An Improved Online Algorithm for the Traveling Repairperson Problem on a Line

Marcin Bienkowski
Institute of Computer Science, University of Wrocław, Poland
marcin.bienkowski@cs.uni.wroc.pl

Hsiang-Hsuan Liu
Institute of Computer Science, University of Wrocław, Poland
alison.hhliu@cs.uni.wroc.pl

Abstract

In the online variant of the traveling repairperson problem (TRP), requests arrive in time at points of a metric space $\mathcal{X}$ and must be eventually visited by a server. The server starts at a designated point of $\mathcal{X}$ and travels at most at unit speed. Each request has a given weight and once the server visits its position, the request is considered serviced; we call such time *completion time* of the request. The goal is to minimize the weighted sum of completion times of all requests.

In this paper, we give a 5.429-competitive deterministic algorithm for line metrics improving over 5.829-competitive solution by Krumke et al. (TCS 2003). Our result is obtained by modifying the schedule by serving requests that are close to the origin first. To compute the competitive ratio of our approach, we use a charging scheme, and later evaluate its properties using a factor-revealing linear program which upper-bounds the competitive ratio.

2012 ACM Subject Classification Theory of computation → Online algorithms; Theory of computation → Scheduling algorithms

Keywords and phrases traveling repairperson problem, competitive analysis, minimizing completion time, factor-revealing LP

Digital Object Identifier 10.4230/LIPIcs.MFCS.2019.6

Funding Supported by Polish National Science Centre grant 2016/22/E/ST6/00499.

1 Introduction

The traveling repairperson problem (TRP) is a variant of the traveling salesperson problem (TSP), where the goal is to minimize the *total latency* instead of a more standard objective of minimizing the total length of a route. In the TRP, there are m requested points of a given metric space $\mathcal{X}$ and they must be eventually visited by a server. Request r_j is a triple (p_j, a_j, w_j), where $p_j \in \mathcal{X}$ denotes request position, $a_j \geq 0$ its release time and w_j its weight.

The server starts at a designated point of $\mathcal{X}$ called *origin* and travels at most at unit speed. That is, for any two times $t < t'$ the distance between positions of the server at times t and t' is at most $t' - t$. Each request r_j must be eventually serviced by moving the server to point p_j. The request cannot be serviced before its release time a_j; we call the time when it is eventually serviced its *completion time* and we denote it C_j. The goal is to find a route for the server (a *schedule*) that minimizes the cost, defined as the weighted sum of completion times, i.e., $\sum_{j=1}^{m} w_j \cdot C_j$.

The TRP has a natural online variant. There, an online algorithm Alg, at time t, knows only requests that arrived before or at time t. The number of requests m is also not known by an algorithm a priori. In the online setting, the goal is to minimize the competitive ratio [11], defined as the maximum over all inputs of the Alg-to-Opt cost ratio, where Opt denotes the optimal offline algorithm.

1.1 Previous work

The online variant has been first investigated by Feuerstein and Stougie [13]. They considered the case where $\mathcal{X}$ is a real line and, adapting an algorithm for the so-called cow-path problem [7], presented a 9-competitive solution. They also gave a lower bound (that holds also already for a line) of $1 + \sqrt{2}$. The result has been subsequently improved by Krumke et al. [18], who gave a deterministic algorithm Interval attaining competitive ratio of $(1 + \sqrt{2})^2 < 5.829$ and a randomized 3.874-competitive solution. Their algorithm works for an arbitrary metric space.

A natural extension to the TRP is a so-called *dial-a-ride problem*, where each request is an object with a source and a destination and the goal is to transport the object [9, 13, 18]. There, the server may have a fixed capacity allowing it to store at most k objects, or this capacity may be infinite. The 5.829-competitive deterministic algorithm by Krumke et al. [18] extends also to this variant and no better algorithms are known even for specific metric spaces.

Some papers considered an extension of the TRP to $k \geq 1$ servers. Bonifaci and Stougie showed how to adapt the algorithm Interval by Krumke et al. [18] to this setting without an increase of the competitive ratio [10]. However, the best known lower bound for multiple servers is 2 (i.e., smaller than the lower bound for the one-server case) [10]. Furthermore, for the particular case of line metrics, the ratio converges to 1 with growing k, and is between $1 + 1/(2k - 1)$ [14] and $1 + O((\log k)/k)$ [10].

Another strand of papers considered different objectives, such as minimizing the total makespan [3, 4, 5, 6, 8, 9] or maximum flow time [16, 17, 19], with a special focus on line metrics. Finally, the offline variant (also known as *minimum latency problem*) has been extensively studied both from the computational hardness and approximation algorithms perspectives, see, e.g., [1, 2, 12, 15, 20, 21].

1.2 Our contribution

In this paper, we focus on the TRP on line metrics and give a 5.429-competitive algorithm Reroute. This improves the long standing record of 5.829 achieved by the algorithm Interval by Krumke et al. [18].

Similarly to the algorithm Interval, Reroute partitions an input into phases of geometrically increasing lengths, and in each phase greedily tries to service the set of pending requests of maximum weight. However, our algorithm Reroute tries to modify the route, so to ensure that (i) either it services requests only in the initial part of the phase, (ii) or in the later part of the phase, it services only requests that are far away from the origin. As such requests cannot be serviced early by any algorithm (also by Opt), this allows us to charge the cost of Reroute against the cost of Opt in a more efficient way.

For the analysis, we construct a charging scheme that maps the total weight serviced by Reroute in particular time intervals to the total weight serviced in appropriate intervals by Opt. This yields a set of linear inequalities that need to hold for any input instance. On this basis, we create a maximization LP (linear program), whose objective value is an upper bound on the competitive ratio. Finally, to bound the value of such factor-revealing LP, we explicitly construct a solution to its dual.

2 Algorithms Interval and ReturnFirst

As our algorithm is built on the phase-based approach of the algorithm Interval proposed by Krumke et al. [18], we start with a description of the latter.

Let $f = \min_j\{\max\{p_j, a_j\}\}$ be the earliest time at which Opt may service a request. (Note that an online algorithm can learn f before or at time f.) Without loss of generality, we may assume that there are no requests that arrive at time 0 at the origin, and hence $f > 0$.

We partition time into phases. Phase 0 starts at time 0 and ends at time f. Phase $i \geq 1$ starts at time $f \cdot \alpha^{i-1}$ and ends at time $f \cdot \alpha^i$, where $\alpha = 1 + \sqrt{2}$. At the beginning of any phase $i \geq 1$, Interval computes and executes a schedule that

- starts at the final server position from the last phase;
- stops at distance at most $f \cdot \alpha^{i-1}$ from the origin;
- has length at most $f \cdot (\alpha^i - \alpha^{i-1})$;
- among schedules satisfying the previous three conditions maximizes the total weight of serviced requests (which are pending when the phase starts).

To simplify the notation, in the rest of the paper, we assume that $f = 1$.[1] We start with a slight modification of the algorithm Interval, called ReturnFirst (RetF). At the beginning of any phase $i \geq 1$, RetF computes and executes a schedule that

- starts at the final server position from the last phase;
- in the first part of the schedule (called *return part*), the server returns to the origin;
- the second part of the schedule (called *serving part*) starts at the origin, is of length at most α^{i-1}, and among such schedules maximizes the total weight of serviced requests (which are pending when the phase starts).

▶ **Observation 1.** *At the beginning of each phase $i \geq 1$,* RetF *has its server at a distance at most α^{i-2} from the origin. Furthermore, the total length of both parts of the schedule is at most the phase length.*

Proof. The first property is clearly satisfied at the beginning of phase 1, which is started with the server at the the origin. In the subsequent phases, this property follows inductively: as in phase $i-1$ the serving part of the schedule starts at the origin and has length at most α^{i-2}, at be beginning of phase i the server distance to the origin is at most α^{i-2}.

The second property follows as the total length of the planned schedule in phase i is at most $\alpha^{i-2} + \alpha^{i-1} = \alpha^i - \alpha^{i-1}$, which is the length of the phase. Note that this property holds as long as $\alpha \geq 1 + \sqrt{2}$. ◀

While RetF may produce schedules that are worse than those of Interval, using similar arguments to those of [18], one can show that RetF is α^2-competitive. In particular, the following bound holds both for Interval and RetF; we present its proof for completeness.

In our arguments, we use $\textsc{Alg}_i$ and $\textsc{Opt}_i$ to denote the total weight of requests serviced by an online algorithm and Opt, respectively, in a phase i. Observe that for RetF and Interval, $\textsc{Alg}_0 = 0$.

▶ **Lemma 2** ([18]). *Let L be the index of the last phase in which* Opt *services any request. Then,* RetF *services all requests within the first $L+1$ phases, and for any phase $j \in \{1, \ldots, L+1\}$, it holds that $\sum_{i=j}^{L+1} \textsc{Alg}_i \leq \sum_{i=j-1}^{L} \textsc{Opt}_i$.*

[1] All terms occurring in the proof, both related to distances and to time, have a multiplicative factor f, which cancels out when the competitive ratio is computed.

Proof. Consider the schedule S_k of OPT in phases $0, 1, \ldots, k$, where $k \in \{0, \ldots, L\}$. Schedule S_k starts at the origin, its length is equal to α^k and the total weight serviced by S_k is $w(S_k) = \sum_{i=0}^{k-1} \textsc{Opt}_i$. In phase $k+1$, when RETF chooses the route for the serving part, S_k is among feasible options. If RETF chose such route, then each request serviced by schedule S_k in OPT's solution is serviced by RETF (in phase $k+1$ or already in earlier phases). Thus, the total weight serviced by RETF in phases $1, \ldots, k+1$ would be at least $w(S_k)$. As RETF chooses the schedule for the serving part which is at least as good as S_k, it holds that

$$\sum_{i=1}^{k+1} \textsc{Alg}_i \geq w(S_k) = \sum_{i=0}^{k} \textsc{Opt}_i \,. \tag{1}$$

If we set $k = L$, then the above inequality implies that RETF services all requests already in the first $L+1$ phases, i.e.,

$$\sum_{i=1}^{L+1} \textsc{Alg}_i = \sum_{i=0}^{L} \textsc{Opt}_i \,. \tag{2}$$

The lemma for $j = 1$ follows by (2), and the lemma for $j \geq 2$ follows by subtracting (1) from (2) and setting $k = j - 2$. ◀

3 Modifying the schedule

We now take a closer look at the structure of schedules produced by RETF in particular phases. We show that the schedule produced in a given phase can be modified, so that it services the same set of requests as RETF, but either it ends substantially earlier than the phase end or from some time it services only far requests.

▶ **Lemma 3.** *Fix any $c \in [0, 1]$ and let $\ell = \alpha^{i-1} + c \cdot \alpha^{i-2}$. Fix a schedule S for phase i produced by the algorithm* RETF*. On the basis of S and c, it is possible to construct schedule $\tilde{S}$ that services the same set of requests as S and*

- *Either the length of $\tilde{S}$ is at most ℓ,*
- *or after executing the prefix of $\tilde{S}$ of length ℓ, the server distance from the origin is at least $(1/3) \cdot (\alpha - 1 + 5c) \cdot \alpha^{i-2}$, and afterwards the server travels away from the origin (with unit speed).*

Proof. In the following proof, server positions are real numbers, with zero denoting the origin. Let p denote the server position at the beginning of phase i. Note that we may assume that $p \geq 0$ without loss of generality. As in the proof of Observation 1, it can be inductively shown that in the produced schedule $\tilde{S}$, the server ends phase $i-1$ at most at distance α^{i-2} from the origin. Thus, $p \leq \alpha^{i-2}$.

Let $[b, u]$ be the interval containing all points visited by the serving part of S. As this part starts at zero, $b \leq 0 \leq u$. Let $x \in \{b, u\}$ be the interval endpoint closer to the origin and y be the further one (with ties broken arbitrarily). Observe that the shortest possible schedule that starts at zero and services all requests from interval $[b, u]$ has length $2 \cdot |x| + |y|$. As the serving part of S services this interval and its length is at most α^{i-1}, it holds that

$$2 \cdot |x| + |y| \leq \alpha^{i-1} \text{ and thus } |x| \leq \alpha^{i-1}/3 \,. \tag{3}$$

We use (3) extensively in our bounds below.

We define two possible schedules S_{xy} and S_{yx} for phase i; we show that at least one of them satisfies the requirements of the lemma.

- S_{xy} starts at p, goes to x (possibly going through zero if p and x are on the opposite sides of the origin), and then proceeds through 0 to y. Its length is $|p - x| + |x| + |y|$.

- S_{yx} starts at p, goes to y (possibly going through zero if p and y are on the opposite sides of the origin), and then proceeds through 0 to x. Its length is $|p-y|+|y|+|x|$.

We consider four cases depending on values of x, y, p and c. Note that all these values are known by RetF at the beginning of phase i.

Case 1. If $x<0$ and $p+3\cdot|x| \geq (\alpha-c)\cdot\alpha^{i-2}$, we set $\tilde{S}=S_{yx}$.
We show that the length of $\tilde{S}$ is at most ℓ. As $x<0$, it holds that $y\geq 0$. If $y\leq p$, the length of $\tilde{S}$ is $(p-y)+y+|x| = p+|x| \leq \alpha^{i-2}+\alpha^{i-1}/3 < \alpha^{i-1} \leq \ell$. Otherwise $y>p$, and then the length of $\tilde{S}$ is

$$\begin{aligned}(y-p)+y+|x| &= 2\cdot y - p + |x| \leq 2\cdot(\alpha^{i-1}-2\cdot|x|) - p + |x| \\ &= 2\cdot\alpha^{i-1} - (p+3\cdot|x|) \leq 2\cdot\alpha^{i-1} - (\alpha-c)\cdot\alpha^{i-2} = \ell\,.\end{aligned}$$

Case 2. If $x<0$ and $p+3\cdot|x| < (\alpha-c)\cdot\alpha^{i-2}$, we set $\tilde{S}=S_{xy}$.
If the length of $\tilde{S}$ is at most ℓ, then the lemma follows. Otherwise, we analyze the prefix of $\tilde{S}$ of length ℓ. It contains the server movement from p to x and then to 0: this holds because the total length of this movement is $p+2\cdot|x| < (\alpha-c)\cdot\alpha^{i-2} \leq \alpha^{i-1} \leq \ell$. Thus, after having executed the prefix of $\tilde{S}$ of length ℓ, the server is traveling away from the origin towards y and its position is equal to

$$\begin{aligned}\ell - (p+2\cdot|x|) &= \alpha^{i-1} + c\cdot\alpha^{i-2} - (2/3)\cdot(p+3\cdot|x|) - p/3 \\ &> \alpha^{i-1} + c\cdot\alpha^{i-2} - (2/3)\cdot(\alpha-c)\cdot\alpha^{i-2} - \alpha^{i-2}/3 \\ &= (1/3)\cdot(\alpha-1+5c)\cdot\alpha^{i-2}\,.\end{aligned}$$

Case 3. If $x\geq p\geq 0$, we set $\tilde{S}=S_{xy}$.
The length of $\tilde{S}$ is then $(x-p)+x+|y| = 2\cdot x+|y|-p \leq \alpha^{i-1}-p \leq \ell$.

Case 4. If $p>x\geq 0$, we set $\tilde{S}=S_{xy}$.
The reasoning here is similar to the one from Case 2. If the length of $\tilde{S}$ is at most ℓ, then the lemma follows. Otherwise, we analyze the prefix of $\tilde{S}$ of length ℓ. It contains the server movement from p to 0 through x: this holds because the length of this movement is equal to $p\leq\alpha^{i-2}<\ell$. Thus, after having executed the prefix of $\tilde{S}$ of length ℓ, the server is traveling away from the origin towards y and its distance from the origin is equal to

$$\ell - p \geq \alpha^{i-1} + c\cdot\alpha^{i-2} - \alpha^{i-2} = (\alpha+c-1)\cdot\alpha^{i-2} > (1/3)\cdot(\alpha-1+5c)\cdot\alpha^{i-2}\,.$$

The last inequality follows as $\alpha > 1+c$. ◀

4 Algorithm Reroute

Our algorithm Reroute(β) is parameterized with a constant $\beta\in[2/\alpha,1]$ and follows the phase framework of RetF. At the beginning of any phase $j\geq 1$, Reroute(β) computes the schedule S in the same way RetF would do, modifies it according to Lemma 3 using $c=\beta\cdot\alpha^2-2\cdot\alpha\in[0,1]$ obtaining schedule $\tilde{S}$, and then executes $\tilde{S}$ within phase j.

For any real $\xi\geq[\beta,1]$, we define

$$\tau_\beta(\xi) = \frac{5\cdot\beta\cdot\alpha^2-9\alpha-1}{3\alpha} + (\xi-\beta)\cdot\alpha\,. \tag{4}$$

The following lemma shows that requests that are serviced late by Reroute(β) in a given phase are far away from the origin, and hence cannot be serviced too early by Opt.

▶ **Lemma 4.** *Fix $\beta \in [2/\alpha, 1]$. For any phase i and any value $\xi \geq \beta$, in time interval $(\xi \cdot \alpha^i, \alpha^i]$,* Reroute($\beta$) *services only requests whose distance to the origin is at least $\tau_\beta(\xi) \cdot \alpha^{i-1}$.*

Proof. By the definition of Reroute(β), its schedule $\tilde{S}$ for phase i is constructed as described in Lemma 3 with $c = \beta \cdot \alpha^2 - 2\alpha \in [0, 1]$. Let $\ell = \alpha^{i-1} + c \cdot \alpha^{i-2} = \beta \cdot \alpha^i - \alpha^{i-1}$. Recall that phase i starts at time α^{i-1}. By Lemma 3, two cases are possible.

- If the length of $\tilde{S}$ is at most ℓ, then the execution of $\tilde{S}$ ends at or before time $\alpha^{i-1} + \ell = \beta \cdot \alpha^i$. Then, the lemma follows trivially as Reroute(β) does not service anything in the remaining part of phase i, i.e., in the time interval $(\beta \cdot \alpha^i, \alpha^i]$.
- Otherwise, the length of $\tilde{S}$ is larger than ℓ. Then, at time $\alpha^{i-1} + \ell = \beta \cdot \alpha^i$, the server still executes $\tilde{S}$, is at distance at least $(1/3) \cdot (\alpha - 1 + 5c) \cdot \alpha^{i-2}$ from the origin and it travels away from the origin with unit speed. Within time interval $[\beta \cdot \alpha^i, \xi \cdot \alpha^i]$, the server either finishes executing $\tilde{S}$ or it increases its distance to the origin by $\xi \cdot \alpha^i - \beta \cdot \alpha^i$. In the former case, the lemma follows trivially, and in the latter case, the server distance to the origin at time $\xi \cdot \alpha^i$ is at least

$$\begin{aligned} \frac{\alpha - 1 + 5c}{3} \cdot \alpha^{i-2} + (\xi - \beta) \cdot \alpha^i &= \frac{5 \cdot \beta \cdot \alpha^2 - 9\alpha - 1}{3} \cdot \alpha^{i-2} + (\xi - \beta) \cdot \alpha^i \\ &= \tau_\beta(\xi) \cdot \alpha^{i-1} . \end{aligned}$$

  From time $\xi \cdot \alpha^i$, the server continues to travel away from the origin, and thus the lemma follows. ◀

4.1 Relating Reroute to Opt

We analyze the performance of algorithm Reroute(β) for a fixed parameter β, such that $2/\alpha \leq \beta \leq 1$. Moreover, we choose β, so that $\tau_\beta(\beta) \geq 1/\alpha$. In our analysis we use a parameter ξ, such that $\beta \leq \xi \leq 1$ and $\tau_\beta(\beta) \leq \tau_\beta(\xi) \leq 1$. Concrete values of β and ξ will be fixed later.

Let L be the index of the last phase in which Opt services a request. On the basis of β and ξ, we partition both Alg_i (the total weight of requests serviced in phase $i \in \{1, \ldots, L+1\}$ by Reroute) and Opt_i (the total weight of requests serviced in phase $i \in \{0, \ldots, L\}$ by Opt) into three parts:

- A_i^a: the weight serviced by Reroute(β) in time interval $(\alpha^{i-1}, \beta \cdot \alpha^i]$,
- A_i^b: the weight serviced by Reroute(β) in time interval $(\beta \cdot \alpha^i, \xi \cdot \alpha^i]$,
- A_i^c: the weight serviced by Reroute(β) in time interval $(\xi \cdot \alpha^i, \alpha^i]$,
- O_i^a: the weight serviced by Opt in time interval $[\alpha^{i-1}, \tau_\beta(\beta) \cdot \alpha^i)$,
- O_i^b: the weight serviced by Opt in time interval $[\tau_\beta(\beta) \cdot \alpha^i, \tau_\beta(\xi) \cdot \alpha^i)$,
- O_i^c: the weight serviced by Opt in time interval $[\tau_\beta(\xi) \cdot \alpha^i, \alpha^i)$.

Note that the validity of this partitioning requires that $1/\alpha \leq \beta \leq \xi \leq 1$ and $1/\alpha \leq \tau_\beta(\beta) \leq \tau_\beta(\xi) \leq 1$. We slightly modify the definition of A_0^a and O_L^c to include also the initial and final time points, i.e., to be the weight serviced by Reroute(β) in $[\alpha^0, \beta \cdot \alpha^1]$ and the weight serviced by Opt in $[\tau_\beta(\xi) \cdot \alpha^L, \alpha^L]$, respectively. Clearly, $\text{Alg}_i = \text{A}_i^a + \text{A}_i^b + \text{A}_i^c$ and $\text{Opt}_i = \text{O}_i^a + \text{O}_i^b + \text{O}_i^c$.

As Reroute services the same set of requests as RetF, the guarantee of Lemma 2 applies to the schedule produced by Reroute(β) as well. In particular, Reroute(β) finishes servicing all requests till the end of phase $L + 1$ (it does not service anything in phase 0) and for any phase $j \in \{1, \ldots, L + 1\}$, it holds that

$$\sum_{i=j}^{L+1} \left(\text{A}_i^a + \text{A}_i^b + \text{A}_i^c \right) \leq \sum_{i=j-1}^{L} \left(\text{O}_i^a + \text{O}_i^b + \text{O}_i^c \right) . \tag{5}$$

Fix any phase $j \in \{1, \ldots, L+1\}$ and consider all requests contributing to the sum $\sum_{i=j}^{L+1}(\mathrm{A}_i^b + \mathrm{A}_i^c)$. By Lemma 4, each such request has to be serviced by Opt at time $\tau_\beta(\beta) \cdot \alpha^{j-1}$ or later (because its distance to the origin is at least $\tau_\beta(\beta) \cdot \alpha^{j-1}$). Thus,

$$\sum_{i=j}^{L+1} \left(\mathrm{A}_i^b + \mathrm{A}_i^c\right) \le \mathrm{O}_{j-1}^b + \mathrm{O}_{j-1}^c + \sum_{i=j}^{L} \left(\mathrm{O}_i^a + \mathrm{O}_i^b + \mathrm{O}_i^c\right) . \tag{6}$$

Similarly, consider all requests contributing to the sum $\mathrm{A}_j^c + \sum_{i=j+1}^{L+1}(\mathrm{A}_i^b + \mathrm{A}_i^c)$. Again, by Lemma 4, each such request has to be serviced by Opt at time $\tau_\beta(\xi) \cdot \alpha^{j-1}$ or later (because its distance to the origin is at least $\tau_\beta(\xi) \cdot \alpha^{j-1}$). Hence,

$$\mathrm{A}_j^c + \sum_{i=j+1}^{L+1} \left(\mathrm{A}_i^b + \mathrm{A}_i^c\right) \le \mathrm{O}_{j-1}^c + \sum_{i=j}^{L} \left(\mathrm{O}_i^a + \mathrm{O}_i^b + \mathrm{O}_i^c\right) . \tag{7}$$

Finally, observe that the total cost of Reroute(β) can be upper-bounded by $\sum_{i=1}^{L+1}(\beta \cdot \alpha^i \cdot \mathrm{A}_i^a + \xi \cdot \alpha^i \cdot \mathrm{A}_i^b + \alpha^i \cdot \mathrm{A}_i^c)$ and the cost of Opt can be lower-bounded by $\sum_{i=0}^{L}(\alpha^{i-1} \cdot \mathrm{O}_i^a + \tau_\beta(\beta) \cdot \alpha^i \cdot \mathrm{O}_i^b + \tau_\beta(\xi) \cdot \alpha^i \cdot \mathrm{O}_i^c)$.

4.2 Factor-revealing LP

Let us now consider what happens when an adversary constructs an instance $\mathcal{I}$ for the algorithm Reroute. When Opt is run on $\mathcal{I}$, this defines L as the index of the last phase when Opt services a request and this also defines non-negative values of variables $\mathrm{O}_i^a, \mathrm{O}_i^b, \mathrm{O}_i^c$ for $i \in \{0, \ldots, L\}$. When Reroute(β) is run on $\mathcal{I}$, this defines non-negative values of variables $\mathrm{A}_i^a, \mathrm{A}_i^b, \mathrm{A}_i^c$ for $i \in \{1, \ldots, L+1\}$. As shown above, these variables satisfy inequalities (5), (6) and (7). Moreover, the goal of the adversary is to maximize the ratio between the cost of Reroute(β) and the cost of Opt.

This maximization problem may become only easier for the adversary if instead of creating an actual input sequence, the adversary simply chooses L and the non-negative values of variables $\mathrm{A}_i^a, \mathrm{A}_i^b, \mathrm{A}_i^c, \mathrm{O}_i^a, \mathrm{O}_i^b, \mathrm{O}_i^c$ for $i \in \{0, \ldots, L\}$, satisfying inequalities (5), (6) and (7), so to maximize the objective value of Reroute-to-Opt cost ratio.

The values of all variables can be multiplied by a fixed value without changing the objective value. Thus, instead of maximizing the cost ratio, the adversary may maximize total cost of Reroute(β) with an additional constraint ensuring that the total cost of Opt is at most 1.

This leads to the following factor-revealing linear program $\mathcal{P}(L, \beta, \xi)$, whose optimal value $P^*(L, \beta, \xi)$ is an upper bound on the competitive ratio of Reroute(β) for a given L. This relation holds for any choice of parameter $\xi \in [\beta, 1]$. The goal of $\mathcal{P}(L, \beta, \xi)$ is to maximize

$$\sum_{i=1}^{L+1} \left(\beta \cdot \alpha^i \cdot \mathrm{A}_i^a + \xi \cdot \alpha^i \cdot \mathrm{A}_i^b + \alpha^i \cdot \mathrm{A}_i^c\right)$$

subject to the following constraints

$$\sum_{i=j}^{L+1}\left(\mathrm{A}_i^a+\mathrm{A}_i^b+\mathrm{A}_i^c\right)\le\sum_{i=j-1}^{L}\left(\mathrm{O}_i^a+\mathrm{O}_i^b+\mathrm{O}_i^c\right)\qquad\text{for all } j\in\{1,\dots,L+1\}$$

$$\sum_{i=j}^{L+1}\left(\mathrm{A}_i^b+\mathrm{A}_i^c\right)\le\mathrm{O}_{j-1}^b+\mathrm{O}_{j-1}^c+\sum_{i=j}^{L}\left(\mathrm{O}_i^a+\mathrm{O}_i^b+\mathrm{O}_i^c\right)\qquad\text{for all } j\in\{1,\dots,L+1\}$$

$$\mathrm{A}_j^c+\sum_{i=j+1}^{L+1}\left(\mathrm{A}_i^b+\mathrm{A}_i^c\right)\le\mathrm{O}_{j-1}^c+\sum_{i=j}^{L}\left(\mathrm{O}_i^a+\mathrm{O}_i^b+\mathrm{O}_i^c\right)\qquad\text{for all } j\in\{1,\dots,L+1\}$$

$$\sum_{i=0}^{L}\left(\alpha^{i-1}\cdot\mathrm{O}_i^a+\tau_\beta(\beta)\cdot\alpha^i\cdot\mathrm{O}_i^b+\tau_\beta(\xi)\cdot\alpha^i\cdot\mathrm{O}_i^c\right)\le 1$$

and non-negativity of the variables. Note that the left hand side of the last inequality is a lower bound on the cost of Opt

It remains to upper-bound the value of $P^*(L,\beta,\xi)$. Such upper bound is given by the value of any feasible solution to the dual program $\mathcal{D}(L,\beta,\xi)$. The goal of $\mathcal{D}(L,\beta,\xi)$ is to minimize $\mathcal{R}$ subject to the following constraints

$$\sum_{i=1}^{j}q_i^a\ge\beta\cdot\alpha^j\qquad\text{for all } j\in\{1,\dots,L+1\}\tag{8}$$

$$\sum_{i=1}^{j-1}\left(q_i^a+q_i^b+q_i^c\right)+q_j^a+q_j^b\ge\xi\cdot\alpha^j\qquad\text{for all } j\in\{1,\dots,L+1\}\tag{9}$$

$$\sum_{i=1}^{j}\left(q_i^a+q_i^b+q_i^c\right)\ge\alpha^j\qquad\text{for all } j\in\{1,\dots,L+1\}\tag{10}$$

$$\alpha^{j-1}\cdot\mathcal{R}\ge\sum_{i=1}^{j}\left(q_i^a+q_i^b+q_i^c\right)+q_{j+1}^a\qquad\text{for all } j\in\{0,\dots,L\}\tag{11}$$

$$\tau_\beta(\beta)\cdot\alpha^j\cdot\mathcal{R}\ge\sum_{i=1}^{j}\left(q_i^a+q_i^b+q_i^c\right)+q_{j+1}^a+q_{j+1}^b\qquad\text{for all } j\in\{0,\dots,L\}\tag{12}$$

$$\tau_\beta(\xi)\cdot\alpha^j\cdot\mathcal{R}\ge\sum_{i=1}^{j+1}\left(q_i^a+q_i^b+q_i^c\right)\qquad\text{for all } j\in\{0,\dots,L\}\tag{13}$$

and non-negativity of the variables. Note that sets of inequalities (8), (9), (10), (11), (12) and (13) correspond to sets of variables A_i^a, A_i^b, A_i^c, O_i^a, O_i^b and O_i^c, respectively, in the primal program $\mathcal{P}(L,\beta,\xi)$.

▶ **Lemma 5.** *There exist values β and ξ, such that for any L, there exists a feasible solution to $\mathcal{D}(L,\beta,\xi)$ whose value is at most $2\sqrt{2}+13/5<5.429$.*

Proof. We choose

$$\beta=(9\alpha+4)/(5\alpha^2)=(3+\sqrt{2})/5\approx 0.883 \text{ and}$$
$$\xi=\beta+(1-\beta)/\alpha=(4\sqrt{2}-1)/5\approx 0.931\,.$$

For these values of β and ξ, it holds that

$$\tau_\beta(\beta)=1/\alpha=\sqrt{2}-1\approx 0.414 \text{ and}$$
$$\tau_\beta(\xi)=\tau_\beta(\beta)+(\xi-\beta)\cdot\alpha=1/\alpha+(2-\sqrt{2})/5=(5+\sqrt{2})/(5+5\sqrt{2})\approx 0.531\,.$$

We set the dual variables as follows:

$$\begin{aligned}
q_1^a &= \beta \cdot \alpha\,, \\
q_1^b &= (\xi - \beta) \cdot \alpha\,, \\
q_1^c &= (1 - \xi) \cdot \alpha\,, \\
q_j^a &= \beta \cdot (\alpha - 1) \cdot \alpha^{j-1} && \text{for } j \in \{2, \ldots, L+1\}\,, \\
q_j^b &= 0 && \text{for } j \in \{2, \ldots, L+1\}\,, \\
q_j^c &= (1 - \beta) \cdot (\alpha - 1) \cdot \alpha^{j-1} && \text{for } j \in \{2, \ldots, L+1\}\,.
\end{aligned}$$

Our choice of β, ξ and dual variables satisfy (8), (9) and (10) conditions of $\mathcal{D}(L, \beta, \xi)$ with equality. Actually, (8), (10) and (9) for $j = 1$ hold with equality for any choice of β and ξ. For $j \in \{2, \ldots, L+1\}$, the left hand side of (9) is equal to $\alpha^{j-1} + \beta \cdot (\alpha - 1) \cdot \alpha^{j-1} = (\beta \cdot \alpha + 1 - \beta) \cdot \alpha^{j-1}$, and the right hand side is equal to $\xi \cdot \alpha^j$; these values coincide for $\xi = \beta + (1 - \beta)/\alpha$.

Given the fixed values of the dual variables, we choose $\mathcal{R}$ as the minimum value satisfying inequalities (11), (12) and (13). Substituting the chosen values of the dual variables in these inequalities and using $\tau_\beta(\beta) = 1/\alpha$, yields

$$\begin{aligned}
\mathcal{R} &\geq (\beta \cdot \alpha)/\alpha^{-1} = \beta \cdot \alpha^2 && \text{by (11) for } j = 0\,, \\
\mathcal{R} &\geq (\alpha^j + \beta \cdot (\alpha - 1) \cdot \alpha^j)/\alpha^{j-1} = \beta \cdot \alpha^2 + \alpha - \beta \cdot \alpha && \text{by (11) for } j \geq 1\,, \\
\mathcal{R} &\geq (\beta \cdot \alpha + (\xi - \beta) \cdot \alpha)/\tau_\beta(\beta) = \xi \cdot \alpha^2 && \text{by (12) for } j = 0\,, \\
\mathcal{R} &\geq (1 + \beta \cdot (\alpha - 1)) \cdot \alpha^j/(\tau_\beta(\beta) \cdot \alpha^j) = \beta \cdot \alpha^2 + \alpha - \beta \cdot \alpha && \text{by (12) for } j \geq 1\,, \\
\mathcal{R} &\geq \alpha^{j+1}/(\tau_\beta(\xi) \cdot \alpha^j) = \alpha/\tau_\beta(\xi) && \text{by (13) for } j \geq 0\,.
\end{aligned}$$

Thus, using $\beta \leq \xi$ and $\xi = \beta + (1 - \beta)/\alpha$, we obtain

$$\begin{aligned}
\mathcal{R} &= \max\left\{\beta \cdot \alpha^2,\ \beta \cdot \alpha^2 + \alpha - \beta \cdot \alpha,\ \xi \cdot \alpha^2,\ \frac{\alpha}{\tau_\beta(\xi)}\right\} \\
&= \max\left\{\xi \cdot \alpha^2,\ \frac{\alpha}{\tau_\beta(\xi)}\right\} \\
&= \max\left\{2\sqrt{2} + 13/5,\ (15 + 10\sqrt{2})/(5 + \sqrt{2})\right\} \\
&= 2\sqrt{2} + 13/5 < 5.429\,,
\end{aligned}$$

which concludes the proof. ◀

▶ **Theorem 6.** *For $\beta = (3 + \sqrt{2})/5 \approx 0.883$, the competitive ratio of* ReRoute(β) *for the traveling repairperson problem is at most $2\sqrt{2} + 13/5 < 5.429$.*

Proof. Fix any input sequence $\mathcal{I}$, run Opt on $\mathcal{I}$, and partition its execution into phases. Let L be the index of the last phase in which Opt services a request.

Let $\xi = \beta + (1 - \beta)/\alpha$. As discussed above, the competitive ratio of ReRoute(β) is upper-bounded by the optimal value $P^*(L, \beta, \xi)$ of the maximization program $\mathcal{P}(L, \beta, \xi)$. By weak duality, the feasible solution to the dual minimization program $\mathcal{D}(L, \beta, \xi)$ of value $2\sqrt{2} + 13/5$ proposed in Lemma 5 is an upper bound on the optimal primal solution $P^*(L, \beta, \xi)$. Hence, $2\sqrt{2} + 13/5$ is an upper bound on the competitive ratio of ReRoute(β). ◀

5 Final remarks

Our computer-based experiments show that further partitioning of phases into more than three intervals does not lead to an improvement of the competitive ratio.

Furthermore, it is possible to show that our solution to the dual program is asymptotically best possible and the ratio cannot be improved by simply choosing better parameters β and ξ. That is, with growing L, the optimal value of the dual $\mathcal{D}(L, \beta, \xi)$ converges to the value $2\sqrt{2} + 13/5$ given by Lemma 5, as demonstrated in the lemma below.

▶ **Lemma 7.** *Fix any L. For any values of β and ξ, satisfying $1/\alpha \le \beta \le \xi \le 1$ and $1/\alpha \le \tau_\beta(\beta) \le \tau_\beta(\xi) \le 1$, the value of any feasible solution to the dual program $\mathcal{D}(L, \beta, \xi)$ is at least $(2\sqrt{2} + 13/5)/(1 + \alpha^{-L})$.*

Proof. Fix any $j \in \{1, \dots, L\}$. By combining requirement (11) with (10), we obtain

$$\begin{aligned}(\mathcal{R} - \alpha) \cdot \alpha^{j-1} &= \mathcal{R} \cdot \alpha^{j-1} - \alpha^j \\ &\ge \sum_{i=1}^{j} \left(q_i^a + q_i^b + q_i^c\right) + q_{j+1}^a - \sum_{i=1}^{j} \left(q_i^a + q_i^b + q_i^c\right) \\ &= q_{j+1}^a \,.\end{aligned}$$

Summing this relation over all $j \in \{1, \dots, L\}$ and using (8) yields

$$(\mathcal{R} - \alpha) \cdot \frac{\alpha^L - 1}{\alpha - 1} \ge \sum_{j=1}^{L} q_{j+1}^a = -q_1^a + \sum_{j=1}^{L+1} q_j^a \ge \beta \cdot \alpha^{L+1} - q_1^a \,.$$

Now we observe that (11) for $j = 1$ implies $q_1^a \le \mathcal{R}/\alpha < \mathcal{R}/(\alpha - 1)$. By substituting this in the inequality above and multiplying both sides by $\alpha - 1$, we get

$$(\mathcal{R} - \alpha) \cdot (\alpha^L - 1) \ge \beta \cdot (\alpha - 1) \cdot \alpha^{L+1} - \mathcal{R} \,.$$

Finally, we divide both sides by α^L, obtaining

$$\mathcal{R} - \alpha \ge (\mathcal{R} - \alpha) \cdot (\alpha^L - 1)/\alpha^L \ge \beta \cdot (\alpha - 1) \cdot \alpha - \mathcal{R}/\alpha^L \,,$$

and therefore,

$$\mathcal{R} \cdot (1 + \alpha^{-L}) \ge \beta \cdot \alpha^2 + \alpha - \beta \cdot \alpha \,.$$

As in our construction we require $\tau_\beta(\beta) \ge 1/\alpha$, it holds that $\beta \ge (9\alpha + 4)/(5\alpha^2) = (3 + \sqrt{2})/5$. Combining this bound with the value of $\alpha = 1 + \sqrt{2}$ yields

$$\mathcal{R} \ge (\beta \cdot \alpha^2 + \alpha - \beta \cdot \alpha)/(1 + \alpha^{-L}) \ge (2\sqrt{2} + 13/5)/(1 + \alpha^{-L}) \,. \qquad \blacktriangleleft$$

References

1 Foto N. Afrati, Stavros S. Cosmadakis, Christos H. Papadimitriou, George Papageorgiou, and Nadia Papakostantinou. The Complexity of the Travelling Repairman Problem. *Informat. Theor. Appl.*, 20(1):79–87, 1986. doi:10.1051/ita/1986200100791.

2 Aaron Archer and Anna Blasiak. Improved Approximation Algorithms for the Minimum Latency Problem via Prize-Collecting Strolls. In *Proc. 21st ACM-SIAM Symp. on Discrete Algorithms (SODA)*, pages 429–447, 2010. doi:10.1137/1.9781611973075.36.

3 Norbert Ascheuer, Sven Oliver Krumke, and Jörg Rambau. Online Dial-a-Ride Problems: Minimizing the Completion Time. In *Proc. 17th Symp. on Theoretical Aspects of Computer Science (STACS)*, pages 639–650, 2000. doi:10.1007/s10951-005-6811-3.

4 Giorgio Ausiello, Esteban Feuerstein, Stefano Leonardi, Leen Stougie, and Maurizio Talamo. Serving Requests with On-line Routing. In *Proc. 4th Scandinavian Symposium and Workshops on Algorithm Theory (SWAT)*, pages 37–48, 1994. doi:10.1007/3-540-58218-5_4.

5 Giorgio Ausiello, Esteban Feuerstein, Stefano Leonardi, Leen Stougie, and Maurizio Talamo. Competitive Algorithms for the On-line Traveling Salesman. In *Proc. 4th Int. Workshop on Algorithms and Data Structures (WADS)*, pages 206–217, 1995. doi:10.1007/3-540-60220-8_63.

6 Giorgio Ausiello, Esteban Feuerstein, Stefano Leonardi, Leen Stougie, and Maurizio Talamo. Algorithms for the On-Line Travelling Salesman. *Algorithmica*, 29(4):560–581, 2001. doi:10.1007/s004530010071.

7 Ricardo A. Baeza-Yates, Joseph C. Culberson, and Gregory J. E. Rawlins. Searching in the Plane. *Information and Computation*, 106(2):234–252, 1993. doi:10.1006/inco.1993.1054.

8 Alexander Birx and Yann Disser. Tight Analysis of the Smartstart Algorithm for Online Dial-a-Ride on the Line. In *Proc. 36th Symp. on Theoretical Aspects of Computer Science (STACS)*, pages 15:1–15:17, 2019. doi:10.4230/LIPIcs.STACS.2019.15.

9 Antje Bjelde, Yann Disser, Jan Hackfeld, Christoph Hansknecht, Maarten Lipmann, Julie Meißner, Kevin Schewior, Miriam Schlöter, and Leen Stougie. Tight Bounds for Online TSP on the Line. In *Proc. 28th ACM-SIAM Symp. on Discrete Algorithms (SODA)*, pages 994–1005, 2017. doi:10.1137/1.9781611974782.63.

10 Vincenzo Bonifaci and Leen Stougie. Online k-Server Routing Problems. *Theory of Computing Systems*, 45(3):470–485, 2009. doi:10.1007/s00224-008-9103-4.

11 Allan Borodin and Ran El-Yaniv. *Online Computation and Competitive Analysis*. Cambridge University Press, 1998.

12 Kamalika Chaudhuri, Brighten Godfrey, Satish Rao, and Kunal Talwar. Paths, Trees, and Minimum Latency Tours. In *Proc. 44th IEEE Symp. on Foundations of Computer Science (FOCS)*, pages 36–45, 2003. doi:10.1109/SFCS.2003.1238179.

13 Esteban Feuerstein and Leen Stougie. On-line single-server dial-a-ride problems. *Theoretical Computer Science*, 268(1):91–105, 2001. doi:10.1016/S0304-3975(00)00261-9.

14 Irene Fink, Sven Oliver Krumke, and Stephan Westphal. New lower bounds for online k-server routing problems. *Information Processing Letters*, 109(11):563–567, 2009. doi:10.1016/j.ipl.2009.01.024.

15 Michel X. Goemans and Jon M. Kleinberg. An improved approximation ratio for the minimum latency problem. *Math. Program.*, 82:111–124, 1998. doi:10.1007/BF01585867.

16 Dietrich Hauptmeier, Sven Oliver Krumke, and Jörg Rambau. The Online Dial-a-Ride Problem under Reasonable Load. In *Proc. 4th Int. Conf. on Algorithms and Complexity (CIAC)*, pages 125–136, 2000. doi:10.1007/3-540-46521-9_11.

17 Sven Oliver Krumke, Willem de Paepe, Diana Poensgen, Maarten Lipmann, Alberto Marchetti-Spaccamela, and Leen Stougie. On Minimizing the Maximum Flow Time in the Online Dial-a-Ride Problem. In *Proc. 3rd Workshop on Approximation and Online Algorithms (WAOA)*, pages 258–269, 2005. doi:10.1007/11671411_20.

18 Sven Oliver Krumke, Willem de Paepe, Diana Poensgen, and Leen Stougie. News from the online traveling repairman. *Theoretical Computer Science*, 295:279–294, 2003. doi:10.1016/S0304-3975(02)00409-7.

19 Sven Oliver Krumke, Luigi Laura, Maarten Lipmann, Alberto Marchetti-Spaccamela, Willem de Paepe, Diana Poensgen, and Leen Stougie. Non-abusiveness Helps: An O(1)-Competitive Algorithm for Minimizing the Maximum Flow Time in the Online Traveling Salesman Problem. In *Proc. 5th Int. Workshop on Approximation Algorithms for Combinatorial Optimization (APPROX)*, pages 200–214, 2002. doi:10.1007/3-540-45753-4_18.

20 René Sitters. The Minimum Latency Problem Is NP-Hard for Weighted Trees. In *Proc. 9th Int. Conf. on Integer Programming and Combinatorial Optimization (IPCO)*, pages 230–239, 2002. doi:10.1007/3-540-47867-1_17.

21 René Sitters. Polynomial time approximation schemes for the traveling repairman and other minimum latency problems. In *Proc. 25th ACM-SIAM Symp. on Discrete Algorithms (SODA)*, pages 604–616, 2014. doi:10.1137/1.9781611973402.46.