On the k-Server Conjecture

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

Christos Papadimitriou
Computer Science and Engineering
University of California, San Diego
christos@cs.ucsd.edu

May 22, 1995

Abstract

We prove that the *work function algorithm* for the k-server problem has competitive ratio at most $2k-1$. Manasse, McGeoch, and Sleator [24] conjectured that the competitive ratio for the k-server problem is exactly k (it is trivially at least k); previously the best known upper bound was exponential in k. Our proof involves three crucial ingredients: A *quasiconvexity property* of work functions, a *duality lemma* that uses quasiconvexity to characterize the configurations that achieve maximum increase of the work function, and a *potential function* that exploits the duality lemma.

1 Introduction

The k-server problem [24, 25] is defined on a metric space $\mathcal{M}$, which is a (possibly infinite) set of points with a symmetric distance function d (non-negative real function) that satisfies the triangle inequality: For all points x, y, and z

$$\begin{aligned} d(x,x) &= 0 \\ d(x,y) &= d(y,x) \\ d(x,y) &\le d(x,z)+d(z,y) \end{aligned}$$

On the metric space $\mathcal{M}$, k servers reside that can move from point to point. A possible position of the k servers is called a *configuration*; that is, a configuration is a multiset of k points of $\mathcal{M}$. We use capital letters for configurations; we also use $D(X, Y)$ for the minimum distance to move the servers from configuration X to configuration Y. We always assume that the k servers are initially at a fixed configuration A_0. For a multiset X and a point a we use $X + a$ for $X \cup \{a\}$ and $X - a$ for $X - \{a\}$. Finally, we use $C(X)$ for the sum of all distances of all pairs of points points in X.

The reader may be wondering why we need to consider configurations to be multisets instead of sets, since it seems reasonable to assume that no two servers occupy the same point simultaneously. As a matter of fact, one can rewrite the proofs in this paper by considering configurations to be sets. The reason we have chosen to use multisets is to be able to use algebraic expressions of the form $A - a + b$, which would necessitate a case analysis in the framework of sets. Other than this, there is no concrete reason for using multisets, and it may be convenient for the reader to consider configurations to be simply sets.

A *request sequence* ρ is a sequence of points of the metric space $\mathcal{M}$ to be serviced by the k servers; servicing a request entails moving some server to the point of request. In particular, if $\rho = r_1 r_2 \ldots r_n$ is a request sequence, then the k servers service ρ by passing through configurations A_0, A_1, A_2, ..., A_n with $r_j \in A_j$. At step j, the cost of servicing request r_j is the cost of moving the k servers from A_{j-1} to A_j; that is, $D(A_{j-1}, A_j)$. The cost for servicing ρ is the sum of the cost for all steps.

Since an on-line algorithm cannot base its decisions on future requests, our choice of A_j must depend only on A_0 and the subsequence of requests $r_1 r_2 \ldots r_j$. On the other hand, an off-line algorithm would know the whole request sequence in advance and consequently in this case A_j depends on A_0 and $r_1 r_2 \ldots r_n$. Let $\mathrm{opt}(A_0, \rho)$ denote the optimal off-line cost for servicing a request sequence ρ starting at the initial configuration A_0. Similarly, let $\mathrm{cost}(A_0, \rho)$ denote the cost for servicing ρ of some on-line algorithm. The competitive ratio of the on-line algorithm is roughly the worst case ratio $\mathrm{cost}(A_0, \rho)/\mathrm{opt}(A_0, \rho)$ [26]. In order to remove any dependency on the initial configuration a more careful definition is necessary: The competitive ratio of the on-line algorithm is the infimum of all c such that for all initial configurations A_0 and for all request sequences ρ

$$\mathrm{cost}(A_0, \rho) \leq c \cdot \mathrm{opt}(A_0, \rho) + C$$

where C may depend on the initial configuration A_0 but not on the request sequence ρ. An on-line algorithm with competitive ratio c is called *c-competitive.*

In metric spaces $\mathcal{M}$ with k or fewer points an on-line algorithm can initially cover all points with its servers; it never again moves them and therefore, its competitive ratio is 1. The problem becomes interesting for metric spaces with at least $k+1$ points. In [24], it was shown that no on-line algorithm can have competitive ratio less that k and the following conjecture was posed:

Conjecture 1 (The k-Server Conjecture) *For every metric space there is an on-line algorithm with competitive ratio k.*

It was also showed that the conjecture holds for two special cases: when $k = 2$ and when the metric space has exactly $k + 1$ points. The *Paging problem* —the special case when all distances are the same— had already been shown to have a k-competitive algorithm in [26]. The k-server conjecture attracted much of interest because of its simplicity, its elegance and its apparent centrality in the study of on-line problems.

For some time, it was open whether *any finite* competitive ratio at all was possible for all metric spaces. It was shown in [14] that indeed there is an algorithm with finite competitive ratio for all metric spaces. Unfortunately, the competitive ratio of the algorithm in [14] increases exponentially with k: It is $\Theta((k!)^3)$. This was improved somewhat in [16], where it was shown that a natural memoryless randomized algorithm, the *harmonic algorithm*, has a competitive ratio $O(k2^k)$. Using the derandomization technique of [1], this establishes that there exists a deterministic algorithm with competitive ratio $O(k^2 4^k)$. The result of [16] was improved slightly in [2] to $O(2^k \log k)$, establishing a deterministic competitive ratio of $O(4^k \log^2 k)$, which was the best known competitive ratio for the general case before this paper. Specifically for the 3-server problem, the best known upper bound was an 11-competitive algorithm for any metric space [10].

The lack of significant progress towards the k-server conjecture led to the study of special cases of the problem. One of the first results in this area [3] was a proof that the harmonic algorithm for 3 servers is competitive (although with a terribly high competitive ratio; this result preceded the work of [14, 16]). Attacking the problem in special metric spaces led to a k-competitive algorithm for the line [6], which was extended to trees [7]. Finally, an $O(k^3)$ competitive deterministic algorithm for the circle was presented in [15].

One of the problems with the known competitive algorithms for the k-server problem is that they are not space-efficient (the algorithm proved $2k-1$-competitive in this paper is no exception). In order to address this problem, [13] considered memoryless randomized algorithms and showed a competitive ratio of k for the special class of resistive metric spaces. By derandomization, this results in a $O(k^2)$ deterministic competitive ratio for resistive or approximately resistive metric spaces (one of them is the circle). Especially for the 2-server problem, [17] and [11] gave a 10-competitive and a 4-competitive efficient deterministic algorithm respectively and [8] showed that the harmonic algorithm is 3-competitive. We should also mention a series [4, 18, 19] of lower bound results for the randomized version of the k-server problem against an *oblivious adversary* and the absence of any interesting upper bound for this case.

In this paper we come very close to proving the k-server conjecture: we establish an upper bound of $2k-1$ (Theorem 1). Previous attacks on this and other on-line problems involved a *potential function*, a numerical invariant that enables the inductive proof. Our technique is based on more complex invariants, which provide valuable information about the structure of the reachable *work functions*. There are two invariants that proved crucial: A *quasiconvexity* property of the work functions, and a *duality* condition. Actually, quasiconvexity is used only in the proof of duality, and the main result follows from a potential function and the duality condition. We believe that these concepts may be of some general value and applicability. For example, using a similar technique but a different potential the exact k-server conjecture was proved for the special case of metric spaces with $k+2$ points [20, 22].

2 The Work Function Algorithm

The algorithm we employ is the *work function algorithm*, a rather natural idea for this problem that was first made explicit in the work of Chrobak and Larmore [9] and discovered independently by Fortnow, Karloff and Vishwanathan and by McGeogh and Sleator. It has already been successfully applied to other problems [5, 12]. In [9], it was shown that the Work Function Algorithm is 2-competitive for $k = 2$. One of the ingredients of our technique is the notion of the *extended cost*, a concept very similar to the *pseudocost* of [9].

Consider an optimal off-line algorithm B servicing a request sequence

$\rho = \rho_1\rho_2$. After servicing the request sequence ρ_1 the k servers of algorithm B occupy some position X. The cost of servicing ρ can be divided into two parts: the cost of servicing the request sequence ρ_1 starting at the initial configuration and ending up at X and the cost of servicing ρ_2 starting at X. An on-line algorithm A that knows algorithm B cannot know the position X, because X may depend on the future request sequence ρ_2. However, algorithm A can compute the cost of servicing ρ_1 of any possible optimal off-line algorithm. In particular, algorithm A can compute the optimal cost of servicing ρ_1 starting at A_0 and ending up at configuration Y, for every possible configuration Y. This leads to the following definition:

Definition 1 (Work function) *Fix a metric space $\mathcal{M}$ and an initial configuration A_0. For a request sequence ρ define the work function w_ρ from configurations to the nonnegative real numbers: $w_\rho(X)$ is the optimal cost of servicing ρ starting at A_0 and ending up at configuration X.*

We usually omit the subscript ρ from w_ρ when it is obvious from the context. Furthermore, for a work function $w = w_\rho$ we refer to $w' = w_{\rho r}$ as the resulting work function after request r, when ρ and r are understood from the context.

Intuitively, the importance of work functions stems from the almost obvious fact that they encapsulate all the useful information about the past; what an on-line algorithm needs to remember is w_ρ, not ρ, because any other algorithm can be transformed to one with this property without deteriorating its competitiveness.

The initial work function $w_e(X)$ of a configuration X is merely the cost of moving the servers from the initial configuration A_0 to the configuration X: $w_e(X) = D(A_0, X)$.

The value $w_{\rho r}(X)$ for some configuration X can be computed as follows: Clearly, if $r \in X$ then $w_{\rho r}(X) = w_\rho(X)$. Otherwise, if $r \notin X$, some server moved from r to some point $x \in X$ and therefore $w_{\rho r}(X) = w_{\rho r}(X - x + r) + d(r, x) = w_\rho(X - x + r) + d(r, x)$. Combining the two cases, we get:

Fact 1 *Let w be a work function; then the resulting work function w' after request r is*
$$w'(X) = \min_{x \in X}\{w(X - x + r) + d(r, x)\}.$$

We also get:

Fact 2 *If w is a work function and r is the most recent request, then for all configurations X*

$$w(X) = \min_{x \in X}\{w(X - x + r) + d(r, x)\}.$$

Recall that in the definition of $w(X)$ we require that servers end up at configuration X; this can be done by moving first to configuration Y and then to X. So we have:

Fact 3 *For a work function w and two configurations X, Y*

$$w(X) \leq w(Y) + D(X, Y).$$

Consider a work function w and the resulting work function w' after request r. By Fact 3 we get

$$w'(X) = \min_{x \in X}\{w(X - x + r) + d(r, x)\} \geq w(X)$$

which translates to:

Fact 4 *Let w be a work function and let w' be the resulting work function after request r. Then for all configurations X: $w'(X) \geq w(X)$.*

Consider a request sequence ρ and let A be the configuration of some on-line algorithm after servicing ρ. Presumably, the most natural on-line algorithm for the k-server problem is the *greedy algorithm*, which moves the closest server to a request, that is, it moves its servers to a new configuration A', with $r \in A'$, that minimizes $D(A, A')$. It is easy to see that the greedy algorithm, being too conservative, has no bounded competitive ratio. At the other end of the spectrum lies the *retrospective algorithm*: It moves its servers to a configuration A', with $r \in A'$, that minimizes $w_{\rho r}(A')$. The idea is that the off-line algorithm that has its servers at A' seems the best so far. It appears that a combination of these two algorithms may be a good idea; the work function algorithm combines the virtues of both of them:

Definition 2 (Work Function Algorithm) *Let ρ be a request sequence and let A be the configuration of an on-line algorithm after servicing ρ. The work function algorithm services a new request r by moving its servers to a configuration A', with $r \in A'$, that minimizes $w_{\rho r}(A') + D(A, A')$.*

As usual, let $w = w_\rho$ and $w' = w_{\rho r}$. Notice that since $r \in A'$ we can replace $w_{\rho r}(A')$ with $w(A')$ in the above definition. Moreover, because of the triangle inequality we can assume that $A' = A - a + r$ for some $a \in A$; $A' = A - a + r$ minimizes $w(A') + D(A, A')$. Using this we see that $w'(A) = \min_{x \in A}\{w(A - x + r) + d(x, r)\} = w(A') + d(a, r)$.

The cost of the work function algorithm to service request r is simply $d(a, r)$. In order to bound the competitive ratio of the work function algorithm, we must also consider the cost of an optimal off-line algorithm. Instead, it has proved convenient to define the *off-line pseudocost,* a simple and surprisingly accurate estimate of the off-line cost. The off-line pseudocost of the move from configuration A to A' is defined to be $w'(A') - w(A)$. It is easy to see that, by summing over all moves, the total off-line pseudocost is equal to the total off-line cost (in the worst case the final configuration of the on-line algorithm is the same with the final configuration of the optimal off-line algorithm; if this is not the case, by extending the request sequence with requests in the final configuration of the off-line algorithm, the off-line cost remains unaffected while the on-line cost increases).

Consider now the sum of the off-line pseudocost and the on-line cost:

$$w'(A') - w(A) + d(a, r)$$

which is equal to $w'(A) - w(A)$. This quantity is bounded by its maximum over all possible configurations. Therefore, the off-line pseudocost plus the on-line cost is bounded above by

$$\max_X \{w'(X) - w(X)\}$$

We call this quantity the *extended cost* of a move. The total extended cost is the sum of the extended cost of all moves. We say that the extended cost *occurs* on a configuration A when A maximizes the quantity in the extended cost.

Clearly, by the definition of the competitive ratio we have:

Fact 5 *If the total extended cost is bounded above by $c+1$ times the off-line cost plus a constant then the work function algorithm is c-competitive.*

The extended cost is an overestimation of the actual on-line cost (plus the optimal off-line cost). It was first introduced in [9] in a somehow different form (they called it *on-line pseudocost*). The advantage of using extended cost instead of real cost is that we don't have to deal at all with the configuration of the on-line servers. Instead, in order to prove that the work function algorithm is competitive, we only have to show that a certain inequality holds for all work functions. Its disadvantage, of course, is that it may overestimate the cost of the work function algorithm (although in view of our main result, the overestimation factor cannot be more than two).

3 Quasiconvexity and Duality

Facts 2 and 3 provide some properties of the work functions. Unfortunately, other functions can satisfy both of them; that is, there are functions that satisfy them and are different from w_ρ for all request sequences ρ (and for all initial configurations A_0). In order to study the behavior of the work function algorithm, it is important to understand better the properties of work functions. One very useful property is that all work functions are *quasiconvex*:

Definition 3 *A function w is called quasiconvex if for all configurations A, B there exists a bijection $h : A \to B$ such that for all bipartitions of A into X, Y:*

$$w(A) + w(B) \geq w(X \cup h(Y)) + w(h(X) \cup Y) \tag{1}$$

It is perhaps useful to visualize quasiconvexity as a discrete variant of convexity, in that the inequality above recalls the definition of convex functions ($0 \leq x \leq 1$): $w(A) + w(B) \geq w(x \cdot A) + w((1-x) \cdot B)$. In the same way that convexity guarantees that all optimal solutions lie in a compact set, (iterated application of) quasiconvexity implies that optimal configurations are transformable into one another via sequences of swaps. Notice that the union ($\cup$) in the definition denotes the union of multisets.

Before we show that all work functions are quasiconvex, we need the following lemma, which provides a stronger form of the quasiconvexity condition by restricting the set of possible bijections.

Lemma 1 *If there exists a bijection h that satisfies the conditions in Definition 3 then there exists a bijection h' that satisfies the same conditions and $h'(x) = x$ for all $x \in A \cap B$.*

Proof. Let h be a bijection from A to B that satisfies the conditions of the definition above and maps the maximum number of elements in $A \cap B$ to themselves. Assume that for some $a \in A \cap B$ we have $h(a) \neq a$. Define a bijection h' that agrees with h everywhere except that

$$h'(a) = a \qquad \text{and} \qquad h'(h^{-1}(a)) = h(a)$$

(h' interchanges the values of h on a and $h^{-1}(a)$).

Consider now a bipartition of A into X and Y and assume (without loss of generality) that $h^{-1}(a) \in X$. If $a \in X$ then $h(X) = h'(X)$ and $h(Y) = h'(Y)$ and (1) holds for h'. Otherwise, when $a \notin X$, we derive the quasiconvexity condition for X and Y from the quasiconvexity condition for $X' = X + a$ and $Y' = Y - a$ as follows:

Since, $h(Y') = h'(Y')$ and $h(X') = h'(X')$ we have that $X' \cup h(Y') = X' \cup h'(Y') = (X + a) \cup h'(Y - a) = X \cup h'(Y)$ and similarly $h(X') \cup Y' = h'(X) \cup Y$. From these we get

$$\begin{aligned} w(A) + w(B) &\geq w(X' \cup h(Y')) + w(h(X') \cup Y') \\ &= w(X \cup h'(Y)) + w(h'(X) \cup Y) \end{aligned}$$

Therefore, h' satisfies the quasiconvexity condition. Because h' maps at least one more element in $A \cap B$ to itself than h, it contradicts the assumption that h maps the maximum number of elements in $A \cap B$ to themselves.

We conclude that $h(a) = a$ for all $a \in A \cap B$, and the lemma holds. $\square$

We are now in a position to show the following important lemma:

Lemma 2 (Quasiconvexity Lemma) *All work functions are quasiconvex.*

Proof. We use induction on the number of requests.

Recall that the initial work function $w_e(X)$ of a configuration X is equal to $D(A_0, X)$, where A_0 is the initial configuration. So we have that

$$w(A) + w(B) = D(A_0, A) + D(A_0, B)$$

Fix two matchings $M(A_0, A)$ and $M(A_0, B)$ that realize the minima of $D(A_0, A)$ and $D(A_0, B)$. Each point x_j in A_0 is matched to some point a_j in A and b_j in B. Consider the bijection $h : A \to B$ that maps each a_j to b_j. For any bipartition of A into X and Y, $w(X + h(Y)) + w(h(X) + Y)$ is equal to the sum of two minima matchings between A_0, $X + h(Y)$ and A_0, $h(X) + Y$. Since we can rearrange the matchings $M(A_0, A)$ and $M(A_0, B)$ to obtain two matchings (not necessarily minima) between A_0, $X + h(Y)$ and A_0, $h(X) + Y$, it follows that $w(A) + w(B) \geq w(X + h(Y)) + w(h(X) + Y)$.

For the induction step, assume that w is quasiconvex. We want to show that the resulting w' after request r is also quasiconvex.

Fix two configurations A and B. Using Fact 1 to express w' in terms of w we get that $w'(A) = w(A - a + r) + d(r, a)$ for some $a \in A$; similarly

$w'(B) = w(B - b + r) + d(r, b)$ for some $b \in B$. The induction hypothesis is that w is quasiconvex, so there exists a bijection h from $A - a + r$ to $B - b + r$ that satisfies the quasiconvexity condition. Furthermore, Lemma 1 allows us to assume that $h(r) = r$.

Consider now a bijection $h' : A \to B$, that agrees with h everywhere except that $h'(a) = b$. We will show that h' satisfies the requirements of the quasiconvexity condition of w'. Consider a bipartition of A into X and Y and without loss of generality assume that $a \in X$. We have:

$$\begin{aligned}
& w'(A) + w'(B) \\
& \quad = w(A - a + r) + w(B - b + r) + d(r, a) + d(r, b) \\
& \quad = w((X - a + r) \cup Y) + w(B - b + r) + d(r, a) + d(r, b) \\
& \quad \geq w((X - a + r) \cup h(Y)) + w(h(X - a + r) \cup Y) \\
& \quad \quad + d(r, a) + d(r, b) \\
& \quad = w((X - a + r) \cup h'(Y)) + w((h'(X) - b + r) \cup Y) \\
& \quad \quad + d(r, a) + d(r, b) \\
& \quad \geq w'(X \cup h'(Y)) + w'(h'(X) \cup Y)
\end{aligned}$$

where the first inequality is based on the quasiconvexity of w and the second one on Fact 1. So, w' is quasiconvex and the lemma follows. □

Now we use the quasiconvexity condition to prove the next two lemmata. In fact, we use the weaker condition:

$$\forall a \in A \quad \exists b \in B : \quad w(A) + w(B) \geq w(A - a + b) + w(B - b + a)$$

We need a definition first:

Definition 4 *A configuration A is called a minimizer of a point a with respect to w, if A minimizes the expression $w(X) - \sum_{x \in X} d(a, x)$, that is*

$$w(A) - \sum_{x \in A} d(a, x) = \min_X \{w(X) - \sum_{x \in X} d(a, x)\}$$

Lemma 3 *Let w be a work function. Consider a new request at r and the resulting work function w'. If A is a minimizer of r with respect to w then A is also a minimizer of r with respect to w'.*

Proof. It suffices to show that for all configurations B:

$$w'(B) - \sum_{b \in B} d(r, b) \geq w'(A) - \sum_{a \in A} d(r, a)$$

or equivalently:

$$w'(B) - \sum_{b \in B} d(r,b) + w(A) \geq w'(A) - \sum_{a \in A} d(r,a) + w(A)$$

In order to show this we need the following: From Fact 1 we get that there exists $b' \in B$ such that

$$w'(B) = w(B) + d(r,b').$$

Using quasiconvexity, we get that there exists $a' \in A$ such that

$$w(B - b' + r) + w(A) \geq w(B - b' + a') + w(A - a' + r).$$

Finally, since A is a minimizer of r we have that

$$w(B - b' + a') - \sum_{b \in B - b' + a'} d(r,b) \geq w(A) - \sum_{a \in A} d(r,a).$$

Putting all these together:

$$\begin{aligned}
&w'(B) + w(A) - \sum_{b \in B} d(r,b) \\
&\quad = w(B - b' + r) + d(r,b') + w(A) - \sum_{b \in B} d(r,b) \\
&\quad = w(B - b' + r) + w(A) - \sum_{b \in B - b' + r} d(r,b) \\
&\quad \geq w(B - b' + a') + w(A - a' + r) - \sum_{b \in B - b' + r} d(r,b) \\
&\quad = w(B - b' + a') + w(A - a' + r) + d(r,a') - \sum_{b \in B - b' + a'} d(r,b) \\
&\quad \geq w(A) + w(A - a' + r) + d(r,a') - \sum_{a \in A} d(r,a) \\
&\quad \geq w(A) + w'(A) - \sum_{a \in A} d(r,a)
\end{aligned}$$

where the last inequality is based on Fact 1. The lemma follows. □

The following lemma has the same premises with Lemma 3, but a different conclusion:

Lemma 4 *Let w be a work function. Consider a new request at r and the resulting work function w'. If A is a minimizer of r with respect to w then the extended cost occurs at A, that is*

$$w'(A) - w(A) = \max_X \{w'(X) - w(X)\}$$

Proof. The proof is rather similar to the proof of Lemma 3. Notice first that it suffices to show that for all configurations B:

$$w'(A) + w(B) \geq w'(B) + w(A)$$

By Fact 1 we get that there exists $a' \in A$ such that

$$w'(A) = w(A - a' + r) + d(r, a').$$

Using quasiconvexity, we also get that there exists $b' \in B$ such that

$$w(A - a' + r) + w(B) \geq w(A - a' + b') + w(B - b' + r).$$

Finally, since A is a minimizer of r with respect to w: $w(A - a' + b') - \sum_{a \in A - a' + b'} d(r, a) \geq w(A) - \sum_{a \in A} d(r, a)$, which is equivalent to

$$w(A - a' + b') + d(r, a') \geq w(A) + d(r, b').$$

Combining all these we get:

$$\begin{aligned} w'(A) + w(B) &= w(A - a' + r) + d(r, a') + w(B) \\ &\geq w(A - a' + b') + d(r, a') + w(B - b' + r) \\ &\geq w(A) + d(r, b') + w(B - b' + r) \\ &\geq w(A) + w'(B) \end{aligned}$$

Again, the last inequality is based on Fact 1. □

Lemmata 3 and 4 can be combined into the following result which characterizes where the extended cost occurs.

Lemma 5 (Duality Lemma) *Let w be a work function and let w' be the resulting work function after request r. Then any minimizer A of r with respect to w is also a minimizer of r with respect to w', and the extended cost of servicing the request r occurs on A.*

We call this the "duality lemma" because it relates a maximum (extended cost) to a minimum (minimizer).

4 A Potential for $(2k-1)$-Competitiveness

We are now ready for the last act of the proof, the definition of an appropriate potential. For configurations $U = \{u_1, \ldots, u_k\}$ and $B_i = \{b_{i1}, \ldots, b_{ik}\}$, $i = 1, \ldots, k$, let

$$\Psi(w, U, B_1, \ldots, B_k) = kw(U) + \sum_{i=1}^{k} \left(w(B_i) - \sum_{j=1}^{k} d(u_i, b_{ij}) \right)$$

Let $\Phi(w)$ denote its minimum value over all configurations U and B_i, $i = 1, \ldots, k$; $\Phi(w)$ is called the *potential* of the work function w[1].

The next two lemmata provide some properties of $\Phi(w)$.

Lemma 6 *For any work function w, the minimum value $\Phi(w)$ of $\Psi(w, U, B_1, \ldots, B_k)$ is achieved for some U that contains the most recent request r.*

Proof. By Fact 2, for some $i \in 1 \ldots k$:

$$w(U) = w(U - u_i + r) + d(r, u_i)$$

If we substitute this to $\Psi(w, U, B_1, \ldots, B_k)$, using the k triangle inequalities $d(r, u_i) - d(u_i, b_{ij}) \geq -d(r, b_{ij})$ we get

$$\Psi(w, U, B_1, \ldots, B_k) \geq \Psi(w, U - u_i + r, B_1, \ldots, B_k)$$

and the lemma follows since $r \in U - u_i + r$. $\square$

The next lemma estimates the potential of the initial work function.

Lemma 7 *For the initial work function $w_e(X) = D(A_0, X)$:*

$$\Phi(w_e) = -2C(A_0)$$

Proof. It is not hard to see that the lemma follows if the minimum value $\Phi(w_e)$ of $\Psi(w, U, B_1, \ldots, B_k)$ is achieved when $U = A_0$ and $B_j = A_0$ for $j = 1, \ldots, k$. Consider a point $u_i \in U$. In the minimum matching $D(A_0, U)$, u_i is matched to some point $a \in A_0$. By using the k triangle inequalities $d(u_i, b_{ij}) \leq d(a, u_i) + d(a, b_{ij})$ we see that we can replace u_i with a without increasing the value of $\Psi(w, U, B_1, \ldots, B_k)$. Therefore, the minimum $\Phi(w_e)$ of $\Psi(w, U, B_1, \ldots, B_k)$ is achieved for $U = A_0$. Similarly, we can show that $B_i = A_0$ for $i = 1, \ldots, k$ and the lemma follows. $\square$

We are now ready to prove our main result:

[1]Our potential differs from what is usually termed as "potential function" in the literature of on-line problems by a constant multiple of the optimal off-line cost.

Theorem 1 *The competitive ratio of the Work Function Algorithm is at most* $(2k-1)$.

Proof. Consider a work function w and let w' be the resulting work function after request r.

According to Lemma 6, the minimum value $\Phi(w')$ of $\Psi(w', U, B_1, \ldots, B_k)$ is achieved for $u_i = r$, for some i. Let A be a minimizer of r with respect to w. Then by Lemma 5, A is also a minimizer of r with respect to w' and it is not difficult to see that the minimum value of $\Psi(w', U, B_1, \ldots, B_k)$ is unaffected if we fix $B_i = A$. Fix the remaining points u_j and b_{jl}, where $\Psi(w', U, B_1, \ldots, B_k)$ achieves its minimum. Let $\Psi_{w'}$, Ψ_w denote the values of Ψ on these points with respect to w' and w. From the definition of $\Phi(w)$ we get that $\Phi(w) \leq \Psi_w$. Obviously then,

$$\Phi(w') - \Phi(w) \geq \Psi_{w'} - \Psi_w \tag{2}$$

Consider now the expression $\Psi_{w'} - \Psi_w$. All distances appearing in the definition of $\Psi_{w'}$ appear also in the definition of Ψ_w, because they are defined on the same set of configurations U, B_j, $j = 1, \ldots, k$. Therefore they cancel out. By Fact 4, $w'(U) \geq w(U)$ and $w'(B_j) \geq w(B_j)$, $j = 1, \ldots, k$. From this we get:

$$\Psi_{w'} - \Psi_w \geq w'(A) - w(A) \tag{3}$$

Putting (2) and (3) together:

$$\Phi(w') - \Phi(w) \geq w'(A) - w(A)$$

According to Lemma 5, the extended cost is $w'(A) - w(A)$, because A is a minimizer of r with respect to w. Thus, we conclude that the extended cost to service request r is bounded above by $\Phi(w') - \Phi(w)$. Summing over all moves we get that the total extended cost is bounded above by $\Phi(w_\rho) - \Phi(w_e)$, where w_e and w_ρ are the initial and the final work functions, respectively.

Let A_0 and A_n be the initial and final configurations (recall that without loss of generality the off-line algorithm ends up in the same configuration with the on-line algorithm). We have

$$\begin{aligned} \Phi(w_\rho) &\leq \Psi(w_\rho, A_n, A_n, \ldots, A_n) \\ &= 2kw_\rho(A_n) - 2C(A_n) \\ &\leq 2kw_\rho(A_n) \end{aligned}$$

The value of $\Phi(w_e)$ is given by Lemma 7, $\Phi(w_e) = -2C(A_0)$. Therefore, the extended cost is at most $2kw_\rho(A_n) + 2C(A_0)$. Because the off-line cost is $w_\rho(A_n)$, the total extended cost is bounded above by $2k$ times the off-line cost plus a constant depending only on the initial configuration. Using Fact 5, we conclude that the Work Function Algorithm is $(2k-1)$-competitive. □

5 Research Directions

We believe that the k-server conjecture is true (and that in fact the work-function algorithm is k-competitive); however, it now seems that substantial extension of our proof will be needed for its precise settlement. A possible research direction that would achieve potentially interesting partial results would extend the special cases of metric spaces for which the k-server conjecture holds. One such special case (metric spaces with $k+2$ points [22]) was in fact a precursor of the present proof.

Finally, much work remains to be done on bridging the gap between the performance of on-line algorithms and the computational processes (such as paging algorithms) that they are supposed to model. Two extensions of competitive analysis that make some progress in this direction are proposed amd explored in [23].

References

[1] S. Ben-David, A. Borodin, R. Karp, G. Tardos, and A. Wigderson. On the power of randomization in on-line algorithms. *Algorithmica*, 11(1):2–14, January 1994.

[2] Y. Bartal and E. Grove. The Harmonic k-server algorithm is competitive. *Personal Communication.*

[3] P. Berman, H. J. Karloff, and G. Tardos. A competitive three-server algorithm. *Proceedings of the First Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 280–90, 1990.

[4] A. Blum, H. Karloff, Y. Rabani, and M. Saks. A decomposition theorem and bounds for randomized server problems. *Proceedings 33rd Annual Symposium on Foundations of Computer Science*, pages 197–207, 1992.

[5] W. R. Burley. Traversing layered graphs using the work function algorithm. *Technical report CS93-319, Dept. of Computer Science and Engineering University of California, San Diego*, 1993.

[6] M. Chrobak, H. J. Karloff, T. Payne, and S. Vishwanathan. New results on server problems. *SIAM Journal on Discrete Mathematics*, 4:172–81, 1991.

[7] M. Chrobak and L. L. Larmore. An optimal on-line algorithm for k-servers on trees. *SIAM Journal on Computing*, 20(1):144–8, February 1991.

[8] M. Chrobak and L. L. Larmore. Harmonic is 3-competitive for two servers. *Theoretical Computer Science*, 98(2):339–46, May 1992.

[9] M. Chrobak and L. L. Larmore. The server problem and on-line games. *On-line algorithms: proceedings of a DIMACS workshop. DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 7:11–64, 1992.

[10] M. Chrobak and L. L. Larmore. Generosity helps or an 11-competitive algorithm for three servers. *Journal of Algorithms*, 16(2):234–63, March 1994.

[11] M. Chrobak and L. L. Larmore. On fast algorithms for two servers. *Journal of Algorithms*, 12(4):607–14, December 1991.

[12] M. Chrobak, L. L. Larmore, N. Reingold, and J. Westbrook. Page migration algorithms using work functions. *Algorithms and Computation. 4th International Symposium, ISAAC '93 Proceedings*, pages 406–15, 1993.

[13] D. Coppersmith, P. Doyle, P. Raghavan, and M. Snir. Random walks on weighted graphs and applications to on-line algorithms. *Journal of the Association for Computing Machinery*, 40(3):421–53, July 1993.

[14] A. Fiat, Y. Rabani, and Y. Ravid. Competitive k-server algorithms. *Proceedings. 31st Annual Symposium on Foundations of Computer Science*, pages 454–63 vol.2, 1990.

[15] A. Fiat, Y. Rabani, Y. Ravid, and B. Schieber. A deterministic $O(k^3)$-competitive k-server algorithm for the circle. *Algorithmica*, 11(6):572–78, June 1994.

[16] E. Grove. The Harmonic online k-server algorithm is competitive. *Proceedings 23rd Annual ACM Symposium on Theory of Computing*, pages 260–66, 1991.

[17] S. Irani and R. Rubinfeld. A competitive 2-server algorithm. *Information Processing Letters*, 39(2):85–91, July 1991.

[18] A. R. Karlin, M. S. Manasse, L. A. McGeoch, and S. Owicki. Competitive randomized algorithms for nonuniform problems. *Algorithmica*, 11(6):542–71, June 1994.

[19] H. Karloff, Y. Rabani, and Y. Ravid. Lower bounds for randomized k-server and motion-planning algorithms. *SIAM Journal on Computing*, 23(2):293–312, April 1994.

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

[21] E. Koutsoupias and C. H. Papadimitriou. On the k-server conjecture. *Proceedings 26th Annual ACM Symposium on Theory of Computing*, pages 507–11, 1994.

[22] E. Koutsoupias and C. H. Papadimitriou. The 2-evader problem. *In preparation.*

[23] E. Koutsoupias and C. H. Papadimitriou. Beyond competitive analysis. In *Proc. 35th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 394–400, 1994.

[24] M. S. Manasse, L. A. McGeoch, and D. D. Sleator. Competitive algorithms for on-line problems. *Proceedings 20th Annual ACM Symposium on Theory of Computing*, pages 322–33, 1988.

[25] M. S. Manasse, L. A. McGeoch, and D. D. Sleator. Competitive algorithms for server problems. *Journal of Algorithms*, 11(2):208–30, June 1990.

[26] D. D. Sleator and R. E. Tarjan. Amortized efficiency of list update and paging rules. *Comm. ACM*, 28(2):202–208, 1985.