The generalized two-server problem

René Sitters[1] and Leen Stougie[23⋆]

[1] Max-Planck-Institute für Informatik
Stuhlsatzenhausweg 85, 66123 Saarbrücken, Germany
sitters@mpi-inf.mpg.de
[2] Department of Mathematics and Computer Science
Technische Universiteit Eindhoven
P.O.Box 513, 5600 MB Eindhoven, The Netherlands
l.stougie@tue.nl
[3] CWI, P.O.Box 94079, 1090 GB Amsterdam, the Netherlands

Abstract. We consider the generalized on-line two-server problem in which at each step each server receives a request, which is a point in a metric space. One of the servers has to be moved to its request. Thus, each of the servers is moving in his own metric space. The special case in which both metric spaces are the real line is known as the CNN-problem. It has been a well-known open question in on-line optimization 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 generalized two-server problem on any metric space.
The basic result in this paper is a characterization of competitiveness for metrical service systems that seems much easier to use when looking for a competitive algorithm. The existence of a competitive algorithm for the generalized two-server problem follows rather easily from this result.

1 Introduction

In the *generalized k-server problem* we are given k servers each of which is moving in some metric space $\mathbb{M}_i$, $i = 1, \ldots, k$, starting in some given point $O_i \in \mathbb{M}_i$. They are to serve requests $r \in \mathbb{M}_1 \times \mathbb{M}_2 \times \cdots \times \mathbb{M}_k$ which arrive one by one. A request $r = (z_1, z_2, \ldots, z_k)$ is served by moving, for at least one i, the server in space $\mathbb{M}_i$ to the point z_i. The decision as to which server to move to the next request is irrevocable and has to be taken without any knowledge about future requests. The cost of moving the i-th server to z_i is equal to the distance travelled by this server from his current location to z_i. The objective is to minimize the total cost to serve all given requests.

We measure the performance of an on-line algorithm through competitive analysis. An on-line algorithm is *c-competitive* if, for any request sequence σ, the algorithm's cost is at most c ($c \geq 1$) times the cost of the optimal solution of the corresponding off-line problem plus an additive constant independent of σ. We say that an algorithm is *competitive* if it is c-competitive for some constant $c \geq 1$.

We can see the generalized k-server problem as a single server problem, by moving the server in the metric space $\mathbb{M} = \mathbb{M}_1 \times \cdots \times \mathbb{M}_k$, and interpreting the server positions and the requests in the description above as the "coordinates" of the single server and the requests. Interpreted in this way, it is a special case of a *metrical service system*. In a metrical service system each request

⋆ Part of this research has been funded by the Dutch BSIK/BRICKS project.

can be any subset of the metric space, and is served by moving the one server to one of the points in the request. Metrical service systems were introduced by Manasse, McGeoch, and Sleator [14] who used the term *forcing task systems* and independently by Chrobak and Larmore [6] to provide a formalism for investigating a wide variety of on-line optimization problems. A precise definition is given in Section 2. In the same section we derive the basic theorem of this paper. It provides a sufficient condition for the existence of constant competitive algorithms for general metrical service systems. The result on the generalized two-server problem then becomes a matter of verifying this condition.

The generalized k-server problem is a natural generalization of the well-known *k-server problem* for which $\mathbb{M}_1 = \mathbb{M}_2 = \cdots = \mathbb{M}_k$ and $z_1 = z_2 = \cdots = z_k$ at each time step. The k-server problem was introduced by Manasse, McGeoch and Sleator [14], 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 $k = 2$ [14], for some special metric spaces such as the line, the star, and for all spaces with at most $k+2$ points [1,4,5]. For $k \geq 3$ the current best upper bound of $2k-1$ is given by Koutsoupias and Papadimitriou [12].

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 [9] prove that for any metric space with at least $k+1$ points 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 (in which all internode distances are one) and $k = 2$ Feuerstein et al. [8] give a 6.275-competitive algorithm, which was improved by Chrobak and Sgall [7] who 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 was known yet even for $k = 2$. It is easy to see that the generalized k-server problem is a generalization of the weighted k-server problem as well.

The generalized two-server problem in which both servers move on the real line has become well-known as the CNN-problem. Koutsoupias and Taylor [13] emphasize the importance of the CNN-problem as one of the simplest problems in a rich class of so-called sum-problems [2]. In the sum-problem each of a set of systems gets a request and only one system has to serve this request.

Koutsoupias and Taylor [13] prove a lower bound of $6 + \sqrt{17}$ on the competitive ratio of any deterministic on-line algorithm for the generalized two-server problem, through an instance of the weighted two-server problem on the real line. They also conjecture that the *generalized work function algorithm* has constant competitive ratio for the generalized two-server problem. For the generalized two-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 Section 3 we answer this question affirmatively, by designing an algorithm and prove an upper bound of 879 on its competitive ratio[4]. Our algorithm is a combination of the well-known balance algorithm and the generalized work function algorithm. The result is merely checking the condition of the general theorem for metrical service systems in Section 2, announced above.

Optimal off-line solutions of metrical service systems can easily be found by dynamic programming (see [2]), which yields an $O(k^2 n^k)$ time algorithm for the generalized k-server problem. For the classical k-server problem this running time can be reduced to $O(kn^2)$ by formulating it as a

[4] An extended abstract [15], co-authored by the authors of this paper, shows a 10^5-competitive algorithm.

min-cost flow problem [4]. No such improvement should be expected for the generalized k-server problem since the problem is NP-hard, as we will show in Section 4.

2 Competitiveness of metrical service systems

A metrical service system $\mathcal{S} = (\mathbb{M}, \mathcal{R})$ is specified by a metric space $\mathbb{M}$ with distance function $d : \mathbb{M}^2 \to \mathbb{R}^+$ and a set $\mathcal{R}$ of all possible requests. Each request $r \in \mathcal{R}$ is a subset of $\mathbb{M}$. An instance of the system consists of an initial server position $\mathcal{O} \in \mathbb{M}$ and a sequence $\sigma = r_1, r_2, \ldots$ of requests. Every request r_i must be served immediately and irrevocably by moving the one server to a point $s_i \in r_i$, before the future requests, $r_{i+1}, r_{i+2}, \ldots$, are given. The cost of the solution is the length of the path in $\mathbb{M}$ followed by the server.

The *generalized work function algorithm* is an important tool in the construction of a competitive algorithm for metrical service systems. The generalized work function algorithm was introduced independently by several people and has been shown to be competitive for several on-line problems. In fact we go along with Koutsoupias and Taylor [13] in conjecturing that it is competitive for the generalized two-server problem as well. The generalized work function algorithm bases its moves on the position of the on-line server and the values of the *work function*.

Definition 1. *Given a metrical service system $\mathcal{S} = (\mathbb{M}, \mathcal{R})$ with origin $\mathcal{O} \in \mathbb{M}$, and request sequence σ we define the* work function $W_\sigma : \mathbb{M} \to \mathbb{R}^+$. *For any point $s \in \mathbb{M}$, $W_\sigma(s)$ is the length of the shortest path that starts in $\mathcal{O}$, ends in s and serves σ.*

We assume here that the work function is always well-defined, which might not be true if the metric space is infinite. Thus, we assume that for any $\sigma = r_1, \ldots, r_n$ and any point $s \in \mathbb{M}$ there are points $s_i \in r_i$ $(i = 1, \ldots, n)$ such that $d(\mathcal{O}, s_1) + d(s_1, s_2) + \cdots + d(s_{n-1}, s_n) + d(s_n, s) \leq d(\mathcal{O}, t_1) + d(t_1, t_2) + \cdots + d(t_{n-1}, t_n) + d(t_n, s)$ for any set of points $t_i \in r_i$ $(i = 1, \ldots, n)$. Notice that this implies $W_{r_1,\ldots,r_i}(s_i) = d(\mathcal{O}, s_1) + d(s_1, s_2) + \cdots + d(s_{i-1}, s_i)$ for all $i \in \{1, \ldots, n\}$.

We will use the following properties, which are rather obvious. The work function on the empty string ϵ of requests is $W_\epsilon(s) = d(\mathcal{O}, s)$ for all $s \in \mathbb{M}$. The work function is Lipschitz continuous: for any two points s and s' in $\mathbb{M}$, $|W_\sigma(s) - W_\sigma(s')| \leq d(s, s')$. It exhibits monotonicity with respect to the request sequence for every $s \in \mathbb{M}$: given any request sequence σ and any new request r, $W_{\sigma,r}(s) \geq W_\sigma(s)$, for all $s \in \mathbb{M}$. Equality holds for all $s \in r$.

Given a work function W_σ we say that point s is *dominated* by point t if $W_\sigma(s) = W_\sigma(t) + d(s, t)$. We define the *support* of W_σ as $\mathrm{supp}(W_\sigma) = \{t \in \mathbb{M} : t \text{ is not dominated by any other point}\}$. If $W_{\sigma,r}$ is a well-defined work function then $\mathrm{supp}(W_{\sigma,r}) \subseteq r$, since for any point $s \notin r$ there exists a point $t \in r$ such that $W_{\sigma,r}(s) = W_{\sigma,r}(t) + d(t, s)$. For more properties and a deeper analysis of work functions we refer to [3],[6].

The generalized work function algorithm is a work function-based algorithm parameterized by some constant $\lambda \geq 1$. We call it λ-WFA. For any request sequence σ and any new request r, λ-WFA moves the server from the position s it had after serving σ to point

$$s' = \mathrm{argmin}_{t \in r}\{W_{\sigma,r}(t) + (1/\lambda)d(s, t)\}. \tag{1}$$

Note that this minimum may not be well-defined if the request r contains infinitely many points of the metric space. In Theorem 1 we assume that the minimum is always attained for some $s' \in r$. For $\lambda = 1$ the algorithm is known as the (standard) work function algorithm, which is $2k - 1$ competitive for the classical k-server problem [12]. For $\lambda > 1$ the algorithm remains unchanged if

we replace $\text{argmin}_{t\in r}$ by $\text{argmin}_{t\in\mathbb{M}}$. To see this let $t \in \mathbb{M}\backslash r$ be an arbitrary point not in the set r of request points. Then there is a point $t' \in r$ such that $W_{\sigma,r}(t) = W_{\sigma,r}(t') + d(t',t)$, implying

$$\begin{aligned} W_{\sigma,r}(t) + (1/\lambda)d(s,t) &= W_{\sigma,r}(t') + d(t',t) + (1/\lambda)d(s,t) \\ &> W_{\sigma,r}(t') + (1/\lambda)(d(t',t) + d(s,t)) \\ &\geq W_{\sigma,r}(t') + (1/\lambda)(d(s,t'). \end{aligned}$$

Thus, for $\lambda > 1$, an alternative definition of λ-WFA is to move the server from s to point

$$s' = \text{argmin}_{t\in\mathbb{M}}\{W_{\sigma,r}(t) + (1/\lambda)d(s,t)\}. \tag{2}$$

In this paper we always assume $\lambda > 1$.

The generalized work function algorithm is competitive for several classical systems, such as the k-server problem [12] or the k-point request problem [3]. However, the analysis of this algorithm is in general complicated. In this section we derive properties of an algorithm A, which by itself may not be competitive for a metrical service system $\mathcal{S}$, such that A combined with the generalized work function algorithm results in a competitive algorithm for $\mathcal{S}$.

2.1 A competitive, work function-based algorithm

The on-line algorithm we propose works in *phases*. The last move in each phase is based on the generalized work function algorithm. We refer to these moves as the λ-WFA-*moves* of the on-line algorithm, though strictly speaking they differ from the move defined by (1) as we will point out later. All other moves in a phase are dictated by some other algorithm A.

The phases of the algorithm induce a partition of the request sequence $\sigma = r_1, r_2, \ldots$ into $r_1, \ldots, r_{i_1}; r_{i_1+1}, \ldots, r_{i_2}; r_{i_2+1}, \ldots, r_{i_3}; \ldots; r_{i_h+1}, \ldots, r_{i_{h+1}}; \ldots$ where (using $i_0 = 0$) $r_{i_{h-1}+1}, \ldots, r_{i_h}$ is the subsequence of consecutive requests served in the h-th phase.

We denote by $A(s,\sigma)$ the cost of algorithm A starting in s and serving request sequence σ. By $\text{OPT}(s,\sigma)$ we denote the cost of the shortest path starting in s and serving request sequence σ.

In the description of a generic phase h of the algorithm we denote by O_h the starting point of phase h, i.e., the endpoint of phase $h-1$, the position of the server after the $h-1$-th λ-WFA-move, serving request $r_{i_{h-1}}$. Hence $O_1 = \mathcal{O}$. We denote by σ_h the sequence of requests served in the first h phases, i.e., $\sigma_h = r_1, \ldots, r_{i_h}$, for $h \geq 1$.

Apart from the parameter $\lambda > 1$ in the definition of λ-WFA we employ a parameter $\gamma \geq 1$ in the description of the on-line algorithm.

Phase h of ONLINE(A, γ, λ):

Given request r_k ($k \geq i_{h-1} + 1$), if $A(O_h, r_{i_{h-1}+1}, \ldots, r_k) \leq \gamma\text{OPT}(O_h, r_{i_{h-1}+1}, \ldots, r_k)$, move the server according to A and wait for the next request. Otherwise, move the server to a point $s \in \mathbb{M}$ that minimizes $W_{r_1,\ldots,r_k}(s) + (1/\lambda)d(O_h, s)$. In this case $i_h = k$, $O_{h+1} = s$ and phase $h+1$ is started.

Two observations are worth making here. First we emphasize that the work function employed in the λ-WFA-move is defined over the *complete* input sequence, released so far, whereas the decision whether to continue to make A-moves or not is based on a comparison with the optimal solution during phase h, as if the sequence started in phase h with O_h as the origin. The second observation is that the move at the end of the phase is in the strict sense not a λ-WFA-move as defined in

(1) or (2), as the distance $d(O_h, s)$ is not from the current position of the server here, but from the origin O_h of the current phase. Still, we will continue calling it a λ-WFA-move, or refer to it alternatively as a work function step.

We are now ready to state our main theorem. The idea behind it is best explained by using adversaries serving the same request sequence, but knowing the sequence in advance.

If algorithm A is not competitive for the metrical service system, then this means that it is not competitive against a single adversary. Here we introduce an alternative model in which the algorithm A is playing against a number of adversaries simultaneously, where each of them has to serve the full sequence and moreover they are not allowed to operate too near to each other in a sense which is defined precisely below. By the latter restriction, if there is a sufficiently high number of adversaries in the game, then some of them will have to choose relatively expensive paths in order to stay sufficiently far away from relatively cheap paths followed by others. A may not be competitive against a single adversary, but it may be competitive against a number of such adversaries if we compare the cost of A to the sum (not average) of the costs of the adversaries.

To formalize the notion of two paths being too near, we introduce *dependency* of two paths. The definition should cover the situation that two paths cross, but should be more general than that. We use $|T|$ to denote the (weighted) length of a path T, as we will do throughout the paper.

Definition 2. *Paths T_1 from s_1 to s_1' and T_2 from s_2 to s_2' are called dependent if $|T_1| + |T_2| \geq d(s_1, s_2') + d(s_2, s_1')$. Otherwise the paths are said to be independent.*

If T_1 and T_2 go through the same point t, then the paths are dependent since $|T_1| + |T_2| \geq d(s_1, t) + d(t, s_1') + d(s_2, t) + d(t, s_2') \geq d(s_1, s_2') + d(s_2, s_1')$. On the other hand, two paths could be dependent without sharing a point of the metric space. Nevertheless it appears to be useful to call two paths dependent if they are close in the above sense.

The theorem says that if the cost of A is bounded by a constant times the length of the optimal path starting in the same point as A plus a constant times the sum of the lengths of m pairwise independent adversary paths, then algorithm ONLINE, which employs A within the phases in the way described above, is competitive against the length of the optimal path serving all requests and starting in $\mathcal{O}$.

Theorem 1. *Let $\mathcal{S}$ be a metrical service system on which the work function and λ-WFA are well-defined. If there exist an on-line algorithm A for $\mathcal{S}$, and constants $c_1 \geq 1, c_2 \geq 0$ and $m \geq 2$ such that for any point $s \in \mathbb{M}$, sequence ρ and pairwise independent paths $T_1, T_2, \ldots T_m$ that serve ρ*

$$A(s, \rho) \leq c_1 \, \textsc{Opt}(s, \rho) + c_2 \sum_{i=1}^{m} |T_i|, \tag{3}$$

then $\textsc{Online}(A, \gamma, \lambda)$ with $\gamma = c_1 + c_2$ and $\lambda = m$ is $15(c_1 + c_2)m2^m$-competitive for $\mathcal{S}$.

The proof follows from combining an appropriate lower bound on the optimal cost with an appropriate upper bound on the algorithm's cost, which are derived in the following two subsections separately. In both bounds the notion of *pseudo-cost* in a phase plays a central role.

Definition 3. *The pseudo-cost of the algorithm $\textsc{Online}(A, \gamma, \lambda)$ in phase h is defined by $\nabla_h = W_{\sigma_h}(O_{h+1}) + (1/\lambda)d(O_h, O_{h+1}) - W_{\sigma_{h-1}}(O_h)$.*

Pseudo-cost has proven to be a useful concept in the analysis of the (generalized) work function algorithm [3,12]. Typically, pseudo-cost is defined for a single λ-WFA-move. In our case, the pseudo-cost captures the cost of the λ-WFA-move in each phase. As we shall prove in Lemma 13, the total cost of all these moves is approximately λ times the sum over the pseudo-costs. On the other hand, the pseudo-cost measures a local increase in the work function. If the metrical service system allows for a competitive algorithm, then this local increase can be used, through an appropriate potential function, to bound the optimal solution from below. Summarizing, the notion of pseudo-cost provides a link between the total cost of the work function steps and the cost of the optimal solution.

Let N be the number of phases which ONLINE uses for serving the entire request sequence σ. The first $N-1$ phases end with a work function step. We assume that phase N ends before a work function step is made. We can do this without loss of generality when we allow the last phase to be empty (hence with ONLINE cost zero).

Theorem 1 is trivially true if ONLINE happens not to take any λ-WFA-move, i.e., the whole sequence is served in one phase. In this case the competitive ratio is γ. Therefore, we assume that at least one λ-WFA-move is made.

The proof of Theorem 1 follows rather easily from the combination of a lower bound on the optimal solution given in Lemma 8 (page 13) and an upper bound on the cost of algorithm ONLINE given in Lemma 14 (page 16). The lemmas are only cited in the proof of Theorem 1 below; they are proven separately in Subsections 2.2 and 2.3, respectively.

Proof (Theorem 1). A lower bound on the optimal cost of serving the sequence σ_{N-1}, that is, the sequence $\sigma = \sigma_N$ minus the requests of the last phase, is given by Lemma 8:

$$\text{OPT}(\mathcal{O}, \sigma_{N-1}) \geq \frac{\lambda - 1}{\beta_1(\lambda + 1)} \sum_{h=1}^{N-1} \nabla_h, \tag{4}$$

with $\beta_1 = 2\left(\frac{2\lambda}{\lambda-1}\right)^{m-2}$. An upper bound on the algorithm's cost for serving the complete sequence σ is given by Lemma 14:

$$\text{ONLINE}(\mathcal{O}, \sigma) \leq \left(\frac{4\gamma\lambda}{\lambda - 1} + (2\gamma + 1)\lambda\right) \sum_{h=1}^{N-1} \nabla_h - \lambda(2\gamma + 1)\text{OPT}(\mathcal{O}, \sigma_{N-1}) + \frac{2\gamma\lambda}{\lambda - 1}\text{OPT}(\mathcal{O}, \sigma). \tag{5}$$

Combining both bounds yields that $\text{ONLINE}(\mathcal{O}, \sigma)$ is bounded from above by

$$\text{ONLINE}(\mathcal{O}, \sigma) \leq \left(\left(\frac{4\gamma\lambda}{\lambda - 1} + (2\gamma + 1)\lambda\right) \frac{\beta_1(\lambda + 1)}{\lambda - 1} - (2\gamma + 1)\lambda\right) \text{OPT}(\mathcal{O}, \sigma_{N-1}) + \frac{2\gamma\lambda}{\lambda - 1}\text{OPT}(\mathcal{O}, \sigma).$$

Since $\lambda > 1$, we have $\frac{\beta_1(\lambda+1)}{\lambda-1} \geq 1$, hence the expression before $\text{OPT}(\mathcal{O}, \sigma_{N-1})$ is non-negative, and we can use $\text{OPT}(\mathcal{O}, \sigma_{N-1}) \leq \text{OPT}(\mathcal{O}, \sigma)$ to arrive at:

$$\text{ONLINE}(\mathcal{O}, \sigma) \leq \left(\left(\frac{4\gamma\lambda}{\lambda - 1} + (2\gamma + 1)\lambda\right) \frac{\beta_1(\lambda + 1)}{\lambda - 1} - (2\gamma + 1)\lambda + \frac{2\gamma\lambda}{\lambda - 1}\right) \text{OPT}(\mathcal{O}, \sigma). \tag{6}$$

Now choose $\lambda = m \geq 2$, by which $(2\gamma+1)\lambda > 2\gamma\lambda/(\lambda-1)$. Writing $\beta_1 = 2\left(\frac{2\lambda}{\lambda-1}\right)^{m-2} = \frac{m-1}{m}\left(\frac{2m}{m-1}\right)^{m-1}$, inequality (6) simplifies to

$$\begin{aligned}\textsc{Online}(\mathcal{O},\sigma) &\leq \left(\tfrac{4\gamma(m+1)}{m-1} + (2\gamma+1)(m+1)\right)\left(\tfrac{2m}{m-1}\right)^{m-1}\textsc{Opt}(\mathcal{O},\sigma)\\ &< \left(\tfrac{4\gamma(m+1)}{m-1} + (2\gamma+1)(m+1)\right)e2^{m-1}\textsc{Opt}(\mathcal{O},\sigma)\\ &< 15\gamma m 2^m \textsc{Opt}(\mathcal{O},\sigma).\end{aligned}$$

□

In Section 3 we show that for the generalized two-server problem the balance algorithm satisfies the premise of this theorem with $c_1 = 2$, $c_2 = 4$ and $m = 3$. A competitive algorithm follows then directly from the theorem.

2.2 A lower bound

For the proof of (4) we use a potential function argument. Our potential function is the same as the one used by Burley [3] in his analysis of the k-point request problem. The function relies on the concept of *slack* of a point relative to another point. Intuitively, the slack of a point s with respect to a point t is the amount that the work function value in s can increase before the generalized work function algorithm moves from s to t. More precisely, the generalized work function algorithm, being in point s after serving sequence σ, moves away from s after a new request r is given if there is a point t such that $W_{\sigma,r}(t) + (1/\lambda)d(s,t) \leq W_{\sigma,r}(s)$. The slack is the difference between the left and right side of this inequality. Formally, given a request sequence σ we define the slack of a point $s \in \mathbb{M}$ relative to a point $t \in \mathbb{M}$ as

$$V_\sigma(s;t) = W_\sigma(t) + (1/\lambda)d(s,t) - W_\sigma(s). \tag{7}$$

We notice that this slack function is not symmetric, which we emphasize in the notation by writing a semicolon between the two arguments of the function. It does satisfy the triangle inequality, i.e., $V_\sigma(s_1;s_2) + V_\sigma(s_2;s_3) \geq V_\sigma(s_1;s_3)$ for any sequence σ and for any three points s_1, s_2 and s_3. Also notice that $V_\sigma(s;s) = 0$ for any σ and any $s \in \mathbb{M}$. For notational convenience we extend the definition of slack to that of slack of a point s with respect to a *finite set of points* S:

$$V_\sigma(s;S) = \min_{t\in S} V_\sigma(s;t). \tag{8}$$

We define potential function $\Phi_\sigma : \mathbb{M}^m \to \mathbb{R}$ as

$$\Phi_\sigma(s_1,\ldots,s_m) = \beta_1 W_\sigma(s_1) - \sum_{i=2}^{m}(\beta_i \min_{j:j<i} V_\sigma(s_i;s_j)),$$

with $\beta_1 = 2\left(\frac{2\lambda}{\lambda-1}\right)^{m-2}$ and $\beta_i = \left(\frac{2\lambda}{\lambda-1}\right)^{m-i}$ $(i = 2,\ldots,m)$. We define

$$\Psi_\sigma = \min_{s_1,\ldots,s_m\in\mathbb{M}}\{\Phi_\sigma(s_1,\ldots,s_m)\}.$$

The following lemma shows that, for any request sequence σ, the value of the optimal solution serving σ is at least Ψ_σ/β_1.

Lemma 1. *$\Psi_\sigma \le \beta_1 \min_{s\in\mathbb{M}} W_\sigma(s) = \beta_1 \textsc{Opt}(\mathcal{O},\sigma)$ for every request sequence σ.*

Proof. Notice that for every $s \in \mathbb{M}$ we have $\Psi_\sigma \le \Phi_\sigma(s,\ldots,s) = \beta_1 W_\sigma(s)$. The equality is simply implied by the definition of the work function. □

The rest of the analysis is devoted to deriving a lower bound on Ψ_σ. In Lemma 3 we prove $\Psi_\epsilon = 0$, where, as before, we use ϵ to denote the empty string. In Lemma 7 we bound $\Psi_{\sigma_{h+1}} - \Psi_{\sigma_h}$ for each phase h. Hence, using $\Psi_{\sigma_0} = \Psi_\epsilon$, summing over all phases except the last one yields $\sum_{h=1}^{N-1} \Psi_h - \Psi_{h-1} = \Psi_{\sigma_{N-1}} - \Psi_\epsilon = \Psi_{\sigma_{N-1}}$, which, using the above Lemma 1, will result, in Lemma 8, in the desired lower bound (4) on $\textsc{Opt}(\mathcal{O},\sigma_{N-1})$.

By definition we have $\beta_m = 1$. The following additional equalities are easily verified.

(i) $\beta_1 = (1-1/\lambda) + (1+1/\lambda)\sum_{j=2}^{m} \beta_j$,

(ii) $\beta_i = 1 + (1+1/\lambda)/(1-1/\lambda) \sum_{j=i+1}^{m} \beta_j$, for $i \in \{1,2,\ldots,m-1\}$.

When considering off-line solutions we are only interested in optimal solutions or so-called adversary paths. Consider an adversary being in s' after having served the consecutive sequences σ and ρ. The length of the path followed by the adversary is exactly $W_{\sigma,\rho}(s')$. Let s be the point in which the adversary was located after having served σ. The length of the path up to that point is $W_\sigma(s)$. Thus, the length of the path between s and s' is $W_{\sigma,\rho}(s') - W_\sigma(s) \ge d(s,s')$. The next lemma shows that when adversary i ($i = 1,\ldots,m$) moves from point s_i, after serving σ, to point s'_i, after serving σ,ρ, then the increase in the function Φ with respect to the ordered sets of points $s_1\ldots,s_m$ and $s'_1\ldots,s'_m$ is at least a constant times the sum of the lengths of the adversary paths.

Lemma 2. *If sequences σ and ρ and points $s_1,\ldots,s_m$ and $s'_1,\ldots,s'_m$ satisfy $W_{\sigma,\rho}(s'_i) - W_\sigma(s_i) \ge d(s_i,s'_i)$ for all i, then*

$$\Phi_{\sigma,\rho}(s'_1,\ldots,s'_m) - \Phi_\sigma(s_1,\ldots,s_m) \ge (1-1/\lambda)\sum_{i=1}^{m}(W_{\sigma,\rho}(s'_i) - W_\sigma(s_i)).$$

Proof. To simplify notation we denote $W_{\sigma,\rho}(s'_i) - W_\sigma(s_i)$ by Δ_i for any $i \in \{1,\ldots,m\}$. First we bound $\min_{j:j<i} V_{\sigma,\rho}(s'_i;s'_j) - \min_{j:j<i} V_\sigma(s_i;s_j)$ for $i = 2,\ldots,m$. Take any arbitrary i and let $k = \operatorname{argmin}_{j:j<i} V_\sigma(s_i;s_j)$. Then

$$\begin{aligned}
\min_{j:j<i} V_{\sigma,\rho}(s'_i;s'_j) - \min_{j:j<i} V_\sigma(s_i;s_j) &= \min_{j:j<i} V_{\sigma,\rho}(s'_i;s'_j) - V_\sigma(s_i;s_k)\\
&\le V_{\sigma,\rho}(s'_i;s'_k) - V_\sigma(s_i;s_k)\\
&= W_{\sigma,\rho}(s'_k) - W_\sigma(s_k) + W_\sigma(s_i) - W_{\sigma,\rho}(s'_i) + \tfrac{1}{\lambda}(d(s'_k,s'_i) - d(s_k,s_i))\\
&= \Delta_k - \Delta_i + \tfrac{1}{\lambda}(d(s'_k,s'_i) - d(s_k,s_i))\\
&\le \Delta_k - \Delta_i + \tfrac{1}{\lambda}(d(s'_k,s_k) + d(s'_i,s_i))\\
&\le (1+1/\lambda)\Delta_k - (1-1/\lambda)\Delta_i\\
&\le (1+1/\lambda)\textstyle\sum_{j=1}^{i-1}\Delta_j - (1-1/\lambda)\Delta_i.
\end{aligned}$$

The first equality holds by definition of k. For the second equality we simply use (7), the definition of slack. The second inequality follows from the triangle inequality, and the third inequality is obtained by applying the premise of the lemma: $\Delta_i \geq d(s_i, s'_i)$ for all $i = 1, \ldots, m$. Now we use the derived inequality to bound the increase of the potential function from below.

$$\begin{aligned}
&\Phi_{\sigma,\rho}(s'_1, \ldots, s'_m) - \Phi_\sigma(s_1, \ldots, s_m) \\
&\qquad = \beta_1(W_{\sigma,\rho}(s'_1) - W_\sigma(s_1)) - \sum_{i=2}^{m} \beta_i(\min_{j:j<i} V_{\sigma,\rho}(s'_i; s'_j) - \min_{j:j<i} V_\sigma(s_i; s_j)) \\
&\qquad \geq \beta_1\Delta_1 - \sum_{i=2}^{m} \beta_i \left((1+1/\lambda) \sum_{j=1}^{i-1} \Delta_j - (1-1/\lambda)\Delta_i \right) \\
&\qquad = \beta_1\Delta_1 - (1+1/\lambda) \sum_{i=2}^{m} \beta_i \sum_{j=1}^{i-1} \Delta_j + (1-1/\lambda) \sum_{i=2}^{m} \beta_i\Delta_i \\
&\qquad = \beta_1\Delta_1 - (1+1/\lambda) \sum_{i=2}^{m} \beta_i\Delta_1 - (1+1/\lambda) \sum_{i=2}^{m} \left(\beta_i \sum_{j=2}^{i-1} \Delta_j \right) + (1-1/\lambda) \sum_{i=2}^{m} \beta_i\Delta_i \\
&\qquad = (\beta_1 - (1+1/\lambda) \sum_{i=2}^{m} \beta_i)\Delta_1 - (1+1/\lambda) \sum_{i=2}^{m} \sum_{j=i+1}^{m} \beta_j\Delta_i + (1-1/\lambda) \sum_{i=2}^{m} \beta_i\Delta_i \\
&\qquad = (\beta_1 - (1+1/\lambda) \sum_{i=2}^{m} \beta_i)\Delta_1 + \sum_{i=2}^{m} \left((1-1/\lambda)\beta_i - (1+1/\lambda) \sum_{j=i+1}^{m} \beta_j \right) \Delta_i \\
&\qquad = (1-1/\lambda)\Delta_1 + \sum_{i=2}^{m} (1-1/\lambda)\Delta_i.
\end{aligned}$$

The last equality follows directly from inserting $\beta_m = 1$ and equalities (i) and (ii) for the β_i's. □

Lemma 3. *For the empty string ϵ we have $\Psi_\epsilon = 0$.*

Proof. $\Phi_\epsilon(\mathcal{O}, \ldots, \mathcal{O}) = 0$. Now apply Lemma 2 with $\sigma = \rho = \epsilon$, and $s_1 = s_2 = \cdots = s_m = \mathcal{O}$. The premise of Lemma 2 is satisfied since for any point $s'_i \in \mathbb{M}$ we have $W_\epsilon(s'_i) - W_\epsilon(\mathcal{O}) = W_\epsilon(s'_i) = d(s'_i, \mathcal{O})$. Hence for any ordered set of points $s'_1, \ldots, s'_m$ we have $\Phi_\epsilon(s'_1, \ldots, s'_m) = \Phi_\epsilon(s'_1, \ldots, s'_m) - \Phi_\epsilon(\mathcal{O}, \ldots, \mathcal{O}) \geq (1 - 1/\lambda) \sum_{i=1}^{m} (W_\epsilon(s'_i) - W_\epsilon(\mathcal{O})) \geq 0$. □

In Lemma 7 we derive a lower bound on the change in value of Ψ in any phase h which finishes with a work-function move, i.e, all phases except the last one. In the proof we use Lemma 2 and the following three preliminary lemmas. Recall the definition of the pseudo-cost ∇_h of phase h on page 5, and remember that we defined O_h as the location of the on-line server at the beginning of phase h.

Lemma 4. $\nabla_h \leq W_{\sigma_h}(s) + (1/\lambda)d(O_h, s) - W_{\sigma_{h-1}}(O_h)$ *for every point $s \in \mathbb{M}$.*

Proof. By definition of the λ-WFA-move of algorithm ONLINE we have for every $s \in \mathbb{M}$ that $W_{\sigma_h}(s) + (1/\lambda)d(O_h, s) \geq W_{\sigma_h}(O_{h+1}) + (1/\lambda)d(O_h, O_{h+1}) = \nabla_h + W_{\sigma_{h-1}}(O_h)$. □

Lemma 5. *Let ρ_h be the sequence of requests served in phase h; then*

$$\nabla_h \leq \frac{\lambda+1}{\lambda} \textsc{Opt}(O_h, \rho_h).$$

Proof. Let t be the endpoint of a path that starts in O_h, serves ρ_h and has minimum length, i.e., a path with length $\textsc{Opt}(O_h, \rho_h)$. Application of Lemma 4 and using $W_{\sigma_h}(t) \leq W_{\sigma_{h-1}}(O_h) + \textsc{Opt}(O_h, \rho_h)$ yields

$$\begin{aligned} \nabla_h &\leq W_{\sigma_h}(t) + (1/\lambda)d(O_h, t) - W_{\sigma_{h-1}}(O_h) \\ &\leq W_{\sigma_{h-1}}(O_h) + \textsc{Opt}(O_h, \rho_h) + (1/\lambda)d(O_h, t) - W_{\sigma_{h-1}}(O_h) \\ &= \textsc{Opt}(O_h, \rho_h) + (1/\lambda)d(O_h, t) \\ &\leq (1 + 1/\lambda)\textsc{Opt}(O_h, \rho_h). \end{aligned} \tag{9}$$

□

Lemma 6. *Let sequence σ, ordered set of points $s_1, \ldots, s_m$, and number $q \in \{2, \ldots, m\}$ satisfy $\min_{j:j<q} V_\sigma(s_q; s_j) \leq 0$; then replacing s_q by some point $u_q \in \mathbb{M}$ for which $\min_{j:j<q} V_\sigma(u_q; s_j) \geq 0$ cannot increase the value of Φ_σ:*

$$\Phi_\sigma(s_1, \ldots, s_{q-1}, s_q, s_{q+1}, \ldots, s_m) - \Phi_\sigma(s_1, \ldots, s_{q-1}, u_q, s_{q+1}, \ldots, s_m) \geq \min_{j:j<q} V_\sigma(u_q; s_j) \geq 0.$$

Proof. Given the ordered set of points $s_1, \ldots, s_m$ we will use the notation S_i for the set of the first i points: $S_i = \{s_1, \ldots, s_i\}$. Using the definition of slack of a point with respect to a set this allows us to write $V_\sigma(s_i; S_{i-1})$ for $\min_{j:j<i} V_\sigma(s_i; s_j)$. To facilitate the exposition further we write $u_1, \ldots, u_m$ for $s_1, \ldots, s_{q-1}, u_q, s_{q+1}, \ldots, s_m$ (thus, $s_i = u_i$ for all $i \neq q$) and U_i for $\{u_1, \ldots, u_i\}$ (thus, $S_i = U_i$ for $i \leq q-1$).

By the definition of Φ we have

$$\begin{aligned} \Phi_\sigma(s_1, \ldots, s_m) - \Phi_\sigma(u_1, \ldots, u_m) = \beta_1(W_\sigma(s_1) - W_\sigma(u_1)) + \\ \textstyle\sum_{i=2}^{m} \beta_i \left(V_\sigma(u_i; U_{i-1}) - V_\sigma(s_i; S_{i-1})\right). \end{aligned} \tag{10}$$

Since $s_i = u_i$ for all $i \neq q$ and $S_i = U_i$ for all $i \leq q-1$, we have $V_\sigma(u_i; U_{i-1}) = V_\sigma(s_i; S_{i-1})$ $\forall i = 2, \ldots q-1$. Hence the right hand side of (10) simplifies to

$$\beta_q \left(V_\sigma(u_q; S_{q-1}) - V_\sigma(s_q; S_{q-1})\right) + \sum_{i=q+1}^{m} \beta_i \left(V_\sigma(s_i; U_{i-1}) - V_\sigma(s_i; S_{i-1})\right). \tag{11}$$

Consider the term in the summation for arbitrary $i \in \{q+1, \ldots, m\}$. Since for $i \geq q+1$, $U_{i-1} = (S_{i-1} \backslash s_q) \cup u_q$ we have $U_{i-1} \cup s_q = S_{i-1} \cup u_q$ $\forall i = q+1, \ldots, m$ and therefore

$$V_\sigma(s_i; U_{i-1}) \geq V_\sigma(s_i; U_{i-1} \cup s_q) = V_\sigma(s_i; S_{i-1} \cup u_q) = \min\{V_\sigma(s_i; S_{i-1}), V_\sigma(s_i; u_q)\}.$$

Hence, for all $i \in \{q+1, \ldots, m\}$

$$\begin{aligned} \beta_i \left(V_\sigma(s_i; U_{i-1}) - V_\sigma(s_i; S_{i-1})\right) &\geq \beta_i \left(\min\{V_\sigma(s_i; S_{i-1}), V_\sigma(s_i; u_q)\} - V_\sigma(s_i; S_{i-1})\right) \\ &= \beta_i \min\{0, V_\sigma(s_i; u_q) - V_\sigma(s_i; S_{i-1})\}. \end{aligned} \tag{12}$$

Using the triangle inequality of the slack below in the second inequality we get for all $i \in \{q+1, \ldots, m\}$

$$\begin{aligned} V_\sigma(s_i; S_{i-1}) &\leq V_\sigma(s_i; S_{q-1}) \\ &= \min_{j:j<q} \{V_\sigma(s_i; s_j)\} \\ &\leq \min_{j:j<q} \{V_\sigma(s_i; u_q) + V_\sigma(u_q; s_j)\} \\ &= V_\sigma(s_i; u_q) + V_\sigma(u_q; S_{q-1}) \end{aligned}$$

Inserting this bound in (12) and then using $V_\sigma(u_q; S_{q-1}) \geq 0$, a premise of the lemma, yields

$$\beta_i \left(V_\sigma(s_i; U_{i-1}) - V_\sigma(s_i; S_{i-1})\right) \geq \min\{0, -\beta_i V_\sigma(u_q; S_{q-1})\} = -\beta_i V_\sigma(u_q; S_{q-1}). \tag{13}$$

Combining (10), (11) and (13), and using $V_\sigma(s_q; S_{q-1}) \leq 0$ (premise of the lemma), we obtain

$$\begin{aligned}
\Phi_\sigma(u_1, \ldots, u_m) - \Phi_\sigma(s_1, \ldots, s_m) &\geq \beta_q V_\sigma(u_q; S_{q-1}) - \sum_{i=q+1}^{m} \beta_i V_\sigma(u_q; S_{q-1}) \\
&\geq (\beta_q - \sum_{i=q+1}^{m} \beta_i) V_\sigma(u_q; S_{q-1}) \\
&\geq V_\sigma(u_q; S_{q-1}),
\end{aligned}$$

where we use property (ii) of β_q for the last inequality. □

The next lemma says that the increase in the function Ψ over a phase is at least $(\lambda - 1)/(\lambda + 1)$ times the pseudo-cost of the phase.

Lemma 7. *Under the conditions of Theorem 1, we have for all phases $h \in \{1, \ldots, N-1\}$ that $\Psi_{\sigma_h} - \Psi_{\sigma_{h-1}} \geq \frac{\lambda-1}{\lambda+1} \nabla_h$.*

Proof. Take an arbitrary phase $h \in \{1, \ldots, N-1\}$. Let ρ_h denote the subsequence of requests served in this phase. In this proof we suppress subindices referring to the number of the phase: we write O, ρ, and ∇ for O_h, ρ_h, and ∇_h, respectively, and write W, Φ, Ψ, and V for $W_{\sigma_{h-1}}$, $\Phi_{\sigma_{h-1}}$, $\Psi_{\sigma_{h-1}}$, and $V_{\sigma_{h-1}}$, respectively. Similarly, we write W', Φ', Ψ', and V' for W_{σ_h}, Φ_{σ_h}, Ψ_{σ_h}, and V_{σ_h}, respectively.

Assume that Φ' attains its minimum at the points $s'_1, \ldots, s'_m$, i.e., $\Psi' = \Phi'(s'_1, \ldots, s'_m)$. For each s'_i there is a point s_i (not necessarily unique) and a path T_i from s_i to s'_i such that T_i serves ρ and

$$W'(s'_i) - W(s_i) = |T_i| \geq d(s_i, s'_i) \qquad (i = 1, \ldots, m). \tag{14}$$

Hence we may apply Lemma 2 to obtain

$$\Phi'(s'_1, \ldots, s'_m) - \Phi(s_1, \ldots, s_m) \geq (1 - 1/\lambda) \sum_{i=1}^{m} (W'(s'_i) - W(s_i)). \tag{15}$$

We distinguish two cases.

Case 1: *The paths $T_1, T_2, \ldots, T_m$ are pairwise independent.*

Remember in what follows that O is O_h and not $\mathcal{O}$. Since Online completed the phase with a work function step we have $A(O, \rho) > \gamma \textsc{Opt}(O, \rho)$. On the other hand, condition (3) in Theorem 1 implies $A(O, \rho) \leq c_1 \textsc{Opt}(O, \rho) + c_2 \sum_{i=1}^{m} |T_i|$. Combining both inequalities yields $\gamma \textsc{Opt}(O, \rho) = (c_1 + c_2) \textsc{Opt}(O, \rho) < c_1 \textsc{Opt}(O, \rho) + c_2 \sum_{i=1}^{m} |T_i|$. Hence,

$$\textsc{Opt}(O, \rho) < \sum_{i=1}^{m} |T_i| = \sum_{i=1}^{m} (W'(s'_i) - W(s_i)),$$

which together with (15), remembering that $\Psi' = \Phi'(s'_1, \ldots, s'_m)$, implies

$$\begin{aligned}
\Psi' - \Psi &= \Phi'(s'_1, \ldots, s'_m) - \Psi \\
&\geq \Phi'(s'_1, \ldots, s'_m) - \Phi(s_1, \ldots, s_m) \\
&> (1 - 1/\lambda) \textsc{Opt}(O, \rho).
\end{aligned}$$

By Lemma 5 we have $\textsc{Opt}(O,\rho) \geq \frac{\lambda}{\lambda+1}\nabla$, implying

$$\Psi' - \Psi > (1-1/\lambda)\frac{\lambda}{\lambda+1}\nabla = \frac{\lambda-1}{\lambda+1}\nabla.$$

Case 2: *There is a pair T_p, T_q $(p<q)$ of dependent paths.*
Using Definition 2, of dependency, we have $W'(s'_p) - W(s_p) + W'(s'_q) - W(s_q) = |T_p| + |T_q| \geq d(s_p, s'_q) + d(s_q, s'_p)$. Hence, $W'(s'_p) - W(s_q) \geq d(s_q, s'_p)$ or $W'(s'_q) - W(s_p) \geq d(s_p, s'_q)$. Assume that the latter is true. The other case is analogous although makes notation slightly more complicated. By this assumption and by (14) the series $s_1, \ldots, s_{q-1}, s_p, s_{q+1}, \ldots, s_m$ and $s'_1, \ldots, s'_{q-1}, s'_q, s'_{q+1}, \ldots, s'_m$ satisfy the premises of Lemma 2, whence

$$\begin{aligned} \Phi'(s'_1, \ldots, s'_m) - \Phi(s_1, \ldots, s_{q-1}, s_p, s_{q+1}, \ldots, s_m) & \\ \geq (1-1/\lambda)\left(W'(s'_q) - W(s_p) + \sum_{i=1, i\neq q}^{m} (W'(s'_i) - W(s_i))\right) & \\ \geq (1-1/\lambda)(W'(s'_i) - W(s_i)) \quad \forall\, i \neq q. & \end{aligned} \tag{16}$$

The last inequality is true since $W'(s'_i) - W(s_i) \geq d(s_i, s'_i) \geq 0$ $\forall i \neq q$, and $W'(s'_q) - W(s_p) \geq d(s_p, s'_q) \geq 0$.

If $W'(s'_i) - W(s_i) \geq (\lambda/(\lambda+1))\nabla$ for some $i \neq q$, then, again remembering that $\Psi' = \Phi'(s'_1, \ldots, s'_m)$, (16) immediately implies

$$\begin{aligned} \Psi' - \Psi &\geq \Phi'(s'_1, \ldots, s'_m) - \Phi(s_1, \ldots, s_{q-1}, s_p, s_{q+1}, \ldots, s_m) \\ &\geq (1-1/\lambda)(W'(s'_i) - W(s_i)) \geq \frac{\lambda-1}{\lambda+1}\nabla. \end{aligned}$$

Hence, from now on we concentrate on the situation

$$W'(s'_i) - W(s_i) \leq (\lambda/(\lambda+1))\nabla \quad \forall\, i \neq q. \tag{17}$$

As in the proof of the previous lemma we define $S_i = \{s_1, \ldots, s_i\}$. Notice that $V(s_p; S_{q-1}) \leq V(s_p; s_p) = 0$, since we assumed $p < q$. On the other hand, we will prove below that $V(O; S_{q-1}) \geq 0$. Lemma 6 then tells us that Φ will reduce if we replace s_p (on position q) by point O. More precisely,

$$\Phi(s_1, \ldots, s_{q-1}, s_p, s_{q+1}, \ldots, s_m) - \Phi(s_1, \ldots, s_{q-1}, O, s_{q+1}, \ldots, s_m) \geq V(O; S_{q-1}). \tag{18}$$

Now we will show that $V(O; S_{q-1}) \geq 0$. Let $k \in \{1, \ldots, q-1\}$ be such that $V(O; S_{q-1}) = V(O; s_k)$. The bound below follows from Lemma 4 (first inequality), the triangle inequality (second inequality), and $W'(s'_k) - W(s_k) \geq d(s_k, s'_k)$ (third inequality):

$$\begin{aligned} V(O; S_{q-1}) &= V(O; s_k) \\ &= W(s_k) + \tfrac{1}{\lambda} d(O, s_k) - W(O) \\ &= W'(s'_k) + \tfrac{1}{\lambda} d(O, s'_k) - W(O) + (W(s_k) - W'(s'_k)) + \tfrac{1}{\lambda}(d(O, s_k) - d(O, s'_k)) \\ &\geq \nabla + (W(s_k) - W'(s'_k)) + \tfrac{1}{\lambda}(d(O, s_k) - d(O, s'_k)) \\ &\geq \nabla + (W(s_k) - W'(s'_k)) - \tfrac{1}{\lambda} d(s_k, s'_k) \\ &\geq \nabla - (1 + \tfrac{1}{\lambda})(W'(s'_k) - W(s_k)) \end{aligned} \tag{19}$$

The non-negativity follows directly, using (17):

$$V(O; S_{q-1}) \geq \nabla - (1 + 1/\lambda)\lambda/(\lambda+1)\nabla = \nabla - \nabla = 0.$$

Finally we combine (16)– (19). Remember again that $\Psi' = \Phi'(s'_1, \ldots, s'_m)$ to see the first inequality below. The second inequality is derived by applying (16), (18) and (19), and the third inequality follows from (17).

$$\begin{aligned}
\Psi' - \Psi &\geq \Phi'(s'_1, \ldots, s'_m) - \Phi(s_1, \ldots, s_{q-1}, O, s_{q+1}, \ldots, s_m) \\
&= \Phi'(s'_1, \ldots, s'_m) - \Phi(s_1, \ldots, s_{q-1}, s_p, s_{q+1}, \ldots, s_m) + \\
&\qquad \Phi(s_1, \ldots, s_{q-1}, s_p, s_{q+1}, \ldots, s_m) - \Phi(s_1, \ldots, s_{q-1}, O, s_{q+1}, \ldots, s_m) \\
&\geq (1 - 1/\lambda)(W'(s'_k) - W(s_k)) + \nabla - (1 + 1/\lambda)(W'(s'_k) - W(s_k)) \\
&= \nabla - (2/\lambda)(W'(s'_k) - W(s_k)) \\
&\geq \nabla - (2/(\lambda + 1))\nabla \\
&= ((\lambda - 1)/(\lambda + 1))\nabla.
\end{aligned}$$

□

Accumulating over the consecutive phases we obtain lower bound (4) on $\textsc{Opt}(\mathcal{O}, \sigma_{N-1})$.

Lemma 8.

$$\textsc{Opt}(\mathcal{O}, \sigma_{N-1}) \geq \frac{\lambda - 1}{\beta_1(\lambda + 1)} \sum_{h=1}^{N-1} \nabla_h,$$

where $\beta_1 = 2\left(\frac{2\lambda}{\lambda - 1}\right)^{m-2}$.

Proof. Combining Lemmas 1, 3, and 7 yields

$$\beta_1 \textsc{Opt}(\mathcal{O}, \sigma_{N-1}) \geq \Psi_{\sigma_{N-1}} = \Psi_{\sigma_{N-1}} - \Psi_\epsilon = \sum_{h=1}^{N-1} (\Psi_{\sigma_h} - \Psi_{\sigma_{h-1}}) \geq \frac{\lambda - 1}{\lambda + 1} \sum_{h=1}^{N-1} \nabla_h.$$

□

2.3 An upper bound

We start the derivation of the upper bound (5) with two preliminary lemmas.

Lemma 9. *For every* $h \in \{1, \ldots, N\}$ *and* $s \in \mathbb{M}$ *we have* $W_{\sigma_{h-1}}(s) \geq W_{\sigma_{h-1}}(O_h) - (1/\lambda)d(O_h, s)$.

Proof. The lemma clearly holds for $h = 1$. If $h \geq 2$ then we know that point O_h is the endpoint of the λ-WFA-move in the previous phase. Hence by definition of ONLINE,

$$W_{\sigma_{h-1}}(O_h) + (1/\lambda)d(O_h, O_{h-1}) \leq W_{\sigma_{h-1}}(s) + (1/\lambda)d(s, O_{h-1}),$$

for every point $s \in \mathbb{M}$. Using the triangle inequality we obtain

$$W_{\sigma_{h-1}}(s) \geq W_{\sigma_{h-1}}(O_h) + (1/\lambda)(d(O_h, O_{h-1}) - d(s, O_{h-1})) \geq W_{\sigma_{h-1}}(O_h) - (1/\lambda)d(O_h, s),$$

for every point $s \in \mathbb{M}$. □

Lemma 10. *Let* ρ_h *be the sequence of requests served in phase* h*; then*

$$\textsc{Opt}(O_h, \rho_h) \leq \frac{2\lambda}{\lambda - 1}\nabla_h + d(O_h, O_{h+1}).$$

Proof. We adopt the shorthand notation from the proof of Lemma 7: $\rho = \rho_h$, $\nabla = \nabla_h$, $W = W_{\sigma_{h-1}}, W' = W_{\sigma_h}, O = O_h$, and $O' = O_{h+1}$.

Consider a path T that starts in $\mathcal{O}$, serves the sequence σ_h, ends in O' and has minimal length, i.e., has length $W'(O')$. Let s be the point in which this path starts phase h. Hence, the length of this path within phase h is $W'(O') - W(s)$. Obviously,

$$d(O', s) \leq W'(O') - W(s) \tag{20}$$

Clearly, the length of the path that goes straight from O to s and then proceeds to O', following T, is an upper bound on $\textsc{Opt}(O, \rho)$:

$$\begin{aligned} \textsc{Opt}(O, \rho) &\leq d(O, s) + W'(O') - W(s) \\ &\leq d(O, O') + d(O', s) + W'(O') - W(s) \\ &\leq d(O, O') + 2(W'(O') - W(s)), \end{aligned} \tag{21}$$

where (20) is used for the last inequality. Further, we know from Lemma 9 that

$$W(s) \geq W(O) - (1/\lambda)d(O, s).$$

Combining this with $\nabla = W'(O') - W(O) + (1/\lambda)d(O, O')$ (Definition 3) yields

$$\begin{aligned} W'(O') - W(s) &\leq \nabla - (1/\lambda)d(O, O') + (1/\lambda)d(O, s) \\ &\leq \nabla + (1/\lambda)d(O', s)) \\ &\leq \nabla + (1/\lambda)(W'(O') - W(s)), \end{aligned}$$

where (20) is used for the last inequality. Hence,

$$W'(O') - W(s) \leq (\lambda/(\lambda - 1))\nabla. \tag{22}$$

Combining (21) and (22) completes the proof. □

In deriving the upper bound on Online's cost we first bound in Lemma 11 the costs in each phase h which ends with a work function move, i.e., $h \in \{1, \ldots, N-1\}$. Denote the cost of the A-moves in each such phase h by $C_h^A = A(O_h, r_{i_{h-1}+1}, \ldots, r_{i_h-1})$. The last request r_{i_h} in such a phase h is served by a work function step instead, which yields a cost that we denote by C_h^W. The total cost in the phase is $C_h = C_h^A + C_h^W$.

Lemma 11. *For all phases $h \in \{1, \ldots, N-1\}$ we have*

$$C_h \leq 4\gamma(\lambda/(\lambda - 1))\nabla_h + (2\gamma + 1)d(O_h, O_{h+1}).$$

Proof. Consider any phase $h \in \{1, \ldots, N-1\}$. Again, we use the shorthand notation of the previous lemma. The cost C_h^W of the work function step at the end of phase h is bounded from above by $C_h^A + d(O, O')$. Moreover, by definition of Online, we have $C_h^A = A(O, r_{i_{h-1}+1}, \ldots, r_{i_h-1}) \leq \gamma\textsc{Opt}(O, r_{i_{h-1}+1}, \ldots, r_{i_h-1}) \leq \gamma\textsc{Opt}(O, \rho)$. Hence,

$$C_h \leq 2C_h^A + d(O, O') \leq 2\gamma\textsc{Opt}(O, \rho) + d(O, O'). \tag{23}$$

Applying Lemma 10 yields

$$C_h \leq 2\gamma\textsc{Opt}(O, \rho) + d(O, O') \leq 2\gamma\left(\frac{2\lambda}{\lambda - 1}\nabla + d(O, O')\right) + d(O, O').$$

□

Lemma 12. *The cost C_N of the Nth (and last) phase is at most $(2\gamma\lambda/(\lambda-1))\textsc{Opt}(\mathcal{O},\sigma)$.*

Proof. Remember that we assumed that the last phase ends before a work function step is made. If there are no requests in the last phase, then $C_N = 0$ and the lemma is trivially true. Otherwise, $C_N = C_N^A$, and by definition of Online

$$C_N^A \le \gamma\textsc{Opt}(O_N,\rho), \tag{24}$$

where ρ is the request sequence of the last phase. The length of the path that starts in O_N, goes straight to $\mathcal{O}$, and then optimally serves the whole sequence σ is an upper bound on $\textsc{Opt}(O_N,\rho)$:

$$\begin{aligned}\textsc{Opt}(O_N,\rho) &\le d(O_N,\mathcal{O}) + \textsc{Opt}(\mathcal{O},\sigma)\\ &\le W_{\sigma_{N-1}}(O_N) + \textsc{Opt}(\mathcal{O},\sigma).\end{aligned} \tag{25}$$

We shall bound $W_{\sigma_{N-1}}(O_N)$. Again, we use Lemma 9 and apply the triangle inequality. For any point $s \in \mathbb{M}$

$$\begin{aligned}W_{\sigma_{N-1}}(s) &\ge W_{\sigma_{N-1}}(O_N) - (1/\lambda)d(s,O_N)\\ &\ge W_{\sigma_{N-1}}(O_N) - (1/\lambda)(d(s,\mathcal{O}) + d(\mathcal{O},O_N))\\ &\ge W_{\sigma_{N-1}}(O_N) - (1/\lambda)(W_{\sigma_{N-1}}(s) + W_{\sigma_{N-1}}(O_N))\\ &= ((\lambda-1)/\lambda)W_{\sigma_{N-1}}(O_N) - (1/\lambda)W_{\sigma_{N-1}}(s),\end{aligned}$$

which implies

$$W_{\sigma_{N-1}}(O_N) \le ((\lambda+1)/(\lambda-1))W_{\sigma_{N-1}}(s)\ \forall s \in \mathbb{M}. \tag{26}$$

Choosing s as the endpoint of an optimal path starting at $\mathcal{O}$ and serving all requests, i.e., all of σ_N, we have $W_{\sigma_{N-1}}(s) \le W_{\sigma_N}(s) = \textsc{Opt}(\mathcal{O},\sigma_N)$. We combine this with (24), (25), and (26) to obtain

$$\begin{aligned}C_N = C_N^A &\le \gamma\textsc{Opt}(O_N,\rho)\\ &\le \gamma(W_{\sigma_{N-1}}(O_N) + \textsc{Opt}(\mathcal{O},\sigma))\\ &\le \gamma(((\lambda+1)/(\lambda-1))\textsc{Opt}(\mathcal{O},\sigma) + \textsc{Opt}(\mathcal{O},\sigma))\\ &= (2\gamma\lambda/(\lambda-1))\textsc{Opt}(\mathcal{O},\sigma).\end{aligned}$$

□

Lemma 13. $\sum_{h=1}^{N-1} d(O_h,O_{h+1}) \le \lambda\sum_{h=1}^{N-1}\nabla_h - \lambda\textsc{Opt}(\mathcal{O},\sigma_{N-1})$.

Proof. For each phase $h \in \{1,\ldots,N-1\}$, by the definition of pseudo-cost,

$$(1/\lambda)d(O_h,O_{h+1}) = \nabla_h + W_{\sigma_{h-1}}(O_h) - W_{\sigma_h}(O_{h+1}). \tag{27}$$

Hence, using as before $\sigma_0 = \epsilon$, the empty string, we have

$$\begin{aligned}\sum_{h=1}^{N-1} d(O_h,O_{h+1}) &= \lambda\sum_{h=1}^{N-1}\nabla_h + \lambda\sum_{h=1}^{N-1}(W_{\sigma_{h-1}}(O_h) - W_{\sigma_h}(O_{h+1}))\\ &= \lambda\sum_{h=1}^{N-1}\nabla_h + \lambda W_\epsilon(\mathcal{O}) - \lambda W_{\sigma_{N-1}}(O_N).\end{aligned}$$

The proof is completed by using $W_\epsilon(\mathcal{O}) = 0$ and $W_{\sigma_{N-1}}(O_N) \ge \textsc{Opt}(\mathcal{O},\sigma_{N-1})$. □

The above results combine to give the upper bound (5) on $\textsc{Online}(\mathcal{O},\sigma)$ in the proof of Theorem 1:

Lemma 14.

$$\textsc{Online}(\mathcal{O},\sigma) \leq \left(\frac{4\gamma\lambda}{\lambda-1} + (2\gamma+1)\lambda\right)\sum_{h=1}^{N-1}\nabla_h - \lambda(2\gamma+1)\textsc{Opt}(\mathcal{O},\sigma_{N-1}) + \frac{2\gamma\lambda}{\lambda-1}\textsc{Opt}(\mathcal{O},\sigma).$$

Proof. First we combine Lemmas 11 and 12:

$$\begin{aligned}\textsc{Online}(\mathcal{O},\sigma) &= \sum_{h=1}^{N-1} C_h + C_N \\ &\leq \sum_{h=1}^{N-1}\left(\tfrac{4\gamma\lambda}{\lambda-1}\nabla_h + (2\gamma+1)d(O_h,O_{h+1})\right) + \tfrac{2\gamma\lambda}{\lambda-1}\textsc{Opt}(\mathcal{O},\sigma) \\ &= \tfrac{4\gamma\lambda}{\lambda-1}\sum_{h=1}^{N-1}\nabla_h + (2\gamma+1)\sum_{h=1}^{N-1} d(O_h,O_{h+1}) + \tfrac{2\gamma\lambda}{\lambda-1}\textsc{Opt}(\mathcal{O},\sigma)\end{aligned}$$

Applying the inequality of Lemma 13 completes the proof. □

3 The generalized two-server problem

In the generalized two-server problem we are given a server, whom we will call the $\mathbb{X}$-server, moving in a metric space $\mathbb{X}$, starting from point $x_0 \in \mathbb{X}$, and a server, the $\mathbb{Y}$-server, moving in a metric space $\mathbb{Y}$, starting in $y_0 \in \mathbb{Y}$. Requests $(x,y) \in \mathbb{X}\times\mathbb{Y}$ are presented on-line one by one and are served by moving one of the servers to the corresponding point in its metric space. The objective is to minimize the sum of the distances travelled by the two servers. This problem can easily be modelled as a metrical service system: There is one server moving in the product space $\mathbb{X}\times\mathbb{Y}$ and any pair $(x,y) \in \mathbb{X}\times\mathbb{Y}$ defines a request $r = \{\{x\}\times\mathbb{Y}\} \cup \{\mathbb{X}\times\{y\}\}$. For any two points (x_1,y_1) and (x_2,y_2) in $\mathbb{X}\times\mathbb{Y}$ we define $d((x_1,y_1),(x_2,y_2)) = d^{\mathbb{X}}(x_1,x_2) + d^{\mathbb{Y}}(y_1,y_2)$, where $d^{\mathbb{X}}$ and $d^{\mathbb{Y}}$ are the distance functions of the metric spaces $\mathbb{X}$ and $\mathbb{Y}$.

Lemma 15. *The work function and λ-WFA are well-defined for the generalized two-server problem.*

Proof. Let $\sigma = (x_0,y_0),\ldots,(x_n,y_n)$ be a request sequence for some generalized two-server problem, where we assume, without loss of generality, that the first request is given at the origin $\mathcal{O} = (x_0,y_0)$. Consider an arbitrary path serving σ and ending at some point (x,y). If at some request both servers move, then the move of the server that does not serve the request can be postponed to the next request at no extra cost. Hence, there exists a path ending at (x,y) of at most the same length on which only one server is moved at each request, possibly with the exception of the last request. Since the number of paths that move only one server with each request is 2^n there exists a path ending at (x,y) that has minimal length, whence the work function is well-defined. More precisely, the endpoint of any such path is in the set $S = \{x_n\}\times\{y_0,\ldots,y_{n-1}\}\cup\{x_0,\ldots,x_{n-1}\}\times\{y_n\}$. Hence $W_{\mathcal{O},\sigma}(x,y) = W_{\mathcal{O},\sigma}(s) + d(s,(x,y))$ for some $s \in S$ implying $\mathrm{supp}(W_{\mathcal{O},\sigma}) \subseteq S$. Since S has only a finite number $(2n)$ of elements the generalized work function algorithm λ-WFA is well-defined. □

Thus, a part of the conditions of Theorem 1 is satisfied. We still need to define an algorithm A that satisfies (3). That means, we must show that there are constants c_1, c_2 and m such that, for any instance, the algorithm's cost is at most c_1 times the cost of the optimal solution plus c_2 times the total cost of m independent solutions. As such an algorithm we have chosen the simple Balance algorithm, which keeps track of the costs incurred by the $\mathbb{X}$- and $\mathbb{Y}$-server and tries to balance their

costs. The BALANCE algorithm is not (constant-)competitive for our problem as it is known not to be competitive for the two-server problem [14]. However, we will show that it satisfies condition (3) with $c_1 = 2$, $c_2 = 4$ and $m = 3$.

We define BALANCE starting in (x_0, y_0) and serving the request sequence $\sigma = (x_1, y_1), (x_2, y_2), \ldots$. Let $B_j^{\mathbb{X}}$ and $B_j^{\mathbb{Y}}$ be the total costs made by, respectively, the $\mathbb{X}$- and the $\mathbb{Y}$-server after the j-th request has been served and let $B_j := B_j^{\mathbb{X}} + B_j^{\mathbb{Y}}$. We denote the positions of the servers after serving the j-th request by $(\hat{x}_j, \hat{y}_j)$.

BALANCE

If $B_j^{\mathbb{X}} + d^{\mathbb{X}}(\hat{x}_j, x_{j+1}) \leq B_j^{\mathbb{Y}} + d^{\mathbb{Y}}(\hat{y}_j, y_{j+1})$, then move the $\mathbb{X}$-server to x_{j+1}. Else move the $\mathbb{Y}$-server to y_{j+1}.

The following lemmas give an upper bound on the cost of BALANCE. We denote by $P_{ij}^{\mathbb{X}}$ $(0 \leq i < j)$ the length of the path $x_i, x_{i+1}, \ldots, x_j$. We define $P_{ij}^{\mathbb{Y}}$ $(0 \leq i < j)$ in a similar way.

Lemma 16. *If* BALANCE *is applied to the request sequence* $(x_1, y_1), \ldots, (x_j, y_j)$ *starting from* (x_0, y_0)*, then* $B_j \leq 2\min\{P_{0j}^{\mathbb{X}}, P_{0j}^{\mathbb{Y}}\}$ $\forall j \geq 0$.

Proof. Clearly, $B_j^{\mathbb{X}} \leq P_{0j}^{\mathbb{X}}$ and $B_j^{\mathbb{Y}} \leq P_{0j}^{\mathbb{Y}}$. Let the i-th request (x_i, y_i) be the last request served by the $\mathbb{X}$-server. Then, $B_j^{\mathbb{X}} = B_i^{\mathbb{X}} \leq B_{i-1}^{\mathbb{Y}} + d^{\mathbb{Y}}(\hat{y}_{i-1}, y_i) \leq P_{0i}^{\mathbb{Y}} \leq P_{0j}^{\mathbb{Y}}$. Hence, $B_j^{\mathbb{X}} \leq \min\{P_{0j}^{\mathbb{X}}, P_{0j}^{\mathbb{Y}}\}$. Similarly it is shown that $B_j^{\mathbb{Y}} \leq \min\{P_{0j}^{\mathbb{X}}, P_{0j}^{\mathbb{Y}}\}$. □

Let $\text{OPT}((x_0, y_0), \sigma)$ denote the cost of an optimal path serving sequence σ and starting from (x_0, y_0).

Lemma 17. *If* BALANCE *is applied to the request sequence* $\sigma = (x_1, y_1), \ldots, (x_j, y_j)$ *starting from* (x_0, y_0)*, then* $B_j \leq 2\text{OPT}((x_0, y_0), \sigma) + 4\min\{P_{1j}^{\mathbb{X}}, P_{1j}^{\mathbb{Y}}\}$.

Proof. If the optimal path uses only one server then $\text{OPT}((x_0, y_0), \sigma) = \min\{P_{0j}^{\mathbb{X}}, P_{0j}^{\mathbb{Y}}\} \geq B_j/2$ by Lemma 16. So assume the optimal path uses both servers and assume without loss of generality $P_{1j}^{\mathbb{X}} \leq P_{1j}^{\mathbb{Y}}$. Since the $\mathbb{X}$-server of the optimal path serves at least one request, say x, we have $\text{OPT}((x_0, y_0), \sigma) \geq d^{\mathbb{X}}(x_0, x) \geq d^{\mathbb{X}}(x_0, x_1) - d^{\mathbb{X}}(x_1, x) \geq P_{01}^{\mathbb{X}} - P_{1j}^{\mathbb{X}}$. Again using Lemma 16 yields $B_j \leq 2P_{0j}^{\mathbb{X}} = 2P_{01}^{\mathbb{X}} + 2P_{1j}^{\mathbb{X}} \leq 2\text{OPT}((x_0, y_0), \sigma) + 4P_{1j}^{\mathbb{X}} = 2\text{OPT}((x_0, y_0), \sigma) + 4\min\{P_{1j}^{\mathbb{X}}, P_{1j}^{\mathbb{Y}}\}$. □

Lemma 18. *If* T_1*,* T_2 *and* T_3 *are pairwise independent paths each serving the request sequence* $\sigma = (x_1, y_1), \ldots, (x_j, y_j)$*, then* $|T_1| + |T_2| + |T_3| \geq \min\{P_{1j}^{\mathbb{X}}, P_{1j}^{\mathbb{Y}}\}$.

Proof. For each $i \in \{1, 2, 3\}$ let $T_i^{\mathbb{X}}$ and $T_i^{\mathbb{Y}}$ be the projections of path T_i on, respectively, metric space $\mathbb{X}$ and metric space $\mathbb{Y}$, and let $p_i = (p_i^{\mathbb{X}}, p_i^{\mathbb{Y}})$, $q_i = (q_i^{\mathbb{X}}, q_i^{\mathbb{Y}})$ be, respectively, the beginning and ending point of T_i.

First we claim that for any pair T_1, T_2 and T_2, T_3 and T_1, T_3 there can be no two requests such that one request is served by the two corresponding $\mathbb{X}$-servers and the other request is served by the two $\mathbb{Y}$-servers. To see this assume on the contrary that paths $T_1^{\mathbb{X}}$ and $T_2^{\mathbb{X}}$ share a point $x \in \mathbb{X}$ and paths $T_1^{\mathbb{Y}}$ and $T_2^{\mathbb{Y}}$ share a point $y \in \mathbb{Y}$. Then for $T_1^{\mathbb{X}}$ and $T_2^{\mathbb{X}}$ we have, applying the triangle inequality, $|T_1^{\mathbb{X}}| + |T_2^{\mathbb{X}}| \geq d^{\mathbb{X}}(p_1^{\mathbb{X}}, x) + d^{\mathbb{X}}(x, q_1^{\mathbb{X}}) + d^{\mathbb{X}}(p_2^{\mathbb{X}}, x) + d^{\mathbb{X}}(x, q_2^{\mathbb{X}}) \geq d^{\mathbb{X}}(p_1^{\mathbb{X}}, q_2^{\mathbb{X}}) + d^{\mathbb{X}}(p_2^{\mathbb{X}}, q_1^{\mathbb{X}})$. Similarly, $|T_1^{\mathbb{Y}}| + |T_2^{\mathbb{Y}}| \geq d^{\mathbb{Y}}(p_1^{\mathbb{Y}}, q_2^{\mathbb{Y}}) + d^{\mathbb{Y}}(p_2^{\mathbb{Y}}, q_1^{\mathbb{Y}})$. Hence,

$$\begin{aligned} |T_1| + |T_2| &= |T_1^{\mathbb{X}}| + |T_1^{\mathbb{Y}}| + |T_2^{\mathbb{X}}| + |T_2^{\mathbb{Y}}| \\ &\geq d^{\mathbb{X}}(p_1^{\mathbb{X}}, q_2^{\mathbb{X}}) + d^{\mathbb{X}}(p_2^{\mathbb{X}}, q_1^{\mathbb{X}}) + d^{\mathbb{Y}}(p_1^{\mathbb{Y}}, q_2^{\mathbb{Y}}) + d^{\mathbb{Y}}(p_2^{\mathbb{Y}}, q_1^{\mathbb{Y}}) \\ &= d(p_1, q_2) + d(p_2, q_1), \end{aligned}$$

which would imply that T_1 and T_2 are dependent according to Definition 2, contrary to the premise of the lemma. Thus, in particular it is impossible that two requests exist such that one is served by the $\mathbb{X}$-servers of T_1 and T_2 and the other is served by the $\mathbb{Y}$-servers of T_1 and T_2. The same applies to the pairs T_2, T_3 and T_1, T_3, which proves our claim.

Second, we claim that we may assume without loss of generality that the $\mathbb{X}$-server of T_1 shares no request with the $\mathbb{X}$-server of T_2 and shares no request with the $\mathbb{X}$-server of T_3. To see this consider a graph on three vertices t_1, t_2 and t_3 and at most one edge between any two vertices. More specifically, between t_i and t_j there is a red edge if the $\mathbb{X}$-servers of T_i and T_j share a request, a blue one if the $\mathbb{Y}$-servers of T_i and T_j share a request, and no edge if neither the $\mathbb{X}$- nor $\mathbb{Y}$-servers of T_i and T_j share a request. Above we argued that for no pair of paths T_i and T_j do both their $\mathbb{X}$-servers and their $\mathbb{Y}$-servers share a request, and hence between any pair of vertices t_i and t_j there cannot be both a blue and a red edge. Hence there are at most three edges in this graph and there must be a vertex that is not incident to both a blue and a red edge. Without loss of generality assume that t_1 is not incident to any red edge. Translated back into the path language, this says that the $\mathbb{X}$-server of T_1 shares no request with the $\mathbb{X}$-server of T_2 and shares no request with the $\mathbb{X}$-server of T_3, which was indeed our second claim.

If the $\mathbb{Y}$-server of T_1 serves all requests then the lemma obviously holds. Otherwise, some request $r = (x_i, y_i)$ is served by the $\mathbb{X}$-server of T_1. By our second claim request r must be served by the $\mathbb{Y}$-servers of T_2 and T_3. But then, since the $\mathbb{Y}$-servers of T_2 and T_3 share a request, the $\mathbb{X}$-servers of T_2 and T_3 are not allowed to share *any* request by our first claim. Combined with our second claim, we are in the situation that for none of the pairs T_1, T_2 and T_2, T_3 and T_1, T_3 is there a request that is served by the two corresponding $\mathbb{X}$-servers. In other words, on the three tours each request is served by at most one of the three $\mathbb{X}$-servers, and therefore by at least two of the three $\mathbb{Y}$-servers. This implies that for each two consecutive requests at least one of the three $\mathbb{Y}$-servers serves both requests. Hence, $|T_1| + |T_2| + |T_3| \geq P^{\mathbb{Y}}_{1j}$. □

The combination of the previous lemmas and Theorem 1 shows that the generalized two-server problem allows for a constant competitive ratio.

Lemma 19. *Algorithm* Online(Balance, γ, λ) *with* $\gamma = 6$ *and* $\lambda = 3$ *is 2160-competitive for the generalized two-server problem.*

Proof. By Lemma 15 the work function and λ-WFA are well-defined. Combining Lemma 17 and Lemma 18 we see that Balance satisfies premise (3) of Theorem 1 with $c_1 = 2, c_2 = 4$ and $m = 3$. Hence, Theorem 1 implies that Online(Balance, c_1+c_2, m) is $15(c_1+c_2)m2^m = 2160$-competitive for the generalized two-server problem. □

Inserting the values $\gamma = 6$ and $\lambda = 3$ directly into bound (6) we obtain the lower competitive ratio of 879.

4 Concluding remarks

Unfortunately, Theorem 1 does not provide a competitive algorithm for the generalized k-server problem for $k \geq 3$ just as easily as for $k = 2$. The question as to whether a competitive algorithm exists for $k \geq 3$ remains unresolved. We believe that the generalized work function algorithm is competitive for the generalized k-server problem for any $k \geq 1$ and $\lambda > 1$.

A problem to which Theorem 1 applies directly is the k-point request problem. In this metrical service system the set of possible requests consists of all k-element subsets of the metric space. For any set of $k+1$ or more paths that serve the same request sequence there must be two paths that share a point and are therefore dependent. So for this problem the conditions of Theorem 1 are satisfied for any algorithm A and $\gamma \geq 0$ if $m = k+1$. Hence, the generalized work function algorithm itself must be competitive. We can derive a bound of $(\lambda+1)(2\lambda/(\lambda-1))^k - \lambda$ on its competitive ratio by substituting $\gamma = 0$ in (6). Burley [3] showed a slightly better bound $((\lambda+1)(2\lambda/(\lambda-1))^{k-1} - \lambda)$ and proved that this is tight. Our proof simplifies a lot if we restrict the analysis to the k-point request problem and let ONLINE be the generalized work function algorithm. In that case Lemma 11 and Lemma 12 are redundant. Case 2 of Lemma 7 simplifies a lot and case 1 becomes redundant which allows for a better choice of the numbers β_i. The modification of our proof to the k-point request problem gives exactly Burley's proof.

Condition (3) of Theorem 1 is imposed on an algorithm A. It would be more interesting to give a (non-trivial) sufficient condition for a metrical service system to have a finite competitive ratio. For example, Friedman and Linial [10] showed that a competitive algorithm exists if each request is a convex set in $\mathbb{R}^2$. Even more interesting would be to find a sufficient condition on the system for competitiveness of the generalized work function algorithm.

We conclude with the NP-hardness proof of the off-line version of the generalized k-server problem. The proof is straightforward from the *exact 3-cover* problem. We first remind the reader to the definition of the latter problem. For NP-Completeness of this problem we refer to [11].

Exact 3-cover:
An instance is given by a set Z with $|Z| = 3q$ and a collection C of 3-element subsets of Z. The question is whether C contains an exact cover for Z, i.e., a subcollection $C' \subseteq C$ such that every element of Z occurs in exactly one member of C'.

Theorem 2. *The (offline) generalized k-server problem is NP-hard.*

Proof. Take any instance I of the exact 3-cover problem (with the notation as defined above). We define an instance I' of the generalized k-server problem. For each 3-element subset $c^i \in C = \{c^1, c^2, \ldots, c^{|C|}\}$ we define a metric space with its server. The $|C|$ metric spaces are identical and consist of an origin O^i, one point P^i at distance 1 from the origin, and one point Q^i at distance $q+1$ from the origin. For each element $z_j \in Z$ we define one request $(r_j^1, \ldots, r_j^i, \ldots, r_j^{|C|})$, which is a $|C|$-tuple of points, one in each metric space. If $z_j \in c^i$ then $r_j^i = P^i$ and $r_j^i = Q^i$ otherwise. A small example is included below.

If there exists an exact 3-cover C' then we move all servers that correspond to a triple $c^i \in C'$ simultaneously to their point P^i, where they will stay during the whole sequence. Every request will be covered by exactly one server. The cost of this solution is q. On the other hand, if there is a solution with cost at most q, then no server moves to a point Q^i, since otherwise the cost would be at least $q+1$. Each server serves at most three requests. To serve all $3q$ requests at least q servers have to move to their point P^i. Since we assumed the cost to be at most q, exactly q servers must move. This set of q servers induces an exact 3-cover.

To illustrate the reduction we give a small example. Let $Z = \{1,2,3,4,5,6\}$ and let C consist of the subsets $c^1 = \{1,2,3\}$, $c^2 = \{4,5,6\}$, $c^3 = \{2,3,4\}$, and $c^4 = \{3,4,5\}$. Then the requests will be $r_1 = (P^1, Q^2, Q^3, Q^4), r_2 = (P^1, Q^2, P^3, Q^4), r_3 = (P^1, Q^2, P^3, P^4), r_4 = (Q^1, P^2, P^3, P^4), r_5 = (Q^1, P^2, Q^3, P^4)$, and $r_6 = (Q^1, P^2, Q^3, Q^4)$. The optimal solution is to move servers 1 and 2 (corresponding with c^1 and c^2) to, respectively, P^1 and P^2 and has cost 2. □

Acknowledgement We acknowledge many detailed comments of anonymous referees, from which we benefitted greatly in writing the present version.

References

1. Yair Bartal and Elias Koutsoupias, *On the competitive ratio of the work function algorithm for the k-server problem*, Theoretical Computer Science **324** (2004), 337–345.
2. Allan Borodin, Nathan Linial, and Michael Saks, *An optimal online algorithm for metrical task systems*, Journal of the ACM **39** (1992), 745–763.
3. William R. Burley, *Traversing layered graphs using the work function algorithm*, Journal of Algorithms **20** (1996), 479–511.
4. Marek Chrobak, Howard Karloff, Tom H. Payne, and Sundar Vishwanathan, *New results on server problems*, SIAM Journal on Discrete Mathematics **4** (1991), 172–181.
5. Marek Chrobak and Lawrence L. Larmore, *An optimal online algorithm for k servers on trees*, SIAM Journal on Computing **20** (1991), 144–148.
6. ———, *Metrical service systems: Deterministic strategies*, Tech. Report UCR-CS-93-1, Department of Computer Science, University of California at Riverside, 1992.
7. Marek Chrobak and Jiří Sgall, *The weighted 2-server problem*, Theoretical Computer Science **324** (2004), 289–312.
8. Esteban Feuerstein, Steve Seiden, and Alejandro Strejlevich de Loma, *The related server problem*, Unpublished manuscript, 1999.
9. Amos Fiat and Moty Ricklin, *Competitive algorithms for the weighted server problem*, Theoretical Computer Science **130** (1994), 85–99.
10. Joel Friedman and Nathan Linial, *On convex body chasing*, Discrete Computational Geometry **9** (1993), 293–321.
11. Richard M. Karp, *Reducibility among combinatorial problems*, Complexity of computer computations (R.E. Miller and J.W. Thatcher, eds.), Plenum Press, New York, 1972, pp. 85–103.
12. Elias Koutsoupias and Christos Papadimitriou, *On the k-server conjecture*, Journal of the ACM **42** (1995), 971–983.
13. Elias Koutsoupias and David Scot Taylor, *The cnn problem and other k-server variants*, Theoretical Computer Science **324** (2004), 347–359.
14. Lyle A. McGeoch Mark S. Manasse and Daniel D. Sleator, *Competitive algorithms for server problems*, Journal of Algorithms **11** (1990), 208–230.
15. René Sitters, Leen Stougie, and Willem de Paepe, *A competitive algorithm for the general 2-server problem.*, ICALP03, Lecture Notes in Computer Science, vol. 2719, Springer, 2003, pp. 624–636.