Online k-server routing problems[*]

Vincenzo Bonifaci[†] Leen Stougie[‡]

November 23, 2007

Abstract

In an online k-server routing problem, a crew of k servers has to visit points in a metric space as they arrive in real time. Possible objective functions include minimizing the makespan (k-Traveling Salesman Problem) and minimizing the sum of completion times (k-Traveling Repairman Problem). We give competitive algorithms, resource augmentation results and lower bounds for k-server routing problems in a wide class of metric spaces. In some cases the competitive ratio is dramatically better than that of the corresponding single server problem. Namely, we give a $1 + O((\log k)/k)$-competitive algorithm for the k-Traveling Salesman Problem and the k-Traveling Repairman Problem when the underlying metric space is the real line. We also prove that a similar result cannot hold for the Euclidean plane.

1 Introduction

In a k-server routing problem, k servers (vehicles) move in a metric space in order to visit a set of points (cities). Given a schedule, that is, a sequence of movements of the servers, the time at which a city is visited for the first time by one of the servers is called the *completion time* of the city. The objective is to find a schedule that minimizes some function of the completion times.

We study k-server routing problems in their *online* version, where decisions have to be taken without having any information about future requests. New requests may arrive while processing previous ones. This online model is often called the *real time* model, in contrast to the *one-by-one* model, which is the

[*]An extended abstract of this work has appeared in the proceedings of the 4th Workshop on Approximation and Online Algorithms, September 2006.

[†]Dept. Computer and Systems Science, University of Rome "La Sapienza", Rome, Italy; Dept. Mathematics and Computer Science, Eindhoven University of Technology, Eindhoven, the Netherlands. Email: bonifaci@dis.uniroma1.it. Research partly supported by the Dutch Ministry of Education, Culture and Science through a Huygens scholarship.

[‡]Dept. Mathematics and Computer Science, Eindhoven University of Technology, Eindhoven, the Netherlands; Center for Mathematics and Computer Science (CWI), Amsterdam, the Netherlands. Email: leen@win.tue.nl. Research partly supported by MRT Network ADONET of the European Community (MRTN-CT-2003-504438) and the Dutch BSIK/BRICKS project.

more common model in texts about online optimization [6], but inadequate for server routing problems. The same real time model is also the natural model and indeed is used for machine scheduling problems [24]. In fact, many of the algorithms for online routing problems are adaptations of online machine scheduling algorithms.

Competitive analysis [6] has become the standard way to study online optimization problems: an online algorithm A is said to be *c-competitive* if, for any instance σ, the cost of A on σ is at most c times the offline optimum cost on σ. This worst-case measure can be seen as the outcome of a game between the online algorithm and an offline *adversary*, that is trying to build input instances for which the cost ratio is as large as possible.

There is an abundant amount of literature on offline server routing problems, both in past and recent times [8, 11, 13, 15, 20]. Online single server routing problems have a recent but growing literature. The first paper by Ausiello et al. [3] introduced the model for the online traveling salesman problem. Later works investigated competitiveness of the more general dial-a-ride problems, in which requests consist of objects that need to be transported from a source to a destination [1, 12], and studied different objective functions or different adversarial models [2, 4, 5, 14, 18, 19, 22]. An overview of single server results is contained in the thesis [21].

Prior to this publication, there was essentially no work on online multi-server routing problems, except for some isolated algorithms [1, 4]. We give competitive algorithms and negative results for online multi-server routing problems, with the objective of minimizing either *makespan* or *sum of completion times*. We consider the variant known as *nomadic*, in which the servers are not required to return at the origin after serving all requests; the above cited previous results apply to the other variant, known as the *homing* traveling salesman problem. Apart from being the first work dedicated to multi-server online routing problems, the results are somewhat unexpected. We give the first results of online problems for which multiple server versions admit lower competitive ratios than their single server counterparts. This is typically not the case for problems in the one-by-one model; for example, it is known that in the famous *k-server* problem [23] the competitive ratio necessarily grows linearly with k.

It may also be useful to draw a comparison with machine scheduling, which is closer to routing problems in many ways. In scheduling, multiple machine problems have been the subject of much research [24]. In the one-by-one model, competitive ratios increase with increasing number of machines. In real time online scheduling, nobody has been able to show smaller competitive ratios for multiple machine problems than for the single machine versions, though here lower bounds do not exclude that such results exist (and indeed people suspect they do) [9, 10].

The rest of our paper is structured as follows. After introducing our model in Section 2, we give in Section 3 competitive algorithms and lower bounds for both the k-Traveling Salesman and the k-Traveling Repairman in general spaces. For these algorithms, the upper bounds on the competitive ratio match those of the best known algorithms for the single server versions. In Section 4, we show that in the case of the real line we have an almost optimal algorithm for large k. The same result cannot hold in the Euclidean plane, as we show in Section 5. We give our conclusions in Section 6.

2 Preliminaries

We assume a real time online model, in which requests arrive over time in a metric space $\mathbb{M}$. Every *request* is a pair $(r, x) \in \mathbb{R}_+ \times \mathbb{M}$ where r is the *release date* of the request and x the location of the request. All the information about a request with release date r, including its existence, is revealed only at time r. Thus, an online algorithm does not know the moment when all requests have been released.

An algorithm controls k vehicles or *servers*. Initially, at time 0, all these servers are located in a distinguished point $o \in \mathbb{M}$, the origin. The algorithm can then move the servers around the space at speed at most 1. (We do not consider the case in which servers have different maximum speeds; in compliance with machine scheduling vocabulary we could say that the servers are identical and work in parallel.) To process, or *serve*, a request, a server has to visit the associated location, but not earlier than the release date of the request. Introduced by Ausiello et al. [3], in the on-line routing literature this version was called the *nomadic* TSP, as opposed to the *homing* TSP, in which the server has to return to the origin after serving all requests. Since we study here only the former version we omit the adjective.

We consider so-called *path metric* spaces, in which the distance d between two points is equal to the length of the shortest path between them. We also require the spaces to be continuous, in the sense that $\forall x, y \in \mathbb{M}\ \forall a \in [0, 1]$ there is $z \in \mathbb{M}$ such that $d(x, z) = ad(x, y)$ and $d(z, y) = (1 - a)d(x, y)$. A discrete space, like a weighted graph, can be extended to a continuous path metric space in the natural way; the continuous space thus obtained is said to be *induced* by the original space. We recall that a function $d : \mathbb{M}^2 \to \mathbb{R}_+$ is a *metric* if it satisfies: definiteness ($\forall x, y \in \mathbb{M}$, $d(x, y) = 0 \Leftrightarrow x = y$); symmetry ($\forall x, y \in \mathbb{M}$, $d(x, y) = d(y, x)$); triangle inequality ($\forall x, y, z \in \mathbb{M}$, $d(x, z) + d(z, y) \geq d(x, y)$). When referring to a *general space*, we mean any element of our class of continuous, path metric spaces. We will also be interested in special cases, namely the real line $\mathbb{R}$ and the real halfline $\mathbb{R}_+$, both with the origin o at 0, and the plane $\mathbb{R}^2$, with o at $(0, 0)$.

Defining the *completion time* of a request as the time at which the request has been served, the *k-traveling salesman problem* (k-TSP) has objective minimizing the *maximum* completion time (the *makespan*), and the *k-traveling repairman problem* (k-TRP) has objective minimizing the *sum* of completion times.

We will use σ to denote a sequence of requests. Given σ, a feasible *schedule* for σ is a sequence of moves of the servers such that all requests in σ are served. $\textsc{ol}(\sigma)$ is the cost online algorithm $\textsc{ol}$ incurs on σ, and $\textsc{opt}(\sigma)$ the optimal offline cost on σ. $\textsc{ol}$ is said to be *c-competitive* if $\textsc{ol}(\sigma) \leq c \cdot \textsc{opt}(\sigma)$ for all σ. The *competitive ratio* of $\textsc{ol}$ is the smallest c such that $\textsc{ol}$ is c-competitive.

Problem	Lower Bound	Upper Bound
k-TSP	2	$1+\sqrt{2}$
k-TRP	2	$(1+\sqrt{2})^2$
k-TSP ($\mathbb{R}^2$)	4/3	$1+\sqrt{2}$
k-TRP ($\mathbb{R}^2$)	5/4	$(1+\sqrt{2})^2$
k-TSP ($\mathbb{R}$)	$1+\Omega(1/k)$	$1+O((\log k)/k)$
k-TRP ($\mathbb{R}$)	$1+\Omega(1/k)$	$1+O((\log k)/k)$

Table 1: The competitive ratio of multiserver routing problems.

We use $s_1, \ldots, s_k$ to denote the k servers, and write $s_j(t)$ for the position of server s_j at time t, and $d_j(t)$ for $d(s_j(t), o)$. Finally, given a path P in $\mathbb{M}$, we denote its length by $\ell(P)$.

All the lower bounds we prove hold for randomized algorithms against an oblivious adversary [6]. In order to prove these results, we frequently resort to the following form of Yao's principle [7, 25].

Theorem 2.1 (Yao's principle)**.** *Let $\{\text{OL}_y : y \in \mathcal{Y}\}$ denote the set of deterministic online algorithms for an online minimization problem. If X is a distribution over input sequences $\{\sigma_x : x \in \mathcal{X}\}$ such that*

$$\inf_{y\in\mathcal{Y}} \mathbb{E}_X[\text{OL}_y(\sigma_x)] \geq c\,\mathbb{E}_X[\text{OPT}(\sigma_x)]$$

for some real number $c \geq 1$, then c is a lower bound on the competitive ratio of any randomized algorithm against an oblivious adversary.

A summary of our results is presented in Table 1.

3 Algorithms for general metric spaces

In this section, we give competitive algorithms and lower bounds for the k-TSP and the k-TRP in general spaces. Our results will be formulated in a more general *resource augmentation* framework [16]. We define the (k, k^*)-TSP and (k, k^*)-TRP exactly as the k-TSP and the k-TRP, except that we measure the performance of an online algorithm with k servers relative to an optimal offline algorithm with $k^* \leq k$ servers.

Sections 3.1 and 3.2 give an algorithm for the (k, k^*)-TSP and the (k, k^*)-TRP respectively. A lower bound for both problems is proved in Section 3.3.

3.1 The k-Traveling Salesman Problem

Theorem 3.1. *There is a deterministic online algorithm for the (k, k^*)-TSP with competitive ratio*

$$1 + \sqrt{1 + 1/2^{\lfloor k/k^* \rfloor - 1}}.$$

Algorithm 1 Group Return Home (GRH)

Divide the servers into $g = \lfloor k/k^* \rfloor$ disjoint sets (*groups*) of k^* servers each. Any remaining server is not used by the algorithm.

Initially, all servers wait at o. Every time a new request arrives, all servers not at o return to the origin at full speed. Once all of the servers in one of the groups, say group G (ties broken arbitrarily), are at o, compute a set of k^* paths $\{P_1, \ldots, P_{k^*}\}$ starting at o, covering all unserved requests and minimizing $\max_i \ell(P_i)$. Then, for $i = 1, \ldots, k^*$, the i-th server in G follows path P_i at the highest possible speed while remaining at a distance at most αt from o at any time t, for some constant $\alpha \in (0, 1]$. Servers in other groups continue to head towards o (or wait there) until a new request is released.

The algorithm achieving this bound is called Group Return Home (Algorithm 1). The algorithm tries to always keep a group of k^* servers near the origin, so it can start quickly a new schedule if needed. This involves a tradeoff with the speed at which the current active group is able to complete its schedule. The following Lemma shows that indeed there is always a group relatively close to the origin. Define the *distance of a group to the origin* at time t as the maximum distance of a server in the group to o at time t.

Lemma 3.2. *At any time t, in the schedule generated by* GRH*, let $G_1(t), \ldots, G_g(t)$ be the g groups in order of nondecreasing distance to o. Then the distance of $G_i(t)$ to o is at most $2^{i-g}\alpha t$.*

Proof. We prove the lemma by induction on the number of requests. That is, we show that if the lemma holds at the release date t of some request, it will hold until the release date t^+ of the next request. Obviously, the lemma is true up to the time the first request is given, since all servers remain at o.

Suppose a request is given at time t. By induction, we know that there are groups $G_1(t), \ldots, G_g(t)$ such that each server of group $G_i(t)$ is at distance at most $2^{i-g}\alpha t$ from o. For the rest of the proof we fix the order of the groups as the order they have at time t and write G_i instead of $G_i(t)$. Let $D_i(\tau)$ be the distance of group G_i to the origin, that is $D_i(\tau) = \max_{s \in G_i} d(s(\tau), o)$.

Between time t and $t' = t + D_1(t)$, the lemma holds since all servers are getting closer to o. We show that the lemma holds at $t' + \delta$ for all $0 < \delta < t^+ - t'$. Notice that $D_1(t' + \delta) \leq \delta$ since every server moves at most at unit speed.

If $\delta \in (0, 2^{1-g}\alpha t]$, we know that $D_1(t' + \delta) \leq 2^{1-g}\alpha t$, so the lemma holds with the groups in the same order as before.

Now, let $\delta \in (2^{i-1-g}\alpha t, 2^{i-g}\alpha t]$ for $2 \leq i \leq g$. Then at time $t' + \delta$, group G_j is already at o for each $1 < j < i$. For group G_i, $D_i(t' + \delta) \leq 2^{i-g}\alpha t - 2^{i-1-g}\alpha t = 2^{i-1-g}\alpha t$. For group G_1, $D_1(t' + \delta) \leq 2^{i-g}\alpha t$. For groups G_{i+1} through G_g, $D_{i+1}(t' + \delta) \leq 2^{i+1-g}\alpha t, \ldots, D_g(t' + \delta) \leq 2^0 \alpha t$. So the lemma holds for these values of δ.

The last case is $\alpha t < \delta < t^+ - t'$. In this case all groups except G_1 are at o, and because of the speed constraint $D_1(t' + \delta) \leq \alpha(t' + \delta)$. Thus the lemma holds. □

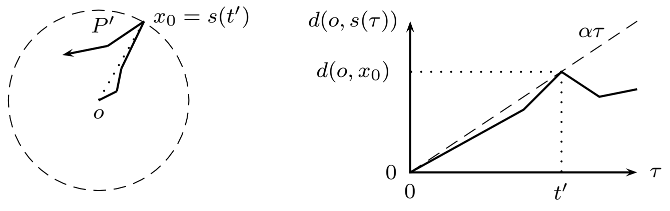

Figure 1: The path followed by s (left) and the variation of $d(o, s)$ (right).

Proof of Theorem 3.1. Let t be the release date of the last request and let G_1 be the group minimizing the distance to the origin at time t. Using Lemma 3.2 we know that $D_1(t) \leq 2^{1-g}\alpha t$. Group G_1 will return to the origin and then follow the offline set of paths $\{P_1, \dots, P_{k^*}\}$. Notice that $\text{OPT}(\sigma) \geq t$, since no schedule can end before the release date of a request, and $\text{OPT}(\sigma) \geq \max_i \ell(P_i)$ because of the optimality of P_i.

Let s be the server in G_1 that achieves the makespan. If s does not limit its speed after time t, we have $\text{GRH}(\sigma) \leq t + D_1(t) + \max_i \ell(P_i) \leq (2 + 2^{1-g}\alpha)\text{OPT}(\sigma)$.

Otherwise, let t' be the last time at which the invariant $d(o, s(\tau)) \leq \alpha\tau$ is satisfied with equality. Then the rate of growth of $d(o, s)$ must be at least α just before t', and less than α just after t' (see Figure 1). Thus, s must serve some request at time t'. If x_0 is the location of this request, $t' = (1/\alpha)d(x_0, o)$ and after time t', s continues following the remaining part of its path, call it P', at full speed. Hence, $\text{GRH}(\sigma) = t' + \ell(P')$. Since $\text{OPT}(\sigma) \geq \max_i \ell(P_i) \geq d(o, x_0) + \ell(P')$ this yields $\text{GRH}(\sigma) \leq (1/\alpha)\text{OPT}(\sigma)$.

Thus, the competitive ratio is at most $\max\{2 + 2^{1-g}\alpha, 1/\alpha\}$ and choosing α in order to minimize it gives $\alpha = \sqrt{2^{g-1}(2^{g-1}+1)} - 2^{g-1}$ and the desired competitive ratio. $\square$

Corollary 3.3. *There is a deterministic* $(1+\sqrt{2})$*-competitive online algorithm for the* k*-TSP.*

3.2 The k-Traveling Repairman Problem

Theorem 3.4. *There is a deterministic online algorithm for the* (k, k^*)*-TRP with competitive ratio* $2 \cdot 3^{1/\lfloor k/k^* \rfloor}$.

We call the algorithm achieving the bound Group Interval (Algorithm 2), as it can be seen as a multi-server generalization of algorithm Interval [18]. The algorithm divides the servers into groups of size k^*, each group operating in phases of geometrically increasing length. Intuitively, the geometrical increase allows the algorithm to schedule, during a phase, at least as many requests as the optimal solution served in the *previous* phase.

The algorithm is well defined since $\beta \geq 1$ for any $\alpha \in (1, 3^{1/g}]$, and moreover the time between two departures from o of the same group is enough for the

Algorithm 2 Group Interval (GI)

Divide the servers into $g = \lfloor k/k^* \rfloor$ disjoint sets (groups) of k^* servers each. Any remaining server is not used by the algorithm.
Let L be the earliest time that any request can be completed (wlog $L > 0$). For $i = 0, 1, \ldots$, define $B_i = \alpha^i L$ where $\alpha \in (1, 3^{1/g}]$ will be fixed in the analysis.
At time B_i, compute a set of paths $S_i = \{P_1^i, \ldots, P_{k^*}^i\}$ for the set of yet unassigned requests released up to time B_i with the following properties:
(i) every P_j^i starts at the origin o;
(ii) $\max_j \ell(P_j^i) \leq B_i$;
(iii) S_i maximizes the number of requests served among all schedules satisfying the first two conditions.
The requests in S_i are now considered assigned.
Let $\beta = 2/(\alpha^g - 1)$. Starting at time βB_i, the j-th server in the (i mod g)-th group follows path P_j^i, then returns to o at full speed.

group to complete its first schedule and return to the origin: $\beta B_{i+g} - \beta B_i = \beta(\alpha^g - 1)B_i = 2B_i$.

To give the proof of Theorem 3.4, we start with two preliminary lemmas.

Lemma 3.5 ([18])**.** *Let $a_i, b_i \in \mathbb{R}$ for $i = 1, \ldots, p$, for which*

(i) $\sum_{i=1}^{p} a_i = \sum_{i=1}^{p} b_i$, *and*

(ii) $\sum_{i=1}^{p'} a_i \geq \sum_{i=1}^{p'} b_i$ *for all* $1 \leq p' \leq p$.

Then the $\sum_{i=1}^{p} \tau_i a_i \leq \sum_{i=1}^{p} \tau_i b_i$ *for any nondecreasing sequence of real numbers* $0 \leq \tau_1 \leq \tau_2 \leq \ldots \leq \tau_p$.

Lemma 3.6. *Let R_i be the set of requests served by the set of paths S_i computed by Group Interval at time B_i, $i = 1, 2, \ldots$ and let R_i^* be the set of requests in the optimal offline solution that are completed in the time interval $(B_{i-1}, B_i]$. Then*

$$\sum_{i=1}^{q} |R_i| \geq \sum_{i=1}^{q} |R_i^*| \quad \text{for all } q = 1, 2, \ldots.$$

Proof. The proof is similar to that of Lemma 4 in [18]. We first argue that for any $q \geq 1$ we can obtain, from the optimal offline solution S^*, a set S of k^* paths, each starting at the origin and of length at most B_q, which serve all requests in $\cup_{i=1}^{q} R_i^*$.

Consider the optimal offline solution S^*. Start at the origin and consider the path followed by each server in S^* for the first B_q time units. This gives a set S of paths of length at most B_q which together serve all the requests in $\cup_{i=1}^{q} R_i^*$. Since the servers move at unit speed, the endpoints of the paths in S are within distance B_q from the origin.

We now consider phase q and show that by the end of phase q, at least $\sum_{i=1}^{q} |R_i^*|$ requests have been scheduled by Group Interval. Notice that the set S obtained as above satisfies the conditions (i) and (ii) required by Group Interval. Moreover, the set S serves all requests in $\cup_{i=1}^{q} R_i^*$. Since Group Interval picks a set of paths maximizing the number of requests served, among all sets satisfying condition (i) and (ii), in phase q Group Interval can serve at least all requests from

$$\left(\bigcup_{i=1}^{q} R_i^*\right) \setminus \left(\bigcup_{i=1}^{q-1} R_i\right).$$

Consequently, the number of requests served in schedules $S_1, \ldots, S_q$ of Group Interval is at least $|\cup_{i=1}^{q} R_i^*| = \sum_{i=1}^{q} |R_i^*|$ as claimed. □

Proof of Theorem 3.4. Let $\sigma = \sigma_1 \ldots \sigma_m$ be any sequence of requests. By construction of Group Interval, each request in R_i is served no later than time $(1+\beta)B_i$. Now, let p be such that the optimal offline schedule completes in the interval $(B_{p-1}, B_p]$. Summing over all phases $1, \ldots, p$ yields

$$\text{GI}(\sigma) \le (1+\beta) \sum_{i=1}^{p} B_i |R_i| = (1+\beta) \cdot \alpha \sum_{i=1}^{p} B_{i-1} |R_i|. \tag{1}$$

From Lemma 3.6 we know that $\sum_{i=1}^{q} |R_i| \ge \sum_{i=1}^{q} |R_i^*|$ for $q = 1, 2, \ldots$ We also know that $\sum_{i=1}^{p} |R_i| = \sum_{i=1}^{p} |R_i^*|$. Applying Lemma 3.5 to the sequences $a_i := |R_i|$, $b_i := |R_i^*|$, $\tau_i := B_{i-1}$, $i = 1, \ldots, p$ yields in (1)

$$\text{GI}(\sigma) \le (1+\beta) \cdot \alpha \sum_{i=1}^{p} B_{i-1} |R_i| \le (1+\beta) \cdot \alpha \sum_{i=1}^{p} B_{i-1} |R_i^*|. \tag{2}$$

Let C_j^* be the optimal offline completion time of request σ_j. For each σ_j denote by $(B_{\phi_j}, B_{\phi_j+1}]$ the interval that contains C_j^*. This inserted in (2) yields

$$\text{GI}(\sigma) \le (1+\beta) \cdot \alpha \sum_{j=1}^{m} B_{\phi_j} \le (1+\beta) \cdot \alpha \sum_{j=1}^{m} C_j^* = (1+\beta) \cdot \alpha \cdot \text{OPT}(\sigma).$$

Choosing $\alpha = 3^{1/g}$ (so that $\beta = 1$) proves the theorem. □

Corollary 3.7. *There is a deterministic $(1+\sqrt{2})^2$-competitive online algorithm for the k-TRP.*

Proof. The upper bound proved in Theorem 3.4 is

$$(1+\beta)\alpha = \left(1 + \frac{2}{\alpha^g - 1}\right)\alpha$$

where $\alpha \in (1, 3^{1/g}]$. When $g = 1$, this expression attains its minimum if $\alpha = 1 + \sqrt{2}$ (for $g \ge 2$, basic calculus shows that the best choice is $\alpha = 3^{1/g}$ as in Theorem 3.4). □

3.3 Lower bounds

Theorem 3.8. *The competitive ratio of any randomized online algorithm for the (k, k^*)-TSP or the (k, k^*)-TRP is at least* 2.

Proof. Consider the metric space induced by a star graph with m unit-length rays, the origin being the center of the star. No request is given until time 1. At time 1, the adversary gives a request on an edge chosen uniformly at random, at distance 1 from the origin. The expected makespan for the adversary is 1. For the online algorithm, we say that a server *guards* a ray if at time 1 the server is located on the ray, but not at the center of the star. Then the makespan is at least 2 if no server guards the ray where the request is released, and at least 1 otherwise. But k servers can guard at most k rays, so

$$\mathbb{E}[\textsc{ol}(\sigma)] \geq 2 \cdot \left(1 - \frac{k}{m}\right) + 1 \cdot \frac{k}{m} \geq 2 - \frac{k}{m}$$

and the result follows by Yao's principle, since m can be arbitrarily large. $\square$

Notice that this lower bound is independent of the values k and k^*. Consequently, the upper bounds of Sections 3.1 and 3.2 are essentially best possible when $k \gg k^*$, as in that case they both approach 2.

4 Algorithms for the real line

4.1 An asymptotically optimal algorithm

Theorem 4.1. *There is a deterministic online algorithm with competitive ratio $1 + O((\log k)/k)$ for both the k-TSP and the k-TRP on the real line.*

As a preliminary, we prove a similar result on the *halfline*. The idea behind the algorithm is to keep the servers, at any time t, well distributed within the interval $[0, t]$, so that any new location can be efficiently reached. In particular it is sufficient that a request (r, x) is served at a time that is small compared to either r or x, as both r and x are lower bounds on the cost of the optimal solution.

Lemma 4.2. *Let g_k be the unique real root greater than 1 of the equation $z^k(z - 1) = 3z - 1$. Then* GPS *(Algorithm 3) is g_k-competitive for k-TSP and k-TRP on the halfline.*

Proof. First, notice that the modified release date of a request (as defined in the description of the algorithm) is a lower bound on its completion time. Thus it is enough to prove that, for every request (r, x), the time at which it is served is at most $g_k r'$.

For $1 < j < k$, we say that a request (r, x) is in *zone j* if $\alpha_j \leq x/r' < \alpha_{j+1}$. We also say that a request is in zone 1 if $x/r' < \alpha_2$, and that it is in zone k if $x/r' \geq \alpha_k$. By construction, every request is in some zone and a request in zone j will be eventually served by server s_j (Figure 2).

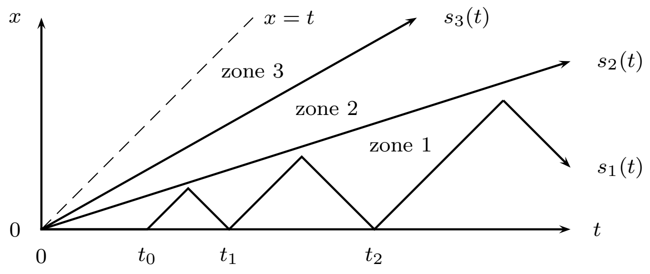

Figure 2: Sample run of algorithm GPS on the halfline ($k = 3$).

Algorithm 3 Geometric Progression Speeds (GPS)

As a preprocessing step, the algorithm delays every request (r, x) for which $x \geq r$ to time x; that is, the release date of each request (r, x) is reset at $r' := \max\{r, x\}$ (the *modified release date*).
Then, let g_k be the unique root greater than 1 of the equation $z^k(z-1) = 3z-1$ and define $\alpha_j = g_k^{j-k-1}$ for $j \in \{2, 3, \ldots, k\}$. For every $j > 1$, server s_j departs at time 0 from o at speed α_j and never turns back (Figure 2). The first server s_1 waits in o until the first request (r_0, x_0) is released with $0 < x_0 < s_2(r'_0)$. For $i \geq 0$, define $t_i = g_k^i r'_0$. During any interval $[t_{i-1}, t_i]$, s_1 moves at full speed first from o to $(g_k - 1)t_{i-1}/2$ and then back to o.

For a request (r, x) in a zone j with $1 < j < k$, since the request is served by server s_j at time x/α_j and since $x \leq \alpha_{j+1} r$, the ratio between completion time and modified release date is at most $\alpha_{j+1}/\alpha_j = g_k$. Similarly, for a request in zone k, since $x \leq r'$, the ratio between completion time and modified release date is at most $1/\alpha_k = g_k$.

It remains to give a bound for requests in zone 1. Take any such request, that is, a request (r, x) such that $x < \alpha_2 r'$ and suppose it is served at time $\tau \in [t_{i-1}, t_i]$ for some i. If $r' \geq t_{i-1}$, then, since $\tau \leq t_i$, the ratio between τ and r' is at most g_k by definition of t_i, $i \geq 0$.

If $r' < t_{i-1}$, then, since $\tau > t_{i-1}$, only two possible cases remain. First, the situation that $x > \frac{g_k - 1}{2} t_{i-2}$. Since $\tau = t_{i-1} + x$ and $r' \geq x/\alpha_2$, we have

$$\frac{\tau}{r'} \leq \frac{x + t_{i-1}}{x/\alpha_2} \leq \alpha_2 \left(1 + \frac{2 g_k t_{i-2}}{(g_k - 1) t_{i-2}}\right) = \alpha_2 \frac{3g_k - 1}{g_k - 1} = \alpha_2 g_k^k = g_k.$$

In the second situation, $x \leq \frac{g_k - 1}{2} t_{i-2}$. Then r' must be such that s_1 was already on its way back to 0 during $[t_{i-2}, t_{i-1}]$, in particular $r' \geq g_k t_{i-2} - x$. Thus,

$$\tau/r' \leq \frac{g_k t_{i-2} + x}{g_k t_{i-2} - x} \leq \frac{3g_k - 1}{g_k + 1} \leq g_k.$$

□

The algorithm for the real line simply splits the k servers evenly between the two halflines, and uses GPS on each halfline.

Algorithm 4 Split Geometric Progression Speeds (SGPS)

```
Arbitrarily assign $\lceil k/2 \rceil$ servers to $\mathbb{R}_+$ and $\lfloor k/2 \rfloor$ servers to $\mathbb{R}_-$. On each of the
two halflines, apply Algorithm 3 independently (that is, ignoring the requests
and the servers on the other halfline).
```

Lemma 4.3. *For any $k \geq 2$,* SGPS *(Algorithm 4) is $g_{\lfloor k/2 \rfloor}$-competitive for the k-TSP and the k-TRP on the line.*

Proof. The only lower bounds on the offline cost that we used in the proof of Lemma 4.2 were the distance of every request from o and the release date of every request. They are valid independent of the number of offline servers. In particular, they hold if the number of offline servers is twice the number of online servers. Thus, we can analyze the competitiveness of the online servers on each of the two halflines separately and take the worst of the two competitive ratios. □

Lemma 4.4. *For any $k \geq 1$, $g_k \leq 1 + \frac{2\log k+3}{k}$.*

Proof. We defined g_k as the unique root greater than 1 of $z^k = 1 + \frac{2z}{z-1}$. Since $\lim_{z\to\infty} z^k > \lim_{z\to\infty} 1 + \frac{2z}{z-1}$, it suffices to prove that $z_0 := 1 + \frac{2\log k+3}{k}$ satisfies $z_0^k \geq 1 + \frac{2z_0}{z_0-1}$. The binomial theorem and the standard fact that $\binom{k}{j} \geq \frac{k^j}{j^j}$ yield

$$\begin{aligned} z_0^k - 1 &= \sum_{j=1}^{k} \binom{k}{j} \frac{(2\log k+3)^j}{k^j} \geq \sum_{j=1}^{k} \frac{(2\log k+3)^j}{j^j} \geq \sum_{j=1}^{\lfloor \log k \rfloor + 1} \Big(\frac{2\log k+3}{j}\Big)^j \\ &\geq \sum_{j=1}^{\lfloor \log k \rfloor + 1} 2^j \geq 2^{\log k+1} - 2 = 2k - 2. \end{aligned}$$

Now it can be verified that for all $k > 2$, $2k - 2 > \frac{2k}{2\log k+3} + 2 = \frac{2z_0}{z_0-1}$. Finally, the bound also holds for $k \in \{1, 2\}$ as seen by explicitly finding g_1 and g_2. □

Theorem 4.1 now follows from Lemma 4.3 and Lemma 4.4.

4.2 Lower bounds

Theorem 4.5. *The competitive ratio of any randomized online algorithm for the k-TSP or the k-TRP on the line is at least $1 + 1/2k = 1 + \Omega(1/k)$.*

Proof. The adversary gives a single request at time 1, in a point drawn uniformly at random from the interval $[-1, 1]$. The expected optimal cost is obviously 1. Thus, by Yao's principle it suffices to show that $\mathbb{E}[\text{OL}(\sigma)] \geq 1 + 1/2k$.

In order to bound $\mathbb{E}[\text{OL}(\sigma)]$, let $f(x) = \min_{j\in\{1,\dots,k\}} d(x, s_j(1))$ (Figure 3). Notice that $1 + f(x)$ is a lower bound on the cost paid by the online algorithm, assuming that the request was given at x. In terms of expected values,

$$\mathbb{E}[\text{OL}(\sigma)] \geq \mathbb{E}[1 + f(x)] = 1 + \frac{1}{2}\int_{-1}^{1} f(x)dx.$$

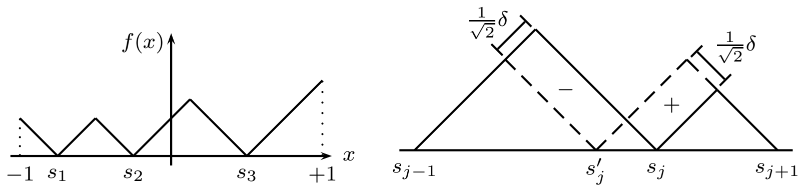

Figure 3: The function $f(x)$ (left) and an illustration of the proof (right).

The claim follows if we can prove a lower bound of $1/k$ for $\int_{-1}^{1} f(x)dx$. Finding such a lower bound is equivalent to finding the configuration of k points $s_1, \ldots, s_k \in [-1, 1]$ that minimizes the value of the area below f.

Assume without loss of generality that $s_1 \leq s_2 \leq \ldots \leq s_k$. We first show that in the minimal configuration the points $s_1, \ldots, s_k$ are evenly spread within the interval $[s_1, s_k]$. Consider any s_j, $1 < j < k$, and assume wlog $s_j \geq (s_{j-1} + s_{j+1})/2$. If we moved the point s_j to $s'_j := (s_{j-1} + s_{j+1})/2$ and let $\delta := s_j - s'_j$, the value of the area below f would vary by $\frac{1}{2}\delta(s_{j+1} + s_{j-1} - 2s_j) \leq 0$ (Figure 3, right). Therefore, we can assume that each server s_j $(1 < j < k)$ is located halfway between s_{j-1} and s_{j+1}, or equivalently that the servers are uniformly distributed across $[s_1, s_k]$.

Similarly one can prove that, in the minimal configuration, $d(-1, s_1) = d(s_k, 1)$. Let $D := d(s_k, 1)$. Then $\int_{-1}^{1} f(x)dx = D^2 + (1 - D)^2/(k - 1)$, which attains its minimum, $1/k$, when $D = 1/k$, as can be seen for example by basic calculus. The claim follows. □

5 Lower bounds on the plane

Comparing the results in Section 3 with those in Section 4, we see that while in general spaces the competitive ratio of both the k-TSP and the k-TRP always remains lower bounded by 2, on the real line we can achieve $1 + o(1)$ asymptotically. A natural question is whether on a low-dimensional space like the Euclidean plane we can also achieve $1 + o(1)$ competitiveness. In this section we answer this question negatively.

Theorem 5.1. *The competitive ratio of any randomized online algorithm for the k-TSP on the plane is at least* $4/3$.

Proof. As a crucial ingredient of the proof we introduce a new kind of request, which is located in a single point x of the space but has an arbitrarily long processing time p (this processing time can be divided among the servers processing the request). We show how this can be emulated in the Euclidean plane with arbitrarily good precision by giving a high enough number of requests packed inside an arbitrarily small square around x.

Fix some arbitrary $\epsilon > 0$. Consider a square with sidelength $s = \sqrt{\epsilon p}$ centered around x. The square can be partitioned in s^2/ϵ^2 smaller squares of sidelength ϵ. In the center of each of these smaller squares we give a request. Notice that the distance between any pair of such requests is at least ϵ. Thus, the sum of the times required for any k servers to serve all requests is at least $(\frac{s^2}{\epsilon^2} - k)\epsilon$, no matter where the servers start (the $-k\epsilon$ term reflects the possible saving each server could have by starting arbitrarily close to the *first* request he serves).

For ϵ tending to zero, the requests converge to the point x and the total processing time needed converges to p. If the starting points of the servers are most favourable, an algorithm could finish serving all requests in time p/k.

We show how to use such a "long" request to achieve our lower bound. At time 1, the adversary gives a long request of processing time $p = 2k$ in a point drawn uniformly at random from $\{(1,0),(-1,0)\}$. The expected optimal cost is $1 + p/k = 3$. By Yao's principle, it remains to prove that $\mathbb{E}[\mathrm{OL}(\sigma)] \geq 4$.

Since there is a single long request, we can assume without loss of generality that all the online servers will move to the request and contribute to serving it. Since $p = 2k$, the server that will contribute most to the service will have to spend time at least $2k/k = 2$ in x, and this is enough for any other server to arrive and give a contribution (since at time 1 no server can be farther than 2 from x).

Suppose without loss of generality that the servers are numbered in order of nondecreasing distance to x and let $d_i = d(x, s_i(1))$. We have $\mathrm{OL}(\sigma) \geq 1 + t_0$, with t_0 the time needed for the servers to completely serve the request, i.e., the time needed to reduce its remaining processing time to zero. During this time, each server first moves towards the request, and then contributes to its service. The total time needed by all servers is thus the processing time plus all the distances:

$$k \cdot t_0 = p + \sum_{i=1}^{k} d_i.$$

Now notice that $\mathbb{E}[d_i]$, the expected distance of server s_i from x, is at least 1 for all i. Hence,

$$\begin{aligned}\mathbb{E}[\mathrm{OL}(\sigma)] \geq 1 + \mathbb{E}[t_0] &\geq 1 + p/k + \mathbb{E}\left[\frac{1}{k}\sum_i d_i\right] \\ &= 3 + \frac{1}{k}\sum_i \mathbb{E}[d_i] \\ &\geq 3 + \frac{1}{k}k \\ &= 4.\end{aligned}$$

□

A similar technique gives an analogous lower bound for the k-TRP on the plane.

Theorem 5.2. *The competitive ratio of any randomized online algorithm for the k-TRP on the plane is at least* $5/4$.

Proof. We use the same input distribution used in the proof of Theorem 5.1. For the costs, it is equivalent but easier to consider average completion time instead of the sum of completion times. Then, the expected optimal cost can be easily seen to be 2 since the k servers can uniformly process the request of length p (recall that $p = 2k$) during time interval $[1, 3]$.

To lower bound the online cost, consider $w \in [0, p]$ and let $C(w)$ be the first time at which a total work of w has been completed by the online algorithm. Then

$$\textsc{ol}(\sigma) = \frac{1}{p} \int_0^p C(w)dw.$$

Now if we define t_0 as in the proof of Theorem 5.1 we have $C(p) = 1 + t_0$, while $C(\epsilon) > 1$ for any $\epsilon > 0$. Moreover, the function C is concave, since the speed at which the long request is processed can only increase as more servers arrive, so the value of the integral above is at least $p(1 + t_0/2)$. Thus

$$\mathbb{E}[\textsc{ol}(\sigma)] \geq \mathbb{E}\left[\frac{1}{p} \cdot p\left(1 + \frac{t_0}{2}\right)\right] = \mathbb{E}[1 + t_0/2] \geq 5/2$$

since we already showed in the proof of Theorem 5.1 that $\mathbb{E}[t_0] \geq 3$. The claim follows by Yao's principle. $\square$

We remark that for small values of k one can prove better lower bounds than those in Theorems 5.1 and 5.2. For example, it is possible to prove a $3/2$ lower bound on the competitive ratio of the 2-TSP on the plane.

Theorem 5.3 ([17])**.** *The competitive ratio of any deterministic online algorithm for the* 2*-TSP on the plane is at least* 3/2*.*

Proof. If at time 1 both online servers are on the same side of the origin, say, on $\mathbb{R}_-$, we issue a single request of length 0 at $+1$. Then, the optimum makespan is 1, while the online algorithm needs time at least 2. So, let us assume that at time 1, one online server is at $-x \leq 0$ and the other is at $y \geq 0$, where $y \leq x$. We issue a request of length $x + y$ at $+1$ at time 1. Clearly $\textsc{opt}(\sigma) = 1 + (x + y)/2$. The first online server can be at $+1$ at time $2 - y$, while the second reaches this point not earlier than time $2 + x$, which is $x + y$ units of time later. So, the second server can only be at $+1$ when the whole service is already done and $\textsc{ol}(\sigma) \geq 2 + x$. We have

$$\frac{\textsc{ol}(\sigma)}{\textsc{opt}(\sigma)} \geq \frac{2 + x}{1 + (x + y)/2} = 2 - \frac{2y}{2 + x + y} \geq 2 - \frac{y}{y + 1} \geq \frac{3}{2}.$$

$\square$

6 Conclusions

After analyzing the differences between multiple and single server variants, we can conclude that sometimes having multiple servers is more beneficial to the online algorithm than to the offline adversary. In some cases, including the traveling repairman problem on the line, the online algorithms can approach the offline cost when there are enough servers. In more general spaces, these extremely favorable situation cannot occur. Still in some intermediate cases, like the Euclidean plane, it is conceivable that the competitive ratios become lower than those of the corresponding single server problems. We leave the analysis of the competitive ratio in these situations as an open problem.

Acknowledgements

We would like to thank Sven Krumke for commenting an early version of the paper and for suggesting Theorem 5.3. We are also grateful to Maarten Lipmann for useful discussions.

References

[1] N. Ascheuer, S. O. Krumke, and J. Rambau. Online dial-a-ride problems: Minimizing the completion time. In H. Reichel and S. Tison, editors, *Proc. 17th Symp. on Theoretical Aspects of Computer Science*, volume 1770 of *Lecture Notes in Computer Science*, pages 639–650. Springer-Verlag, 2000.

[2] G. Ausiello, V. Bonifaci, and L. Laura. The on-line asymmetric traveling salesman problem. *Journal of Discrete Algorithms*, 2007. In press.

[3] G. Ausiello, E. Feuerstein, S. Leonardi, L. Stougie, and M. Talamo. Algorithms for the on-line travelling salesman. *Algorithmica*, 29(4):560–581, 2001.

[4] M. Blom, S. O. Krumke, W. E. de Paepe, and L. Stougie. The online TSP against fair adversaries. *INFORMS Journal on Computing*, 13(2):138–148, 2001.

[5] V. Bonifaci. *Models and Algorithms for Online Server Routing*. PhD thesis, Technical University Eindhoven, The Netherlands, 2007. Available at `http://www.dis.uniroma1.it/~bonifaci/papers/phdthesis-tue.pdf`.

[6] A. Borodin and R. El-Yaniv. *Online Computation and Competitive Analysis*. Cambridge University Press, 1998.

[7] A. Borodin and R. El-Yaniv. On randomization in online computation. *Information and Computation*, 150:244–267, 1999.

[8] K. Chaudhuri, B. Godfrey, S. Rao, and K. Talwar. Paths, trees, and minimum latency tours. In *Proc. 44th Symp. on Foundations of Computer Science*, pages 36–45, 2003.

[9] B. Chen and A. P. A. Vestjens. Scheduling on identical machines: How good is LPT in an on-line setting? *Operations Research Letters*, 21(4):165–169, 1997.

[10] J. R. Correa and M. R. Wagner. LP-based online scheduling: From single to parallel machines. In *Integer programming and combinatorial optimization*, volume 3509 of *Lecture Notes in Computer Science*, pages 196–209. Springer-Verlag, 2005.

[11] J. Fakcharoenphol, C. Harrelson, and S. Rao. The k-traveling repairman problem. In *Proc. 14th Symp. on Discrete Algorithms*, pages 655–664, 2003.

[12] E. Feuerstein and L. Stougie. On-line single-server dial-a-ride problems. *Theoretical Computer Science*, 268(1):91–105, 2001.

[13] G. N. Frederickson, M. S. Hecht, and C. E. Kim. Approximation algorithms for some routing problems. *SIAM Journal on Computing*, 7(2):178–193, 1978.

[14] D. Hauptmeier, S. O. Krumke, and J. Rambau. The online dial-a-ride problem under reasonable load. In G. Bongiovanni, G. Gambosi, and R. Petreschi, editors, *Proc. 4th Italian Conference on Algorithms and Complexity*, volume 1767 of *Lecture Notes in Computer Science*, pages 125–136. Springer-Verlag, 2000.

[15] R. Jothi and B. Raghavachari. Minimum latency tours and the k-traveling repairmen problem. In M. Farach-Colton, editor, *Proc. 6th Symp. Latin American Theoretical Informatics*, volume 2976 of *Lecture Notes in Computer Science*, pages 423–433. Springer-Verlag, 2004.

[16] B. Kalyanasundaram and K. Pruhs. Speed is as powerful as clairvoyance. *Journal of the ACM*, 47(4):214–221, 2000.

[17] S. O. Krumke, 2006. Personal communication.

[18] S. O. Krumke, W. E. de Paepe, D. Poensgen, and L. Stougie. News from the online traveling repairman. *Theoretical Computer Science*, 295(1-3):279–294, 2003.

[19] S. O. Krumke, L. Laura, M. Lipmann, A. Marchetti-Spaccamela, W. E. de Paepe, D. Poensgen, and L. Stougie. Non-abusiveness helps: an $O(1)$-competitive algorithm for minimizing the maximum flow time in the online traveling salesman problem. In K. Jansen, S. Leonardi, and V. V. Vazirani, editors, *Proc. 5th Int. Workshop on Approximation Algorithms for Combinatorial Optimization*, volume 2462 of *Lecture Notes in Computer Science*, pages 200–214. Springer-Verlag, 2002.

[20] E. L. Lawler, J. K. Lenstra, A. Rinnooy Kan, and D. B. Shmoys, editors. *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization.* Wiley, Chichester, England, 1985.

[21] M. Lipmann. *On-Line Routing.* PhD thesis, Technical University Eindhoven, The Netherlands, 2003.

[22] M. Lipmann, X. Lu, W. E. de Paepe, R. A. Sitters, and L. Stougie. On-line dial-a-ride problems under a restricted information model. *Algorithmica*, 40(4):319–329, 2004.

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

[24] J. Sgall. On-line scheduling. In A. Fiat and G. J. Woeginger, editors, *Online Algorithms: the State of the Art*, pages 196–231. Springer-Verlag, 1998.

[25] L. Stougie and A. P. A. Vestjens. Randomized on-line scheduling: How low can't you go? *Operations Research Letters*, 30(2):89–96, 2002.