Weak adversaries for the k-server problem

Elias Koutsoupias*
Computer Science Department
University of California, Los Angeles
elias@cs.ucla.edu

Abstract

We study the k-server problem when the off-line algorithm has fewer than k servers. We give two upper bounds of the cost $\mathrm{WFA}(\rho)$ of the Work Function Algorithm. The first upper bound is $k\mathrm{OPT}_h(\rho)+(h-1)\mathrm{OPT}_k(\rho)$, where $\mathrm{OPT}_m(\rho)$ denotes the optimal cost to service ρ by m servers. The second upper bound is $2h\mathrm{OPT}_h(\rho) - \mathrm{OPT}_k(\rho)$ for $h \leq k$. Both bounds imply that the Work Function Algorithm is $(2k-1)$-competitive. Perhaps more important is our technique which seems promising for settling the k-server conjecture. The proofs are simple and intuitive and they do not involve potential functions. We also apply the technique to give a simple condition for the Work Function Algorithm to be k-competitive; this condition results in a new proof that the k-server conjecture holds for $k = 2$.

1. Introduction

The standard framework for evaluating on-line problems is competitive analysis in which we compare the performance of an on-line algorithm with the performance of the off-line (adversary) algorithm. It is natural to try to extend this framework by curtailing the resources of the adversary. For the k-server [10] problem, a natural approach is to compare the performance of the on-line algorithm against an off-line algorithm that has less than $h \leq k$ servers. To be more precise, let's use the notation $\mathrm{A}_m(\rho)$ to denote the cost of servicing request sequence ρ by algorithm A that uses m servers. We then want to compare the on-line cost $\mathrm{A}_k(\rho)$ with the optimal cost $\mathrm{OPT}_h(\rho)$ of the optimal off-line algorithm that uses a different number h of servers.

For the paging problem, the special case of the k-server problem on uniform metric spaces, this approach was successful: Sleator and Tarjan in their seminal paper [11] showed that the competitive ratio is $k/(k-h+1)$. Even the weighted caching problem, a natural generalization of the paging problem, has the same competitive ratio [12]: there are on-line algorithms A_k with $\mathrm{A}_k(\rho) \leq \frac{k}{k-h+1}\mathrm{OPT}_h(\rho) +$ const, and the ratio $\frac{k}{k-h+1}\mathrm{OPT}_h(\rho)$ is the best the possible [10].

Unfortunately, it was soon realized that no such result is possible for the general k-server problem: Even for the line, the competitive ratio may not even depend on k. For example, as Bar-Noy and Schieber showed, the ratio against $h = 2$ servers is at least 2, no matter how large k is (see [3], page 175). Compare this with the positive results for the original k-server problem ($h = k$): We know that the competitive ratio is between k [10] and $2k-1$ [6, 7, 8] for any metric space. The k-server conjecture though remains unsettled; the conjecture states that for all metric spaces the competitive ratio is exactly k. We have only resolved the conjecture, in the affirmative of course, for some special cases (for $k = 2$ [10], for tree metrics [4], for metric spaces with $k + 2$ points [9], and recently for $k = 3$ in the 2-dimensional Manhattan metric [2]).

In this paper, we present the first positive results on the k-server problem with weak-adversaries ($h \neq k$). Instead of comparing the on-line cost $\mathrm{A}_k(\rho)$ with $\mathrm{OPT}_h(\rho)$ only, we compare it with a weighted sum of $\mathrm{OPT}_h(\rho)$ and $\mathrm{OPT}_k(\rho)$. We show (Theorem 1) that the cost of the Work Function Algorithm can be bounded by combination of the two off-line costs:

$$\mathrm{WFA}_k(\rho) \leq k\mathrm{OPT}_h(\rho) + (h-1)\mathrm{OPT}_k(\rho) + \text{const}, \quad (1)$$

for any h (even for $h > k$) and k. The special case $h = k$, gives that the Work Function Algorithm has competitive ratio $2k-1$. This is a new simpler and proof of the result of [6, 8] and it does not involve a potential function.

We apply the same technique and give a generalization of the Quasiconvexity lemma [8] for a combination of work functions that involve h and k servers. We use the new quasiconvexity lemma to show a relation between extended costs that involve different number of servers. In particular, it shows that the extended cost is monotone: more servers can only reduce the extended cost. Although this seems natural, it came as a surprise to us that it holds for any in-

*Research supported in part by National Science Foundation grant CCR-9521606.

Authorized licensed use limited to: Bodleian Libraries of the University of Oxford. Downloaded on November 08,2026 at 20:35:47 UTC from IEEE Xplore. Restrictions apply.

dividual request. Using this monotonicity property of the extended costs we show that

$$\text{WFA}_k(\rho) \leq 2h\text{OPT}_h(\rho) - \text{OPT}_k(\rho) + \text{const.}$$

This bound, unlike (1), holds only for $h \leq k$. We can improve this when $h = 2$ to

$$\text{WFA}_k(\rho) \leq 3\text{OPT}_2(\rho) - \text{OPT}_k(\rho) + \text{const.}$$

Finally, we applied the technique to attack the k-server conjecture. We succeeded only in (re)proving the special case of $k = 2$, but we believe that our approach may be fruitful.

All our proofs are graphical or pictorial. The ideas are extremely simple, but the text versions of these pictorial proofs are unsatisfactorily lengthy. The underlying idea in all proofs is to find appropriate paths in a given graph metric. As a result, we don't have to guess or use potential functions.

2. Preliminaries

We consider request sequences $\rho = r_1 r_2 \dots r_n$. We let ρ_i to denote the prefix of ρ of length i; in particular, $\rho_0 = \epsilon$ is the empty sequence. We shall assume that there is a fixed initial configuration (multiset of k points). In the case of algorithms with different number of servers, the initial configuration of h servers is a subset of the initial configuration of k servers whenever $h \leq k$. We often assume that the initial configuration consists of k (or h) copies of a fixed point r_0.

The work function $w_\rho(X)$ is defined to be the optimal cost to service the request sequence ρ and then move to configuration X. For simplicity we write w_i instead of w_{ρ_i}. The Work Function Algorithm works as follows: Let A_{i-1} be its configuration just before servicing request r_i. To service r_i it moves to configuration A_i that contains r_i and minimizes $w_i(A_i) + d(A_{i-1}, A_i)$. For a more thorough exposition see [8]. The Work Function Algorithm can be interpreted as the *greedy* algorithm which assumes that the *future is a mirror image of the past.*

It was suggested in [5] that instead of bounding the actual cost of the Work Function Algorithm, it suffices to bound its *extended cost* (it is called *pseudocost* in [5]). The extended cost for request r_i is equal to the maximum increase of the work function: $\max_X\{w_i(X) - w_{i-1}(X)\}$. They showed that the extended cost is at most equal to the on-line plus the off-line (optimal) cost. Therefore, to prove that the Work Function Algorithm is c-competitive, it suffices to bound the extended cost by $(c+1)\text{OPT}(\rho)+\text{const}$. The great advantage of using the extended cost is that we can ignore completely the configuration of the on-line algorithm. To show a competitive ratio c it suffices to prove the following property of work functions:

$$\sum_{i=1}^{n} \max_X\{w_i(X) - w_{i-1}(X)\} \leq (c+1) \min_Y\{w_n(Y)\} + \text{const.} \tag{2}$$

It was shown in [8] that the Work Function Algorithm is $(2k-1)$-competitive. The proof is based on some fundamental properties (Quasiconvexity and Duality) of work functions. Structural properties of work functions, such as Quasiconvexity and Duality, are very useful in analysing on-line algorithms. They seem indispensable for settling the k-server conjecture. For the paging problem, the special easy case of the k-server problem, the structure of work functions has been completely characterized [8].

The Duality property of k-server work functions characterizes the configurations that achieve the maximum $\max_X\{w_i(X) - w_{i-1}(X)\}$. In particular, it states that the expression is maximized by a *minimizer* of r_i with respect to w_{i-1}. A minimizer of r_i with respect to w_{i-1} is a configuration that minimizes $\min_X\{w_{i-1}(X) - \sum_{x\in X} d(r_i, x)\}$.

The Duality Lemma suggests that we could prove the k-server conjecture by finding appropriate expressions (potential functions) that "contain" an appropriate minimizer. It would be helpful if we could find a minimizer that is completely independent of the work function. But is this possible? For some special metric spaces this is indeed possible. A typical such metric space is the circle (the distance between two points is the length of the shortest arc between them). The crucial observation is that if X is a minimizer of r_i then if a point x of X is "pushed away" from r_i the resulting configuration is also a minimizer of r_i. The reason is that in the expression $w_{i-1}(X) - \sum_{x\in X} d(r_i, x)$ any increase of $w_{i-1}(X)$ is canceled by the decrease of $-d(r_i, x)$. As a result, if we push all points of a minimizer as far away as possible from r_i, they will become the antipode (diametrically opposite) $\overline{r}_i$ of r_i. Hence the configuration $\overline{r}_i^k$ (k copies of $\overline{r}_i$) is a minimizer of r_i for any work function. A similar metric space is the line whose properties are explored in [1] to show that on this metric space the Work Function Algorithm is k-competitive.

In fact, any metric space M with bounded diameter δ can be extended to a symmetric metric space M' with diameter $\Delta = 2\delta$. By symmetry we mean the property: Each point a has an antipode $\bar{a}$ such that for any point b: $d_{M'}(a,b) + d_{M'}(b,\bar{a}) = \Delta$. One way to extend M is to take two copies of it and define $d_{M'}(a,b) = d_M(a,b)$ if a and b are in the same copy and $d_{M'}(a,b) = \Delta - d_M(\bar{a},b)$ otherwise. It is trivial to verify that the triangle inequality is satisfied by M'.

For symmetric metric spaces, the sufficient condition (2) for the Work Function Algorithm to be c-competitive be-

Authorized licensed use limited to: Bodleian Libraries of the University of Oxford. Downloaded on November 08,2026 at 20:35:47 UTC from IEEE Xplore. Restrictions apply.

comes:

$$\sum_{i=1}^{n}(w_i(\overline{r}_i^k) - w_{i-1}(\overline{r}_i^k)) \leq (c+1)w_n(\overline{r}_n^k) + \text{const.} \quad (3)$$

where we used also the fact that $\min_Y \{w_n(Y)\}$ is within a constant from $w_n(\overline{r}_n^k)$.

We shall deal only with symmetric metric spaces. Clearly all finite metric spaces have bounded diameter and can be extended to a symmetric metric space. Our results can be extended to any metric space (by simply showing that the constants do not depend on the diameter of the metric space).

We also use the notion of a k-path: A k-path is simply a way to service a request sequence (not necessarily in an optimal way). We frequently use the term "k-path $w_\rho(X)$" to mean a k-path (again not necessarily optimal) that services ρ and ends up at X. Since, we can rewrite (3) in the form

$$\sum_{i=1}^{n} w_i(\overline{r}_i^k) \leq \sum_{i} w_{i-1}(\overline{r}_i^k) + (c+1)w_n(\overline{r}_n^k) + \text{const.}$$

it suffices to exhibit a way to obtain the k-paths corresponding to the left hand side using the k-paths corresponding to the right hand side. Almost all results in this paper exploit this simple idea.

3. Work Function Algorithm against weak adversaries

Our first result relates the cost of the Work Function Algorithm to the optimal costs when h or k servers are available.

Theorem 1 *For any metric space and any request sequence ρ, the cost $\text{WFA}_k(\rho)$ of the Work Function Algorithm of k servers is at most $k\text{OPT}_h(\rho) + (h-1)\text{OPT}_k(\rho) + \phi_\epsilon$ for any h, where ϕ_ϵ is a constant (it depends only on the initial configurations).*

Proof. We need to show that the extended cost is at most $k\text{OPT}_h(\rho) + h\text{OPT}_k(\rho) + \text{const.}$ Equivalently, we should show

$$\sum_{i=1}^{n} w_i(\overline{r}_i^k) \leq \sum_{i=1}^{n} w_{i-1}(\overline{r}_i^k) + k\text{OPT}_h(\rho) + h\text{OPT}_k(\rho) + \text{const.} \quad (4)$$

It is very helpful to consider a graphical representation of the above expression. Picture the sequence of requests $r_0, r_1, \ldots, r_n$ and their antipodes $\overline{r}_0, \overline{r}_1, \ldots, \overline{r}_n$ as in Figure 1. Then $w_{i-1}(\overline{r}_i^k)$ is a k-path that starts at r_0^k, visits $r_1, \ldots, r_{i-1}$, and ends up at $\overline{r}_i^k$.

The underlying idea of the proof is strikingly simple: Let G be the multigraph consisting of the edges of the k-paths

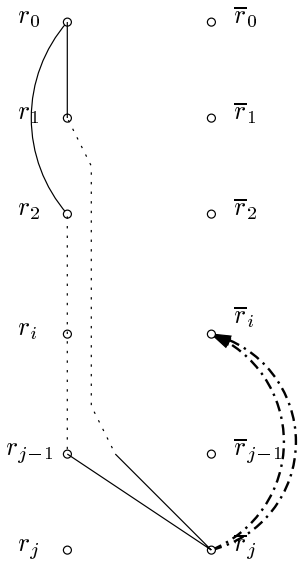

Figure 1. Creating $w_i(\overline{r}_i^k)$ from $w_{j-1}(\overline{r}_j^k)$

of the right hand side of (4). Instead of trying to show that (4) holds in the original metric space, it suffices to show that it holds in the *graph metric* of G. In other words, using the triangle inequality, we want to form $\sum_{i=1}^{n} w_i(\overline{r}_i^k)$ —the left hand side of (4)— from the edges of $\sum_{i=1}^{n} w_{i-1}(\overline{r}_i^k) + k\text{OPT}_h(\rho) + h\text{OPT}_k(\rho)$ —the right hand side of (4).

In fact, we do something much more specific. We form each k-path $w_i(\overline{r}_i^k)$ by taking a k-path $w_{j-1}(\overline{r}_j^k)$ and using some additional edges. These additional edges come from the h-path $\text{OPT}_h(\rho)$ (we need k copies of each additional edge). In particular, we express $w_i(\overline{r}_i^k)$ as $w_{j-1}(\overline{r}_j^k) + kd(\overline{r}_j, \overline{r}_i)$, for some $j > i$. In words, we create a k-path for $w_i(\overline{r}_i^k)$ as follows: the k servers start at r_0, visit requests $r_1, \ldots, r_i, \ldots, r_{j-1}$, move to $\overline{r}_j$ and from there to $\overline{r}_i$. Let us call the edge $(\overline{r}_j, \overline{r}_i)$ *back* edge of the resulting k-path. Notice that we need k copies of each back edge.

The crucial point is the relation between i and j, i.e., which back edges we use. We use $k\text{OPT}_h(\rho)$ as our source of back edges. For a given i, we choose j so that the edge (r_i, r_j) belongs to the h-path of $\text{OPT}_h(\rho)$ (we use the fact that $d(\overline{r}_j, \overline{r}_i) = d(r_j, r_i)$). Here is a slightly different way: Back edges are provided by the optimal h-path that services $\bar{\rho} = \overline{r}_1 \ldots \overline{r}_n$. The total weight (distance) of back edges is equal (within a constant) to $k\text{OPT}_h(\rho)$.

We have to take care also of the fact that not all i's can have corresponding j's: If i is one of the last nodes (requests) of the h-path of back edges then there is no corresponding j. There are h such nodes. This is the reason for the additional term $h\text{OPT}_k(\rho)$ in the right hand side of (4). In summary, a k-path $w_i(\overline{r}_i^k)$ is formed by $w_{j-1}(\overline{r}_j^k) + kd(\overline{r}_i, \overline{r}_j)$ where j is the next vertex after i in

Authorized licensed use limited to: Bodleian Libraries of the University of Oxford. Downloaded on November 08,2026 at 20:35:47 UTC from IEEE Xplore. Restrictions apply.

the h-path $\text{OPT}_h(\rho)$, unless i has no next vertex, in which case it is formed by $\text{OPT}_k(\rho)$.

A careful accounting shows that the proof holds for any metric space (not necessarily finite); the additive term const is independed of the diameter of the metric space. □

The $2k-1$ upper-bound of [8] is the special case of Theorem 1 when $h = k$:

Corollary 1 (Koutsoupias and Papadimitriou, 1994) *The Work Function Algorithm has competitive ratio at most* $2k-1$.

Also, by letting h to be equal to the number of points of the metric space, and observing that in this case $\text{OPT}_h(\rho) = 0$, we get:

Corollary 2 *The Work Function Algorithm has competitive ratio at most $n-1$ on any metric space of n points. In particular, the Work Function Algorithm is k-competitive for metric spaces of $k+1$ points.*

3.1. Quasiconvexity and implications

In this section we generalize the Quasiconvexity lemma of [8] to work functions of h and k servers. The special case (when $h = k$) of the following lemma was shown in [8] and enabled the proof that the Work Function Algorithm is $(2k-1)$-competitive.

Lemma 1 (Quasiconvexity) *For any h-configuration A and k-configuration B, with $h \leq B$, and any point $a \in A$, there is a point $b \in B$ such that*

$$w_n(A) + w_n(B) \geq w_n(A - a + b) + w_n(B - b + a).$$

Proof. The proof has the same underlying idea: show that the relation holds not in the original metric space but in the graph metric space of $w_n(A) + w_n(B)$. In other words, consider an h-path and a k-path of $w_n(A) + w_n(B)$ and show how to obtain paths for $w_n(A-a+b)+w_n(B-b+a)$.

Fix an h-path and a k-path of $w_n(A) + w_n(B)$. Consider a point $a \in A$. We define an alternating path as follows: Start at a, follow an h-edge backwards (towards the first request), then follow a k-edge forward (towards the last request) and repeat (see Figure 2). It is trivial to see that this is a simple path that can only end up at some point $b \in B$. The assumption that the initial h configuration is a subset of the initial k-configuration is crucial here —whenever the aletrnating path visits a node of the initial h configuration, there is always a forward k-edge to continue. For the same reason, the constraint $h \leq k$ in the lemma is essential, otherwise the alternating path may be trapped at a point of the initial h-configuration that does not belong to the initial k-configuration.

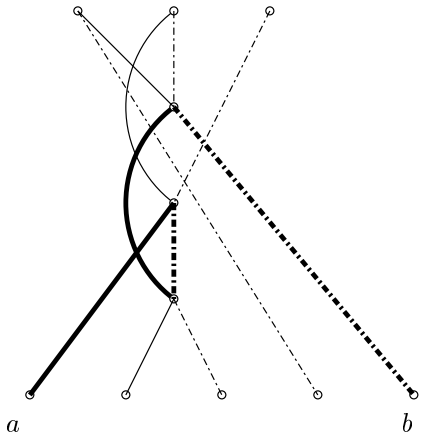

Figure 2. Alternating path

If we exchange the h-edges with the k-edges of the alternating path, we get a new h-path that ends up at $A - a + b$ and a new k-path that ends up at $B - b + a$. The resulting paths have length at most $w_n(A - a + b) + w_n(B - b + a)$. Since they use exactly the edges of the original paths we get

$$w_n(A) + w_n(B) \geq w_n(A - a + b) + w_n(B - b + a).$$

□

We can strengthen the Quasiconvexity Lemma by noticing that alternating paths that originate at different points are edge-disjoint. Therefore, we can exchange more than one point between A and B. More preciselly, there is a 1-to-1 mapping $f : A \mapsto B$ such that for any subset H of A:

$$w_n(A)+w_n(B) \geq w_n(f(H)\cup(A\backslash H))+w_n(H\cup(B\backslash f(H)).$$

It is interesting to note that because the above proof does not use the triangle inequality, the Quasiconvexity Lemma holds even in non-metric spaces.

We can now use the Quasiconvexity Lemma to show the following surprising result.

Lemma 2 *For each request, the extended cost of h servers is greater or equal to the extended cost of k servers when $h \leq k$.*

Proof. We have

$$\begin{aligned} w_n(\overline{r}_n^h) + w_{n-1}(\overline{r}_n^k) &= \\ w_{n-1}(r_n\overline{r}_n^{h-1}) + d(r_n,\overline{r}_n) + w_{n-1}(\overline{r}_n^k) &\geq \\ w_{n-1}(\overline{r}_n^h) + w_{n-1}(r_n\overline{r}_n^{k-1}) + d(r_n,\overline{r}_n) &= \\ w_{n-1}(\overline{r}_n^h) + w_n(\overline{r}_n^k). & \end{aligned}$$

The first equality results from expressing $w_n(\overline{r}_n^h)$ in terms of w_{n-1}. Similarly the last equality results from expressing

Authorized licensed use limited to: Bodleian Libraries of the University of Oxford. Downloaded on November 08,2026 at 20:35:47 UTC from IEEE Xplore. Restrictions apply.

$w_n(\overline{r}_n^k)$ in terms of w_{n-1}. Finally, the inequality follows from quasiconvexity.

Therefore, the theorem holds: $w_n(\overline{r}_n^h) - w_{n-1}(\overline{r}_n^h) \geq w_n(\overline{r}_n^k) - w_{n-1}(\overline{r}_n^k)$. □

An immediate consequence is that the Work Function Algorithm has a nice monotonicity property: If more on-line servers are available, the on-line cost can only decrease and this holds for any request sequence ρ. More precisely, for every ρ, the extended cost of the Work Function Algorithm WFA_k does not exceed the extended cost of WFA_h when $h \leq k$. By Corollary 1, the latter is bounded by $2h\text{OPT}_h(\rho)$ which proves the following theorem.

Theorem 2 *For $h \leq k$ and any request sequence ρ,*

$$\text{WFA}_k(\rho) \leq 2h\text{OPT}_h(\rho) - \text{OPT}_k(\rho) + \textit{const},$$

where const does not depend on ρ. For the special case of $h = 2$, the bound is better:

$$\text{WFA}_k(\rho) \leq 3\text{OPT}_2(\rho) - \text{OPT}_k(\rho) + \textit{const}.$$

The special case of the theorem ($h = 2$) follows from the fact that the k-server conjecture holds for $k = 2$ using the extended cost [5] (see also Theorem 3]).

4. The 2-server problem

There are a lot of different proofs that the k-server conjecture holds for $k = 2$; the first one appeared in [10]. In fact, most of them show directly or indirectly that the Work Function algorithm is 2-competitive and [5] goes further and shows that it suffices to consider the extended cost.

We offer here one more proof of this theorem. Its main characteristic is that it does not use any potential function. Of course, a careful reader can always recover a hidden potential and turn the proof into an inductive one. We believe however that the k-server conjecture can be settled by generalizing appropriately this proof —although we don't know in which direction.

We discuss the ideas behind the proof for the general case of k servers. Only the last lemma in this section is specific for $k = 2$. The idea of the proof is very similar with the one in Theorem 1. We want to find a way to transform the set of k-paths $w_{i-1}(\overline{r}_i^k)$ into the set of k-paths $w_i(\overline{r}_i^k)$ in an appropriate graph metric. In the proof of Theorem 1, we formed the k-path $w_i(\overline{r}_i^k)$ by extending a k-path $w_{j-1}(\overline{r}_j^k)$ with k copies of a back edge, for some appropriate $j > i$. This approach seems to "waste" the last part of the k-path $w_{j-1}(\overline{r}_j^k)$. Can we be more prudent? We will show that the answer is positive and that it leads to a 2-competitive algorithm when $k = 2$.

Once the k-servers of $w_{j-1}(\overline{r}_j^k)$ service request r_i, we could immediately redirect them to $\overline{r}_i$ using back edges (see

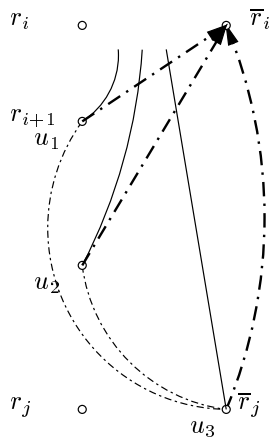

Figure 3. Redirecting $w_{j-1}(\overline{r}_j^k)$ to $w_i(\overline{r}_i^k)$.

Figure 3). This is exactly what we do: We form the k-path $w_i(\overline{r}_i^k)$ from some k-path $w_{j-1}(\overline{r}_j^k)$ where $j > i$. Let $\{u_1, \ldots, u_k\}$ be the multiset of k points visited by the k servers in $w_{j-1}(\overline{r}_j^k)$ immediately after r_i has been visited. Apparently, each u_l is in $\{r_{i+1}, \ldots, r_{j-1}, \overline{r}_j\}$. We want to use back edges $(u_l, \overline{r}_i)$ to form the k-path $w_i(\overline{r}_i^k)$. But where do the back edges $(u_l, \overline{r}_i)$ come from? They are unused edges of some other k-paths. In the same manner, the last unused part of the k-path $w_{j-1}(\overline{r}_j^k)$ —from each u_l to $\overline{r}_j$— can be used as back edges for other k-paths. This, in fact, is the crucial part: we use the last unused part of the k-paths as back edges for other paths. To be more accurate, if (x, y) is an unused edge, we use the antipode edge $(\bar{x}, \bar{y})$ as back edge. What are the conditions that allow the k-paths $w_i(\overline{r}_i^k)$ to be formed in this manner? It works out that the above requirements are equivalent to a surprisingly simple (necessary and sufficient) condition that mysteriously relates k-paths and $(k+1)$-paths.

The condition is that there exists a $(k+1)$-path P which partially agrees with the last part of the k-paths $w_{j-1}(\overline{r}_j)$. To be more specific, let us denote by $Q[i, j]$ the set of edges of multipath Q that have both nodes in $\{r_{i+1}, \ldots, r_{j-1}\}$. Then the condition is:

$$\text{for each edge } (r_i, r_j) \text{ of } P\text{: } P[i, j] \equiv w_{j-1}(\overline{r}_j)[i, j]. \quad (5)$$

Fix an edge (r_i, r_j) of P and consider an edge $e = (r_a, r_b)$ or $e = (r_a, \overline{r}_j)$ of the k-path $w_{j-1}(\overline{r}_j)$. If $a \leq i$ then e should be a forward edge which will be part of the k-path $w_i(\overline{r}_i^k)$; otherwise, the antipode edge $\bar{e}$ should be a back edge which will be part of the k-path $w_a(\overline{r}_a^k)$. It is straightforward to check that this simple condition guarantees that each edge of the k-paths $w_{j-1}(\overline{r}_j)$ is used at most once, either as a back or forward (non-back) edge.

We can show that if there is a path that satisfies (5) then the extended cost is bounded by $(k+1)\text{OPT}(\rho) + \text{const}$.

Authorized licensed use limited to: Bodleian Libraries of the University of Oxford. Downloaded on November 08,2026 at 20:35:47 UTC from IEEE Xplore. Restrictions apply.

Lemma 3 *Fix a request sequence $\rho = r_1 \dots r_n$. If there is a k-path P that services ρ such that $P[i,j] \equiv w_{j-1}(\overline{r}_j)[i,j]$, for each edge $(r_i, r_j) \in P$, then* $\text{WFA}(\rho) \leq k\text{OPT}(\rho) + const.$

Proof. For every i such that there is a j with $(r_i, r_j) \in P$, we can form the path $w_i(\overline{r}_i^k)$ from $w_{j-1}(\overline{r}_j^k)$ and k back edges. There are exactly $k+1$ i's that have no next j. For these i's we need a new k-path. Let $F = \{i : \text{ there is no j such that } (r_i, r_j) \in P\}$; F has cardinality $k+1$. We have

$$\sum_i w_i(\overline{r}_i^k) \leq \sum_j w_{j-1}(\overline{r}_j^k) + \sum_{i \in F} w_i(\overline{r}_i^k).$$

The lemma follows since $\text{OPT}(\rho)$ is within a constant from $w_i(\overline{r}_i^k)$. □

The above lemma holds for any k. We don't know how to use it (or extend it) to prove the k-server conjecture. We can however show that it gives a proof of the 2-server conjecture that involves no potential function. More precisely, condition (5) is always satisfied when $k = 2$:

Lemma 4 *Fix a request sequence $\rho = r_1 \dots r_n$. For any set of 2-paths $w_{j-1}(\overline{r}_j^2)$, $j = 1, \dots, n$, there is a 3-path P that services ρ such that $P[i,j] \equiv w_{j-1}(\overline{r}_j)[i,j]$, for each edge $(r_i, r_j) \in P$.*

Proof. We build the 3-path P backwards. Suppose that we have $P[i,n]$ which agrees with the appropriate last parts of $w_{j-1}(\overline{r}_j^2)$ for all $j > i$. We want to find $P[i-1,n]$. That is, we want to extend $P[i,n]$ by an edge (r_i, r_j) for some $j > i$. There are three candidate j's (because P is a 3-path) of which one is $i+1$. Let j_1 and j_2 be the other two candidate j's. If the edge (r_i, r_{i+1}) belongs to both $w_{j_1-1}(\overline{r}_{j_1}^2)$ and $w_{j_2-1}(\overline{r}_{j_2}^2)$ then we extend $P[i,j]$ by the edge (r_i, r_{i+1}), that is: $P[i-1,n] = P[i,n] \cup \{(r_i, r_{i+1})\}$. The edge (r_i, r_{i+1}) belongs to P, $w_{j_1-1}(\overline{r}_{j_1}^2)$ and $w_{j_2-1}(\overline{r}_{j_2}^2)$, and therefore (5) is satisfied. Otherwise, by symmetry we can assume that (r_i, r_{i+1}) does not belong to $w_{j_1-1}(\overline{r}_{j_1}^2)$. Then we let $P[i-1,n] = P[i,n] \cup \{(r_i, r_{j_2})\}$. It is not hard to see that (5) is satisfied in this case too.

We left for last the problem on how to start building P (the basis case of the backwards induction). The inductive step uses $w_{j_1-1}(\overline{r}_{j_1}^2)$ and $w_{j_2-1}(\overline{r}_{j_2}^2)$. But what are these initially, since there is no $j > n$? We can simply assume that there are such paths (any paths). Here is a more concrete suggestion. Extend the request sequence by two more requests $r_{n+1} = r_n$ and $r_{n+2} = \overline{r}_n$. The 3-path P starts at the last three requests (r_n, r_{n+1} and r_{n+2}). Now, $j_1 = r_{n+1}$ and $j_2 = r_{n+2}$. □

The two lemmata imply that the Work Function Algorithm is 2-competitive when $k = 2$.

Theorem 3 (Manasse, McGeogh, and Sleator, 1988) *For any request sequence ρ,*

$$\text{WFA}_2(\rho) \leq 2\text{OPT}_2(\rho) + const.$$

5. Open problems

There are numerous open problems left. The most important one, of course, is whether the technique of this paper can resolve the k-server conjecture. We don't know if there is an analogue of Lemma 4 for $k > 2$. If not, what is the appropriate direction of generalizing Lemma 3?

Improving the upper bounds against weaker adversaries ($h < k$) is another important research direction. Also, there are essentially no known lower bounds for $h > 2$. As mentioned in the introduction for $h = 2$ the competitive ratio is 2 for any $k \geq 2$.

References

[1] Y. Bartal and E. Koutsoupias. The work function algorithm for the line. In preparation, 1999.

[2] W. Bein, M. Chrobak, and L. L. Larmore. The 3-server problem in the plane. In *Proc. 7th European Symp. on Algorithms*, 1999.

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

[4] M. Chrobak and L. L. Larmore. An optimal online algorithm for k servers on trees. *SIAM Journal on Computing*, 20:144–148, 1991.

[5] M. Chrobak and L. L. Larmore. The server problem and on-line games. In *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, volume 7, pages 11–64, 1992.

[6] E. Koutsoupias. *On-line algorithms and the k-server conjecture*. PhD thesis, University of California, San Diego, La Jolla, California, June 1994.

[7] E. Koutsoupias and C. Papadimitriou. On the k-server conjecture. In *Proc. 26th Symp. Theory of Computing*, pages 507–511, 1994.

[8] E. Koutsoupias and C. Papadimitriou. On the k-server conjecture. *Journal of the ACM*, 42:971–983, 1995.

[9] E. Koutsoupias and C. Papadimitriou. The 2-evader problem. *Information Processing Letters*, 57(5):249–252, Mar. 1996.

[10] M. Manasse, L. A. McGeoch, and D. Sleator. Competitive algorithms for online problems. In *Proc. 20th Symp. Theory of Computing*, pages 322–333, 1988.

[11] D. Sleator and R. E. Tarjan. Amortized efficiency of list update and paging rules. *Communications of the ACM*, 28:202–208, 1985.

[12] N. Young. *Competitive paging and dual-guided on-line weighted caching and matching algorithms*. PhD thesis, Department of Computer Science, Princeton University, 1991.

Authorized licensed use limited to: Bodleian Libraries of the University of Oxford. Downloaded on November 08,2026 at 20:35:47 UTC from IEEE Xplore. Restrictions apply.