DIMACS Technical Report 2001-34
October 2001

A Randomized On–line Algorithm for the k–Server Problem on a Line

by

Béla Csaba[1]
Email: csaba@mpi-sb.mpg.de
Address: Max-Planck-Institut für Informatik,
Im Stadtwald 66123 Saarbrücken, Germany.

Sachin Lodha[2]
Email: lodha@cs.rutgers.edu
Address: Department of Computer Science,
Rutgers University New Bruswick, New Jersey 08903

[1]Part of this research was done while the author was a graduate student at Rutgers University.
[2]This work was supported by DIMACS which was funded by the NSF under grant STC 91-19999.

ABSTRACT

We give a $O(n^{\frac{2}{3}} \log n)$–competitive randomized k–server algorithm when the underlying metric space is given by n equally spaced points on a line. For $n = k + o(k^{\frac{3}{2}} / \log k)$, this algorithm is $o(k)$–competitive.

1 Introduction

The k–server problem is defined as follows: We are given k mobile servers that occupy k points of a metric space M. We assume that M is finite, $|M| = n > k$, and $k > 1$. At each time step a request, a point of M, appears. An algorithm must serve this request by moving one of its servers to the requested point (if that is vacant). The algorithm is charged for a cost which is equal to the distance moved by the server. The algorithm A is *on–line* if it serves the request without knowing what the future requests will be.

The standard way of measuring the performance of an on–line algorithm is *competitive analysis*. An on–line algorithm A is c–competitive if there exists a constant K such that for all request sequence ρ,

$$A(\rho) \quad \leq \quad c \cdot opt(\rho) + K, \tag{1}$$

where $A(\rho)$ is the cost of A on ρ and $opt(\rho)$ denotes the optimal off–line cost of ρ.

This measure first appeared in [14] for the paging problem. The general k–server problem was introduced in [12]. They proved that the competitive ratio of any on–line algorithm is at least k $(1 < k < n)$. They formulated the *k–server conjecture*, *i.e.* there exists a k–competitive on–line algorithm for every metric space M. This conjecture is still unsettled, the best known general upper bound on the competitive ratio is $2k - 1$ [10]. However, for some special cases, there are k–competitive on–line algorithms (*e.g.* tree metric spaces [7], $n = k + 2$ [5, 11], *etc.*).

In several branches of computer science, randomization does help. For the k–server problem very little is known about the competitive ratio of randomized on–line algorithms. The competitive ratio of a randomized algorithm R is c if the following holds: There exists a constant K such that for all request sequence ρ,

$$E[R(\rho)] \leq c \cdot opt(\rho) + K,$$

where $E[R(\rho)]$ denotes the expected cost of R on ρ. In [13], an H_k–competitive randomized on–line algorithm is given for the paging problem (H_k is the k^{th} harmonic number. $H_k \approx$ *log* k). This is the best possible: [8] shows that H_k is a lower bound in this case. Another special case for which there is a good randomized on–line algorithm is the case of $p(n)$–unbalanced metric spaces [6].

For general metric spaces, the best results so far were achieved by the help of randomized algorithms for the metrical task system problem. In a series of papers ([1], [2], [3] and [9]), a powerful method was developed for the randomized MTS problem. Translating these results for the k–server problem, the smallest randomized competitive ratio for general metric spaces is $min(2k-1, O(\log \binom{n}{k} \log\log \binom{n}{k} \log n \log\log n))$, see [9]. Note that $\Omega(\sqrt{\frac{log\ k}{loglogk}})$ lower bound is given in [6] for the competitive ratio in general metric spaces.

Here we investigate the randomized k–server problem for the special case when the underlying metric space is given by n equally spaced points on a line. The case when the metric space is a line has received special attention. First the lower bound $\Omega(\frac{log\ k}{loglog\ k})$ is shown in

[6]. There are two interesting results concerning upper bounds (algorithms). For the special case when the underlying metric space is the real line and $k = 2$, [4] exhibits a randomized on–line algorithm with competitive ratio $\frac{155}{78} < 2$. Note that in this case the size of the metric space, $|M|$, can be infinite. Another algorithm is for a line segment of length n with equidistant point set having competitive ratio $O(\log \binom{n}{k} \log n)$, see [9].

Our contribution is a $O(n^{\frac{2}{3}} \log n)$–competitive randomized on–line algorithm when the underlying metric space is given by n equally spaced points on a line. For $n = k + o(k^{3/2}/\log k)$, this algorithm is $o(k)$–competitive. Therefore it improves the best results so far for the range $n - k \in [ck, o(k^{3/2})/\log k]$. We remark that our method is directly suited for the k–server problem on a line, we do not use previous results on the metrical task systems.

Throughout the paper, we consider the k–server problem in an equivalent form of *l–evader problem*: There are $l = n - k$ evaders on the metric space M with n points. When there is no server on a point $x \in M$, then there is an evader on x, and vice versa. While in the k–server problem the satisfaction of a request is moving a server there, in the l–evader problem we must move away the evader residing on the request. Off–line and on–line algorithms, competitivity are defined in the same way as in the k–server setup. Note that no two evader can occupy the same point of M: that would result in more servers. It is easy to see that the l–evader problem is an equivalent formulation of the k–server problem. We will define and analyze our algorithm as a randomized l–evader algorithm. More on the l–evader problem can be found in [11].

We will consider an *oblivious adversary model*: there is an adversary A, who knowing the randomized algorithm R, but without the knowledge of its random choices, constructs the request sequence ρ. It pays optimally for it ($A(\rho) = opt(\rho)$) and compares it with the expected cost of R.

2 Definition of the Algorithm

We start by determining a block system on the line segment. First of all, we *partition* it into $t = n/b$ blocks, each of length $b = n^{\frac{1}{3}}$. Secondly, we pick a random number s uniformly from the interval $[0, b-1]$. We will shift the blocks by this number: the length of the leftmost block will be s, the length of the rightmost block will be $b - s$, all the other blocks will have the same length, b. For a block B, we denote its length by $|B|$.

R divides the course of satisfaction into phases. During the satisfaction of a request sequence ρ, the evaders of R will have two type of motions: steps inside a block or randomly jumping out into one of the *available* blocks. The rules for what makes a block available and how to determine the end of a phase will be discussed later.

Let us provide the rules for taking steps inside a block. Assume that there are r evaders in block B. We will consider first the case $r < |B|$, the other case, $r = |B|$, will be handled differently. Let ρ be a request sequence. If the i^{th} request of ρ, ρ_i, is not in B, then no evader will move in B. Otherwise, if $\rho_i \in B$ and there is an evader on ρ_i, that evader will go to the closest vacant position according to its *direction*. Every evader has a direction, which is "right" in the beginning, and from time to time it changes from "right" to "left" and vice versa. Suppose that evader e is residing on ρ_i and its direction is "right". Then e will move to the closest vacant position to the right of ρ_i. If there is no such position, e changes its direction to "left" and will go to the closest vacant position to the left of ρ_i. Similarly, e switches direction from "left" to "right", if it has to move, and there is no vacant position on the left. These are the only cases an R–evader changes its direction. We say that e moved a *round* when it switches direction from "left" to "right". Note that we have ensured that no two evaders will take the same position in any block.

When $r = |B|$ and there is a request for a position in B, that evader residing on the request will jump out into an available block.

For deciding which are the available blocks and how to change blocks we will assign $|B|$ *auxiliary blocks* $AB_1, \ldots, AB_{|B|}$ to each block B. Recall that R divides the satisfaction of a request sequence into phases. During a phase, every block will get *marks* based on working of *auxiliary evaders* moving in corresponding *auxiliary blocks*. These marks will help R to decide on the availability of a block.

At the start of the algorithm, R puts x *auxiliary evaders* into the auxiliary block AB_x for every block B and every x ($1 \leq x \leq |B|$). The evaders in AB_x are placed on the leftmost x positions in AB_x and their starting direction is "right". Say that from the beginning of the actual phase the adversary gave R the request sequence ρ'. If $x = |B|$ and there is a request in ρ' for some position in B, then B will be $|B|$–marked. For $x < |B|$, the auxiliary evaders will move in AB_x when satisfying ρ' as we described above for the R–evaders. We will count the number of rounds taken by all the auxiliary evaders of AB_x from the beginning of the phase. If the sum of these rounds in AB_x reaches b^2, B will be x–marked. Note that a block can have several marks. We will always consider the smallest one. Thus, for each block, its mark is monotonically decreasing with time during a phase. Observe that the marking procedure is *deterministic*, there is *no* random choice in it.

Now we are ready to give the description of R.

Initialization: R chooses a random shift $s \in [0, b-1]$ and determines the block system. R places its evaders on the leftmost positions inside their blocks. The auxiliary evaders also take the leftmost positions in the auxiliary blocks.

Description of a Phase:

- If there is a request for a position in block B, take a step in B and in AB_j ($1 \leq j \leq |B|$) according to the rules described earlier and update B's marks, if necessary.

- Suppose that B has become r'–marked. If there are $r \geq r'$ evaders in B, then eject $r - r' + 1$ evaders, one by one, from B — *if possible*, eject among those which already changed block during this actual phase. Use *First In First Out* policy for their ejection.

- Determine the available blocks for the ejected evaders. A block having r'' R–evaders is available if the smallest mark of the block is greater than $r'' + 1$. Pick randomly, uniformly a block, say C, from the available ones for the first ejected evader, that evader jumps into it. Rearrange the evaders of the block (including the new one) to have the configuration of the evaders of $AC_{r''+1}$. Rearrange the evaders in block B according to the configuration of the evaders of AB_{r-1}. For each of the remaining ejected evaders, repeat the above process: randomly pick an available block for jumping in and rearrange the evaders in it and those in block B according to the actual configuration of the evaders in the appropriate auxiliary blocks.

- If there is no available block for an ejected evader, the phase ends. Erase marks of all the blocks and begin a new phase.

We finish the definition of R. As we already stated, R starts by determining the block system on the line segment before the first request appears. The shift s is *not known* to A. The evaders are placed according to the initial configuration. R rearranges them to their leftmost positions inside each block. Then the first phase starts when the first request appears.

3 Notation

We use following notation throughout the remainder of the paper.

- We will assume that our blocks are arbitrarily numbered from 1 to $t = \frac{n}{b}$. So we can talk about i^{th} block, B_i.

- C_p denotes the *blockwise configuration* of the adversary evaders at the end of Phase p, *i.e.* C_p is a t–tuple: it has the number r in the i^{th} position, $C_p(i) = r$, if at the end of Phase p there are r A–evaders in B_i.

- Similarly D_p denotes the blockwise configuration of the R–evaders at the end of Phase p.

4 Analysis of R

We are going to prove the following theorem.

Theorem 1 *R is a $O(n^{\frac{2}{3}} \log n)$–competitive randomized k–server algorithm for the underlying metric space given by n equally spaced points on a line. It is $o(k)$–competitive for $n = k + o(k^{\frac{3}{2}} / \log k)$.*

4.1 R Is Distribution Invariant

We show that, despite its random choices, the distribution of R's evaders is the same for all runs at the end of each phase for a fixed shift.

Lemma 1 *Fix the shift s of the block system and let ρ be an arbitrary request sequence. Let B be a block. Suppose that for one run of R there are r evaders in B at the end of Phase p when satisfying ρ. Then B will contain exactly r evaders at the end of Phase p for all ρ–satisfying runs of R. Besides, the phases end at the same time for all runs.*

Proof: We prove the statement by induction on the phase number. It is trivially true before the first request appears because R starts in the initial configuration in each of its runs. Assume that the lemma is true up to Phase $p-1$. Denote the possible runs of R by $f_1, f_2, \ldots, f_u$. Assume that we are now at the end of Phase p for the run f_1 and B has r evaders inside. We also assume that f_1 is the first run which finished Phase p, *i.e.* other runs may not have finished the Phase p so far.

We will denote mark of block j by $m(j)$. $e_x(j)$ will track the current number of evaders in block j in run x. Recall that $m(j)$ is invariant of random choices of algorithm, *i.e.* different runs, for a given request sequence ρ. The current *available space* in block j in run x is $s_x(j) = m(j) - e_x(j) - 1$. Note that $s_x(j) \geq 0$ for any x, j always. We end phase when we cannot fulfill above requirement for every block j.

Since f_1 is the first run to end Phase p, there must be some block i_0 such that its latest mark $m(i_0)$ caused $e_1(i_0) - m(i_0) + 1$ evaders to jump out of block i_0 and there is no *space* left for them. Mathematically,

$$S_1 = \sum_{j, j \neq i_0}^{t} s_1(j) < e_1(i_0) - m(i_0) + 1.$$

Note that S_1 can also be written as

$$\begin{aligned} S_1 &= \sum_{j,j\neq i_0}^{t} m(j) - \sum_{j,j\neq i_0}^{t} (e_1(j) + 1) \\ &= \sum_{j,j\neq i_0}^{t} m(j) - (l - e_1(i_0) + t - 1) \\ &= \left(\sum_{j,j\neq i_0}^{t} m(j) - l - t + 1 \right) + e_1(i_0) \end{aligned}$$

where the first term in above sum is same for different runs at any given time owing to deterministic marking procedure.

Let's examine situation in run f_2 when jump–outs start in block B_{i_0} in run f_1. We consider three cases:

1. $e_1(i_0) = e_2(i_0)$: In both the runs, evaders start jumping out of block, one by one, simultaneously. Suppose we have problem for w^{th} evader jumping out in the run f_1. At this point in time, $S_1 = 0$. That means $e_1(j) = m(j) - 1$ for all j ($j \neq i_0$). Corresponding sum S_2 is also 0, owing to the fact that marks are same and the current number of evaders in block i_0 is same for both f_1 and f_2. So f_2 also ends and for any block j, $e_1(j) = e_2(j)$.

2. $e_1(i_0) > e_2(i_0)$: Here we have to distinguish two cases

   (a) $e_1(i_0) \geq m(i_0) > e_2(i_0)$: There are no jump–outs in f_2, and yet S_1 becoming 0 after w^{th} jump–out in f_1 indicates that S_2 was already negative to begin with — a contradiction!

   (b) $e_2(i_0) \geq m(i_0)$: There will be jump–outs in f_2 now. We can envision this as f_2 waiting for f_1 to bring its $e_1(i_0)$ value to $e_2(i_0)$ and then starting its own jump–outs. If, at all, $e_1(i_0)$ could be reduced to $e_2(i_0)$, then we are in case 1. If not, then we have $S_2 < S_1 = 0$ — a contradiction!

3. $e_1(i_0) < e_2(i_0)$: There are jump–outs in both f_1 and f_2. Since f_1 finishes first, $S_2 \geq S_1 + e_2(i_0) - e_1(i_0)$. We can envision this as f_1 now waiting for f_2 to bring its $e_2(i_0)$ value to $e_1(i_0)$ and then starting its own jump–outs. Again we are in case 1!

This finishes the proof of the lemma. □

Remark: This is a very important lemma which allows us to calculate the exact number of evaders that changed blocks during any Phase p. This number is

$$m_p = \sum_{i=1}^{t} max\{0, D_{p-1}(i) - D_p(i)\}.$$

Since $l = \sum_{i=1}^{t} D_{p-1}(i) = \sum_{i=1}^{t} D_p(i)$, we can also write following expression for m_p

$$m_p \quad = \quad \frac{1}{2}\sum_{i=1}^{t} |D_{p-1}(i) - D_p(i)|. \tag{2}$$

Note that every block and its corresponding all auxiliary blocks have exactly the same configuration at the end of a phase irrespective of random choices made during the phase. Once we fix s and ρ, we fix D_p's. We will like to figure out how "quickly" we move from D_{p-1} to D_p in R. We answer it in the next section.

5 Expected Cost of R

We will now prove a series of lemmas to estimate the expected cost of R.

5.1 Initial Cost

Lemma 2 *R spends at most $O(bn)$ during the* **Initialization**.

Proof: R rearranges all the evaders in each and every block and its corresponding auxiliary blocks to the leftmost positions during the **Initialization**. It could be done in $O(b^2)$ cost per block. Summing over all blocks, cost paid by R is at most $O(b^2) \cdot \frac{n}{b} = O(bn)$. □

Note that this is a one–time constant cost and it can be absorbed in K of equation (1).

5.2 Cost Associated with Jump–outs

Let us now consider a simple game related to the analysis of R. Adversary has a permutation σ of the first t positive integers $[t]$. $\sigma(i)$ denotes i^{th} element of σ and, therefore, $\sigma^{-1}(i)$ denotes the position of i in σ. We don't know σ. In the first round, we randomly, uniformly pick one element of $[t]$: t_1. Let $p_1 = \sigma^{-1}(t_1)$. Then adversary shows us first p_1 elements of σ and discards them. In the second round we randomly, uniformly pick one from the remaining ones, denote it by t_2. In general, in the $(i+1)^{\text{th}}$ round, we pick a number randomly, uniformly from the set $\{\sigma(p_i+1), \sigma(p_i+2), \ldots, \sigma(t)\}$ and denote it by t_{i+1}. We stop whenever we run out of elements, *i.e.* reach $\sigma(t)$. Denote the number of rounds in one possible outcome of the game by l_t — this is a random variable. We will denote its expectation, $E[l_t]$, by L_t.

Lemma 3 $L_t = \sum_{i=1}^{t} \frac{1}{i}$. *That is,* $L_t = O(\log t)$.

Proof: We prove the statement by induction on t. It is trivially true for $t = 1$. Assume now that the lemma is true up to t. Consider the set $\{1, 2, \ldots, t, t+1\}$. Pick t_1 and let $p_1 = \sigma^{-1}(t_1)$. p_1 can be anything between 1 and $t+1$ with the same probability $\frac{1}{t+1}$. From this, we can write recurrence: $L_{t+1} = \frac{1}{t+1} \sum_{r=1}^{t+1} (1 + L_{t+1-r})$; $(L_0 = 0)$. Using the induction hypothesis and simple rearrangement, we get

$$L_{t+1} = L_t + \frac{1}{t+1} = 1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{t+1} = \sum_{i=1}^{t+1} \frac{1}{i}.$$

□

Remark: Note that Lemma 3 is valid even if we allow adversary following freedom: After we pick an element in any round, adversary can "pick" some permutation of the remaining elements from *any* probability space defined on the set of possible permutations of those remaining elements for the next round. This is clearly due to the fact that L_t is only a function of t, completely independent of choice of σ.

Recall that when an R–evader has to jump out of a block, that evader jumps out which is among the ones that jumped into the block in the actual phase, if there is such an evader. This allows us to talk about the number of evaders changing block during a phase. We will like to relate the *number of jumping evaders* and the *expected number of jump–outs* during a phase.

Let's consider j^{th} request, say ρ_j. Suppose it is in block B_i. When R serves it, that might possibly change mark of B_i depending on work done in B_i's auxiliary blocks. So if we keep track of these changes in the marks of the blocks while serving ρ, we get a *unique* marking sequence corresponding to ρ in which marks of each block are *strictly decreasing.* Even though the marking sequence is deterministic, randomization comes into play when we choose arbitrary available block for a jumping evader. So any marking sequence will have lots of possible runs. From Lemma 1, we know that all these runs start and end all phases at the same time and have exactly same distribution of evaders at the end of each and every phase. This allows us to consider restriction of the marking sequence, and therefore associated runs, to any one particular phase p. From the remark at the end of Lemma 1, we know that total number of evaders that change block during phase p is

$$m_p = \frac{1}{2}\sum_{i=1}^{t} |D_{p-1}(i) - D_p(i)|.$$

Let M_p be a random variable which counts the number of jump–outs during Phase p. We will prove following lemma that finds $E[M_p]$.

Lemma 4 *Suppose there are m_p evaders which change block during Phase p. All the others stay in their blocks for the whole phase. Then the expected number of jump–outs in Phase p, $E[M_p]$, is $O(m_p \log n)$.*

Proof: There are m_p jumping evaders, let's use random variable Y_i to count the number of jump–outs of i^{th} evader. Then, $M_p = \sum_{i=1}^{m_p} Y_i$.

We will first find out $E[Y_i]$. Let's consider the original block, say B_j, from which our i^{th} evader, e_i, *starts* jumping out. There are precisely $b_j = |D_{p-1}(j) - D_p(j)|$ evaders that *start* their jump–outs from B_j. Let e_i be w^{th} evader, $1 \le w \le b_j$, that jumps out from B_j. Then it is clear that, irrespective of the run, e_i will start jumping out only when the mark of B_j drops below $D_p(j) + w$ for the *first time*. Let's denote the time of this event in the marking sequence by T_i.

At T_i, depending on the run we are in, there is a certain configuration of evaders in different blocks. Let's suppose that $L \le t$ blocks are available for e_i to jump into, namely, $B_{i_1}, B_{i_2}, \ldots, B_{i_L}$, and the block B_{i_1} has r_{i_1} evaders.

Recall that we use **FIFO** rule for ejecting evaders from any block: An evader will not jump out of a block unless those which jumped into the same block *after it* have already jumped out.

This will mean that if e_i chooses B_{i_q} to jump into, $1 \le q \le L$, then it will jump out only when mark of the block B_{i_q} drops below $r_{i_q} + 1$ for the *first time*. Let's denote the time of this event in the marking sequence by T_{i_q}.

It is easy to see now that we are in the situation described for the game in Lemma 3. Our *original* σ corresponds to the permutation of $i_1, i_2, \ldots, i_L$ that sorts $T_{i_1}, T_{i_2}, \ldots, T_{i_L}$ in the *increasing* order. The evader e_i has no idea about σ and picks any of the L available blocks. On the other hand, adversary has no inkling of e_i's random choices. So $E[Y_i] = O(\log t) = O(\log n)$.

Only difference between our algorithm and the game is that e_i might not have to jump out even if adversary "hits" its block, while we always jump out in the game. Now it trivially follows that

$$E[M_p] = \sum_{i=1}^{m_p} E[Y_i] = O(m_p \log n).$$

□

Lemma 5 *If there are m_p evaders during Phase p which change block, then the expected cost of R associated with jump–outs during Phase p is $O(m_p n \log n)$.*

Proof: Let's say there are M jump outs during phase p. We can associate following costs to a jump out:

1. the number of steps inside the block before jump–out — it is $O(n)$. Reason: Jump–out from block B takes place when its mark, say m, becomes less than or equal to the current number of evaders, say r, in it. This means that auxiliary block B_m finished $b^3 = O(n)$ work. Since we keep the smallest mark, work done in B_r (which is exact replica of B!) is at most work done in B_m, *i.e.* $O(n)$.

2. the distance the evader jumps — it is $O(n)$; and

3. the cost of rearranging the blocks — it is $O(b^2) = o(n)$.

So overall cost is $O(n)$ per jump–out and the cost of R associated with jump–outs during Phase p is $O(Mn)$.

But notice that M is a random variable and, from Lemma 4, $E[M] = O(m_p \log n)$. Therefore, the expected cost of R associated with jump–outs during phase p is $O(m_p n \log n)$.

□

5.3 Overhead Cost

Mere jump–outs don't account for entire cost paid by R. We have to account for

1. work done in a block after final jump–out out of it,

2. work done in a block after final jump–in into it, and

3. work done in a block with no jump–ins and jump–outs.

Each type of work is of order $O(n)$. Work of type 1 and 2 could be easily accounted into cost associated with jump–outs (this will just add at most 2 to the constant in Big–Oh notation of Lemma 5). So we only have to worry about work of type 3.

Definition 1 *A block B_i is called unchanged in Phase p if $D_{p-1}(i) = D_p(i)$.*

Let U_p be the set of indices of the unchanged blocks in Phase p, *i.e.*

$$U_p = \{i | D_{p-1}(i) = D_p(i)\}.$$

Note that, in the light of Lemma 1, U_p is uniquely determined by a fixed s and ρ. Let $u_p = |U_p|$. Unaccounted work of type 3 could only be possible in one or many members of U_p. So its total cost is $O(u_p n)$. We have following lemma.

Lemma 6 *If there are m_p evaders which change block and u_p unchanged blocks during Phase p, then the expected cost of R during Phase p is $O(m_p n \log n + u_p n)$.*

6 Analyzing Adversary Cost

Now we are going to estimate the cost of A in a phase. For that we will consider two copies of a block B, B_1 and B_2. Assume that A has a evaders in B_1, *viz.* $y_1, y_2, \ldots, y_a$, and R has r evaders in B_2, *viz.* $x_1, x_2, \ldots, x_r$. We also assume that $a \geq r$. Let ρ be a request sequence composed of positions in B. We allow any kind of movements for the A–evaders in B_1 which satisfies ρ. The R–evaders in B_2 will move according to our rules described earlier. It is important to note that neither the A–evaders nor the R–evaders will be allowed to jump out of their block. Then the following lemma holds.

Lemma 7 *Let B_1, B_2, a, r and ρ be as above. Suppose that the R–evaders move T rounds combined when satisfying ρ. Then the A–evaders in B_1 will move a distance of at least $a\frac{T}{r}$ combined.*

Proof: When the R–evaders move in B_2, they *meet* the A–evaders. This can be a *cover–and–leave*, *i.e.* the R–evader stopped at a position on which there was an A–evader in B_1 and now leaves it or a *jump–over*, when the R–evader cannot stop on that position because the A–evader is already covered by another R–evader.

The R–evaders move T rounds combined when satisfying ρ which results in $2aT$ meetings with A–evaders. We will give a lower bound on the number of cover–and–leaves. Note that the number of cover–and–leaves is certainly a lower bound on the cost of A: A R–evader leaves a position only when there is a request for that position. Hence A has to move its evader as well.

Let $\mathcal{C}$ denote the current configuration of A–evaders and R–evaders in B_1 and B_2 respectively. We define the potential of $\mathcal{C}$, denoted by ϕ, as the number of positions in B which have an A–evader in B_1 and a R–evader in B_2, *i.e. covered* positions. During the course of satisfaction of ρ, $\mathcal{C}$ and ϕ keep changing. We associate cost to any change in $\mathcal{C}$, it is in terms of *jump–overs*. We monitor ϕ and observer that if

1. ϕ decreases by 1: R *pays* at most $r-1$. Note that this is a cover–and–leave case, so A moves by a distance ≥ 1.

2. ϕ remains unchanged: Suppose R *pays* a non–zero cost. Once again, R won't *pay* more than $r-1$. Note that this is a cover–and–leave case too, so A moves by a distance ≥ 1. We don't care about A's movement if R pays nothing!

3. ϕ increases by 1: R pays at most $r-1$ again. A might not make any move at all. To make life simpler, we will charge R *nothing* in this case, but account for this *unpaid* cost by including it in case 1.

So A moves its evader by ≥ 1 when R pays at most $2(r-1)$ and total cost paid by R is $2aT$. This settles the lemma. $\square$

We use $T = b^2$ rounds as a threshold in marking procedure of R. Also observe that the number of evaders r in any block cannot be bigger than b. We get following Corollary of Lemma 7

Corollary 1 *Let B_1, B_2, a, r and ρ be as above. Suppose that the R–evaders move b^2 rounds combined when satisfying ρ. Then the A–evaders in B_1 will move a distance of at least ab combined.*

Remark: If $r > a$, then it is possible that all the A–evaders are covered by non–moving R–evaders. Hence A takes no steps while R moves any number of rounds.

In the above we assumed that the A–evaders stay in their block. But we do not have any control on them, they may change block too.

Definition 2 *A takes an A–**action** in block B if either the A–evaders move b steps inside B or one of them jumps out of B.*

For example, in Corollary 1, we proved that A takes at least a A–actions in B if R has $r \leq a$ evaders in B, the R–evaders move b^2 rounds, and all the A–evaders and R–evaders stay in B. According to our previous remark, if $a - r + 1$ A–evaders jump out of B, then it is possible that no A–evaders move inside B. But that means $a - r + 1$ A–actions.

Lemma 8 *Let B be a block with a A–evaders and r R–evaders in it at the beginning of the actual phase, at time t_0. R–evaders are not allowed to jump in or jump out. Suppose that from time t_0 the request sequence forced the R–evaders to move b^2 rounds in B. If $a \geq r \geq 1$, then we can associate at least $a - r + 1$ A–actions with B in the actual phase.*

Proof: Suppose that x A–evaders jump out of block B during the phase. If $x \geq a - r + 1$, we are done. Otherwise, $x \leq a - r$. That means B always has at least $a - x \geq r$ A–evaders. So using Corollary 1, we have at least $a - x$ more A–actions. Total number of A–actions, therefore, is $a - x + x = a \geq a - r + 1$. $\square$

Notice that so far we have been insisting that R–evaders don't jump out of their block B. But we cannot assume that during the run of R. What we know for sure though is that the auxiliary evaders in auxiliary blocks of B never jump around! Also rules of their movement are same as those of actual R–evaders. This allows us to relate the mark of block B at the end of phase and A–actions in B during the phase using Lemma 8.

Lemma 9 *Let a be the number of A–evaders in block B at the* **Start of Phase** *p. Let r be the number of R–evaders in B at the* **End of Phase** *p. If $a > r$, then there are at least $a - r$ A–actions in B during Phase p.*

Proof: Due to our marking procedure, the mark of B, say m, at the end of Phase p is at most $r + 1$.[1] Consider copy of B in adversary algorithm and B_m, the m^{th} auxiliary block of B in R. m being the mark of B, the number of rounds in B_m is at least b^2. Since $a \geq r + 1 \geq m \geq 1$, applying Lemma 8 on these two "copies" of B, we prove that the number of A–actions in B during the phase is at least $a - m + 1 \geq a - r$. $\square$

[1] In fact, it is one more than the number of evaders for all, but one, blocks!

Now we are ready to compare the number of A–actions with the expected cost of R. Let m_p denote the number of moving evaders and u_p denote the number of *unchanged* blocks in Phase p. Let a_p denote the number of A–actions in Phase p by adversary. We will make following assumption for the proofs of Lemma 10 and Theorem 1.

Assumption: For every p and its corresponding U_p, following holds: If $i \in U_p$, then

$$|C_{p-1}(i) - C_p(i)| + |C_{p-1}(i) - D_p(i)| \ \ \geq 1. \tag{3}$$

This assumption discards the possibility that no difference exists in block B_i between the adversary and R during Phase p. Otherwise one can easily verify that, in the light of Lemma 7, R performs relatively much better in such blocks. So we are assuming worse scenario. We have following key lemma.

Lemma 10 *Fix the shift s. Let ρ be a request sequence for which R has F phases. Then* $\sum_{p=1}^{F} a_p + \frac{n}{5b} \geq \frac{1}{5}\sum_{p=1}^{F}(m_p + u_p)$.

Proof: Recall, from the remark at the end of Lemma 1, we have equation (2)

$$m_p \;\; = \;\; \frac{1}{2}\sum_{i=1}^{t} |D_{p-1}(i) - D_p(i)|.$$

Similarly we can lower bound the number of A–actions by the number of block changes

$$a_p \;\; \geq \;\; \frac{1}{2}\sum_{i=1}^{t} |C_{p-1}(i) - C_p(i)|. \tag{4}$$

But a_p is at least $\sum_{i=1}^{t} max\{0, C_{p-1}(i) - D_p(i)\}$ by Lemma 9. Therefore,

$$a_p \;\; \geq \;\; \frac{1}{2}\sum_{i=1}^{t} |C_{p-1}(i) - D_p(i)|. \tag{5}$$

Simple triangle inequality, and inequalities (4), (5) and (3) imply

$$a_p \;\; \geq \;\; \frac{1}{4}\sum_{i=1}^{t} |C_p(i) - D_p(i)| \tag{6}$$

$$a_p \;\; \geq \;\; \frac{u_p}{2}. \tag{7}$$

Once again, simple triangle inequality, and inequalities (6), (7), (5) and equation (2) imply

$$\begin{aligned}
8a_{p-1} + 2a_p \;\; &\geq \;\; \sum_{i=1}^{t} |C_{p-1}(i) - D_{p-1}(i)| + 2u_{p-1} \\
&+ \;\; \sum_{i=1}^{t} |C_{p-1}(i) - D_p(i)| \\
&\geq \;\; \sum_{i=1}^{t} |D_{p-1}(i) - D_p(i)| + 2u_{p-1} \\
&= \;\; 2(m_p + u_{p-1}).
\end{aligned} \tag{8}$$

Using equation (8), summing over all phases from 1 to F, and noting that $a_0 = u_0 = 0$ and $u_p \leq t = \frac{n}{b}$, we get

$$\begin{aligned} \sum_{p=1}^{F} a_p + \frac{n}{5b} &\geq \frac{1}{10} \sum_{p=1}^{F} (8a_{p-1} + 2a_p) + \frac{n}{5b} \\ &\geq \frac{1}{5} \sum_{p=1}^{F} (m_p + u_p). \end{aligned}$$

□

Next we determine the expected number of A's block changes (the randomness is in the random choice of the shift s).

Lemma 11 *Let ρ be a request sequence, then the expected number of A's block changes is $opt(\rho)b^{-1}$.*

Proof: We assume that the A–evaders always take steps of length one. This can be achieved by possibly dividing the motion of an evader into individual steps of length one, and if necessary, by switching role between evaders when one passes the other. Consider the indicator random variables $X_1, X_2, \ldots, X_{opt(\rho)}$. $X_i = 1$ if in the i^{th} step the moving A–evader changes block, otherwise it is zero. Then clearly $Pr(X_i = 1) = b^{-1}$ for all i. So the expected number of block changes is $E[\sum_i X_i] = \sum_i Pr(X_i = 1) = opt(\rho)b^{-1}$. □

Now it is easy to estimate the expected number of A–actions.

Lemma 12 *For a request sequence ρ, the expected number of A–actions is at most $2opt(\rho)b^{-1}$*

Proof: By Lemma 11 the expected number of block changes during the satisfaction of ρ is $opt(\rho)b^{-1}$. The same number is clearly an upper bound for the number of other type of A–actions, *viz.* work done inside a block. □

7 Proof of Theorem 1

We are ready to prove our main result, Theorem 1.

Let ρ be a request sequence. Let us denote by m_p^s the number of moving R–evaders in Phase p when the shift of the block system is s and the number of phases for shift s by t_s. Let u_p^s denote the number of *unchanged blocks* during Phase p for shift s. Also $R(\rho)_p^s$ will denote the cost of R in Phase p with shift s and $R(\rho)^s$ the cost of R on ρ with shift s.

By Lemma 12,

$$2opt(\rho)b^{-1} \geq E[\sharp \text{ of A–actions}].$$

Lemma 10 implies that

$$E[\sharp \text{ of A–actions}] + \frac{n}{5b} \geq \frac{1}{5b}\sum_{s=1}^{b}\sum_{p=1}^{t_s}(m_p^s + u_p^s).$$

By Lemma 6, the latter expression is bounded below by

$$\frac{1}{5b}\sum_{s=1}^{b}\sum_{p=1}^{t_s}\left(E[R(\rho)_p^s]\frac{1}{O(n\log n)}\right).$$

By linearity of expectation, the above formula is

$$\frac{1}{O(bn\log n)}\sum_{s=1}^{b}E[R(\rho)^s].$$

Taking the average for shifts and using above inequalities, we have

$$2opt(\rho)b^{-1} + \frac{n}{5b} \geq E[R(\rho)]\frac{1}{O(n\log n)}.$$

That is,

$$O(\frac{n\log n}{b})opt(\rho) + O(\frac{n^2\log n}{b}) \geq E[R(\rho)].$$

So the competitive ratio is $O(n^{\frac{2}{3}}\log n)$. □

Acknowledgment: The authors wish to thank Endre Szemerédi and Péter Hajnal for their helpful remarks and discussions.

References

[1] Y. Bartal. Probabilistic approximations of metric spaces and its algorithmic application. In *Proceedings of the 37th Annual Symposium on Foundations of Computer Science*, pages 184–193, 1996.

[2] Y. Bartal. On approximating arbitrary metrics by tree metrics. In *Proceedings of the 30th Annual ACM Symposium on Theory of Computing*, pages 161–168, 1998.

[3] Y. Bartal, A. Blum, C. Burch, and A. Tomkins. A polylog(n)–competitive algorithm for metrical task systems. In *Proceedings of the 29th Annual ACM Symposium on Theory of Computing*, pages 711–719, 1997.

[4] Y. Bartal, M. Chrobak, and L. Larmore. A randomized algorithm for two servers on the line. *Information and Computation*, 158:53–69, 1999.

[5] Y. Bartal and E. Koutsoupias. On the competitive ratio of the work function algorithm for the k–server problem. In *Proceedings of 17th Annual Symposium on Theoretical Aspects of Computer Science*, pages 605–613, 2000.

[6] Y. Bartal, H. Karloff, Y. Rabani, and M. Saks. A decomposition theorem for task systems and bound for randomized server problems. Manuscript, 2000.

[7] M. Chrobak and L. Larmore. An optimal algorithm for the server problem on trees. *SIAM Journal of Computing*, 20:144–148, 1991.

[8] A. Fiat, H. Karloff, M. Luby, L. McGeoch, D. Sleator, and N. Young. Competitive paging algorithms. *Journal of Algorithms*, 12(4):685–699, 1991.

[9] A. Fiat and M. Mendel. Better algorithms for unfair metrical task systems and applications. In *Proceedings of the 32nd Annual ACM Symposium on Theory of Computing*, pages 725–734, 2000.

[10] E. Koutsoupias and C. Papadimitriou. On the k–server conjecture. *Journal of the ACM*, 42(5):971–983, 1995.

[11] E. Koutsoupias and C. Papadimitriou. The 2–evader problem. *Information Processing Letters*, 57(5):249–252, 1996.

[12] M. Manasse, L. McGeoch, and D. Sleator. Competitive algorithms for server problems. *Journal of Algorithms*, 11(2):208–230, 1990.

[13] L. McGeoch and D. Sleator. A strongly competitive randomized paging algorithm. *Algorithmica*, 6(6):816–825, 1991.

[14] D. Sleator and R. Tarjan. Amortized efficiency of list update and paging rules. *Communications of the ACM*, 28(2):202–208, 1985.