R–LINE: A Better Randomized 2-Server Algorithm on the Line

Lucas Bang, Wolfgang Bein, and Lawrence L. Larmore

Department of Computer Science,
University of Nevada Las Vegas, Nevada 89154, USA
{bang,beinw}@unlv.nevada.edu, larmore@cs.unlv.edu

Abstract. A randomized on-line algorithm is given for the 2-server problem on the line, with competitiveness less than 1.901 against the oblivious adversary. This improves the previously best known competitiveness of $\frac{155}{78} \approx 1.987$ for the problem.

1 Introduction

In the k-server problem, there are k identical mobile servers in a metric space M. At any time, a point $r \in M$ can be "requested," and must be "served" by moving one of the k servers to the point r. The cost of that service is defined to be the distance the server is moved; for a sequence of requests the goal is to serve the requests at small cost. An *online algorithm* for the server problem decides, at each request, which server to move, but does not know the sequence of future requests. We analyze an online algorithm for the server problem in terms of its *competitive ratio*, which essentially gives the ratio of its cost over the cost of an optimal (offline) algorithm which has knowledge of the entire request sequence before making any decisions. More precisely, we say that an online algorithm $\mathcal{A}$ for the server problem is C*-competitive*, if there is a constant K, such that, given any request sequence σ, $cost_{\mathcal{A}}(\sigma) \leq C \cdot cost_{\mathcal{OPT}}(\sigma) + K$, where $cost_{\mathcal{OPT}}(\sigma)$ is the minimum possible cost of any service of σ. If $\mathcal{A}$ is a randomized online algorithm, we express the inequality in terms of expected cost, *i.e.*, $E(cost_{\mathcal{A}}(\sigma)) \leq C \cdot cost_{\mathcal{OPT}}(\sigma) + K$. In the analysis of an online algorithms it is customary to think of the optimal service as performed by an *oblivious adversary*. The optimal cost is then also referred to as the "cost of the adversary," and the movement of the servers in the optimal algorithm as "adversary moves."

The server problem was first proposed by Manasse, McGeoch and Sleator [13] and the problem has been widely studied since then. They also introduced the now well-known k*-server conjecture*, which states that, for each k, there exists an online algorithm for k servers which is k-competitive for any metric space. The conjecture was immediately proved true by the same researchers for $k = 2$, but for larger k, the conjecture remains open, although it has been proved for a number of special classes of metric spaces, such as trees [10], spaces with at most $k + 2$ points [12], and the Manhattan plane for $k = 3$ [6].

T. Erlebach and G. Persiano (Eds.): WAOA 2012, LNCS 7846, pp. 120–130, 2013.

In the randomized case, little is known. Bartal *et al.* [4] have an asymptotic lower bound, namely that the competitiveness of any randomized online algorithm for an arbitrary metric space is $\Omega(\log k / \log^2 \log k)$. It is conjectured that there is an $O(\log k)$ competitive algorithm for general metric spaces. A recent breakthrough is the algorithm by Bartal *et al.* [2], which gives a poly-logarithmic competitive algorithm for finite metric spaces.

Surprisingly, no randomized competitive algorithm for the 2-server problem for general spaces is known to have competitiveness less than 2, although that barrier has been broken for a number of classes of spaces. The competitiveness is known to be $\frac{3}{2}$ for uniform spaces, and Bein *et al.* [7] have shown that there is a randomized algorithm with competitive ratio of at most 1.5897 for all 3-point spaces. Bein *et al.* [8] have recently given a "better than 2" competitive algorithm for crosspolytope spaces using knowledge states [9]. A lower bound of $1 + e^{-1/2} \approx 1.606$ has been shown [11].

We define the (m, n)-*server problem*, for $m > n$, to be the variation where there are m mobile servers in the metric space, and each request must be served by at least n of them. For the 2-server problem on the line, Bartal *et al.* give a barely random online algorithm for the 2-server problem on the line, with competitive ratio $\frac{155}{78} \approx 1.987$ [5]; their method is to define a deterministic online algorithm for the $(6, 3)$-server problem with that competitiveness, from which three deterministic online algorithms are defined. The randomized algorithm is simply to pick one of those three at random, each with probability $\frac{1}{3}$, and then use the chosen algorithm for the entire request sequence.

Our contribution. In this paper, we give a randomized online algorithm for the $(2n, n)$-server problem on the line, for every $n \geq 3$. By Theorem 1 below, we obtain a randomized algorithm for the 2-server problem on the line. As n increases, the competitiveness of our algorithm decreases, and the limiting value is less than 1.901.

2 The Algorithm R–LINE

Our algorithm, R–LINE, is defined to be a randomized algorithm for the $(2n, n)$-server problem, for $n \geq 3$. We make use of the following two theorems from [5]:

Theorem 1. *Given any C-competitive online algorithm for the $(2n, n)$-server problem, we can derive a randomized online algorithm for the 2-server problem that is C-competitive.*

Theorem 2. *Any optimal offline strategy for the $(2n, n)$ server problem keeps the servers in two blocks of n each, assuming that the servers are together in two blocks in the initial configuration.*

By Theorem 2, without loss of generality we can assume that the adversary is using an optimal 2-server algorithm, but serves with cost equal to n times the distance moved. We will use the notation s_i both to refer to the i^{th} server and its location, when no confusion arises. We assume that $s_1 \leq s_2 \leq ... \leq s_{2n-1} \leq s_{2n}$.

We also refer to the adversary's servers as a_1 and a_2, and assume that $a_1 \leq a_2$. The algorithm thus knows the location of one of the adversary's servers, which we call the *visible* server, and which, by a slight abuse of notation, we also call r. We denote the adversary's other server by a, and refer to it as the *hidden* server, since the algorithm does not know where it is.

We define a configuration of servers (R–LINE's as well as the adversary's) to be *satisfying* if at least n of R–LINE's servers are at r. We refer to a satisfying configuration as an S-configuration, and we assume that the initial configuration is an S-configuration.

Every round begins by the adversary choosing a new request point r and moving one of its two servers to r. R–LINE then moves as many of its servers as necessary to r, and the resulting configuration is once again an S-configuration. No R–LINE server will pass another R–LINE server that does not serve. In general, R–LINE deterministically moves zero or more servers to r, and then uses randomization to decide which additional servers to move. R–LINE is *lazy*, meaning that it never moves any server that does not serve the request.

2.1 The Potential

The algorithm R–LINE is given based on a suitable potential, which is used in Section 3 to prove competitiveness. For each fixed $n \geq 3$, we define a competitiveness C for R–LINE as well as a potential ϕ on configurations. This potential will satisfy the following property:

Property 1. If ϕ is the potential at the configuration before a round and ϕ' the potential after the round, and if $cost_{\text{R–LINE}}$ and $cost_{\mathcal{A}dv}$ are the costs incurred by R–LINE and the adversary, respectively, then

$$E(cost_{\text{R–LINE}} + \phi' - \phi) \leq C \cdot cost_{\mathcal{A}dv}$$

where E denotes expected value.

Isolation Indices and Coefficients. For $0 \leq i \leq 2n$ and $0 \leq j \leq 2$, if $1 \leq i + j \leq 2n + 1$, we define $\alpha_{i,j}$, the $(i,j)^{\text{th}}$ *isolation index* of a configuration, to be the length of the longest interval that has exactly i algorithm servers to the left and exactly j adversary servers to the left. More formally,

$$\alpha_{i,j} = \max \begin{cases} \min\{s_{i+1}, a_{j+1}\} - \max\{s_i, a_j\} \\ 0 \end{cases}$$

where we let $s_0 = a_0 = -\infty$ and $s_{2n+1} = a_3 = \infty$ by default. Isolation indices are part of T-theory and a more general definition of isolation indices is given in [1].

For each $0 \leq i \leq 2n$ and $0 \leq j \leq 2$, we define a constant $\eta_{i,j}$, the $(i,j)^{\text{th}}$ *isolation index coefficient*. The isolation index coefficients satisfy a symmetry property, namely $\eta_{i,j} = \eta_{2n-i,2-j}$; furthermore, $\eta_{0,0} = \eta_{2n,n} = 0$. We formally define the potential of a configuration to be

$$\phi = \sum \{\eta_{i,j} \cdot \alpha_{i,j} \ : \ (0 \leq i \leq 2n) \wedge (0 \leq j \leq 2) \wedge (1 \leq i + j \leq 2n + 1)\}$$

Intuitively, $\eta_{i,j}$ is a weight on the isolation index $\alpha_{i,j}$ for any configuration. For each given n, the competitiveness C and the isolation index coefficients $\{\alpha_{i,j}\}$ must satisfy a system of inequalities given in Section 3.

We will first define R–LINE in terms of those constants, and then show that R–LINE is C-competitive if the system of inequalities is satisfied. In Section 4 we outline how to find a solution to these inequalities, and give the values of the constants for $n = 3$.

2.2 Algorithm Description

We now define R–LINE. Between rounds, the configuration of servers is always an S-configuration. When the adversary makes a request at a point r, R–LINE responds by making a sequence of *moves*, each consisting of the movement of one or more servers to r. Thus, during a round, R–LINE makes at most n moves. Not all configurations can arise during execution of R–LINE; in fact, we define two classes of configurations, D-configurations and R-configurations, such that every intermediate configuration of R–LINE belongs to one of those two classes. If the current configuration is a D-configuration, then R–LINE's next move is to move one or more servers deterministically to r, while if the current configuration is an R-configuration, then R–LINE's next move is to choose, using randomization, a set of servers to move to r. In this case there are always two choices – to move one or more servers from the previous request point to r, completing the round, or to move just one server from the other side, possibly not completing the round.

We now define the classes of configurations. Note that, before the current round began, there must have been n algorithm servers at the previous request point, which we call r'. Without loss of generality, $r' \neq r$.

1. **S-Configuration:** there are n algorithm servers at r.
2. **D-Configuration:** the following two conditions hold.
   (a) There are more than n algorithm servers either strictly to the left or strictly to the right of r; that is, $r > s_{n+1}$ or $r < s_n$.
   (b) If there are fewer than n algorithm servers at r', then there is no algorithm server strictly between r' and r, and furthermore, there are at least n algorithm servers at the points r' and r combined.
3. **R-Configuration:**
   (a) There are exactly n algorithm servers on the same side of r as r', that is, either $r' = s_n < r$ or $r < r' = s_{n+1}$.
   (b) There is no algorithm server strictly between r' and r, and furthermore, there are at least n algorithm servers at the points r' and r combined.

We now give an explicit definition of R–LINE. By symmetry, we can assume, without loss of generality, that $r' < r$. The reader might also consult Figure 1 where we illustrate R–LINE through a single round, in a case where $n = 3$.

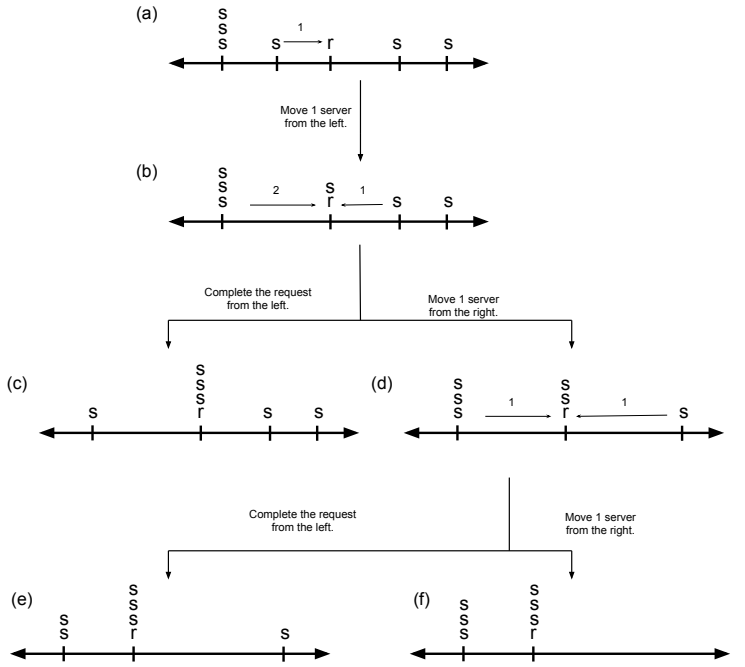

Fig. 1. (a) A D-configuration, where $n = 3$. The request is r, there are three servers located at $r' < r$. The next move is deterministic. (b) An R-configuration. One server has moved to r from the left. The next move is randomized; either move two servers from the left or one from the right. (c) An S-configuration, after two servers moved from the left. The round is over. (d) An R-configuration, after one server moved from the right. The next move is randomized; either move one server from the left or one from the right. (e) An S-configuration, after one server moved from the right. The round is over. (f) An S-configuration, after one server moved from the left. The round is over.

1. If the current configuration is a D-configuration, then there are m algorithm servers to the left of r for some $m > n$. Move the servers $s_{n+1}, \ldots s_m$ to r. If the resulting configuration is an S-configuration, the round is over. Otherwise, the resulting configuration is an R-configuration, and proceed to the next step.
2. If the current configuration is an R-configuration, then $r' = s_n < r \leq s_{n+1} < s_{2n}$. Let p be the number of algorithm servers at r. Then $s_{n+p+1} > r$. R–LINE executes one of two moves; each move is executed with a probability that is determined by solving a 2-person zero-sum game. We compute those probabilities below. The two choices of move are:

   (a) Move s_{n+p+1} to r.
   (b) Move the servers $s_{p+1} \ldots s_n$ to r.

   If the resulting configuration is an S-configuration, the round is over. Otherwise, the resulting configuration is an R-configuration, and repeat this step.

For the randomized step, one of the two choices is selected by using the optimum strategy for a 2-person zero sum game, where R–LINE is the column player, and $\mathcal{A}dv$ is the row player; the choice of the row player is where to place the hidden server. As we show later, we can assume, without loss of generality, that the hidden server is located at either s_n or s_{n+p+1}. Thus, each player has exactly two strategies. Each entry of the payoff matrix is equal to $\Delta\phi + cost = \phi' - \phi + cost$, where ϕ and ϕ' are the potentials before and after the move; and *cost* is the cost of the move, which is equal to the number of servers moved times distance moved, either $(s_{n+p+1} - r)$ or $(n-p)(r-s_n)$.

The payoff matrix is as follows:

$$G =$$

	Move s_{n+p+1}	Move $s_{p+1} \dots s_n$
$a = s_n$	$(\eta_{n+p+1,2} - \eta_{n+p,2} + 1)(s_{n+p+1} - r)$	$(\eta_{p,1} - \eta_{n,1} + n - p)(r - s_n)$
$a = s_{n+p+1}$	$(\eta_{n+p+1,1} - \eta_{n+p,1} + 1)(s_{n+p+1} - r)$	$(\eta_{p,0} - \eta_{n,0} + n - p)(r - s_n)$

3 Proof of Competitiveness

We now present a system of inequalities, which we denote $\mathbb{S}$, which suffice for R–LINE to be C-competitive. We will prove, in Theorem 3, that $\mathbb{S}$ implies C-competitiveness of R–LINE.

$$\forall\ 0 \le i \le 2n:\ |\eta_{i,1} - \eta_{i,0}| \ \le\ n \cdot C \tag{1}$$

$$\forall\ 1 \le i \le n \text{ and } \forall\ 1 \le j \le 2:\ \eta_{i,j} + 1 \ \le\ \eta_{i-1,j} \tag{2}$$

$$\forall\ 1 \le i \le n \text{ and } \forall\ 1 \le j \le 2:\ \eta_{i-1,j-1} \ \le\ \eta_{i,j-1} + 1 \tag{3}$$

$$\forall\ 1 \le i \le n:\ (\eta_{i-1,1} - \eta_{i,1} + 1)(\eta_{n-i,1} - \eta_{n,1} + i) \ \le\ (\eta_{i-1,0} - \eta_{i,0} + 1)(\eta_{n-i,0} - \eta_{n,0} + i) \tag{4}$$

Theorem 3. *For any assignment of values to C and $\eta_{i,j}$ for $0 \le i \le 2n$ and $0 \le j \le 2$ that satisfies the system $\mathbb{S}$,* R–LINE *is C-competitive.*

We prove Theorem 3 with a sequence of lemmas. We will prove that if the system of inequalities $\mathbb{S}$ is satisfied, then the following properties hold. We write $\Delta\phi = \phi' - \phi$, where ϕ is the potential before the move and ϕ' is the potential after the move.

1. For any move by the adversary, $\Delta\phi \le C \cdot cost_{\mathcal{A}dv}$. (Recall that the adversary pays n times the distance moved.)
2. For any deterministic move by R–LINE, $\Delta\phi + cost \le 0$.
3. We may assume the adversary's hidden server is at one of at most two possible locations during a given round, namely at the closest algorithm server to either the left or the right of r.
4. For any randomized move by R–LINE, $\mathrm{E}(\Delta\phi + cost) \le 0$.

We say that a move is *simple* if the move consists of moving a single server (either an algorithm or an adversary server) across an interval, and there is no other server (of either type) located strictly between the end points of that interval.

We also refer to a simple move as a *step*; in general, every movement of servers is a concatenation of steps.

Lemma 1. *If $\mathbb{S}$ holds, then Property 1 holds.*

Proof. By the symmetry of the $\eta_{i,j}$, inequality (1) implies that $|\eta_{i,j} - \eta_{i,j-1}| \leq n \cdot C$ for $j = 1, 2$. Without loss of generality the move is simple, since every move which is not simple is the concatenation of steps. Without loss of generality, the adversary server a_j moves to the right, from x to y, where $x < y$. Since the move is simple, $s_i \leq x$ and $y \leq s_{i+1}$ for some $0 \leq i \leq 2n$,. (Recall the default values $s_0 = -\infty$ and $s_{2n+1} = \infty$.) Thus, $\alpha_{i,j}$ decreases by $y - x$ and $\alpha_{i,j-1}$ increases by $y - x$. The cost to the adversary of this move is $n(y - x)$. By definition of the potential, $\Delta\phi = (\eta_{i,j} - \eta_{i,j-1})(y - x) \leq n \cdot C \cdot (y - x) \leq C \cdot cost_{\mathcal{A}dv}$.

Lemma 2. *If $\mathbb{S}$ holds, then Property 2 holds.*

Proof. For convenience, we assume that $r < r' = s_{n+1}$. There are exactly m algorithm servers to the right of r, for some $m > n$. Servers $s_{2n-m+1} \ldots s_n$ move to r. The move is the concatenation of steps, and it suffices to show that $\Delta\phi \geq cost_{\text{R–LINE}}$ for each of those steps.

Fix one step. During the step, s_i moves from x to y, where $y < x$, for some $2n - m + 1 \leq i \leq n$. The algorithm cost of the step is $x - y$. Pick the maximum j such that $a_j \leq y$. Since $r \leq y$, j is either 1 or 2. The move causes $\alpha_{i,j}$ to decrease by $x - y$ and $\alpha_{i-1,j}$ to increase by the same amount. By inequality (1), and the definition of the potential: $\Delta\phi + cost_{\text{R–LINE}} = (x - y)(\eta_{i,j} - \eta_{i-1,j} + 1) \leq 0$.

Lemma 3. *If $1 \leq i \leq 2n$ and $j = 1, 2$, then $\eta_{i,j} + \eta_{i-1,j-1} \leq \eta_{i,j-1} + \eta_{i-1,j}$*

Proof. Suppose $i \leq n$. Then $1 + \eta_{i,j} \leq \eta_{i-1,j}$ by (2), while $-1 + \eta_{i-1,j-1} \leq \eta_{i,j-1}$ by (3). Adding the two inequalities, we obtain the result.

If $i > n$, then $\eta_{2n-i+1,3-j} + \eta_{2n-i,2-j} \leq \eta_{2n-i+1,2-j} + \eta_{2n-i,3-j}$ by the previous case. By symmetry, we are done.

Lemma 4. *If $\mathbb{S}$ holds, then Property 3 holds.*

Proof. Since a could be any point on the line, the payoff matrix of the game has infinitely many rows. We need to prove that just two of those rows, namely $a = s_n$ and $a = s_{n+p+1}$, dominate the others.

By batching the row strategies, we illustrate the $\infty \times 2$ payoff matrix below.

		Move s_{n+p+1}	Move $s_{p+1} \ldots s_n$
I	$a \leq s_n$	$(\eta_{n+p+1,2} - \eta_{n+p,2} + 1)(s_{n+p+1} - r)$	$(\eta_{p,1} - \eta_{n,1} + n - p)(r - s_n)$
II	$s_n \leq a \leq r$	$(\eta_{n+p+1,2} - \eta_{n+p,2} + 1)(s_{n+p+1} - r)$	$(\eta_{p,1} - \eta_{n,1} + n - p)(r - a)$ + $(\eta_{p,0} - \eta_{n,0} + n - p)(a - s_n)$
III	$r \leq a \leq s_{n+p+1}$	$(\eta_{n+p+1,2} - \eta_{n+p,2} + 1)(s_{n+p+1} - a)$ + $(\eta_{n+p+1,1} - \eta_{n+p,1} + 1)(a - r)$	$(\eta_{p,0} - \eta_{n,0} + n - p)(r - s_n)$
IV	$a \geq s_{n+p+1}$	$(\eta_{n+p+1,1} - \eta_{n+p,1} + 1)(s_{n+p+1} - r)$	$(\eta_{p,0} - \eta_{n,0} + n - p)(r - s_n)$

The row strategy $a = s_n$ trivially dominates all row strategies in Batch I. It also dominates all row strategies in Batch II, because

$$\begin{aligned}\eta_{p,1} - \eta_{n,1} &= \sum_{i=p+1}^{n} (\eta_{i-1,1} - \eta_{i,1}) \\ &\geq \sum_{i=p+1}^{n} (\eta_{i-1,0} - \eta_{i,0}) \quad \text{by Lemma 3} \\ &= \eta_{p,0} - \eta_{n,0}\end{aligned}$$

The row strategy $a = s_{n+p+1}$ trivially dominates all row stages in Batch IV. It also dominates all row strategies in Batch III, because $\eta_{n+p+1,1} - \eta_{n+p,1} \geq \eta_{n+p+1,2} - \eta_{n+p,2}$, which we can similarly prove using Lemma 3.

We make use of a standard game theory lemma taken from [3]. To this end we remind the reader that a *saddle point* of a zero-sum game is defined to be an entry $a_{i,j}$ of the payoff matrix that is both a maximum of its row and a minimum of its column. If a game has a saddle point, then the value the game is the value of the saddle point, and it is optimum for the row player to always play the i^{th} row, and for the column player to always play the j^{th} column.

Lemma 5. *Suppose If* $A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$ *is the payoff matrix for a 2-person zero sum game* G*, and there is no saddle point. Then*

$$v(G) = \frac{\det A}{a_{11} - a_{12} - a_{21} + a_{22}}$$

Furthermore, the optimum strategy for the row player is:

Play row 1 with probability $\frac{a_{22}-a_{21}}{a_{11}-a_{12}-a_{21}+a_{22}}$

Play row 2 with probability $\frac{a_{11}-a_{12}}{a_{11}-a_{12}-a_{21}+a_{22}}$

While the optimum strategy for the column player is:

Play column 1 with probability $\frac{a_{22}-a_{12}}{a_{11}-a_{12}-a_{21}+a_{22}}$

Play column 2 with probability $\frac{a_{11}-a_{21}}{a_{11}-a_{12}-a_{21}+a_{22}}$

Lemma 6. *If* $\mathbb{S}$ *holds, then Property 4 holds.*

Proof. Consider the 2×2 payoff matrix G of Section 2.2. By $\mathbb{S}$, the upper left and lower right entries of G are negative, while the upper right and lower left entries are positive. By Theorem 5, the value of our game is

$$\frac{\det(G)}{(\eta_{n+p+1,2} + \eta_{n+p+1} - \eta_{n+p,2} - \eta_{n+p+1,1}) \cdot (s_{n+p+1} - r) + (\eta_{p,0} + \eta_{n,1} - \eta_{n,0} - \eta_{p,1}) \cdot (r - s_n)}$$

The numerator is non-negative by inequality 4. The denominator is negative, which we can prove by combining inequalities of $\mathbb{S}$ labeled (2) and (3). Thus, $E(\Delta\phi + cost_{\text{R–LINE}}) = v(G) \leq 0$ as claimed.

Theorem 3 follows immediately from Lemmas 1, 2, 4, and 6.

4 Finding a Solution to the Inequalities

We need to find a solution to the system $\mathbb{S}$ for which C is smaller than 2, preferably as small as possible. We first reduce $\mathbb{S}$ to a system, which we call $\mathbb{S}'$, that is easier to work with.

Let $n \geq 3$ be fixed. We introduce variables ϵ_i and δ_i for all $0 \leq i < n$. Our system $\mathbb{S}'$ consists of the following constraints.

$$\begin{aligned}
(2i + \epsilon_{n-i})(2 - \epsilon_i + \epsilon_{i-1}) &= 4i && \forall\ 0 < i < n \\
(2n + \epsilon_0)(2 + \epsilon_{n-1}) &\geq 4n \\
\delta &= -\epsilon_0/2 \\
C &= (2n - \delta)/n \\
\delta_i &= \epsilon_i + 2\delta && \forall\ 0 \leq i < n \\
\alpha_{i,0} &= 3i - \delta_i && \forall\ 0 \leq i < n \\
\alpha_{i,0} &= 2n + i - 2\delta && \forall\ n \leq i \leq 2n \\
\alpha_{i,1} &= 2n - i - \delta && \forall\ 0 \leq i < n \\
\alpha_{i,1} &= i - \delta && \forall\ n \leq i \leq 2n \\
\alpha_{i,2} &= \alpha_{2n-i,0} && \forall\ 0 \leq i \leq 2n
\end{aligned}$$

A solution to the system $\mathbb{S}'$. also provides a solution to $\mathbb{S}$.

4.1 The Case $n = 3$

For $n = 3$ the competitiveness of R–LINE can be calculated in closed form. The solution to $\mathbb{S}'$ for which C is minimum can be obtained by 4th degree polynomial. For $n = 3$, R–LINE has competitiveness $1 + \frac{\sqrt{71+17\sqrt{17}}}{12} \approx 1.98985407$. We also provide the values of the isolation index coefficients. The values of the constants $\delta, \delta_1, \delta_2$, and the $\eta_{i,j}$ are shown in the following tables.

Constants
$\delta = 3 - \frac{\sqrt{71+17\sqrt{17}}}{4} \approx 0.030437789$
$\delta_1 = 2 - \frac{\sqrt{7+\sqrt{17}}}{2} \approx 0.332433987$
$\delta_2 = 4 - \frac{\sqrt{79-7\sqrt{17}}}{2} \approx 0.459581218$

$\eta_{i,j} =$

	0	1	2
0	0	$6 - \delta$	$12 - 2\delta$
1	$3 - \delta_1$	$5 - \delta$	$11 - 2\delta$
2	$6 - \delta_2$	$4 - \delta$	$10 - 2\delta$
3	$9 - 2\delta$	$3 - \delta$	$9 - 2\delta$
4	$10 - 2\delta$	$4 - \delta$	$6 - \delta_2$
5	$11 - 2\delta$	$5 - \delta$	$3 - \delta_1$
6	$12 - 2\delta$	$6 - \delta$	0

The analytic methods used to find the above constants do not easily generalize and so we utilize approximation methods to determine the values of the constants for larger values of n. It is worth noting that Bartal et. al. provided an algorithm for the $(6, 3)$-server problem in [5] with competitiveness $\frac{155}{78} \approx 1.9871795$ which

is better than the result shown here. However, by using larger values of n we achieve a better upper bound on the competitiveness of the 2-server problem.

4.2 The Program

For $n > 3$, we approximate the value of C numerically, using a program to find a solution to $\mathbb{S}'$. Our program computes a function f, where $\delta = f(\epsilon_{\lfloor n/2 \rfloor})$. To find the maximum value of δ, we assumed that f is bimodal,[1] that is, there is some $x^* > 0$ for which $f(x)$ is maximum, and that $f(x)$ is monotone increasing for $0 < x < x^*$ and monotone decreasing for $x > x^*$. We then use a divide and conquer algorithm similar to binary search to find $f(x^*)$.

1. Guess $\epsilon_{\lfloor n/2 \rfloor}$, using our search algorithm.
2. If n is odd, then solve the following equation for $\epsilon_{(n+1)/2}$:

$$(n + 1 + \epsilon_{(n-1)/2})(2 - \epsilon_{(n+1)/2} + \epsilon_{(n-1)/2})$$

3. For all $0 < i < \lfloor \frac{n}{2} \rfloor$ in decreasing order:
   (a) Solve the following equation for ϵ_i:

$$(2(i + 1) + \epsilon_{n-i-1})(2 - \epsilon_{i+1} + \epsilon_i) = 4(i + 1)$$

   (b) Solve the following equation for ϵ_{n-i}:

$$2(n - i) + \epsilon_i 2 - \epsilon_{n-i} + \epsilon_{n-i-1} = 4(n - i)$$

4. Solve the following equation for δ:

$$(2 + \epsilon_{n-1})(2 - \epsilon_1 - 2\delta) = 4$$

5. Verify the following inequality:

$$(2n - 2\delta)(2 + \epsilon_{n-1}) \geq 4n$$

6. If our search interval is small enough, proceed to the last step. Otherwise, return to step 1.

7. $C \leftarrow \frac{2n-\delta}{n}$.

Our calculations show that $C \approx 1.90098671$ for $n = 2000$. Running the program for larger n leads us to believe that

$$\lim_{n\to\infty} C \approx 1.9007617$$

[1] However, the validity of the program does not depend on the bimodality of f.

5 Future Work

There are two possible directions in which to improve the results in this paper. We conjecture that the set of possible distributions can be expanded in order to obtain an even lower competitiveness. But the real gist of our work is to get a "better than 2" result for general spaces. Since our R–LINE is based on a potential defined in terms of isolation indices it is natural that a generalization to arbitrary metric spaces could make use of T-theory, where a more complex potential will be utilized.

Though not claimed in this paper, our preliminary investigation indicates that R–LINE – unlike the Bartal *et al.* algorithm – generalizes to trees.

Furthermore, it is known that Theorem 1 does not extend to $k \geq 3$ for all metric spaces. However, it does extend to $k \geq 3$ for the line, and should extend easily to the circle.

References

1. Bandelt, H.-J., Dress, A.: A canonical decomposition theory for metrics on a finite set. Adv. Math. 92, 47–105 (1992)
2. Bansal, N., Buchbinder, N., Madry, A., Naor, J.: A polylogarithmic-competitive algorithm for the k-server problem. In: Proc. 52nd Symp. Foundations of Computer Science (FOCS), 10 pages. IEEE Computer Society (2011)
3. Barron, E.N.: Game Theory: An Introduction. John Wiley & Sons, New Jersey (2008)
4. Bartal, Y., Bollobás, B., Mendel, M.: A Ramsey-type theorem for metric spaces and its applications for metrical task systems and related problems. In: Proc. 42nd Symp. Foundations of Computer Science (FOCS), pp. 396–405. IEEE (2001)
5. Bartal, Y., Chrobak, M., Larmore, L.L.: A randomized algorithm for two servers on the line (Extended Abstract). In: Bilardi, G., Pietracaprina, A., Italiano, G.F., Pucci, G. (eds.) ESA 1998. LNCS, vol. 1461, pp. 247–258. Springer, Heidelberg (1998)
6. Bein, W., Chrobak, M., Larmore, L.L.: The 3-server problem in the plane. Theoret. Comput. Sci. 287, 387–391 (2002)
7. Bein, W., Iwama, K., Kawahara, J.: Randomized competitive analysis for two-server problems. Algorithms 1, 30–42 (2008)
8. Bein, W., Iwama, K., Kawahara, J., Larmore, L.L., Oravec, J.A.: A randomized algorithm for two servers in cross polytope spaces. Theoretical Computer Science 412(2), 563–572 (2011)
9. Bein, W., Larmore, L., Noga, J., Reischuk, R.: Knowledge state algorithms. Algorithmica 60(3), 653–678 (2011)
10. Chrobak, M., Larmore, L.L.: An optimal online algorithm for k servers on trees. SIAM J. Comput. 20, 144–148 (1991)
11. Chrobak, M., Larmore, L.L., Lund, C., Reingold, N.: A better lower bound on the competitive ratio of the randomized 2-server problem. Inform. Process. Lett. 63, 79–83 (1997)
12. Koutsoupias, E., Papadimitriou, C.: Beyond competitive analysis. In: Proc. 35th Symp. Foundations of Computer Science (FOCS), pp. 394–400. IEEE (1994)
13. Manasse, M., McGeoch, L.A., Sleator, D.: Competitive algorithms for server problems. J. Algorithms 11, 208–230 (1990)