A competitive algorithm for the general 2-server problem

A competitive algorithm for the general 2-server problem

R.A. Sitters[1], L. Stougie[13], and W.E. de Paepe[2]

[1] Department of Mathematics and Computer Science
[2] Department of Technology Management
Technische Universiteit Eindhoven
P.O.Box 513, 5600 MB Eindhoven, The Netherlands
{r.a.sitters, l.stougie, w.e.d.paepe}@tue.nl
[3] CWI, P.O.Box 94079, 1090 GB Amsterdam, the Netherlands

Abstract. We consider the general on-line two server problem in which at each step both servers receive a request, which is a point in a metric space. One of the servers has to be moved to its request. The special case where the requests are points on the real line is known as the CNN-problem. It has been a well-known open question if an algorithm with a constant competitive ratio exists for this problem. We answer this question in the affirmative sense by providing the first constant competitive algorithm for the general two-server problem on any metric space.

1 Introduction

In the *general k-server problem* we are given servers $s_1, \ldots, s_k$, each of which moving in a metric space $\mathbb{M}_i$. Requests $r \in \mathbb{M}_1 \times \mathbb{M}_2 \times \ldots \times \mathbb{M}_k$ are presented on-line one by one. Thus, a request is a k-tuple $r = (z_1, z_2, \ldots, z_k)$ and it is served by moving one of the servers s_i to his corresponding point z_i. The decision which server to move is irrevocable and has to be taken without any knowledge about future requests. The cost of moving server s_i to z_i is equal to the distance travelled by s_i from his current location to z_i. The objective is to minimize the total cost to serve all given requests. The performance of an on-line algorithm is measured through competitive analysis. An online algorithm is c-competitive if, for any request sequence σ, the algorithm's cost are at most c times the cost of the optimal solution of the corresponding off-line problem plus some additive constant independent of σ.

The general k-server problem is a natural generalization of the well-known *k-server problem* for which $M_1 = M_2 = \ldots = M_k$ and $z_1 = z_2 = \ldots = z_k$ at each time step. The k-server problem was introduced by Manasse, McGeoch and Sleator [9], who proved a lower bound of k on the competitive ratio of any deterministic algorithm for any metric space with at least $k+1$ points and posed the well-known k-server conjecture saying that there exists a k-competitive algorithm for any metric space. The conjecture has been proved for $k = 2$ [9] and some special metric spaces [2][3]. For $k \geq 3$ the current best upper bound of $2k - 1$ is given by Koutsoupias and Papadimitriou [7].

The *weighted k-server problem* turns out to be much harder. In this problem a weight is assigned to each server and the total cost is the sum of the weighted distances. Fiat and Ricklin [6] prove that for any metric space there exists a set of weights such that the competitive ratio of any deterministic algorithm is at least $k^{\Omega(k)}$. For a uniform metric space, on which the problem is called the weighted paging problem, Feuerstein et al. [5] give a 6.275-competitive algorithm. For $k = 2$ Chrobak and Sgall [4] provided a 5-competitive algorithm and proved that no better competitive ratio is possible.

A weighted k-server algorithm is called competitive if the competitive ratio is independent of the weights. For a general metric space no competitive algorithm is known yet even for $k = 2$. It is easy to see that the general k-server problem is a generalization of the weighted k-server problem as well.

The general 2-server problem in which both servers move on the real line has become well-known as the CNN-problem[4]. Koutsoupias and Taylor [8] emphasize the importance of the CNN-problem as one of the simplest problems in a rich class of so-called sum of task systems [1]. In the sum-problem each system gets a task (request) and only one system has to fulfill its task. Such problems form a richer class than the k-server problem for modelling purposes (see [8]).

Koutsoupias and Taylor [8] prove a lower bound of $6+\sqrt{17}$ on the competitive ratio of any deterministic on-line algorithm for the general 2-server problem, through an instance of the weighted 2-server problem on the real line. They also conjecture that the *work function algorithm* has constant competitive ratio for the general 2-server problem. It seems to be a bad tradition of multiple-server problems to keep unsettled conjectures. For the general 2-server problem the situation was even worse than for the k-server problem: the question if any algorithm exists with constant competitive ratio remained unanswered.

In this paper we answer this question affirmatively, by designing an algorithm and prove an upper boud of $100,000$ on its competitive ratio. The constant is huge, but our goal was indeed to settle the question. We believe that our result gives new insight in the problem and will lead to more and much better algorithms for the general k-server problem in the near future.

Optimal off-line solutions of metrical task systems can easily be found by dynamic programming (see [1]), which yields a $O(n^2)$ time algorithm for the general two server problem. As a result our algorithm can be implemented to work in polynomial time.

2 A competitive algorithm

A request is given by a pair $r_i = (x_i, y_i)$ with x_i a point in metric space $\mathbb{M}_1 = (\mathbb{X}, d_x)$ and y_i in $\mathbb{M}_2 = (\mathbb{Y}, d_y)$. We suppress the sub-indices on the distance since it will always be clear from the context which of the two measures is meant. We denote the two servers as the x- and y-server. The distance $\delta : (\mathbb{M}_1 \times \mathbb{M}_2)^2 \rightarrow \mathbb{R}$ is defined as $\delta((x_1, y_1), (x_2, y_2)) = d(x_1, x_2) + d(y_1, y_2)$.

We say that an online algorithm for the general two server problem is *lazy* if at any request only the server that services the request moves, and halts in the request. Our online algorithm is not lazy but it is easy to make it into a lazy algorithm by treating all moves made by the algorithm as virtual, and move a server only for real when it serves the next request. The triangle inequality ensures that the real movement is no more than the sum of the virtual moves. Moreover, we allow that the virtual moves are made to points outside the metric space. This is useful when we want to make a virtual move to a point between two points a_1 and a_2 of the metric space. This can easily be done by adding a new points a to the metric space and defining for any other point z in the metric space $d(z, a) = \min\{d(z, a_1) + d(a_1, a), d(z, a_2) + d(a_2, a)\}$, where choose $d(a, a_1)$ and $d(a, a_2)$ such that their sum is $d(a_1, a_2)$.

A tour is defined as a directed path in the product space $\mathbb{M}_1 \times \mathbb{M}_2$, and we denote the length of a tour T by $|T|$. We say that a tour T serves the request sequence $r_1, \ldots, r_n$ if there is a sequence of pairs $(\bar{x}_1, \bar{y}_1), \ldots, (\bar{x}_n, \bar{y}_n)$, that lie on T in this order, such that for all $j \in \{1, \ldots, n\}$, $\bar{x}_j = x_j$ or $\bar{y}_j = y_j$.

Given a configuration $(\hat{x}_0, \hat{y}_0)$ and a request sequence $r_1, r_2, \ldots, r_n$ we denote X_{0j} $(0 < j \leq n)$ as the length of the path $\hat{x}_0, x_1, \ldots, x_j$, and X_{ij} $(1 \leq i < j \leq n)$ as the length of the path $x_i, x_{i+1}, \ldots, x_j$. We denote Y_{ij} $(0 \leq i < j \leq n)$ in a similar way.

[4] The name CNN-problem was suggested by Gerhard Woeginger.

2.1 Basic properties and a sketch

We state two important properties of the general 2-server problem, which have inspired the design of our on-line algorithm. They are stated in Lemmas 1 and 2 and illustrated in Figure 2.1. The figure shows a part of an instance of the CNN-problem consisting of 7 requests. Five possible tours are shown. Tour T_D serves all request with the the x-server, starting from the x-coordinate of the first request, hence $|T_D| = X_{1,7}$. Similarly, $|T_E| = Y_{1,7}$. The other three tours each have length much smaller than $\min\{X_{1,7}, Y_{1,7}\}$. Tour T_A is relative far apart from tours T_B and T_C, whereas the tours T_B and T_C are relative close to one another. Lemma 1 states the impossibility of the

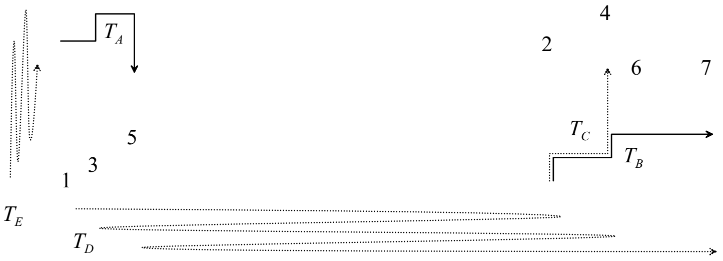

Fig. 1. Part of an instance of the CNN-problem and several feasible tours.

existence of more than two small tours, serving the same request sequence, which are mutually relatively far apart. First we give a notion of closeness of two tours which we employ in this paper. Let T_A and T_B be tours in $\mathbb{M}_1 \times \mathbb{M}_2$. We say that they are *connected* if there are points $x \in \mathbb{X}$ and $y \in \mathbb{Y}$ such that x and y are points on T_A as well as on T_B. (We do not impose that (x, y) is on the tours).

Lemma 1. *Given are three tours T_A, T_B and T_C, each serving a request sequence $r_1, r_2, \ldots, r_n$. If non of the three pairs $\{T_A, T_B\}$, $\{T_B, T_C\}$ and $\{T_A, T_C\}$ is connected, then $|T_A| + |T_B| + |T_C| \geq \min\{X_{1n}, Y_{1n}\}$.*

Proof. Assume without loss of generality that the x-server of T_A shares no request with the x-server of T_B and no request with the x-server of T_C. If the y-server of T_A serves all requests then the lemma obviously holds. So assume request r_i is served by the x-server of T_A. Then T_B and T_C must serve point y_i. But this means that the x-servers of T_B and T_C do not share a request. Hence, the three x-servers share no request. Thus, each request must be served by at least two y-servers and for each two consecutive requests there is a y-server that serves both requests. □

Lemma 2. *If T_A and T_B are connected and (x_{j_1}, y_{j_1}) and (x_{j_2}, y_{j_2}) are points on respectively T_A and T_B, then $\delta((x_{j_1}, y_{j_1}), (x_{j_2}, y_{j_2})) \leq |T_A| + |T_B|$.*

Proof. Let $x \in \mathbb{X}$ and $y \in \mathbb{Y}$ be points that connect both tours. Then,

$$\begin{aligned}\delta((x_{j_1}, y_{j_1}), (x_{j_2}, y_{j_2})) &= d(x_{j_1}, x_{j_2}) + d(y_{j_1}, y_{j_2}) \\ &\leq d(x_{j_1}, x) + d(x, x_{j_2}) + d(y_{j_1}, y) + d(y, y_{j_2}) \\ &\leq |T_A| + |T_B|.\end{aligned}$$

□

The idea behind our competitive algorithm is to try to remain close to an optimal tour, unless the optimal tour is relatively large. The algorithm works in phases, that are separate except that the end positions of the servers in one phase are their starting positions in the next phase. The algorithm is defined such that in each phase it is successful with respect to at least one of the following two goals:
Keeping the tour relatively short in comparison to an optimal tour that serves the same request sequence presented in the phase;
A substantial decrease in distance between the position of its own servers and those of an optimal tour at the end of the phase in comparison to the start of the phase.

In the beginning of each phase the algorithm chooses a reasonable strategy, which we call Balance and which is presented in the following subsection. While applying this strategy the algorithm keeps track of short tours for serving the requests in the phase. If such a short tour emerges then the algorithm tends to move its servers to the server positions of this short tour. If only one such a tour exists or if all short tours are relatively near to each other then the second goal stated above is reached and the phase stops. We know from Lemma 1 that at most two of them may be relatively far apart. In case indeed two such tours exist then the algorithm needs to move its servers to the positions the servers have on one of the short tours. It chooses the one that it is in a certain way nearest.

This choice may turn out to be unfortunate, which is illustrated in Figure 2.1: all requests are given in turn in points 1 and 2. To stay competitive an algorithm, starting the phase from v, must either move to point a or to point b. On an optimal tour the requests could be served at zero cost if this tour started the phase in a or b. If the algorithm moves its servers to a while the servers on an optimal tour appear to be in b then the distance to the optimal tour has even increased after this move. Similarly, if the algorithm moves its servers to b, the optimal servers may turn out to be in a.

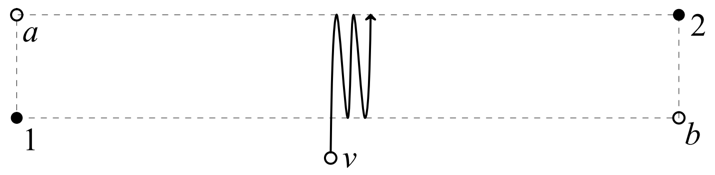

Fig. 2. A difficult choice.

We define the strategy Compete, described in Section 2.3, to avert this potential danger. The achievement of Compete is that, at the end of the phase, small tours, if any exist, have their servers on positions which are concentrated around a single point in $\mathbb{M}_1 \times \mathbb{M}_2$. Once this is achieved the phase is finished by moving the servers in the direction of the positions on the shortest tour at the end of the phase.

It will be clear from the sketch of the algorithm that in each phase, Online carefully chooses its steps in order to stay competitive. If at some moment the requested points are relatively far away for both servers, then the algorithm is forced to make a large step. If this step is much larger than the sum of all preceding steps in the phase, then the phase is terminated immediately and the request is considered as the first request in the new phase.

The precise description of the algorithm, which is found in Section 2.4, is rather technical. The basic ideas have just been sketched, but their implementation allows freedom for specific choices.

The choices we made are motivated by no more and no less than the fact that they allowed us to prove the desired competitiveness. Many other choices are possible, also alternatives for BALANCE and COMPETE and may give even better competitive ratios. However, the main goal of our research was to prove the conjecture that a constant competitive algorithm for the general two-server problem exists. We leave it to future research to find better competitive ratios.

2.2 Algorithm Balance

Algorithm BALANCE is applied at the beginning of each phase of ONLINE and within the subroutine COMPETE. We describe it on a request sequence $r_1, \dots, r_n$ starting from the positions $(\hat{x}_0, \hat{y}_0)$. Let S_j^x and S_j^y be the total costs made by, respectively, the x- and the y-server after serving request r_j, and $S_j := S_j^x + S_j^y$. Let $(\hat{x}_j, \hat{y}_j)$ be the server positions after serving request r_j.

BALANCE:
If $S_j^x + d(\hat{x}_j, x_{j+1}) \leq S_j^y + d(\hat{y}_j, y_{j+1})$, then move the x-server to request x_{j+1}. Else move the y-server to request y_{j+1}.

The following lemma gives an upper bound on the cost of BALANCE.

Lemma 3. $S_j \leq 2\max\{S_j^x, S_j^y\} \leq \min\{X_{0j}, Y_{0j}\}$ $\forall j \in \{0, \dots, n\}$.

Proof. Clearly, $S_j^x \leq X_{0j}$ and $S_j^y \leq Y_{0j}$. Let request r_i, $i \leq j$, be the last request served by the x-server. Then, by definition, $S_j^x = S_i^x \leq S_{i-1}^y + d(\hat{y}_{i-1}, y_i) \leq Y_{0i} \leq Y_{0j}$. Hence, $S_j^x \leq \min\{X_{0j}, Y_{0j}\}$. Similarly it is shown that $S_j^y \leq \min\{X_{0j}, Y_{0j}\}$. □

Lemma 4. $S_{j+1} \leq 3S_j + \min\{d(\hat{x}_0, x_{j+1}), d(\hat{y}_0, y_{j+1})\}$, $\forall j \geq 1$.

Proof. Without loss of generality we assume that $d(\hat{x}_0, x_{j+1}) \leq d(\hat{y}_0, y_{j+1})$. By definition of BALANCE we have $S_{j+1} \leq S_j + \min\{S_j^x + d(\hat{x}_j, x_{j+1}), S_j^y + d(\hat{y}_j, y_{j+1})\} \leq S_j + S_j^x + d(\hat{x}_j, x_{j+1}) \leq S_j + 2S_j^x + d(\hat{x}_0, x_{j+1}) \leq 3S_j + d(\hat{x}_0, x_{j+1})$. □

2.3 Algorithm Compete

We denote the positions of the servers at the beginning of algorithm COMPETE by $(\hat{x}, \hat{y})$. The behavior of the algorithm depends on a parameter $(x^*, y^*) \in \mathbb{M}_1 \times \mathbb{M}_2$, which is regarded as the position of the servers on the alternative short tour. Define $\Delta = \frac{1}{2}\max\{d(\hat{x}, x^*), d(\hat{y}, y^*)\}$. We describe the algorithm in case $\Delta = \frac{1}{2}d(\hat{x}, x^*)$. Interchanging the role of x and y gives the description in case $\Delta = \frac{1}{2}d(\hat{y}, y^*)$. The algorithm works in phases. The only information it takes to the next phase is the current position of its servers. We describe a generic phase on a sequence of requests $r_1, r_2, \dots$. Occasionally both servers make a move after the release of a request. Let S_j^x and S_j^y be the distance travelled by respectively the x- and y-server during the current phase, after both servers made their moves upon the release of request r_j and $(\hat{x}_j, \hat{y}_j)$ their positions at the same time.

A phase of COMPETE(x^*, y^*) :

⟨1⟩ Apply BALANCE, until the release of a request r_j with $d(\hat{x}, x_j) \geq \Delta$ in which case go to Step ⟨2⟩.
⟨2⟩ Apply the following three steps.
  a. If $d(\hat{y}, y_j) < d(\hat{x}, x_j)$, then serve y_j, else serve x_j.

b. If the x-server has not served any request in the phase, and $j > 1$, then move the x-server over a distance S^y_{j-1} towards x_{j-1}.
c. Start a new phase.

The following lemma shows that if the two alternative short tours remain small, then COMPETE remains competitive with the sum of the two short tours. We exploit that COMPETE starts with its servers in the same position as one of the two short tours.

Lemma 5. *Given any request sequence let T be the tour made by Compete(x^*, y^*) starting from position $(\hat{x}, \hat{y})$. Let T_1 and T_2 be tours, both serving the same request sequence and starting in respectively $(\hat{x}, \hat{y})$ and (x^*, y^*). If $|T_1| + |T_2| < \Delta$, then $|T| \leq 10(|T_1| + |T_2|)$.*

Proof. We assume that $\Delta = \frac{1}{2}d(\hat{x}, x^*)$, the other case being similar. Let $(x^{(1)}, y^{(1)})$ and $(x^{(2)}, y^{(2)})$ be the final positions of respectively T_1 and T_2. We give an auxiliary request $(x^{(2)}, y^{(1)})$ at the end, which is served in T_1 and T_2 at no extra cost. Since we assume $|T_1| + |T_2| < \Delta$ we have $d(\hat{x}, x^{(2)}) \geq 2\Delta - d(x^*, x^{(2)}) > \Delta$ and therefore the last phase will end properly, i.e. with a $\langle 2 \rangle$ step.

Consider an arbitrary phase in the algorithm, and suppose this phase contains n requests. We use $(\hat{x}_0, \hat{y}_0)$ for the positions of the servers at the beginning of the phase (in the first phase $(\hat{x}_0, \hat{y}_0) = (\hat{x}, \hat{y})$). Define $S = S^x_n + S^y_n$. Let C^x_1 (C^y_1) and C^x_2 (C^y_2) be the total cost of the x-server (y-server) on, respectively, T_1 and T_2 in the phase and define $C_1 = C^x_1 + C^y_1$, $C_2 = C^x_2 + C^y_2$, and $C = C_1 + C_2$. The positions of the servers in T_1 after serving request r_j, $j \in \{0, \ldots, n\}$ are denoted by (x'_j, y'_j).

We define a potential at the beginning of the phase as $\Psi = 3d(\hat{x}_0, x'_0)$, whence the increase in potential during this phase is $\Delta\Psi = 3d(\hat{x}_n, x'_n) - 3d(\hat{x}_0, x'_0)$. We will prove that $S \leq 10C - \Delta\Psi$. This proves the lemma since taking the sum over all phases yields $|T| \leq 10(|T_1| + |T_2|) - 3d(\hat{x}_N, x'_N) \leq 10(|T_1| + |T_2|)$, where $(\hat{x}_N, x'_N)$ denote the final positions of the x-servers of T and T_1 in the last phase.

Given the condition of the lemma, request r_n is served by the y-servers of T_1, since otherwise $|T_1| \geq d(\hat{x}, x_n) \geq \Delta$. For the same reason, r_n is served by the y-server of T, since otherwise, by definition of Step 2a of COMPETE, $d(\hat{y}, y_n) \geq d(\hat{x}, x_n) \geq \Delta$, implying again $|T_1| \geq \Delta$. Hence, $\hat{y}_0 = y'_0$ and $\hat{y}_n = y'_n = y_n$. To simplify notation we define $y_0 = \hat{y}_0$ $(= y'_0)$. By a similar argument $r_1, \ldots, r_{n-1}$ must be served by the y-server of T_2. We distinguish three cases.

Case 1. The y-server of T_1 serves a request r_k with $k \in \{1, \ldots, n-1\}$. In this case, $C^y_1 \geq d(y_0, y_k) + d(y_k, y_n)$ and $C^y_2 \geq d(y_1, y_k) + d(y_k, y_{n-1})$. By the triangle inequality we have $d(y_0, y_k) + d(y_k, y_n) + C^y_2 \geq d(y_0, y_1) + d(y_n, y_n)$. Hence, $C^y_1 + 2C^y_2 \geq d(y_0, y_1) + d(y_{n-1}, y_n) + C^y_2 \geq S^y_n$. By the use of BALANCE also $d(y_0, y_1) + C^y_2 \geq S^x_n$. Clearly the increase in potential is bounded by $\Delta\Psi \leq 3(C^x_1 + S^x)$. We obtain

$$\begin{aligned}
S &= 4S^x_n + S^y_n - 3S^x_n \\
&\leq 5C^y_1 + 10C^y_2 + 3C^x_1 - \Delta\Psi \\
&\leq 10C - \Delta\Psi.
\end{aligned}$$

Case 2. The y-server of T_1 serves only r_0 and r_n and the x-server of COMPETE *serves no request in the phase.* In this case $C^y_1 = d(y_0, y_n)$, $x'_n = x_{n-1}$ (for $n \geq 2$), and $S^x_n = S^y_{n-1}$. The increase in potential is $\Delta\Psi \leq 3(C^x_1 - S^x_n)$. Therefore,

$$\begin{aligned}
S &= S^x_n + S^y_n \\
&\leq S^x_n + 2S^y_{n-1} + d(y_0, y_n) \\
&= 3S^x_n + d(y_0, y_n) \\
&\leq 3C^x_1 - \Delta\Psi + d(y_0, y_n) \\
&\leq 3C - \Delta\Psi.
\end{aligned}$$

Case 3. The y-server of T_1 serves only r_0 and r_n and the x-server of Compete *serves a request r_j with $j \in \{1, \dots, n-1\}$.* Again, $C_1^y = d(y_0, y_n)$ and $x'_n = x_{n-1}$ (for $n \geq 2$). The increase in potential is $\Delta\Psi \leq 3(C_1^x - d(\hat{x}_0, x_1))$. Clearly $S_n^x \leq d(\hat{x}_0, x_1) + C_1^x$, and by definition of Balance also $S_{n-1}^y \leq d(\hat{x}_0, x_1) + C_1^x$. We obtain,

$$\begin{aligned} S &= S_n^x + S_n^y \\ &\leq S_n^x + 2S_{n-1}^y + d(y_0, y_n) \\ &\leq 6C_1^x - \Delta\Psi + d(y_0, y_n) \\ &\leq 6C - \Delta\Psi. \end{aligned}$$

□

2.4 Algorithm Online.

Algorithm Online works in phases. The only information taken from one phase to the next phase is the position of the servers at the end of the phase. Each phase starts by applying the simple Balance routine, until it becomes clear that there exists a short tour whose servers positions are not far from the starting position of Online in the phase. At that moment, it makes an extra move to the positions of the servers on a short tour. If there is only one such a tour or every two such tours are relatively near to each other, then the phase ends. Otherwise Online switches to the subroutine Compete. We notice that all these short tours do not need to start the phase in the same position as Online. The phase ends with again an extra move. As announced in the sketch of the algorithm the phase can also end prematurely as soon as a request is presented that requires a relatively large move, which is defined in the algorithm by the *exception rule*.

We denote $v_0 = (O, O)$ as the starting position in phase 1, and v_i as the end position in phase i ($i \geq 1$), and hence the starting position in phase $i+1$. The requests given in phase i are denoted by $r_{i1}, r_{i2}, \dots$. We describe a generic phase of Online and to facilitate the exposition we suppress the subindex i of the phase. Thus, we denote the requests in the generic phase by $r_1, r_2, \dots$, the starting position by v_{-1} and the end position by v. We denote S_j ($j \geq 1$) as the cost of serving $r_1, r_2, \dots r_j$ by Balance in Step (1) of the algorithm. Similarly, Z_j we denote the length of the tour by Compete in Step (2).

As indicated above the algorithm occasionally makes more moves than necessary to serve a next request r_j. We denote $(\hat{x}_j, \hat{y}_j)$ as the position of Online after all moves are made, and we denote A_j as the cost of Online until this moment. Additionally, we use the notation $(\hat{x}_0, \hat{y}_0)$ for the initial positions v_{-1} of the Online servers in the phase. The constant η appearing in the description has value $\eta = 1/120$. For the tours T_A, T_B, T_C, and T_D in the description, v_A, v_B, v_C, and v_D represent their end points (end position of their servers).

A phase of Online :

Apply the Steps (1),(2) and (3), with the following *exception rule*: If at any moment a request $r_j = (x_j, y_j)$ ($j \geq 2$) is released for which $\min\{d(\hat{x}_0, x_j), d(\hat{y}_0, y_j)\} \geq 4A_{j-1}$, then return to $v_{-1} = (\hat{x}_0, \hat{y}_0)$ and let (x_j, y_j) be the first request in a new phase.

(1) Apply Balance. At the release of a request r_j for which there is a tour T_A, with $|T_A| < \eta S_j$ and $\delta(v_{-1}, v_A) \leq S_j/3$, return to v_{-1} after serving r_j, and continu with Step (2). Let r_{k_1} be this request.

(2) Let T_B and T_C be tours, serving $r_1, \dots, r_{k_1}$, with $\max\{|T_B|, |T_C|\} < \eta S_{k_1-1}$, and $\delta(v_B, v_C) \geq 16\eta S_{k_1-1}$. If no such tours exist, then define $k_2 = k_1$ and continue with (3). Assume w.l.o.g. that $\delta(v_{-1}, v_B) \leq \delta(v_{-1}, v_C)$. Move the servers to v_B and apply Compete(v_C) until $Z_j \geq S_{k_1-1}$. Let r_{k_2} be this request. Move the servers back to v_{-1}.

(3) Let T_D serve $r_1, \ldots, r_{k_2}$ and have minimum length. If $|T_D| < \eta S_{k_1-1}$, then move from v_{-1} towards v_D over a distance $\frac{1}{3} S_{k_1-1}$. Start a new phase from this position.

We emphasize that Z_j, $(k_1 + 1 \leq j \leq k_2)$ is the cost COMPETE(v_C) makes starting from v_B.

3 Competitive analysis

In the competitive analysis we distinguish two types of phases. The phases of type $\mathbb{I}$ are those that terminated prematurely by the exception rule. The other phases are of type $\mathbb{II}$. Notice that the last phase, which we denote by N, is of type $\mathbb{II}$ since any phase of type $\mathbb{I}$ is followed by at least one more phase.

For the analysis we introduce a potential function Φ that measures the distance of the position of the ONLINE servers to those of the optimal tour. The potential at the beginning of a phase i is $\Phi_{i-1} = 1000 \cdot \delta(v_{i-1}, v^*_{i-1})$. We define the potential at the end of the last phase N to be zero, i.e., $\Phi_N = 0$.

We will prove in Lemma 8 that in each phase of type $\mathbb{II}$ either the ONLINE tour is relatively short with respect to the optimal tour over the requests in the phase or the potential function has decreased substantially over the phase. First, in the next two lemmas we will bound the length of the ONLINE tour in a type $\mathbb{II}$ phase in terms of the bound from Lemma 1, which in a sense bounds the length of tours from below.

Consider an arbitrary phase of type $\mathbb{II}$, and suppose it contains n requests $r_1, \ldots, r_n$. As before, we denote X_{ij} $(0 \leq i < j \leq n)$ as the length of the path $x_i, x_{i+1}, \ldots, x_j$ (For $i = 0$ this path is $\hat{x}_0, x_1, \ldots, x_j$). We denote Y_{ij} in a similar way. To simplify notation we write, in the following lemmas, S shortly for S_{k_1-1}, the length of the tour that BALANCE makes in Step (1) of ONLINE. By $|P|$ we denote the length of the ONLINE tour in the phase.

Lemma 6. *For each phase of type $\mathbb{II}$,* $\min\{X_{1k_1}, Y_{1k_1}\} \geq (1/12 - \eta/2)S$

Proof. Assume w.l.o.g. $X_{1k_1} \leq Y_{1k_1}$ and notice that Lemma 3 implies $S_{k_1} \leq 2\min\{X_{0k_1}, Y_{0k_1}\}$. Tour T_A cannot serve merely y-requests, since in that case $\delta(v_{-1}, v_A) \geq d(\hat{y}_0, y_1) - |T_A| = Y_{0k_1} - Y_{1k_1} - |T_A| \geq Y_{0k_1} - 2|T_A| \geq (1/2 - 2\eta)S_{k_1} > S_{k_1}/3$. Now, let x_j be the last x-request served by T_A. In this case we obtain $S_{k_1}/3 \geq \delta(v_{-1}, v_A) \geq d(\hat{x}_0, x_j) - |T_A| \geq X_{0k_1} - 2X_{1k_1} - |T_A| \geq (1/2 - \eta)S_{k_1} - 2X_{1k_1}$. Hence, $X_{1k_1} \geq (1/12 - \eta/2)S_{k_1} \geq (1/12 - \eta/2)S$. □

Lemma 7. *For each phase of type $\mathbb{II}$, $|P| < 191S$.*

Proof. Notice that Lemma 4 and the exception rule imply $S_{k_1} \leq 3S + \min\{d(\hat{x}_0, x_{k_1}), d(\hat{y}_0, y_{k_1})\} \leq 7S$. The cost made in Step (1) is at most $2S_{k_1} \leq 14S$ and the cost in Step (3) is at most $S/3$. If COMPETE is not applied in the phase, then only Step (1) and (3) add to the cost, whence $|P| \leq 14\frac{1}{3}S$. So assume COMPETE is applied.

First we bound $\delta(v_{-1}, v_B)$. Applying Lemma 6, $|T_A| + |T_B| + |T_C| \leq 7\eta S + \eta S + \eta S < \min\{X_{1k_1}, Y_{1k_1}\}$. Hence, these three tours do not satisfy the property of Lemma 1. By Lemma 2 the tours T_B and T_C cannot be connected, whence T_A must be connected to T_B or T_C. In the first case we have $\delta(v_{-1}, v_B) \leq \delta(v_{-1}, v_A) + |T_A| + |T_B|$, applying Lemma 3. Similarly if T_A is connected to T_C then $\delta(v_{-1}, v_C) \leq \delta(v_{-1}, v_A) + |T_A| + |T_C|$. Since we assumed $\delta(v_{-1}, v_B) \leq \delta(v_{-1}, v_C)$ we have $\delta(v_{-1}, v_B) \leq \delta(v_{-1}, v_A) + |T_A| + \min\{|T_B|, |T_C|\} \leq S_{k_1}/3 + \eta S_{k_1} + \eta S < 4S$.

Next we bound the cost made by COMPETE. Since by definition of ONLINE $Z_{k_2-1} < S$ the total cost after r_{k_2-1} is served is $A_{k_2-1} < 14S + 4S + S = 19S$. If r_{k_2} is served in Step $\langle 1 \rangle$ of COMPETE then the last step was a BALANCE step in a phase of COMPETE. Assume the phase started in the point (x, y). By Lemma 4,

$$\begin{aligned} Z_{k_2} &< 3S + \min\{d(x, x_{k_2}), d(y, y_{k_2})\} \\ &\leq 3S + \delta((x, y), (\hat{x}_0, \hat{y}_0)) + \min\{d(\hat{x}_0, x_{k_2}), d(\hat{y}_0, y_{k_2})\} \\ &\leq 3S + 5S + 4 \cdot 19S \\ &= 84S. \end{aligned}$$

For the third inequality we used the exception rule. Now assume the request is served in Step $\langle 2\rangle$ of COMPETE. The cost of the Step in $\langle 2\rangle$a is at most $Z_{k_2-1} + \min\{d(\hat{x}_{k_1}, x_{k_2}), d(\hat{y}_{k_1}, y_{k_2})\}$, and the cost in $\langle 2\rangle$b is no more than Z_{k_2-1}. Therefore,

$$\begin{aligned} Z_{k_2} &\leq 3Z_{k_2-1} + \min\{d(\hat{x}_{k_1}, x_{k_2}), d(\hat{y}_{k_1}, y_{k_2})\} \\ &< 3S + \delta((\hat{x}_0, \hat{y}_0), (\hat{x}_{k_1}, \hat{y}_{k_1})) + \min\{d(\hat{x}_0, x_{k_2}), d(\hat{y}_0, y_{k_2})\} \\ &\leq 3S + 4S + 4 \cdot 19S \\ &= 83S. \end{aligned}$$

We conclude that the total cost in the phase is no more than $14S + 2(4S + 84S) + S/3 = 190\frac{1}{3}S$. $\square$

In the following crucial lemma we use $|P^*|$ as the length of an optimal tour to serve the request in the phase considered. We use v^*_{-1} and v^* for the starting and finishing positions of the servers on an optimal tour in the phase. In accordance with suppressing the subindex for the phase, we also write Φ_{-1} and Φ for the potential function, respectively at the beginning and at the end of the phase.

Lemma 8. *For each phase of type* $\mathbb{III}$, $2|P| < 10^5|P^*| - \Phi + \Phi_{-1}$.

Proof. First assume the phase considered is not the last phase. We have to show that

$$F \equiv c_1|P^*| + c_2\left(\delta(v_{-1}, v^*_{-1}) - \delta(v, v^*)\right) > 2|P|, \tag{1}$$

with $c_1 = 10^5$ and $c_2 = 10^3$. We distinguish three cases.

Case 1. $|P^*| \geq \eta S$. In this case $\delta(v_{-1}, v^*_{-1}) - \delta(v, v^*) \geq -\delta(v, v_{-1}) - \delta(v^*, v^*_{-1}) \geq -S/3 - |P^*|$. Inequality (1) becomes in this case $F \geq c_1|P^*| - c_2(S/3 + |P^*|) = (c_1 - c_2)|P^*|) - c_2 S/3 > 382S > 2|P|$.

Case 2. $|P^*| < \eta S$ *and* COMPETE *was not applied.* From the proof of Lemma 7 $|P| \leq 14\frac{1}{3}S$. By definition of ONLINE the endpoint of any tour serving requests $r_1, \ldots, r_{k_1-1}$ and with length smaller than ηS, must be at a distance greater than $S/3$ from point v_{-1}. In particular $\delta(v_{-1}, v_D) > S/3$. Since COMPETE was not applied $\delta(v_D, v^*) < 16\eta S$. By the triangle inequality

$$\delta(v_{-1}, v^*) \geq \delta(v_{-1}, v_D) - \delta(v_D, v^*) > \delta(v_{-1}, v_D) - 16\eta S,$$

and

$$\delta(v, v^*) \leq \delta(v_{-1}, v_D) - S/3 + \delta(v_D, v^*).$$

Hence,

$$\begin{aligned} \delta(v_{-1}, v^*_{-1}) - \delta(v, v^*) &\geq \delta(v_{-1}, v^*) - \delta(v, v^*) - \delta(v^*_{-1}, v^*) \\ &> S/3 - 32\eta S - |P^*| \\ &= 8\eta S - |P^*|. \end{aligned}$$

Hence, $F > c_1|P^*| + c_2(8\eta S - |P^*|) = (c_1 - c_2)|P^*| + 8c_2\eta S \geq 8c_2\eta S > 29S > 2|P|$.

Case 3. $|P^*| < \eta S$ *and* COMPETE *was applied.* Let T'_B be an optimal extensions of T_B, i.e. it starts in v_B, serves the requests $r_{k_1+1}, \ldots, r_{k_2}$ and has minimum length. Define T'_C similar as

$T_B^{'}$ with respect to T_C. Now we apply Lemma 5 with the parameters $(\hat{x},\hat{y}) = v_B$, $(x^*,y^*) = v_C$, $T_1 = T_B^{'}, T_2 = T_C^{'}$, and $\Delta = \frac{1}{2}\max\{d(\hat{x},x^*), d(\hat{y},y^*)\} \geq \delta(v_B,v_C)/4 \geq 4\eta S$. Lemma 5 implies

$$|T_B^{'}| + |T_C^{'}| \geq \min\{\Delta, S/10\} \geq 4\eta S. \tag{2}$$

Now let $\widehat{T}_B$ and $\widehat{T}_C$ be arbitrary tours that serve $r_1,\ldots,r_{k_2}$ and are connected with T_B and T_C respectively. Assume that T_B and $\widehat{T}_B$ both serve the requests x_i and y_j for some $i,j \in \{1,\ldots,k_1\}$. A possible extension of T_B is to move the servers to (x_i,y_j) and serve the requests $r_{k_1+1},\ldots,r_{k_2}$ similar to $\widehat{T}_B$. This implies $|T_B^{'}| \leq |T_B| + |\widehat{T}_B|$. Similarly $|T_C^{'}| \leq |T_C| + |\widehat{T}_C|$. Together with (2) this yields

$$|\widehat{T}_B| + |\widehat{T}_C| \geq |T_B^{'}| + |T_C^{'}| - |T_B| - |T_C| \geq 2\eta S. \tag{3}$$

Let T be an arbitrary tour that serves $r_1,\ldots,r_{k_2}$ with $|T| < \eta S$. Since $|T_B| + |T_C| + |T| < 3\eta S < \min\{X_{1k_1}, Y_{1k_1}\}$, these three tours do not satisfy the property of Lemma 1. (To apply Lemma 1 strictly we should consider T restricted to the first k_1 requests.) Since T_B and T_C are not connected (using Lemma 2) tour T must be connected with either T_B or T_C. With (3) this implies that either any such tour T is connected with T_B or any such tour is connected with T_C. Hence the optimal tour P^*, and the tour T_D defined by ONLINE, are both connected with T_B or are both connected with T_C. This implies $\delta(v_D,v^*) \leq |T_D| + \max\{|T_B|,|T_C|\} + |P^*| < 2\eta S + |P^*|$. With the triangle inequality we obtain

$$\begin{aligned}
&\delta(v_{-1},v_{-1}^*) - \delta(v,v^*)\\
&\geq \delta(v_{-1},v^*) - \delta(v,v^*) - \delta(v_{-1}^*,v^*)\\
&\geq (\delta(v_{-1},v_D) - \delta(v,v_D)) - 2\delta(v_D,v^*) - |P^*|\\
&> S/3 - 2(2\eta S + |P^*|) - |P^*|\\
&= 2S/5 - 3|P^*|
\end{aligned}$$

Inequality (1) becomes $F > c_1|P^*| + c_2(2S/5 - 3|P^*|) > c_2 \cdot 2S/5 > 382S > 2|P|$.

It is clear that the inequality of the lemma also holds for the last phase of ONLINE, phase N, in case this phase finishes with Step (3). That the inequality als holds if this phase finishes in one of the two other steps is a matter of case checking, which we omit in this extended abstract. If the phase finishes in one of the two other steps, then a much better inequality can be obtained by going through the analysis above. We omit this in this extended abstract.

□

Constant competitiveness of ONLINE is now an easy consequence. We use P_i and P_i^* for the ONLINE and the optimal tour in phase i, respectively.

Theorem 1. ONLINE *is* 100.000-*competitive for the general two server problem.*

Proof. Consider a phase j of type $\mathbb{I}$. Since ONLINE ends the phase in the same position as it started, any increase in potential is caused by the change of positions of the servers on the optimal tour only. Hence, $10^5 \cdot |P_j^*| - \Phi_j + \Phi_{j-1} \geq 10^5 \cdot |P_j^*| - 10^3|P_j^*| \geq 0$. On the other hand, any phase j of type $\mathbb{I}$ is followed by a phase $j+1$ in which the cost of the first step, is at least twice the total cost $|P_j|$ of phase j. The last phase is of type $\mathbb{III}$ implying that the ONLINE cost over all phases of type $\mathbb{I}$ is at most $\frac{1}{2} + \frac{1}{4} + \ldots = 1$ times the cost over all phases of type $\mathbb{III}$. We conclude that

$$\sum_{j\in\mathbb{I}\cup\mathbb{III}} |P_j| \;\leq\; 2\sum_{j\in\mathbb{III}} |P_j| \;<\; \sum_{j\in\mathbb{III}} 10^5 \cdot |P_j^*| - \Phi_j + \Phi_{j-1} \leq$$

$$\sum_{j\in\mathbb{I}\cup\mathbb{III}} 10^5 \cdot |P_j^*| - \Phi_j + \Phi_{j-1} \;=\; \sum_{j\in\mathbb{I}\cup\mathbb{III}} 10^5 \cdot |P_j^*|.$$

□

References

1. Allan Borodin, Nathan Linial, and Michael Saks, *An optimal online algorithm for metrical task system*, Journal of the ACM **39** (1992), 745–763.
2. Marek Chrobak, Howard Karloff, Tom H. Payne, and Sundar Vishwanathan, *New results on server problems*, SIAM Journal on Discrete Mathematics **4** (1991), 172–181.
3. Marek Chrobak and Lawrence L. Larmore, *An optimal online algorithm for k servers on trees*, SIAM Journal on Computing **20** (1991), 144–148.
4. Marek Chrobak and Jiří Sgall, *The weighted 2-server problem*, The 17th Annual Symposium on Theoretical Aspects of Computer Science (STACS), LNCS 1770, Springer-Verlag, 2000, pp. 593–604.
5. Esteban Feuerstein, Steve Seiden, and Alejandro Strejlevich de Loma, *The related server problem*, Unpublished manuscript, 1999.
6. Amos Fiat and Moty Ricklin, *Competitive algorithms for the weighted server problem*, Theoretical Computer Science **130** (1994), 85–99.
7. Elias Koutsoupias and Christos Papadimitriou, *On the k-server conjecture*, Journal of the ACM **42** (1995), 971–983.
8. Elias Koutsoupias and David Taylor, *The* CNN *problem and other k-server variants*, The 17th Annual Symposium on Theoretical Aspects of Computer Science 2000 (STACS), LNCS 1770, Springer-Verlag, 2000, pp. 581–592.
9. Mark Manasse, Lyle A. McGeoch, and Daniel Sleator, *Competitive algorithms for server problems*, Journal of Algorithms **11** (1990), 208–230.