definition 3: *Let σ be a request sequence on the circle C. For a given initial position of the k servers I, define $\mathsf{cost}_{\mathrm{OPT}}(\sigma, I)$ to be the minimum over all strategies A of the cost* for the *discrete* circle. A *discrete* circle is a metric space that consists of a finite subset of the circle points, unlike the (continuous) circle that consists of an infinite set of points.

In [BBKTW], Ben-David et al. investigate the power of randomization in on-line algorithms. They prove that if there exists a c_k-competitive randomized strategy against an adaptive adversary and a d_k-competitive randomized strategy against an oblivious adversary for some metric space, then there exists a $c_k \cdot d_k$-competitive *deterministic* strategy for this metric space. However, the proof is existential and does not construct this deterministic strategy. Applying this theorem to the strategies given in [Kar, CDRS], we get that there *exists* a $4k^2$-competitive deterministic strategy for the *discrete* circle. Unfortunately, we do not know how to construct such a strategy.

Our on-line strategy (as well as the strategy proposed in [Kar]) uses the k-competitive on-line strategy for the line given in [CKPV]. This deterministic strategy, called the *double-cover* strategy, works as follows. Suppose that a request appears at a point x of the line. If x is to the left of the leftmost server then this server moves to serve the request. Similarly, if x is to the right of the rightmost server then this server moves to serve the request. If x is between two servers, then the closer server moves to serve the request, and the further moves towards the request the same distance made by the closer server. The proof that this strategy is k-competitive uses a potential function argument.

The rest of the paper is organized as follows. In the next section we give an overview of our strategy and in Section 3 we prove that this strategy is $(8k^3 + 4k^2 + 4)$-competitive. We end with some generalizations.

2. Overview of the strategy

¿From now on, we assume without loss of generality that the circle has unit length circumference. We start with some definitions.

Definition 1: *The circle C is the set of points given by the polar coordinates $\{p_\theta = (\frac{1}{2\pi}, \theta) \mid 0 \leq \theta < 2\pi\}$. (From now on, we will use θ as a shorthand for p_θ.) The circle C defines a metric space $\mathcal{M}$ by setting the distance between two points θ_1 and θ_2 as the length of the shortest path in C connecting these points; that is, the minimum between $\frac{|\theta_1-\theta_2|}{2\pi}$ and $1 - \frac{|\theta_1-\theta_2|}{2\pi}$.*

Definition 2: *For $j = 1, \ldots, 4k$, let the arc C_j be the set of points $\{\theta \mid (j-1)\pi/2k < \theta \leq j\pi/2k\}$. For $i = 1, \ldots, 2k$, let S_i be the line segment given by omitting the arc C_{2i} from C.*

An on-line strategy A for moving the servers is *c_k-competitive* for $\mathcal{M}$ if for every request sequence σ, the cost of serving σ using the strategy A is bounded (up to an additive constant) by c_k times the cost of serving σ using the best *off-line* strategy; i.e., a strategy whose decisions are based on an apriori knowledge of the request sequence σ. More formally, define $\mathsf{cost}_{\mathrm{OPT}}(\sigma)$ to be the optimal cost of serving σ, and $\mathsf{cost}_A(\sigma)$ to be the cost of serving $sigma$ using A. Then, A is c_k-competitive if for all request sequences σ,

$$\mathsf{cost}_A(\sigma) \leq c_k \cdot \mathsf{cost}_{\mathrm{OPT}}(\sigma) + R,$$

for some constant R that does not depend on σ (it may depend on k and the metric space).

In this paper we consider the k-server problem on (a metric space defined by) the circle. Specifically, the k mobile servers occupy points of the circle and the cost of a move between two points is the distance between these points as measured on the circle. We give a deterministic on-line strategy for this problem that is $O(k^3)$-competitive.

Our strategy is the best constructive deterministic strategy for $k > 2$. It is the first deterministic strategy designed specifically for the metric space defined by the circle. For $k \geq 3$, applying the known strategies for a general metric space, either the one for an arbitrary k [FRR], or the one for $k = 3$ [BKT], result in inferior strategies for the circle. For $k = 2$, the k-server strategies for a general metric space given in [MMS], [CL], and [IR] are two, four, and ten competitive, respectively, and thus, better than our strategy for this case, that is 116 competitive.

Karp [Kar] gives a randomized strategy for the k-server problem on the circle. In this strategy a random point r is chosen on the circle. Then, the requests are served using the strategy for the line that results after "cutting" the circle at the point r. It is proved in [Kar] that this strategy achieves an expected competitive ratio $2k$ against an *oblivious* adversary; i.e., an adversary that fixes the sequence of requests without the knowledge of the random decisions made by the on-line strategy. Our strategy also involves "cutting" the circle in some specified points and using the on-line strategy for each of the resulting lines. In this sense, our strategy can be viewed as a *derandomization* of the strategy given in [Kar].

Coppersmith et al. [CDRS] give a randomized strategy for the k-server problem on a broad class of metric spaces, called *resistive* metric spaces. This strategy is k-competitive against an *adaptive* on-line adversary; i.e., an adversary that initiates a request based on the previous decisions of the on-line strategy, but has to serve the requests on-line. The circle is not a resistive metric space. However, using a resistive metric space that "approximates" the circle, [CDRS] achieve a $2k$-competitive strategy against an adaptive on-line adversary

A Deterministic $O(k^3)$-Competitive k-Server Algorithm for the Circle

Amos Fiat * Yuval Rabani* Yiftach Ravid* Baruch Schieber †

Abstract

Suppose that k mobile servers are located on a circle. Repeatedly, a request at a point x on the circle appears. To serve this request one of the serves has to be moved to x. The cost of moving a server to x is the distance on the circle between the server's previous location and x. The decision which server to move has to be done *on-line*; that is, without the knowledge of future requests. We give a deterministic on-line algorithm for making these decisions. Our algorithm is $O(k^3)$-competitive: for any sequence of requests, the cost incurred by our algorithm in serving this sequence is bounded (up to an additive constant) by $O(k^3)$ times the cost of serving this sequence using the best *off-line* algorithm; i.e., an algorithm that has apriori knowledge of the whole sequence. Our algorithm is the best deterministic constructive algorithm for $k > 2$.

1. Introduction

The k-server problem is stated as follows. Given a metric space $\mathcal{M}$, suppose that k mobile servers are located at some points of $\mathcal{M}$. Repeatedly, a request at a point $x \in \mathcal{M}$ appears. To serve this request one of the servers has to be moved from its current location to x. The cost of this move is the distance in $\mathcal{M}$ between the current location and x. The decision which server to move has to be done *on-line*; that is, without the knowledge of future requests. On-line problems, and specifically the k-server problem, received a lot attention recently. See, e.g., [ST, BLS, MMS, CDRS, FRR].

*Department of Computer Science, School of Mathematics, Tel-Aviv University, Tel-Aviv 69978, ISRAEL.

†IBM - Research Division, T.J. Watson Research Center, P.O. Box 218, Yorktown, NY 10598.