Time-Optimal k-Server

Fabian Frei
CSAIL, MIT, Cambridge, MA, USA

Dennis Komm
Department of Computer Science, ETH Zürich, Switzerland

Moritz Stocker
Department of Computer Science, ETH Zürich, Switzerland

Philip Whittington
Department of Computer Science, ETH Zürich, Switzerland

Abstract

The time-optimal k-server problem minimizes the time spent instead of the distance traveled when serving n requests, appearing one after the other, with k servers in a metric space. The classical distance model was motivated by a hard disk with k heads. Instead of minimal head movements, the time model aims for optimal reading speeds.

This paper provides a lower bound of $2k - 1$ on the competitive ratio of any deterministic online algorithm for the time-optimal k-server problem on a specifically designed metric space. This lower bound coincides with the best known upper bound on the competitive ratio for the classical k-server problem, achieved by the famous work function algorithm. We provide further lower bounds of $k + 1$ for all Euclidean spaces and k for uniform metric spaces.

Our most technical result, proven by applying Yao's principle to a suitable instance distribution, is a lower bound of $k + H_k - 1$ that holds even for randomized algorithms, which contrasts with the best known lower bound for the classical problem, which is polylogarithmic in k.

We hope to initiate further intensive study of this natural problem.

2012 ACM Subject Classification Theory of computation → K-server algorithms

Keywords and phrases k-server problem, optimizing time instead of distance, deterministic and randomized algorithms, Yao's principle

Digital Object Identifier 10.4230/LIPIcs.ISAAC.2025.32

Related Version *Full Version*: https://doi.org/10.48550/arXiv.2503.05589 [15]

Funding *Fabian Frei*: This research was funded in part by grant number 218001 awarded to Fabian Frei by the Swiss National Science Foundation. Work done in part while at CISPA Helmholtz Center for Information Security and ETH Zurich.

Acknowledgements We thank Peter Rossmanith for his suggestion to study the k-server problem in the time model and Richard Královič for pointing out how the algorithm Robin could be extended to graphs of bounded aspect ratio.

1 Introduction

The *k-server problem*, introduced by Manasse et al. in 1988 [23], has been repeatedly referred to as "the holy grail" of online computation [2, 8, 12]. In particular the k-server conjectures about the best competitive ratios achievable by deterministic and randomized algorithms have inspired intensive research for many decades.

The k-server problem can be considered for any natural k and any given nonempty metric space $\mathcal{M} = (M, d)$ together with an *initial configuration* $C_0 \in M^k$. An instance is a sequence $r_1, \ldots, r_n \in M$ of point requests, revealed one by one. An online algorithm answers each request r_j, knowing only the already revealed requests, with a configuration $C_j \in M^k$

such that $C_j(i) = r_j$ for some i, which we describe as *server* s_i *serving request* r_j. Hence, any solution is a sequence of configurations $C_1, \ldots, C_n \in M^k$ describing the movements of k servers such that each requested point is served by moving a server there. The next request is revealed only once the current request has been served. The *cost* of a solution is traditionally defined as the total of all distances traveled by all servers, yielding the classical, well-researched k-server problem.

Another – no less natural – possibility is to count towards the cost not all but only the *maximum* distance traveled by any server between any two configurations. In this paper, we explore this variant and refer to the different definitions of cost as the *distance model* and the *time model*:

The distance model. This is the well-known classical variant. The cost incurred by the algorithm to serve request r_j is the *sum* of all distances traveled by the servers when changing from configuration C_{j-1} to configuration C_j. The cost of a solution $C_1, \ldots, C_n$ is thus $\sum_{j=1}^{n} \sum_{i=1}^{k} d(C_{j-1}(i), C_j(i))$, the total of all distances moved by all servers. We can imagine paying a fuel cost for all server movements.[1]

The time model. The cost for serving a request is the *maximum* distance traveled by the servers; i.e., the total cost is $\sum_{j=1}^{n} \max_{i=1}^{k} d(C_{j-1}(i), C_j(i))$. Hence, an optimal solution minimizes the total waiting time incurred by the requests until they are served. We thus refer to the problem in this model as the *time-optimal k-server problem*.

In many situations where requests need to be served, it is more important to react as fast as possible rather than moving less. We might consider ambulances or police cars being called to the scene of an emergency, but a good example was already given by the famous seminal paper [23] that used it to introduce the classical k-server problem instead: planning the motions of a hard disk with k heads. It could be argued that rather than caring about minimizing hidden head movements, the typical user wants optimized reading and writing speeds. Despite this, past research has focused almost exclusively on the distance model; see Subsection 1.1.

We hope to initiate a new line of research that focuses on the time model. As usual, the main goal is to determine the best competitive ratios achievable by deterministic and randomized algorithms. We prove several lower bounds in this new model, and design a deterministic algorithm matching these bounds on uniform metric spaces.

1.1 Related Work

In this section, we review known results and open questions for the k-server problem in both models. For the classical distance model, we restrict ourselves to the most important milestones; and at the same time introduce some basic notions and concepts that we make use of later. For the time model, however, we can give a full account.

The Distance Model

We note that on metric spaces with at most k points the algorithm keeping one server on each point is trivially 1-competitive. On a clique with $k + 1$ or more vertices, there is a very simple lower bound of k: for any algorithm, there is an instance that always asks for

[1] We remark that moving more than one server per time step cannot save cost in this model as the algorithm incurs the same cost for any given movement, regardless of when it occurs. However, it is easy in the distance model to convert an algorithm that moves some servers simultaneously into a *lazy* algorithm, which does not do this, but incurs at most the same cost [17].

an unoccupied vertex, causing a cost of 1 with each request, whereas the offline strategy of serving an unoccupied vertex with a server currently at a point that is not among the next $k-1$ requests spends at most a cost of 1 for every k requests. (The k-server problem on cliques of size N is also equivalent to the paging problem with N pages and a cache of size k [6,17].) The lower bound of k can be generalized to all nontrivial metric spaces, that is, to *any* metric space with at least $k+1$ points [6, Thm. 10.1], [17, Thm. 4.4].

Manasse et al. already conjectured that there is a matching upper bound when they introduced the problem in 1988 [23, Conjecture 4].

▶ **Conjecture 1** (*k*-Server Conjecture)**.** *For any metric space and any $k \geq 1$, there is a k-competitive algorithm for the k-server problem in the distance model.*

They also proved the conjecture in the two special cases of only $k = 2$ servers and of k servers on metric spaces with only $k+1$ points [23, Thms. 5, 6]. Note that the journal version of this seminal paper appeared two years later under a different title [24]. This is also how long it took for anyone to establish any upper bound on the competitive ratio that did not grow with the input length n but only with k, albeit with an exponential dependence [13, 14, Thm. 2]. Another four years later, Koutsoupias and Papadimitriou provided the first subexponential bounds by analyzing the so-called *work function algorithm* (WFA, independently proposed as a candidate by multiple researchers [11, Sect. 2.3]), which lowered the upper bound to $2k-1$ [20, Thm. 4.3].

To date, WFA remains the algorithm with the best known competitive ratio for general metric spaces. We are thus left with a constant factor of essentially 2 between the upper and lower bound, despite the decades-long efforts by the research community trying to improve the upper bound or disprove the k-server conjecture. It is conjectured that WFA even attains the best possible competitive ratio of k [20].

But at least for some special metric spaces the k-server conjecture could be proven. A prominent example is the real line. Here, Chrobak et al. [9] were able to give the matching upper bound of k on the competitive ratio by introducing and analyzing the so-called *double coverage* algorithm (DC) with the following, somewhat unintuitive behavior. When a point x is requested, DC moves not only one, but two servers towards x: a closest server to the right of x and a closest server to the left of x. They move at the same speed and both stop once one of them has reached x. This strategy was generalized to trees by Chrobak and Larmore [10]. Lastly, WFA is also known to be k-competitive on metric spaces with at most $k+2$ points, multiray spaces, and for $k = 3$ on trees and circles [12, 16, 21].

For randomized algorithms, the proof of a lower bound of k from the deterministic case does not work anymore. Instead, a lower bound of $\Omega(\log k)$ has been known for a long time. Let $H_k := \sum_{j=1}^{k}(1/j)$ denote the *k-th harmonic number*. On a clique of $k+1$ vertices, no deterministic algorithm is better than H_k-competitive in expectation on the instance distribution that requests any vertex except the one just requested with the same probability of $1/k$. *Yao's principle* [25] now transforms this lower bound for deterministic algorithms on an input distribution into a lower bound for randomized algorithms on fixed instances [6], [17, Thm. 2.11]. This has remained the best lower bound for multiple decades. (A series of results [3–5] showed a lower bound that is only $\Omega(\log k/(\log\log k))$ instead of $\Omega(\log k)$ but in exchange works on any metric space of at least $k+1$ points.) This led to the following folklore conjecture [19, Conj. 2].

▶ **Conjecture 2** (Randomized *k*-Server Conjecture)**.** *For any $k \geq 1$, there is an $\mathcal{O}(\log k)$-competitive randomized algorithm for the k-server problem in the distance model.*

Table 1 Overview of the best known upper and lower bounds (in the upper and lower row, respectively) on the competitive ratio for the k-server problem in the distance and time model, for deterministic algorithms on the line and general metric spaces, and additionally for randomized algorithms on general metric spaces. Note that all lower bounds on general graphs, apart from the deterministic lower bound of k in the distance model, are existential, i.e., they hold on specially constructed graphs.

Distance model			Time model		
Deterministic		Randomized	Deterministic		Randomized
Line	General	General	Line	General	General
k [9]	$2k-1$ [20]	$2k-1$ [20]	$(k^2+k)/2$ [22]	$2k^2-k$ [20, 22]	$2k^2-k$ [20, 22]
k [23]	k [23]	$\Omega((\log k)^2)$ [7]	$k+1$ [Th. 6]	$2k-1$ [Th. 12]	$k+H_k-1$ [Th. 16]

This conjecture was refuted in 2023 by Bubeck et al. [7], who were able to slightly lift the lower bound to $\Omega((\log k)^2)$ for one specific metric space. Note, however, the remaining exponential gap between this improved lower bound and the best known upper bound for randomized algorithms, which is still $2k-1$ and attained by the deterministic WFA.

A remarkable positive result by Bansal et al. [1] is a randomized algorithm that is, on metric spaces with N points, $\mathcal{O}((\log k)^2(\log N)^3 \log\log N)$-competitive in expectation, and thus even with high probability by a general result by Komm et al. [18]. Their algorithm thus obtains a competitive ratio that is polylogarithmic in k whenever N is polynomial in k. A more detailed discussion of the results up until 2009 is found in a survey by Koutsoupias [19].

The most relevant results for the k-server problem in the distance model on general metric spaces can thus be summarized as follows (see also Table 1): The best known upper bound on the competitive ratio is $2k-1$ (for both deterministic and randomized algorithms) and attained by the work function algorithm WFA [20]. The best known lower bound on the competitive ratio is $\Omega((\log k)^2)$ for randomized algorithms [7], and k for deterministic algorithms [23]. A matching deterministic upper bound of k is attained on the line by the double coverage algorithm DC [9], and on trees by the generalized variant DC-TREE [10].

The Time Model

Almost no research has been published on the time model, which allows us to describe all existing results in what follows.

Clearly, changing from the distance model to the time model helps the algorithm, because it can now move servers synchronously at no additional cost. However, this does not imply an improved competitive ratio. Since the advantages of the time model can be utilized by both the online algorithm and the offline solution it competes with, the ratio could, a priori, improve, stay the same, or get worse, and in different ways for different metric spaces.

A first result distinguishing the two variants was given by Koutsoupias and Taylor [22]. As they remark [22, Sect. 5], any upper bound for the distance model implies a k times larger upper bound for the time model: On the one hand, any algorithm for the distance model also works for the time model with the same or even lower cost. On the other hand, the optimal algorithm against which the online algorithm competes saves at most a factor of k in the time model compared to its cost in the distance model. Overall, the competitive ratio increases by at most a factor of k when an algorithm for the distance model is used in the time model. The $(2k-1)$-competitive WFA thus implies an upper bound of $2k^2-k$. For trees it immediately follows that DC-TREE, which is k-competitive in the distance model, is

k^2-competitive in the time model. Koutsoupias and Taylor [22, Thm. 3] lowered this bound to $k(k+1)/2$ with a straightforward adjustment of the potential function used in the analysis of DC-TREE; note that both bounds are in $\mathcal{O}(k^2)$. They concluded with a lower bound of 3 for $k = 2$ servers on the real line [22, Thm. 4], but provided no results for $k \geq 3$.

1.2 Contribution

The goal of this paper is to popularize the time-optimal k-server problem and establish it as a research object that is well worth the community's attention. We believe the k-server problem in the time model to be natural and well motivated. We also hope that the insights gained by analyzing the time model might shed further light on the long-standing open questions about the classical problem.

Our main contribution are lower bounds for the time-optimal k-server problem that are at least as large as and often larger than the best known lower bounds for the classical variant. For deterministic algorithms, we achieve the lower bound of k on all unweighted graphs with at least $k+1$ vertices (which excludes only cases where the competitive ratio is trivially 1) in Lemma 5. We then go above this, albeit just barely, with a lower bound of $k+1$ for a large class of metric spaces that includes all Euclidean spaces, infinite grids, and large cycles (Theorem 6). We then construct a specific metric space (superexponentially large in k, but finite and of diameter only 3) for which we can prove a lower bound of $2k-1$, which is exactly the best known upper bound for the classical k-server problem (Theorem 12).

Our main result for randomized algorithms is a lower bound on the expected competitive ratio for the time-optimal k-server problem (Theorem 16). The bound is $k + H_k - 1$, and thus exponentially larger than the recently proven polylogarithmic lower bound in the distance model [7].

Despite the significantly raised lower bounds, a gap still remains with a linear factor between the upper and lower bounds.

2 Preliminaries

For any nonnegative integer n, we use the notation $[n] \coloneqq \{1, \dots, n\}$. We denote by $H_k \coloneqq \sum_{j=1}^{k} (1/j)$ the k-th harmonic number.

Online Algorithms and Competitive Analysis

An online algorithm receives an instance I consisting of n *requests*, $I = (x_1, \dots, x_n)$, which are presented to ALG in n consecutive time steps; n is by default not known to ALG a priori. Whenever a request x_i is presented, ALG has to provide an *answer* y_i to x_i, where y_i depends only on the prefix $x_1, \dots, x_i$ of already presented requests. The full answer sequence $\text{ALG}(I) = (y_1, \dots, y_n)$ is the solution computed by ALG on I. Any solution $\text{ALG}(I)$ to I is assigned a *cost* denoted by $\text{cost}(I, \text{ALG}(I))$. An *optimal solution* for I, denoted by $\text{OPT}(I)$, is a solution with minimal cost across all solutions to I. Note that $\text{OPT}(I)$ and $\text{cost}(I, \text{OPT}(I))$ can generally be computed only once I is known.

The performance of an online algorithm ALG is measured in terms of its competitive ratio. ALG is called *c-competitive* if there is a constant α so that, for every instance I, it holds that $\text{cost}(I, \text{ALG}(I)) \leq c \cdot \text{cost}(I, \text{OPT}(I)) + \alpha$. Similarly, a randomized online algorithm RAND is said to be *c-competitive in expectation* or have an *expected competitive ratio of c* if there is a constant α so that, for every instance I, it holds that $\mathbb{E}[\text{cost}(I, \text{RAND}(I))] \leq c \cdot \text{cost}(I, \text{OPT}(I)) + \alpha$.

The k-Server Problem

The intuition behind the k-server problem is best described by a scenario where we are given a metric space and k *servers*, each occupying one point in the space. A request x is any point of the space, and an answer must specify at least one server that is moved to x in response. However, it is possible to move other servers as well. A request is revealed once the previous one has been served. The goal is to minimize either the total distance traveled after answering all requests (the distance model) or the overall time spent (the time model).

Formally, let $\mathcal{M} = (M, d)$ be any metric space. (In particular, the distance function $d\colon M^2 \to \mathbb{R}$ is zero on the pairs (x, x) for $x \in M$, otherwise positive, symmetric, and satisfies the triangle inequality, i.e., $\forall x, y, z \in M\colon d(x, z) \leq d(x, y) + d(y, z)$.) Any finite metric space can be described as the complete undirected graph whose vertices are M with an edge weight function $\{u, v\} \mapsto d(u, v)$. Conversely, we can interpret any finite, connected, weighted, and undirected graph as the metric space whose point set is the vertex set and whose distance function maps two points to the length of a shortest path between them.

In particular, any finite, connected, unweighted, and undirected graph also describes a metric space in the same way. From now on we always assume graphs in this paper to be finite, connected, undirected, and unweighted. Other important metric spaces include the real line, the Euclidean plane, and generally the Euclidean space of any given dimension.

For any positive integer k and metric space $\mathcal{M} = (M, d)$, we call a map $C\colon [k] \to M$ a *k-configuration*. In the context of the k-server problem we may describe a k-configuration by saying that server s_i is at point $C(i)$ for any $i \in [k]$. We say that a k-configuration C *covers* or *occupies* a point $p \in M$ with server s_i if $C(i) = p$ for some $i \in [k]$. For a sequence of k-configurations $C_0, \ldots, C_n$, we say, for any $i \in [k]$ and $j \in [n]$, that server s_i moves from $C_{j-1}(i)$ to $C_j(i)$ in step j. Whenever the value of k is clear from the context, which is the case for the remainder of this paper, we omit the k and simply say configuration instead of k-configuration. An *instance* of the k-server problem on $\mathcal{M}$ of length n is a sequence $(r_1, \ldots, r_n)$ of n points in M and an initial configuration $C_0\colon [k] \to M$. A *solution* is a sequence of configurations $(C_1, \ldots, C_n)$ such that C_j covers r_j for all $j \in [n]$. The *cost* of such a solution is $\sum_{i=j}^{n} \sum_{i=1}^{k} d(C_{j-1}(i), C_j(i))$ in the distance model and $\sum_{j=1}^{n} \max\{d(C_{j-1}(i), C_j(i)) \mid i \in [k]\}$ in the time model.

Intricacies of the Time Model

In the distance model, any algorithm can without loss of generality be transformed into one that is *lazy*, i.e., moves only one server at a time. Hence it is common and convenient to argue only about lazy algorithms. In the time model, this simplification is no longer justified; we also need to consider movements of the servers other than the one serving a request. This leads to intricacies peculiar to the time model, which we briefly discuss here.

Consider a graph with integer weights. Suppose that a server incident to an edge of unit length moves across this edge to serve a request at the other end. Suppose that another server is positioned at a vertex incident to a longer edge of length 2. Depending on the situation to be modeled, one might want to allow or disallow a synchronous movement by the second server to the midpoint of the longer edge. In the default formulation, servers can only traverse edges completely; partial movements are impossible. This implies the following perhaps unintuitive behavior: Even if a server that traversed a unit edge to serve a request subsequently moves back to serve the next request, traveling a length of 2 in two steps, it is impossible for another server to traverse an edge of length 2 at the same time without extra cost. Even if the servers move synchronously as far as possible, the cost is at least 3. This

problem does not occur in unweighted graphs, however. Indeed, if it is desired to enable the servers to traverse a long edge over multiple steps, this is easily achieved by transforming a given finite graph with rational weights into an unweighted one by subdividing the edges into segments of a length that divides all occurring weights.

All of our results apply to both variants since Robin (see Definition 3) works also on weighted graphs, and the graphs constructed for our lower bounds are all unweighted.

3 Results

We now present our results for the time-optimal k-server problem: an observation yielding an upper bound in Subsection 3.1, the lower bounds for deterministic algorithms in Subsection 3.2, and a lower bound for randomized algorithms in Subsection 3.3. Some proofs are summarized for our main theorems and removed for more straightforward results and corollaries. They can be found (in more detail) in the full version of the paper [15].

3.1 Upper Bounds

We start with some observations on a simple algorithm.

▶ **Definition 3** (Algorithm Robin)**.** *Label the k servers $s_1, \dots, s_k$ in arbitrary order. If a requested point is already covered by a server, then* Robin *does not change its configuration. The m-th request that requires* Robin *to move a server is served by s_i with $i \equiv_k m$, while all other servers stay idle. In other words,* Robin *moves a server only when it is necessary, only one server at a time, the first k movements are by $s_1, \dots, s_k$ in this order, and then it repeats cyclically.*

Note that Robin never moves servers synchronously, thus the cost is the same in the time model and the distance model. Essentially, Robin is a *marking algorithm* [6,17]. Koutsoupias and Taylor [22] already remarked that Robin is k-competitive on uniform metrics. It is also not hard to see that this observation can be extended to metrics of bounded *aspect ratio*, where the aspect ratio of a metric space is its diameter divided by the minimal distance between distinct points.

▶ **Observation 4.** *On metric spaces with aspect ratio A,* Robin *is Ak-competitive.*

3.2 Lower Bounds for Deterministic Algorithms

In this section, we provide lower bounds for deterministic algorithms on various metric spaces.

Universal Lower Bound of k on Unweighted Graphs

We begin with a bound of k for all unweighted graphs on which the problem is nontrivial.

▶ **Lemma 5.** *Let k be a positive integer. No algorithm for the time-optimal k-server problem can be better than k-competitive on graphs with more than k vertices.*

Proof sketch. We choose any connected set S of $k+1$ vertices of G and consider the metric subspace described by the induced subgraph $G[S]$. The algorithm is given an instance that always requests a point not occupied by any of its servers. We note that any configuration where all k servers occupy k distinct locations can be reached from any other such configuration at cost at most 1, by moving servers along a path connecting two unoccupied points. This allows an optimal offline algorithm to serve any k consecutive requests at cost 1. ◀

Note that, in contrast to the corresponding result for the distance model, it is unclear whether this result can be extended to arbitrary metric spaces with more than k points.

Lower Bound of $k + 1$ on the Line

We now consider a special type of metric space that has received a lot of attention (see Subsection 1.1) for the classical k-server problem: the line. As mentioned above, there are k-competitive algorithms for k-server in the distance model on the real line, the rational line, and the integer line. The following theorem shows that the time-optimal k-server problem is strictly harder in all of these cases.

▶ **Theorem 6.** *For any integer $k \geq 2$, no algorithm for the time-optimal k-server problem has a competitive ratio better than $k + 1$ on the line (real, rational, or integer), on any finite unweighted cycle with an even number of at least $2k + 6$ vertices, or on the continuous one-dimensional sphere $\mathbb{S}^1$.*

Proof. Let ALG be any online algorithm for the given metric space. For a finite unweighted cycle with an even number of $N \geq 2k + 6$ vertices, label the vertices counterclockwise by $0, 1, \ldots, N - 1$. For the sphere $\mathbb{S}^1$, choose $N = 2k + 6$ points evenly spaced along this metric space, label them $0, 1, \ldots, N - 1$ counterclockwise, and re-scale the space such that the distance between consecutive points is 1. For simplicity, we refer to the labeled points as integers and say that an integer *neighbors* another one if they are at distance 1 from each other. In all cases, every even integer neighbors two odd ones and each odd one two even ones. We will construct an instance phase by phase such that there is an optimal algorithm OPT satisfying the following invariant.

▶ **Invariant 7.** *At the beginning of each phase, the k servers of* OPT *occupy k distinct even integers or k distinct odd integers. Moreover, at least two of these integers are consecutive (in the sense of being separated by a distance of exactly 2).*

Without loss of generality, we assume for the following description that the integers occupied by OPT at the start of the phase are all even by choosing an appropriate initial configuration and shifting the labels of the integers by one after each phase. Any phase will consist of potentially repeated requests for k distinct odd integers, among which at least two are consecutive. Moreover, the requests will be chosen such that OPT can cover all of them by moving all of its k servers simultaneously, each by a distance 1 and thus at total cost 1, at the beginning of the phase and with no movements afterwards. Any phase constructed with these properties preserves the invariant for the next phase, allowing us to iterate the process to construct an arbitrarily long instance of arbitrarily high optimal cost.

Since OPT does not move its servers anymore after its phase-initial synchronous movements, we can request any point covered by OPT as many times as we would like without increasing the cost of OPT or changing OPT's configuration at the end of the phase. We can thus enforce the following invariant for ALG without loss of generality.

▷ Claim 8. Once ALG has served a request at some point during some phase, it will always have a server at this point during all further requests of this phase.

Proof of claim. Assume that ALG makes a move such that some integer previously requested in the current phase is no longer covered. Then the constructed instance could be extended by immediately requesting this integer again after this move, at no additional cost to OPT. This can be continued until all the integers previously requested in the current phase are covered again by ALG. ◁

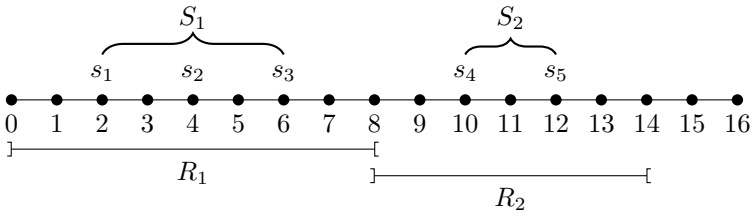

Figure 1 Example of the subdivision of phase-initial server positions and the corresponding ranges used in the proof of Theorem 6. (Note that the figure does not show the entire space, which is a cycle or the infinite line, but only a segment.)

Using this invariant of ALG, we can now also assume the following without loss of generality.

▷ Claim 9. In each phase, k distinct odd integers are requested, each exactly once.

In summary, we can assume the following properties for each phase.

1. At the beginning of the phase, the servers of ALG occupy the same positions as the servers of OPT, namely k *distinct* even integers, at least two of which are consecutive (in the sense of being separated by a distance of 2).
2. During the phase, k distinct odd integers are requested one by one. Each such integer must, once requested, be covered by ALG during all subsequent requests of the phase.
3. Moreover, there is a bipartite matching between the k odd requested points and the k phase-initially covered even integers where each edge of the matching has weight exactly 1.
4. Finally, OPT moves its servers along this matching at cost 1 when the first request of the phase appears and does not move its servers anymore for the rest of the phase.

We now describe how the k requests for any given phase are chosen depending on ALG's behavior in a way that guarantees Invariant 7 to hold for the following phase.

Consider any phase. Partition the phase-initial server positions (which are the same for OPT and ALG) into $\ell \geq 1$ maximal, pairwise disjoint, nonempty sets $S_1, \dots, S_\ell$ of consecutive even integers. We define $k_i := |S_i|$ and note that $\sum_{i=1}^{\ell} k_i = k$. Given a metric space $\mathcal{M} = (M, d)$, a point $c \in M$, and radius $\rho \geq 0$, we call $D_\rho(c) := \{p \in M \mid d(p,c) < \rho\}$ the *open ball* and $D_\rho[c] := \{p \in M \mid d(p,c) \leq \rho\}$ the *closed ball*. For any $m \in [\ell]$, we call $R_m := \bigcup_{i \in S_m} D_2(i)$ the *range* of S_m; Figure 1 shows an example for $k = 5$.

Note that the range R_m contains exactly $k_m + 1$ odd integers. Exactly k_m of them are requested during a phase in the instance family described below. This guarantees that OPT can indeed move all its servers from their phase-initial positions to the points request during this phase at cost 1, and then keep them there for the remainder of the phase, which proves the first part of Invariant 7. We call the range R_m *saturated* if the instance has already requested k_m out of the $k_m + 1$ odd integers of R_m during the current phase, and *unsaturated* otherwise. Recall that once an integer is requested, a server occupies it during all remaining requests of the phase; thus any saturated R_m contains at least k_m servers. A phase ends when all ranges are saturated.

Since there are two consecutive even integers in the phase-initial configuration of OPT, by renaming we can assume without loss of generality that $k_1 \geq 2$ and that $S_1 = \{2, 4, \dots, 2k_1\}$. Since R_1 contains exactly $k_1 + 1 \geq 3$ consecutive odd integers and k_1 of them are requested during the phase, at least two consecutive odd integers are requested. This proves the second part of Invariant 7.

We now describe what the requests in the considered phase depend on. For this, we will call a point *distant* if and only if it has distance at least 1 from all current servers, i.e., if it lies outside of all open unit balls around the current server locations. The first request is 3, which is indeed a distant point since all servers are phase-initially at even integers. From then on, a further request is chosen according to the following *selection procedure* until $k-1$ odd integers have been requested during the phase or the procedure becomes impossible: Request an arbitrary distant odd integer not previously requested during the phase from any unsaturated range R_m except for the integer 1. The sequence of such requests ends only when k_m requests have been made in each range R_m with $1 \neq m \in [\ell]$ and $k_1 - 1$ requests have been made in R_1, or if no such request is possible anymore. We will show that once this process ends, there must be a point in R_1 that no server can reach with cost less than 2 using the following simple claim.

▷ Claim 10. If a range R_m for any $m \in [\ell]$ contains at most k_m servers, it also contains a distant odd integer. If the *truncated range* $R'_1 \coloneqq R_1 \setminus (0, 4]$ contains at most $k_1 - 2$ servers, it contains a distant odd integer.

Proof of claim. A unit ball can contain at most one odd integer. Moreover, unit balls around points outside of a range R_m cannot contain odd integers inside this range, and analogously for the truncated range R'_1. Hence, the range R_m for any $m \in [\ell]$ containing at most k_m servers means that at most k_m out of the $k_m + 1$ odd integers in it are not distant and thus that there is a distant odd integer in R_m. Analogously, the truncated range R'_1 containing at most $k_1 - 2$ servers means that at most $k_1 - 2$ out of the $k_1 - 1$ odd integers in it are not distant, and thus that there is a distant odd integer in R'_1. ◁

By this claim, if the selection procedure ends, any range R_m with $1 \neq m \in [\ell]$ must contain at least k_m servers, whether or not it is saturated, and R'_1 must contain at least $k_1 - 2$ servers. We consider two different cases.

- *Case 1: R'_1 contains at least $k_1 - 1$ servers.* In this case, R_1 contains at least k_1 servers (including the server positioned on 3). This means that all ranges R_m with $1 \neq m \in [\ell]$ must contain exactly k_m servers and must therefore be saturated. Thus, these servers all occupy odd integers and have a distance of at least 2 to the odd integer 1. The same holds for any servers in R'_1.
- *Case 2: R'_1 contains exactly $k_1 - 2$ servers.* In this case, R'_1 contains a distant point. Since the selection procedure ended, a total of $k_1 - 1$ requests must have already been made to R_1 and served. Thus, the $k_1 - 2$ servers in R'_1 occupy distinct odd integers and cannot reach any other odd integer in R_1 with cost less than 2. Of the other ranges R_m with $1 \neq m \in [\ell]$, all but possibly one must be saturated, since they contain exactly k_m servers. Therefore, these servers occupy odd integers and cannot reach odd integers in R_1 with cost less than 2. Finally, one range R_m with $1 \neq m \in [\ell]$ may contain $k_m + 1$ servers, or a single server may occupy a point outside a range. Such a server can only reach either the odd integer 1 or the odd integer $2k_1 - 1$ in R_1 with cost less than 2. This is clear if the metric space is a line.

  If it is a cycle or the continuous sphere, the two ranges would have to meet at both ends. However, this would mean that they contain all odd integers in the metric space, of which there are at least $k + 3$, while they contain a maximum of $(k_1 + 1) + (k_m + 1) \leq k + 2$ odd integers. Thus, since there are two odd integers in R_1 that have not been requested yet, one of them cannot be reached by any server with cost less than 2.

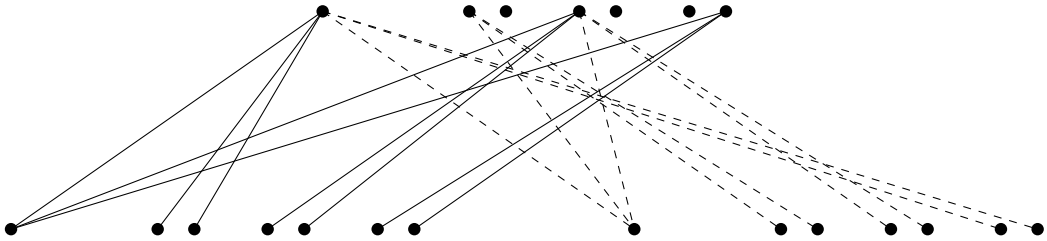

Figure 2 A small part of the construction used to prove Theorem 12 for $k = 3$: one of the $k(k-1)^{k-1} = 12$ blocks of one layer at the top (with the hub point on the left and the fringe points in k groups of $k-1$ to the right), two blocks of the next layer at the bottom left and bottom right, and the edges induced by the corresponding choices of $k-1$ groups and one fringe point per chosen group in the block above; with dashed edges for the choice represented by the bottom right block.

We thus have, in both cases, an odd integer in R_1 that is at distance at least 2 from all servers. Such an integer is requested next, forcing ALG to incur a cost of at least 2 when serving it. From then on, additional distant points in unsaturated ranges R_m with $m \in [\ell]$ (including R_1) are requested. Such points exist by Claim 10. Once k distinct odd integers have been requested, all ranges are saturated and the phase ends.

Recall that the first request was to the odd integer 3. Since R_1 contains $k_1 + 1$ odd integers, k_1 of which have been requested, at least one of the odd integers 1 or 5 must have been requested, and thus indeed at least two consecutive odd integers. This means that the final configuration of the phase is such that Invariant 7 is satisfied for an immediately following phase.

We can thus iterate the process and construct arbitrarily many new phases such that for each phase ALG incurs a cost of at least $k + 1$ while OPT has a cost of exactly 1. ◀

Note that Theorem 6 generalizes the lower bound of 3 given by Koutsoupias and Taylor for the special case of $k = 2$ servers on the real line [22, Thm. 4]. In contrast to the distance model, lower bounds in the time model do not trivially extend to arbitrary metric superspaces. However, this particular result can be extended to other metric spaces, e.g., the Euclidean plane.

▶ **Corollary 11.** *No algorithm can solve the time-optimal k-server problem with a competitive ratio better than $k + 1$ in the Euclidean space of any positive dimension.*

Existential Lower Bound of $2k - 1$ on General Graphs

The most celebrated result for the k-server problem is Koutsoupias and Papadimitriou's proof [20] showing WFA to be $(2k-1)$-competitive in the distance model. Intriguingly, $2k-1$ is also our best *lower* bound for the time model.

▶ **Theorem 12.** *For any $k \geq 1$, there is a finite metric space on which no online algorithm for the time-optimal k-server problem has a competitive ratio better than $2k - 1$.*

Proof. We first give the general idea of the proof. Consider a star with k rays of length 2, and assume that there is a server at each midpoint of a ray. The center of the star is requested first. Any algorithm must move one server there at cost 1. With one of the k servers at the center, one ray no longer contains a server. Its outer point is requested next, causing a cost of at least 2. This process is then repeated $k-1$ times, always requesting the outer point of an unoccupied ray, for a total cost of $2k - 1$.

The main challenge lies in simulating this instance while making the process repeatable. To this end, we construct a metric space given by a finite, unweighted graph.

Description of the metric space. The metric space is a tripartite graph $G = (V, E)$, i.e., its vertices can be partitioned into three disjoint *layers* $L_1 \cup L_2 \cup L_3 = V$ such that the edges can be partitioned as $E_1 \cup E_2 \cup E_3 = E$ with $E_1 \subseteq L_1 \times L_2$, $E_2 \subseteq L_2 \times L_3$, and $E_3 \subseteq L_3 \times L_1$.

On the instances we describe later, the three layers will be used cyclically by any optimal solution: its servers will all move synchronously first from L_1 to L_2, then to L_3, then back to L_1, and so on. Moreover, the three subgraphs $(L_1 \cup L_2, E_1)$, $(L_2 \cup L_3, E_2)$, and $(L_3 \cup L_1, E_3)$ are all isomorphic. In particular, the layers all have the same size. Each layer is partitioned into $B \coloneqq k(k-1)^{k-1}$ uniformly sized *blocks*. Each block in turn consists of one special vertex called its *hub* and $k(k-1)$ vertices that we call its *fringe points*, grouped into k groups each of size $k-1$. We identify the blocks with the integers $[B]$, and similarly denote the groups of each block by $[k]$ and the points of a group by $[k-1]$. We denote the hub of block b in layer L_ℓ by $h(\ell, b)$ and the fringe point n in group g of the same block by $f(\ell, b, g, n)$. Thus, the vertex set is given by

$$V \coloneqq \bigcup_{\ell=1}^{3} \bigcup_{b=1}^{B} \left(\{h(\ell, b)\} \cup \bigcup_{g=1}^{k} \bigcup_{n=1}^{k-1} \{f(\ell, b, g, n)\} \right) .$$

For notational convenience we define $L_4 \coloneqq L_1$ and $L_0 \coloneqq L_3$. To describe the edges of G, we first highlight some uniform aspects of the construction. For each $\ell \in [3]$, the edges E_ℓ between L_ℓ and $L_{\ell+1}$ are constructed identically.

Moreover, the edges from any layer to the next are *block-uniform* in the following sense: for any $\ell \in [3]$, $g \in [k]$, and $n \in [N]$, the neighborhood in $L_{\ell+1}$ is the same for every fringe point $f(\ell, b, g, n)$ and for any hub $h(\ell, b)$ independent of $b \in [B]$.

Now consider any block b of L_ℓ. To each possible choice of $k-1$ fringe points from $k-1$ distinct groups, we assign one block $c \in [B]$ from $L_{\ell+1}$ by an arbitrary but fixed bijection. Such a bijection exists, because there are k possibilities to choose $k-1$ fringe groups and $(k-1)^{k-1}$ ways to choose one fringe point from each of these groups, which means that the total number of possible choices is exactly B, the number of blocks in $L_{\ell+1}$.

Fix such a choice and the corresponding block c in $L_{\ell+1}$. Each of the $k-1$ chosen fringe points, as well as the hub of b, is assigned injectively to one of the k groups of fringe points of c and then made adjacent to all $k-1$ points of the assigned group. For concreteness, let the chosen fringe point of group g in b be assigned to group g in block c, while the hub is assigned to the remaining group of block c that is not represented in the $k-1$ chosen ones. Finally, all $k-1$ chosen fringe points and the hub of b are made adjacent to the hub of c in $L_{\ell+1}$.

More formally, identify the possible choices of $k-1$ fringe points in a block of L_ℓ with $[B]$. For each choice $c \in [B]$, denote by $g_c \in [k]$ the unique index of the group without a chosen point. For any $g \in [k]$ with $g \neq g_c$, denote by $n_c^g \in [k-1]$ the index of the unique point chosen from group g. This means that the fringe points in choice c are exactly $f(\ell, b, g, n_c^g)$ for $g \in [k]$ and $g \neq g_c$. Then we have the following edges between block b, which is in L_ℓ, and $L_{\ell+1}$:

$$\begin{aligned} E(\ell, b) \coloneqq & \bigcup_{c=1}^{B} \{\{h(\ell, b), h(\ell+1, c)\}\} \\ & \cup \bigcup_{c=1}^{B} \bigcup_{n=1}^{k-1} \{\{h(\ell, b), f(\ell+1, c, g_c, n)\}\} \end{aligned}$$

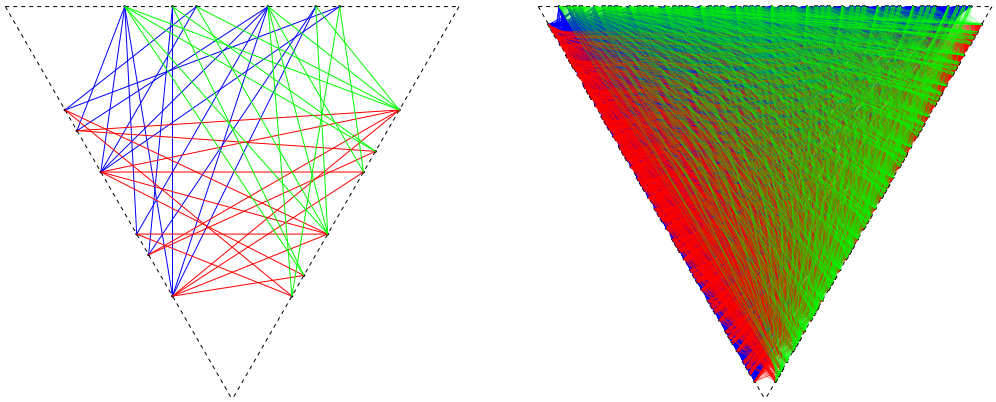

Figure 3 The graph describing the metric space used to prove Theorem 12 for $k = 2$ (left) and $k = 3$ (right). The layers L_1, L_2, and L_3 are arranged counterclockwise with L_1 at the top. Edges from L_1 to L_2 are shown in blue, those from L_2 to L_3 in red, and those from L_3 to L_1 in green.

$$\cup \bigcup_{c=1}^{B} \bigcup_{\substack{g\in[k]\\ g\neq g_c}} \{\{f(\ell, b, g, n_c^g), h(\ell+1, c)\}\}$$
$$\cup \bigcup_{c=1}^{B} \bigcup_{\substack{g\in[k]\\ g\neq g_c}} \bigcup_{n=1}^{k-1} \{\{f(\ell, b, g, n_c^g), f(\ell+1, c, g, n)\}\} .$$

The full edge set is then

$$E \coloneqq \bigcup_{\ell=1}^{3} \bigcup_{b=1}^{B} E(\ell, b) .$$

A visual representation of the graph $G = (V, E)$ for $k = 2$ is shown in Figure 3. For $k = 3$, an example of one block in L_1 and two blocks in L_2 is given in Figure 2, while the full graph is shown in Figure 3. The following claim summarizes some important properties of G. It follows directly from the construction of the edges.

▷ Claim 13. Fix any block b in L_ℓ. We have the following properties:

- **(i)** Any vertex in $L_{\ell-1}$ is adjacent to vertices of at most one fringe group of block b.
- **(ii)** Any fringe point in $L_{\ell+1}$ is adjacent to at most one fringe point of block b.
- **(iii)** Any hub point in $L_{\ell+1}$ is adjacent to exactly one vertex for each of exactly $k-1$ of the k fringe groups of block b.

Description of the instances. We now show that no online algorithm can be better than $(2k-1)$-competitive on the metric space described by G.

Let ALG be any deterministic online algorithm. We describe an instance consisting of distinct phases. In each phase, exactly k distinct points, all from the same layer, are requested, potentially with many repetitions. The instances are designed such that OPT can move all its k servers by a distance of 1 at the beginning of each phase and then serve all requests within this phase at no additional cost, while ALG has cost at least $2k-1$.

At the start of each phase, the servers of OPT are located in a single block b of L_ℓ and more specifically at one of its hub points and $k-1$ points chosen from $k-1$ distinct fringe groups in it.

The phase consists of vertices from the block c in $L_{\ell+1}$ corresponding to the choice of fringe points in block b that form the configuration of OPT at the start of the phase. Specifically, it consists of the hub of block c and $k-1$ fringe points chosen from distinct groups. This means that the process can be repeated since the final configuration is equivalent to the starting position. It also means that OPT can reach the configuration at cost 1: it can move $k-1$ of its servers in block b to fringe points in block c in the unique group they are adjacent to, and move the final server to the hub of block c.

We can assume that within each phase, ALG keeps a server on each previously requested point. Otherwise, the instance will simply request that point again, at no additional cost to OPT.

The instance starts the phase with a request to the hub of block c. ALG must now move a server to that vertex and thus incurs cost 1. Assume that $s \in [k-1]$ fringe points in block c have already been requested by the instance in this phase. We claim that there is at least one fringe point n in block c such that

(a) no vertex from the group in block c containing n was requested in this phase before, and

(b) no server of ALG can reach n with cost less than 2.

By assumption, $s+1$ servers of ALG already occupy requested points in block c. These points are not adjacent to any other point in L_2, and thus in particular not to any point in block c. There are $k-s$ groups in block c that are not yet represented by a request in this phase. Since there are only $k-s-1$ remaining servers, by Claim 13(i) and (ii), there is at least one group in block c such that no fringe point in this group is adjacent to a server that is in $L_{\ell-1}$ or on a fringe point in $L_{\ell+1}$. Within this group, there are $k-1$ points. Any server of ALG that can reach such a point with cost at most 1 is either on that point (in which case it cannot reach any other point in block c with cost at most 1) or on a hub in $L_{\ell+1}$ (in which case, by Claim 13(iii), the same holds).

Since there are $k-s-1$ remaining servers, this means that such a point n exists, unless $s=0$, i.e., if the only point requested in this phase has been the hub, and all servers of ALG occupy hubs in $L_{\ell+1}$ or fringe points in L_ℓ. In this case, by Claim 13(iii), a server on a hub in $L_{\ell+1}$ can reach at most $k-1$ fringe points with cost 1. A server on a fringe point in block c can only reach one such point. Therefore, the total number of fringe points in block c that can be reached in distance 1 by servers on such vertices is at most $(k-1)^2$. This is strictly smaller than the total number of $k \cdot (k-1)$ fringe points in block c; so again, a point n with the required properties must exist.

The instance now requests this point, which ALG must serve with cost at least 2. After $k-1$ such requests, the phase ends. ALG and OPT are in the promised final configuration, and the total cost of ALG in this phase is $1 + (k-1) \cdot 2 = 2k-1$. ◀

▶ **Observation 14.** *The diameter of the metric space used in the proof of Theorem 12 is* 3.

It follows by Observation 4 that there is a $3k$-competitive algorithm on this graph.

3.3 Lower Bound for Randomized Algorithms

In this section, we provide a lower bound for randomized algorithms. We first state Yao's principle for infinite minimization problems [17, Thm. 2.5]. A comprehensive explanation is found in the textbooks by Borodin and El-Yaniv [6], and Komm [17].

▶ **Fact 15** (Yao's Principle). *Let $\mathcal{I}_1, \mathcal{I}_2, \dots$ be an infinite sequence of sets of instances of a given (online) minimization problem. Let $A_1, A_2, \dots$ be a list of all deterministic online algorithms for this problem. Let $\Pr_i$ denote an adversarial probability distribution on the instances in $\mathcal{I}_i$, and let $\mathbb{E}_i$ denote the corresponding expected value function. If there is some constant $c \geq 1$ such that*

(i) $\dfrac{\min_j(\mathbb{E}_i[\mathrm{cost}(A_j(\mathcal{I}_i))])}{\mathbb{E}_i[\mathrm{cost}(\mathrm{OPT}(\mathcal{I}_i))]} \geq c$ *for every positive integer i, and*

(ii) $\lim_{i\to\infty} \mathbb{E}_i[\mathrm{cost}(\mathrm{OPT}(\mathcal{I}_i))] = \infty$,

then there is, for any given constant $\varepsilon > 0$, no randomized online algorithm for the given problem that is $(c-\varepsilon)$-competitive in expectation.

Existential Lower Bound of $k + H_k - 1$ on General Graphs

We now provide a lower bound of $k+\Omega(\log k)$ on the expected competitive ratio of randomized online algorithms. This is achieved on a significant extension of the metric space used in the proof of Theorem 12.

Note that it is easy to find a metric space on which no randomized algorithm can have a competitive ratio better than $\Omega(k)$. In fact, on uniform metric spaces with $N \gg k$, there is a lower bound of $k - o(1)$, which can be shown by requesting points uniformly at random and using Yao's principle. The value of Theorem 16 lies in showing a bound that is *strictly larger* than k, the best known lower bound and conjectured competitive ratio for deterministic algorithms in the classical distance model.

▶ **Theorem 16.** *Let any positive integer $k \geq 1$ be given. For any $\varepsilon > 0$, there is a finite metric space on which no randomized online algorithm for the time-optimal k-server problem has a better constant competitive ratio than $k + H_k - 1 - \varepsilon$.*

Proof sketch. The metric space is described by a graph $G = (V, E)$ that extends the construction used in the proof of Theorem 12. Instead of a single hub point $h(\ell, b)$, each block contains a *hub group* of N hub points $h(\ell, b, n)$ for some large number N. Each of the k fringe groups also contains N fringe points. Correspondingly, the number of blocks is $B = kN^{k-1}$.

We describe a distribution over hard instances on which no deterministic algorithm ALG can be better than $(k + H_k - 1 - \varepsilon)$-competitive, and then apply Yao's principle. The instances are again partitioned into phases, and in each phase, points are requested from the block corresponding to the positions of the servers of OPT, so that OPT has cost 1 per phase.

The instances start with a hub point chosen uniformly at random from this block. The expected cost ALG incurs serving this request is approximately 1 since there are $N \gg k$ possible choices. From then on, one of the $k+1$ groups (either the hub group or a fringe group) is chosen uniformly at random. If no point from that group has been requested yet in that phase, a point from the group is chosen uniformly at random and requested; otherwise, the previous point is requested again.

We then show that, if s distinct points with $1 \leq s < k$ have been requested, the expected cost of ALG is at least $(k+1-s)/(k+1)+1/(k+1)$, by using the equivalent of Claim 13. The expected number of requests until the next distinct point is requested is $(k+1)/(k+1-s)$, so the expected cost of ALG in that timeframe is at least $1+1/(k+1-s)$. Summing up over all s and adding the expected cost of 1 for the first request to a hub point, the expected cost of ALG is therefore

$$1 + \sum_{s=1}^{k-1} \left(1 + \frac{1}{k+1-s}\right) = k + H_k - 1\,. \qquad \blacktriangleleft$$

Note that it can be shown that the graph used in the proof of Theorem 16 has diameter 3, analogously to Observation 14. Thus there is also a deterministic algorithm with competitive ratio $3k$ on that metric space.

4 Conclusion

We hope to initiate a line of research that focuses on the time-optimal k-server problem. We started by proving a series of lower bounds, showing the time model to be harder than the distance model on various metric spaces, including simple cycles and all Euclidean spaces, which implies that the direct analogue of the k-server conjecture in the time model cannot be true. Our strongest deterministic lower bound matches – intriguingly – the best upper bound known for the classical distance-optimal k-server problem, which is attained by the deterministic work function algorithm WFA. A priori, it could thus be true that this algorithm is in fact exactly $(2k-1)$-competitive on general metric spaces in both models.

A natural next step is to find good performance guarantees for WFA in the time-optimal setting. Unfortunately, the analysis of Koutsoupias and Papadimitriou [20] that proved successful for the distance model resists any straightforward adaptation; multiple key concepts, such as the duality between the so-called minimizers and maximizers, do not translate well to the time model. Better upper bounds in the time model will thus probably provide us with several novel techniques.

Our preliminary attempts and experimentally gathered evidence indeed point to a subquadratic upper bound. We believe the lower bound of Theorem 12 to be tight, i.e., we conjecture that a competitive ratio of $2k-1$ is indeed attainable.

▶ **Conjecture 17.** *There is a deterministic algorithm for the time-optimal k-server problem with a competitive ratio of $2k-1$ on general metric spaces.*

For randomized algorithms, we analogously suspect the existence of an expected competitive ratio of c in the distance model to correspond to one of $c+k-1$ in the time model.

References

1. Nikhil Bansal, Niv Buchbinder, Aleksander Mądry, and Joseph (Seffi) Naor. A polylogarithmic-competitive algorithm for the k-server problem (extended abstract). In *Proceedings of the 52th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2011)*, pages 267–276, 2011. doi:10.1109/FOCS.2011.63.
2. Nikhil Bansal, Niv Buchbinder, and Joseph (Seffi) Naor. Metrical task systems and the k-server problem on HSTs. In *Proceedings of the 37th International Colloquium on Automata, Languages and Programming (ICALP 2010)*, volume 6198 of *LNCS*, pages 287–298, 2010. doi:10.1007/978-3-642-14165-2_25.
3. Yair Bartal, Béla Bollobás, and Manor Mendel. Ramsey-type theorems for metric spaces with applications to online problems. *Journal of Computer and System Sciences (JCSS)*, 72(5):890–921, 2006. doi:10.1016/J.JCSS.2005.05.008.
4. Yair Bartal, Nathan Linial, Manor Mendel, and Assaf Naor. On metric Ramsey-type phenomena. In *Proceedings of the 35th Annual ACM Symposium on Theory of Computing (STOC 2003)*, pages 463–472, 2003. doi:10.1145/780542.780610.
5. Avrim Blum, Howard J. Karloff, Yuval Rabani, and Michael E. Saks. A decomposition theorem and bounds for randomized server problems. In *33rd Annual Symposium on Foundations of Computer Science (FOCS 1992)*, pages 197–207, 1992. doi:10.1109/SFCS.1992.267772.
6. Allan Borodin and Ran El-Yaniv. *Online Computation and Competitive Analysis*. Cambridge University Press, Cambridge, 1998. doi:10.5555/290169.

7 Sébastien Bubeck, Christian Coester, and Yuval Rabani. The randomized k-server conjecture is false! In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing (STOC 2023)*, pages 581–594, 2023. doi:10.1145/3564246.3585132.

8 Niv Buchbinder and Joseph (Seffi) Naor. The design of competitive online algorithms via a primal-dual approach. *Foundations and Trends in Theoretical Computer Science*, 3(2–3), 2009. doi:10.1561/0400000024.

9 Marek Chrobak, Howard J. Karloff, Thomas H. Payne, and Sundar Vishwanathan. New results on server problems. *SIAM Journal on Discrete Mathematics*, 4(2):172–181, 1991. doi:10.1137/0404017.

10 Marek Chrobak and Lawrence L. Larmore. An optimal on-line algorithm for k-servers on trees. *SIAM Journal on Computing*, 20(1):144–148, 1991. doi:10.1137/0220008.

11 Marek Chrobak and Lawrence L. Larmore. The server problem and on-line games. In *On-Line Algorithms, Proceedings of a DIMACS Workshop*, volume 7, pages 11–64, 1991. doi:10.1090/DIMACS/007/02.

12 Christian Coester and Elias Koutsoupias. Towards the k-server conjecture: A unifying potential, pushing the frontier to the circle. In *48th International Colloquium on Automata, Languages, and Programming (ICALP 2021)*, volume 198 of *LIPIcs*, pages 57:1–57:20, 2021. doi:10.4230/LIPICS.ICALP.2021.57.

13 Amos Fiat, Yuval Rabani, and Yiftach Ravid. Competitive k-server algorithms (extended abstract). In *31st Annual Symposium on Foundations of Computer Science (FOCS 1990)*, LIPIcs, pages 454–463, 1990. doi:10.1109/FSCS.1990.89566.

14 Amos Fiat, Yuval Rabani, and Yiftach Ravid. Competitive k-server algorithms. *Journal of Computer and System Sciences (JCSS)*, 48(3):410–428, 1994. doi:10.1016/S0022-0000(05)80060-1.

15 Fabian Frei, Dennis Komm, Moritz Stocker, and Philip Whittington. Time-optimal k-server, 2025. doi:10.48550/arXiv.2503.05589.

16 Zhiyi Huang and Hanwen Zhang. Deterministic 3-server on a circle and the limitation of canonical potentials. *Theoretical Computer Science*, 1020:114844, 2024. doi:10.1016/j.tcs.2024.114844.

17 Dennis Komm. *An Introduction to Online Computation – Determinism, Randomization, Advice*. Springer, 2016. doi:10.1007/978-3-319-42749-2.

18 Dennis Komm, Rastislav Královič, Richard Královič, and Tobias Mömke. Randomized online algorithms with high probability guarantees. *Algorithmica*, 84(5):1357–1384, 2022. doi:10.1007/s00453-022-00925-z.

19 Elias Koutsoupias. The k-server problem. *Computer Science Review*, 3(2):105–118, 2009. doi:10.1016/j.cosrev.2009.04.002.

20 Elias Koutsoupias and Christos H. Papadimitriou. On the k-server conjecture. *Journal of the ACM*, 42(5):971–983, 1995. doi:10.1145/210118.210128.

21 Elias Koutsoupias and Christos H. Papadimitriou. The 2-evader problem. *Information Processing Letters*, 57(5):473–482, 1996. doi:10.1016/0020-0190(96)00010-5.

22 Elias Koutsoupias and David Scot Taylor. The CNN problem and other k-server variants. *Theoretical Computer Science (TCS)*, 324(2-3):347–359, 2004. doi:10.1016/J.TCS.2004.06.002.

23 Mark S. Manasse, Lyle A. McGeoch, and Daniel D. Sleator. Competitive algorithms for on-line problems. In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing (STOC 1988)*, pages 322–333, 1988. doi:10.1145/62212.62243.

24 Mark S. Manasse, Lyle A. McGeoch, and Daniel Dominic Sleator. Competitive algorithms for server problems. *Journal of Algorithms*, 11(2):208–230, 1990. doi:10.1016/0196-6774(90)90003-W.

25 Andrew C.-C. Yao. Probabilistic computations: Toward a unified measure of complexity (extended abstract). In *Proceedings of the 18th Annual Symposium on Foundations of Computer Science (FOCS 1977)*, pages 222–227, 1977. doi:10.1109/SFCS.1977.24.