Generosity Helps, or an 11-Competitive Algorithm for Three Servers

Marek Chrobak* Lawrence L. Larmore†

UCR-CS-93-4

*Department of Computer Science, University of California, Riverside. Research partially supported by NSF grant CCR9112067.

†Department of Computer Science, University of California, Riverside. Research partially supported by NSF grant CCR9112067.

Abstract

We propose a new algorithm called *Equipoise* for the k-server problem, and we prove that it is 2-competitive for two servers and 11-competitive for three servers. For $k = 3$, this is a tremendous improvement over previously known constants. The algorithm uses several techniques for designing on-line algorithms – convex hulls, work functions and forgiveness.

1 Introduction

The k-server problem was introduced by Manasse, McGeoch, and Sleator [13]. A given problem instance consists of a positive integer k, and a metric space M. The data consist of an initial configuration $S^0 \in \Lambda^k M$, (where $\Lambda^k X$ = the set of all multisets of order k within a set X) and a sequence of *requests* $r^1, r^2, \ldots r^m \in M$. We think of S^0 as the initial positions of k movable *servers*. After each request r^t we must move one server to r^t in order to "serve" the request. We are also free to move the other servers. Cost is defined as the total distance moved by all servers.

More formally, the output consists of a sequence $S^1, S^2, \ldots S^m \in \Lambda^k M$, such that $r^t \in S^t$, and cost is measured as $\sum_{t=1}^{m} S^{t-1}S^t$, where, for $S, S' \in \Lambda^k M$, SS' is the "minimum matching" distance between the two multisets.

The task is to design an *on-line algorithm* for this problem, that is an algorithm that outputs the move in response to r^t before receiving the next request r^{t+1}. It is not hard to see that for $k \geq 2$ there is no on-line algorithm that computes the optimal schedule. This motivates the search of algorithms that are "almost" optimal, that is, algorithms that compute a schedule within a constant factor of the minimum.

More precisely, an on-line k-server algorithm $\mathcal{A}$ is called *c-competitive* if for each configuration $X \in \Lambda^k M$

$$\sup_{\varrho \in M^*} \{cost_{\mathcal{A}}(X, \varrho) - c \cdot cost_{opt}(X, \varrho)\} \ < \ \infty,$$

where $cost_{\mathcal{A}}(X, \varrho)$ is the cost of the schedule computed by $\mathcal{A}$ starting from X on request sequence ϱ, and $cost_{opt}(X, \varrho)$ is the corresponding optimal schedule.

Manasse, McGeoch, and Sleator [13] proved that if $c < k$ then there does not exist a c-competitive algorithm for k servers. They also gave a 2-competitive algorithm for 2 servers. Another optimal solution was presented in [5]. The problem of whether there is a k-competitive algorithm for k servers if $k \geq 3$ remains open, and is known as the *k-server problem.*

A great deal of effort has been made to improve the time and space efficiency of 2-server algorithms [6, 12], including randomized algorithms [4, 14].

For some special cases, k-competitive algorithms have been found. Chrobak *et al.* [3] gave a k-competitive algorithm for the line and the weighted cache problem. This was generalized later to trees in [7]. Coppersmith *et al.* [9] discovered a randomized algorithm with competitiveness constant k that works for a class of metric spaces called *resistive.*

Although, for arbitrary k, no k-competitive algorithm is known, there are already some results in this direction. Fiat *et al.* [10] have presented a competitive algorithm for k servers whose constant is an exponential function of k. Grove [11] has proven that the harmonic algorithm is competitive for any k (also with an exponential constant). His result gives another competitive deterministic

algorithm via the technique developed in [1].

In this paper we give an 11-competitive algorithm for three servers. Although the algorithm is probably not optimal (the conjectured constant is 3), our constant, 11, is much smaller than previously known values [2, 10, 11] of competitiveness for arbitrary metric spaces. The best of these constants, given by Grove [11], is 576, and the proof is not constructive, *i.e.*, no explicit algorithm is given.

The time complexity of our algorithm is $O(n^2)$ for each step, where n is the number of "relevant" points, *i.e.*, points which are either positions of our servers initially, or have been requested.

Besides the algorithm itself, we present some techniques that we believe should find applications in the design of on-line algorithms (in fact, some of them already have). These include:

Work Functions. A work function describes the optimal cost of serving a given sequence of requests, from a given initial state to any given final state. A work function describes all relevant information about the behavior of the adversary up to now. (A similar concept was used by Manasse, McGeoch and Sleator [13].) The optimal cost of serving the sequence of past requests is the infimum of the current work function.

Forgiveness. A work function gives all potentially useful information about the adversary's past behavior, and it may have a very complicated form. Every local minimum of a work function is a possible configuration of the adversary servers. This makes the design of an algorithm and proof of its competitiveness difficult. The idea of "forgiveness" is to lower the values of the work function on some configurations, causing the work function to have a simpler form. That may, in turn, make us decrease the total we charge the adversary – in fact, it may even cause some of the moves to have *negative* cost allocated to the adversary.

Another, useful, way to view this method is to imagine that we play against a *turbo-adversary* that, due to forgiveness, never pays more than the optimal cost, and sometimes pays less. If we apply an appropriate forgiveness strategy, we gain extra knowledge about possible turbo-adversary configurations. If our algorithm is c-competitive against such a turbo-adversary, then it is certainly c-competitive.

An example of a forgiveness operation is shown in Figure 1, where the space of server configurations is viewed, symbolically, as the real line. Before forgiveness, the optimal algorithm may be in configurations X, Y, Z or T. After forgiveness it can be in configurations U or V. Because of the forgiveness operation, the infimum of the current work function will decrease by d, and thus the adversary cost will also decrease by d.

In this paper we carry the forgiveness operation to the extreme. Our algorithm, *Equipoise*, given a current work function ω, forgives ω to a "characteristic" function[1] of some carefully chosen

[1] A formal definition of this term will be given in section 3. Essentially, if the work function is χ_X, the characteristic function of some configuration X, we may assume, without danger of increasing our cost, that the adversary is at X.

Figure 1: An example of forgiveness.

configuration X. This means that we *know the exact position of the turbo-adversary servers* at every step.

Convex Hulls. It turns out that sometimes it is not possible to find a configuration X such that the forgiveness to χ_X gives the desired effect. We solve this difficulty by embedding M into a bigger metric space called the *convex hull of* M, and finding X in this convex hull. Convex hulls were introduced in [5], where they were used to design an optimal algorithm for two servers.

As it turns out, for two servers, Equipoise is equivalent to our optimal algorithm from [5]. For $k = 3$ we obtain an 11-competitive algorithm. We also show that our analysis is tight, in the sense that, with our forgiveness strategy, Equipoise is no better than 11-competitive.

The algorithm and correctness proofs can be presented without explicit use of work functions and forgiveness. This is due to the extreme nature of our forgiveness strategy, since each work function after forgiveness can be identified with a configuration of our servers. Nevertheless, both tools were instrumental in discovering the algorithm, and are necessary for a complete understanding of the proofs. We give more details in Section 3.

2 The Adversary Pseudo-Cost

Let $\mathcal{A}$ be an arbitrary k-server algorithm. Instead of comparing $\mathcal{A}$'s cost to the optimal cost, we will define a new quantity which we call *adversary pseudo-cost* for each of $\mathcal{A}$'s moves.

Let S be the current configuration of our servers, let r be the new request point and suppose that our algorithm moves one server so that the new configuration is S', with $r \in S'$. $\mathcal{A}$'s cost is $SS' =$

the total distance moved by our servers, while the adversary pseudo-cost is defined to be

$$\nabla(S, r, S') = \min_{X \ni r} \{SX - S'X\}.$$

Note that, without loss of generality, the minimum may be taken only over configurations $r \in X \subseteq \{r\} \cup S$, of which there are just k choices. Note also that, unlike the optimal cost, the adversary pseudo-cost depends not only on the request sequence but also on our strategy $\mathcal{A}$.

Lemma 2.1 *For any sequence of requests, the total adversary pseudo-cost for any k-server algorithm $\mathcal{A}$ is less than or equal to the optimum cost for that request sequence.*

Proof: Let S^0 be the initial configuration of our servers. The adversary's initial configuration A^0 is the same, that is $A^0 = S^0$. Let $\varrho = r^1 \ldots r^m$ be the request sequence. After request r^t our servers are in configuration S^t and the adversary servers are in configuration A^t. By the triangle inequality

$$\begin{aligned}
\sum_{t=1}^{m} \nabla(S^{t-1}, r^t, S^t) &= \sum_{t=1}^{m} \min_{X \ni r^t} \{S^{t-1}X - S^tX\} \leq \sum_{t=1}^{m} (S^{t-1}A^t - S^tA^t) \\
&= \sum_{t=1}^{m} (S^{t-1}A^t - S^{t-1}A^{t-1}) - S^mA^m \leq \sum_{t=1}^{m} A^{t-1}A^t = cost_{opt}(\varrho)
\end{aligned}$$

□

3 Work Functions and Forgiveness

In this section we explain the intuitions behind the definition of the adversary pseudo-cost. In order to do this, we need to introduce two more notions: work functions and forgiveness.

These notions are not formally necessary for the description of the algorithm, but we feel that grasping them will greatly aid the reader in understanding Equipoise.

Work functions. A *work function* is a non-negative real-valued function $\omega : \Lambda^k M \to \mathbf{R}^+$ such that, for any two server configurations X and Y

$$\omega(X) - \omega(Y) \leq XY.$$

We call this the *slope condition*. We can think of $\omega(X)$ as a conditional obligation of the adversary – the amount of cost we can assume the adversary must have incurred if he is at configuration X.

If ω is a work function and $r \in M$, we define another work function $\omega \wedge r$, the *update of ω by r*, by

$$(\omega \wedge r)(X) = \min_{Y \ni r} \{\omega(Y) + XY\}.$$

Note that the minimum can be taken over only the k choices of Y which lie in $X \cup \{r\}$.

Informally, if ω is the system of conditional obligations of the adversary, then $\omega \wedge r$ is the new system of conditional obligations after one more request, r. The update operator can be extended to arbitrary sequences by setting $\omega \wedge \varepsilon = \omega$ (where ε is the empty sequence) and $\omega \wedge (\varrho r) = (\omega \wedge \varrho) \wedge r$. If ω is the *characteristic function* of X, χ_X, defined by $\chi_X(Y) = XY$, we can assume, without danger of increasing the competitiveness, that the adversary is at X. For example, initially, the work function is χ_{S^0}.

We can express the optimal cost of serving a given sequence of request using work functions. If ω is the current work function, the optimal cost of serving the past sequence of requests is simply $\inf \omega$.

An *offset function* is a work function whose infimum is zero. In an algorithm it is more convenient to deal with offset functions instead of work functions. Given an offset function ω we define an operator $\triangle$ as follows:

$$(\omega \triangle r)(X) = (\omega \wedge r)(X) - \inf(\omega \wedge r).$$

Then $\omega \triangle r$ will be the offset function after request r. Also, we can define the optimal cost of serving this request to be $\inf(\omega \wedge r)$. The optimal cost of serving a sequence of requests $\varrho \in M^*$ will be the sum of optimal costs of serving the individual requests which constitute ϱ.

Forgiveness. Consider an algorithm $\mathcal{A}$ that always remembers the current offset function ω and uses this function to decide the next move. Let r be the new request. A *forgiveness operation* is to replace the work function $\omega \wedge r$ by a new function $\mu \leq \omega \wedge r$. Then the new offset function will be $\omega' = \mu - \inf \mu$. The algorithm will then base its next decision on ω', as if ω' were the new offset function, not $\omega \triangle r$. This operation can only decrease the adversary cost, since $\inf \mu \leq \inf(\omega \wedge r)$. Thus if such an algorithm $\mathcal{A}$ is c-competitive with forgiveness then it must be also c-competitive (that is, without forgiveness).

The extreme forgiveness strategy is to choose $\mu = \chi_X + a$, for some configuration X and constant a. Then the cost we charge the adversary will be $\inf \mu = a$. Notice that if χ_X is the current offset function, we can as well assume that the adversary is in configuration X, and $\mathcal{A}$ should move its servers to configuration X, too. (Of course, in order to have a competitive algorithm we need to choose X very carefully. The choice of this configuration will be described in the next section).

Now we will determine how to compute the cost we charge the adversary when we use this forgiveness strategy. Assume that currently our servers are in configuration S, and $\omega = \chi_S$ is the current offset function. Let r be the new request point. The new work function is $\nu = \chi_S \wedge r$. Our algorithm $\mathcal{A}$ chooses a new configuration S', forgives ν to $\mu = \chi_{S'} + a$, where a is largest possible so that $\mu \leq \nu$, and moves its servers to S'. The new offset function is $\omega' = \chi_{S'}$ and, as explained before, the adversary cost of this move is a. Thus we only need to determine the value of a.

First notice that in order to have $\mu \leq \nu$ we only need to make sure that $\mu(X) \leq \nu(X)$ for $X \ni r$ (because of the slope condition). Also, if $r \in X$ then $\nu(X) = \chi_S(X) = SX$. Thus for $X \ni r$ we must

have

$$S'X + a \;=\; \mu(X) \;\leq\; \nu(X) \;=\; SX.$$

And furthermore we want a to be the largest number satisfying the above inequality (so that we charge the adversary as much as possible). Thus $a = \min_{X \ni r}\{SX - S'X\}$, the adversary pseudo-cost.

Overall, this shows that that the pseudo-cost function, defined in the previous section, is precisely the forgiven adversary cost, if in our forgiveness strategy we use characteristic functions.

4 Convex Hulls

Let M be a metric space. By xy we denote the distance between two points $x, y \in M$. We shall use the word *isometry* to mean a one-to-one and onto function from one metric space to another which preserves distances. An *embedding* shall be an isometry from one metric space to a subset of another, *i.e.*, a function which preserves distances, but is not necessarily onto.

Let $\Delta(a, b, c)$ be the predicate, defined for all real a, b, c, which is true if and only if there could exist a triangle whose sides have lengths a, b, c. Thus, $\Delta(a, b, c)$ if and only if $a \leq b + c$, $b \leq c + a$, and $c \leq a + b$. We define $\mathbf{Cl}(M)$, the *closure* of M, to be the set of all real-valued functions u on M for which

$$\forall x, y \in M \;:\; \Delta\left(u(x), u(y), xy\right).$$

The next lemma describes the structure of $\mathbf{Cl}(M)$.

Lemma 4.1 $\mathbf{Cl}(M)$ *is a metric space under the sup-metric:* $uv = \sup_{x \in M} |u(x) - v(x)|$ *for* $u, v \in \mathbf{Cl}(M)$.

Proof: We need only show that distances are finite. For any $x, y \in M$

$$u(x) - v(x) \;\leq\; u(x) - v(x) + [u(y) - u(x) + xy] + [v(y) + v(x) - xy] \;=\; u(y) + v(y)$$

Thus, $\sup(u - v) \leq \inf(u + v)$, hence uv is finite. □

The relation "$\leq$" on $\mathbf{Cl}(M)$, defined by $u \leq v$ iff $u(x) \leq v(x)$ for each $x \in M$, is a partial order. We also write $u < v$ if $u \leq v$ and if $u(x) < v(x)$ for some $x \in M$. We define $\mathbf{CH}(M)$, the *convex hull* of M, to be the set of all members of $\mathbf{Cl}(M)$ which are minimal under this partial order. Note that M embeds isometrically into $\mathbf{CH}(M)$ by mapping each $x \in M$ onto its *characteristic function* χ_x, defined by $\chi_x(y) = xy$. For if $x, y \in M$, then we have $\chi_x \chi_y = \sup_z\{|xz - yz|\} = xy$. To simplify notation we will often identify each x with χ_x, and thus we can write $M \subseteq \mathbf{CH}(M)$.

For some metric spaces the convex hull is identical (isometric) to the original metric space. For example, the convex hull of $\mathbf{R}$ with the usual metric is $\mathbf{R}$ itself. More generally, the convex hull of a tree T (as defined in [7]) is T. The same is true for $\mathbf{R}^n$ with the sup-metric. However it is not true for $\mathbf{R}^n$, $n > 1$, with the Euclidean metric.[2] The structure of convex hulls of finite spaces is considered in Theorem 4.1.

For each $u \in \mathbf{Cl}(M)$, define the *slack function* $u^\star\ M : \to \mathbf{R}^+$ by

$$u^\star(x) = \inf_{y \in M} \frac{u(x) + u(y) - xy}{2}.$$

Note that, by the triangle inequality, $u^\star \geq 0$. Also, $u^\star$ is bounded because for each $x, y \in M$ we have $2 \cdot u^\star(x) \leq u(x) + u(y) - xy \leq 2 \cdot u(y)$, implying that $\sup(u^\star) \leq \inf(u)$.

Lemma 4.2 *Let $u \in \mathbf{Cl}(M)$. Then $u \in \mathbf{CH}(M)$ iff $u^\star = 0$. In particular, if M is finite then $u \in \mathbf{CH}(M)$ iff $\forall x \in M\ \exists y \in M\ :\ u(x) + u(y) = xy$.*

Proof: The second part follows easily from the first, so it is enough to prove only the first part.

If $u > v \in \mathbf{Cl}(M)$, pick $x \in M$ such that $u(x) > v(x)$. Then

$$\begin{aligned} 2 \cdot u^\star(x) &= \inf_{y \in M}\{u(x) + u(y) - xy\} = u(x) - v(x) + \inf_{y \in M}\{v(x) + u(y) - xy\} \\ &\geq u(x) - v(x) + \inf_{y \in M}\{v(x) + v(y) - xy\} > 0. \end{aligned}$$

Conversely, if $u^\star$ is not identically zero, let $v = u - u^\star$. We will show that $v \in \mathbf{Cl}(M)$. Since $u^\star(x) + u^\star(y) \leq u(x) + u(y) - xy$, we have $v(x) + v(y) \geq xy$. We also have to prove that $v(x) - v(y) \leq xy$. Choose an arbitrary $\epsilon > 0$. By the definition of $u^\star$, there exists a point $z \in M$ such that $[u(x) + u(z) - xz]/2 - u^\star(x) \leq \epsilon$. Then we have

$$\begin{aligned} v(x) - v(y) &= u(x) - u(y) + u^\star(y) - u^\star(x) \\ &\leq u(x) - u(y) + \tfrac{1}{2} \cdot [u(y) + u(z) - yz] - \tfrac{1}{2} \cdot [u(x) + u(z) - xz] + \epsilon \\ &= \tfrac{1}{2} \cdot [u(x) - u(y)] + \tfrac{1}{2} \cdot [xz - yz] + \epsilon \leq xy + \epsilon. \end{aligned}$$

Letting $\epsilon \to 0$ we obtain $v(x) - v(y) \leq xy$. This completes the proof that $v \in \mathbf{Cl}(M)$.

Since $v < u$ and $v \in \mathbf{Cl}(M)$, u cannot minimal in $\mathbf{Cl}(M)$, hence $u \notin \mathbf{CH}(M)$. $\square$

Lemma 4.3 *If $v \in \mathbf{Cl}(M)$ then there exists $u \leq v$ such that $u \in \mathbf{CH}(M)$.*

Proof: Let $v_0 = v$, and $v_{k+1} = v_k - v_k^\star$. First we show that $v_{k+1}^\star \leq v_k^\star/2$:

$$\begin{aligned} 2 \cdot v_{k+1}^\star(x) &= \inf_{y \in M}\{v_{k+1}(x) + v_{k+1}(y) - xy\} = \inf_{y \in M}\{v_k(x) + v_k(y) - xy - v_k^\star(x) - v_k^\star(y)\} \\ &\leq \inf_{y \in M}\{v_k(x) + v_k(y) - xy\} - v_k^\star(x) = 2 \cdot v_k^\star(x) - v_k^\star(x) = v_k^\star(x). \end{aligned}$$

[2] If it were, $\mathbf{R}^n$ (by the infinite version of Lemma 4.4) would have to contain a copy of the convex hull of a uniform space with three points, which it does not.

Let $u = \lim_{k\to\infty} v_k$. Then for each k we have $u \le v_k$, implying that $u^\star \le v_k^\star$. This, in turn, yields $u^\star(x) \le \lim_{k\to\infty} v_k^\star(x) = 0$. By Lemma 4.2, $u \in \mathbf{CH}(M)$. $\square$

If x, y, z are points in an arbitrary metric space, we say that y is *between* x and z if $xz = xy + yz$.

Lemma 4.4 *Let X be a finite metric space, and $X' = X \cup \{x'\}$ for $x' \notin X$. Then the function I on $\mathbf{CH}(X)$ defined by the formula*

$$\begin{aligned} I(u)(x) &= u(x) \quad \text{for } x \in X \\ I(u)(x') &= \max_{x\in X}\{xx' - u(x)\} \end{aligned}$$

is an embedding of $\mathbf{CH}(X)$ into $\mathbf{CH}(X')$. Furthermore, if for each $x \in X$ there exists $y \in X$ such that x' is between x and y, then I is an isometry.

Proof: Let $u \in \mathbf{CH}(X)$, and let $v = I(u)$. We first show $v \in \mathbf{Cl}(X')$. It suffices to show that $\Delta(v(x), v(x'), xx')$ for any $x \in X$. Fix $x \in X$, and pick $y \in X$ such that $v(x') = x'y - u(y)$. Then

$$v(x') - xx' \;\le\; v(x') + xy - x'y \;=\; xy - u(y) \;\le\; u(x) \;=\; v(x).$$

Since $u^\star(x) = 0$, we can pick $z \in X$ such that $u(x) + u(z) = xz$. Then

$$v(x') \;\ge\; x'z - u(z) \;=\; x'z - xz + u(x) \;\ge\; u(x) - xx'.$$

Finally

$$v(x) + v(x') \;\ge\; u(x) + xx' - u(x) \;=\; xx'.$$

This shows that $v \in \mathbf{Cl}(X')$.

It remains to show that v is minimal. We use Lemma 4.2. Since $v(x') + v(y) = x'y$, $v^\star(x') = 0$, while $v^\star(x) \le u^\star(x) = 0$ for $x \in X$. Thus $v \in \mathbf{CH}(X')$.

Now we show that I is an embedding. Let $u, u' \in \mathbf{CH}(X)$, $v = I(u)$ and $v' = I(u')$. The inequality $uu' \le vv'$ is obvious, since $X \subseteq X'$. To prove $vv' \le uu'$, we need only show that $|v(x') - v'(x')| \le uu'$. Without loss of generality, $v'(x') \ge v(x')$. Choose $y \in X$ such that $v(x') = x'y - u(y)$. Then

$$v'(x') - v(x') \;\le\; [x'y - u'(y)] - [x'y - u(y)] \;\le\; u'(y) - u(y) \le uu'.$$

Finally we prove the second part of the lemma. Suppose that for any $x \in X$, x' lies between x and some $y \in X$. Let $v \in \mathbf{CH}(X')$ and $u = v_{|X}$. We will show that $u \in \mathbf{CH}(X)$ and that $v = I(u)$.

It is routine to check that $u \in \mathbf{Cl}(X)$. The proof that $u \in \mathbf{CH}(X)$ is based on Lemma 4.2. Fix $x \in X$. Since $v \in \mathbf{CH}(X')$, we have $u(x) + v(z) = xz$ for some $z \in X'$. If $z \in X$, we are done.

Otherwise, if $z = x'$, by our assumption about x', there exists $y \in X$ such that $xx' + x'y = xy$. Then we have

$$\begin{aligned} u(x) + u(y) &= [xx' - v(x')] + u(y) \;=\; [xy - x'y] - v(x') + u(y) \\ &= xy - [v(x') + x'y - v(y)] \;\leq\; xy, \end{aligned}$$

hence $u(x) + u(y) = xy$. We conclude that $u \in \mathbf{CH}(X)$.

We have that $u \in \mathbf{CH}(X)$ by Lemma 4.2. From

$$I(u)(x') \;=\; \min_{x \in X}\{xx' - v(x)\} \leq v(x')$$

we obtain $I(u) \leq v$. By minimality of v, $I(u) = v$. □

Lemma 4.5 *If $X = \{x_1, \ldots x_n\}$ is any finite metric space, and if $U \subseteq \mathbf{CH}(X)$ is finite, say $|U| = m$, then there exists an isometry $J : \mathbf{CH}(U) \subseteq \mathbf{CH}(X)$ such that*

(a) *$J(u) = u$ for any $u \in U$*

(b) *For any given $\xi \in \mathbf{CH}(U)$, $J(\xi)$ can be computed in $O(n(n+m))$ time as follows:*

1. **for** $i := 1$ **to** n **do**

$$v(x_i) \;:=\; \max\left\{ \max_{u \in U}\{u(x_i) - \xi(u)\}\,,\; \max_{j<i}\{x_j x_i - v(x_j)\} \right\};$$

2. $J(\xi) := v$.

Proof: Write $U = \{u_1, \ldots u_m\}$. As previously noted, by a slight abuse of notation, we can identify each point with its characteristic function. Thus, we write $X = \{\chi_{x_1}, \ldots, \chi_{x_n}\} \subseteq \mathbf{CH}(X)$.

Let $U_i = U \cup \{\chi_{x_1}, \ldots, \chi_{x_i}\}$. We iterate the construction of Lemma 4.4 to obtain the embedding

$$I^n \;:\; \mathbf{CH}(U) \;=\; \mathbf{CH}(U_0) \;\subseteq\; \mathbf{CH}(U_1) \;\subseteq\; \ldots \;\subseteq\; \mathbf{CH}(U_n) \;=\; \mathbf{CH}(U \cup X).$$

Let $X_j = X \cup \{u_1, \ldots u_j\}$. We iterate the construction of Lemma 4.4 to obtain the embedding

$$I^m \;:\; \mathbf{CH}(X) \;=\; \mathbf{CH}(X_0) \;\subseteq\; \mathbf{CH}(X_1) \;\subseteq\; \ldots \;\subseteq\; \mathbf{CH}(X_m) \;=\; \mathbf{CH}(U \cup X).$$

For each u_j and for each $x \in X$, there exists $y \in X$ such that $u_j(x) + u_j(y) = xy$, since $u_j \in \mathbf{CH}(X)$. Thus, u_j is between χ_x and χ_y, hence, by Lemma 4.4, I^m is an isometry. We define J to be the composition $(I^m)^{-1} \circ I^n$. It is a routine exercise to verify that the program outputs that same function. □

Theorem 4.1

(a) *If* $|M| = 2$, $\mathbf{CH}(M)$ *is a line segment, with the two points of* M *at the ends.*

(b) *If* $|M| = 3$, $\mathbf{CH}(M)$ *consists of three line segments joined at a point, to form a "Y" shape* (See Figure 2). *The points of* M *are at the ends of the arms.*

(c) *If* $|M| = 4$, $\mathbf{CH}(M)$ *consists of a rectangle with the Manhattan metric, together with a line segment attached by one end to each corner. We call these segments "whiskers". The points of* M *are at the outer ends of the whiskers* (See Figure 3).

(d) *If* $|M| = n$, $\mathbf{CH}(M)$ *can be embedded in* $\mathbf{R}^n$ *(with the sup-norm metric) and consists of the finite union of a number of convex pieces of various dimensions. None of the pieces has dimension greater than* $\lfloor n/2 \rfloor$.

Figure 2: Convex hull of three points x, y, r.

Figure 3: Convex hull of four points x, y, z, r.

Proof: Let $n = |M|$, and write $M = \{x_1, \ldots, x_n\}$. For ease of notation, we write $\ell_{ij} = x_i x_j$. We embed $\mathbf{Cl}(M)$ in $\mathbf{R}^n$ as follows: u maps to the n-tuple $(u(x_1), \ldots, u(x_n))$, which we also write as $(u_1, \ldots, u_n)$. Note that $x_i = \chi_{x_i}$ maps to the n-tuple which has a zero in position i, and whose j^{th} coordinate is ℓ_{ij}.

For convenience, we identify $\mathbf{Cl}(M)$ with its image in $\mathbf{R}^n$. Since the notation "x_i" is already in use for a point in M, we refer to the i^{th} coordinate of $\mathbf{R}^n$ as the "u_i" coordinate. Then $\mathbf{Cl}(M)$ is the convex region of $\mathbf{R}^n$ defined by the system of linear constraints

$$\forall i, j \ : \ u_i - u_j \leq \ell_{ij}$$
$$\forall i, j \ : \ u_i + u_j \geq \ell_{ij}$$

Since the number of constraints is finite, the region is polyhedral. $\mathbf{CH}(M)$ is the subset (which is not necessarily convex) of that region made up of "minimal" faces, under the coordinate-wise comparison partial ordering in $\mathbf{R}^n$. Since $\mathbf{Cl}(M)$ lies entirely within the positive orthant of $\mathbf{R}^n$, and since no coordinate of any $u \in \mathbf{CH}(M)$ can exceed the diameter of M, $\mathbf{CH}(M)$ is bounded.

Figure 4 shows $\mathbf{CH}(M) \subseteq \mathbf{Cl}(M) \subseteq \mathbf{R}^2$ if M has two points. Note that $\mathbf{CH}(M)$ consists of the interval whose endpoints are $x_1 = (0, \ell_{12})$ and $x_2 = (\ell_{12}, 0)$.

Figure 4: Convex hull and closure of two points. Note: x_1 and x_2 denote points in $\mathbf{R}^2$, not coordinates.

M can be thought of as a weighted complete graph of order n. Write e_{ij} for the edge joining x_i with x_j, which we assign the weight ℓ_{ij}. For convenience, we also consider edges e_{ii} of weight 0. An *edge cover* of M is defined to be a set of edges which covers all vertices. Note that an edge cover has order at least $\lceil n/2 \rceil$.

Each edge e_{ij} defines a hyperplane $H_{ij} \subseteq \mathbf{R}^n$, namely the solutions to the equation $u_i + u_j = \ell_{ij}$. If C is a set of edges, define $H_C = \bigcap_{e_{ij} \in C} H_{ij}$. If r_C is the rank of the system of equations $\{u_i + u_j = \ell_{ij} \ : \ e_{ij} \in C\}$, then H_C has dimension $n - r_C$. (Note that $r_C \leq |C|$.) Let $P_C = H_C \cap \mathbf{Cl}(M)$.

If $u \in \mathbf{Cl}(M)$, we define $C_u = \{e_{ij} \ : \ u(x_i) + u(x_j) = \ell_{ij}\}$. C_u is an edge cover if and only if

$u \in \mathbf{CH}(M)$, by Lemma 4.2. Thus, $P_C \subseteq \mathbf{CH}(M)$ if C is an edge cover, and $\mathbf{CH}(M)$ is the union of the P_C for all C which are minimal edge covers.[3]

Each P_C is convex, as it is the intersection of two convex sets, namely H_C and $\mathbf{Cl}(M)$, and is polyhedral, as it is specified by a finite number of linear inequalities. Furthermore, $\mathbf{CH}(M)$ is bounded. Thus $\mathbf{CH}(M)$ is the union of a finite number of convex polyhedra.

Since $r_C = |C| \geq \lceil n/2 \rceil$ if C is a minimal edge cover, the dimension of each $P_C \subseteq \mathbf{CH}(M)$ cannot exceed $\lfloor n/2 \rfloor$. Part (d) of the conclusion is thus established.

We now consider the structure of the convex hull for small spaces, *i.e.* $|M| \leq 4$.

Space with Two Points. The only minimal edge cover of M is $\{e_{12}\}$. Thus $\mathbf{CH}(M) = P_{12}$, which is a line segment of length (in the sup-norm metric) ℓ_{12} in $\mathbf{R}^2$ from $x_1 = (0, \ell_{12})$ to $x_2 = (\ell_{12}, 0)$.

Space with Three Points. Let $a, b, c \geq 0$ be the "slack parameters", defined by the system of equations

$$\ell_{12} \;=\; a + b \qquad \ell_{13} \;=\; a + c \qquad \ell_{23} \;=\; b + c$$

There are three minimal edge covers, namely $A_1 = \{e_{12}, e_{13}\}$, $A_2 = \{e_{12}, e_{23}\}$, and $A_3 = \{e_{13}, e_{23}\}$. Each P_{A_i} is a line segment, and they have a common end-point at $(a, b, c) \in \mathbf{R}^3$. P_{A_1} is the line segment from $x_1 = (0, a + b, a + c)$ to (a, b, c), of length a. P_{A_2} is the line segment from $x_2 = (a + b, 0, b + c)$ to (a, b, c), of length b. P_{A_3} is the line segment from $x_3 = (a + b, b + c, 0)$ to (a, b, c), of length c.

Space with Four Points. There are seven minimal edge covers, but $\mathbf{CH}(M)$ is always the union of polyhedra associated with just five of them. Four of these (which we call "whiskers") have dimension at most 1, and the other has dimension at most 2.

Without loss of generality, $\ell_{13} + \ell_{24} \;\geq\; \ell_{12} + \ell_{34}$, $\ell_{14} + \ell_{23}$.

Define $A_1 = \{e_{12}, e_{13}, e_{14}\}$, $A_2 = \{e_{12}, e_{23}, e_{24}\}$, $A_3 = \{e_{13}, e_{23}, e_{34}\}$, $A_4 = \{e_{14}, e_{24}, e_{34}\}$, and $B = \{e_{13}, e_{24}\}$.

We first prove that each $u \in \mathbf{CH}(M)$ lies in either some P_{A_i} or in P_B. As before, we write u_i for $u(x_i)$. We know C_u is an edge cover, and thus contains a minimal edge cover C. If $C = A_i$, then $u \in P_{A_i}$. If $C = B$, then $u \in P_B$. The only remaining possibilities for C are $\{e_{12}, e_{34}\}$ and $\{e_{14}, e_{23}\}$. Without loss of generality, suppose that $C = \{e_{12}, e_{34}\}$, that is $u_1 + u_2 = \ell_{12}$ and $u_3 + u_4 = \ell_{34}$. Then

$$u_1 + u_2 + u_3 + u_4 \;=\; \ell_{12} + \ell_{34} \;\leq\; \ell_{13} + \ell_{24} \;\leq\; u_1 + u_3 + u_2 + u_4.$$

The inequalities above must then be equalities. Thus $\ell_{13} = u_1 + u_3$ and $\ell_{24} = u_2 + u_4$, hence $B = \{e_{13}, e_{24}\} \subseteq C_u$. This, in turn, means that $u \in P_B$.

H_{A_i} has dimension $4 - |A_i| = 1$, while H_B has dimension $4 - |B| = 2$.

[3] An edge cover is *minimal* if no subset is also an edge cover.

Let $a, b, c, d, e, f \geq 0$ be the solutions to the equations

$$\begin{aligned} \ell_{12} &= a+b+e & \ell_{13} &= a+c+e+f & \ell_{23} &= b+c+f \\ \ell_{14} &= a+d+f & \ell_{24} &= b+d+e+f & \ell_{34} &= c+d+e \end{aligned}$$

Let x_i' be the point at which P_{A_i} meets P_B. Expressing all points using their coordinates in $\mathbf{R}^4$, we have

$$\begin{aligned} x_1 &= (0, a+b+e, a+C+e+f, a+d+f) & x_1' &= (a, b+e, c+e+f, d+f) \\ x_2 &= (a+b+e, 0, b+c+f, b+d+e+f) & x_2' &= (a+e, b, c+f, d+e+f) \\ x_3 &= (a+c+e+f, b+c+f, 0, c+d+e) & x_3' &= (a+e+f, b+f, c, d+e) \\ x_4 &= (a+d+f, b+d+e+f, c+d+e, 0) & x_4' &= (a+f, b+e+f, c+e, d) \end{aligned}$$

We show that P_B is isometric to an $e \times f$ rectangle in the Manhattan plane, which we express as $R_{ef} = [0, e] \times [0, f]$. If $p = (\lambda, \mu) \in R_{ef}$, let $f(p) = x_1' + \lambda(1, -1, -1, 1) + \mu(1, 1, -1, -1) \in \mathbf{R}^4$. It is easy to see that that $f(R_{ef}) = P_B$. In order to prove that f is an embedding, let us choose two points $p = (\lambda, \mu)$, $p' = (\lambda', \mu') \in R_{ef}$. Without loss of generality $\lambda \leq \lambda'$. Then $pp' = \lambda' - \lambda + |\mu' - \mu|$. But, from the definition of the sup-norm metric, and by disregarding identical terms inside the maximum, we have

$$f(p)f(p') = \max\{|\lambda' - \lambda + \mu' - \mu|, |\lambda' - \lambda + \mu - \mu'|\} = \lambda' - \lambda + |\mu' - \mu| = pp'.$$

Figure 5: Convex hull of four points x_1, x_2, x_3, x_4.

Overall, P_B is the rectangle whose corners are x_i' for $i = 1, 2, 3, 4$. P_{A_i} is a line segment connecting x_i to x_i', of length a, b, c, d for $i = 1, 2, 3, 4$, respectively. Figure 5 shows how the pieces fit together. □

Degeneracy. It is important to note that the structures given in Theorem 4.1 can be degenerate. For example, any whisker can have length zero, and in the case of $n = 4$, the rectangle P_B can degenerate to a line segment or even a point.

5 The Equipoise Algorithm

In this section, we introduce the *Equipoise Algorithm* for the k-server problem in an arbitrary metric space, M. For the sake of exposition, it is best to first define Equipoise abstractly in $\mathbf{CH}(M)$, and later present its implementation in finite and then arbitrary metric spaces.

Equipoise in $\mathbf{CH}(M)$. We assume first that Equipoise has k movable servers in $\mathbf{CH}(M)$, where M is a finite metric space. We can also allow the adversary to makes requests in $\mathbf{CH}(M)$. At each request r, all of our k servers may move within $\mathbf{CH}(M)$. The algorithm is memoryless, in the sense that the only inputs used for each move are the previous positions of our servers in $\mathbf{CH}(M)$ and the current request point.

Suppose that $S = \{s_1, \ldots s_k\} \in \Lambda^k \mathbf{CH}(M)$ is the current configuration of the servers and r the new request point. Define $K = S \cup \{r\}$ and $E_{ij} = \mathbf{CH}(\{s_i, s_j\})$, for $1 \leq i < j \leq k$. By Lemma 4.5 we can assume that

$$E_{ij} \subseteq \mathbf{CH}(S) \subseteq \mathbf{CH}(K) \subseteq \mathbf{CH}(M)$$

for $1 \leq i < j \leq k$.

Equipoise will move its servers to a configuration S' that is determined as follows:

Define a weighted complete graph G whose vertices are our current server positions and the weight of the edge $e_{ij} = (s_i, s_j)$ is

$$weight(s_i, s_j) = \frac{s_i s_j + s_i r + s_j r}{2}.$$

Let T be a minimal spanning tree of the graph G, which consists of $k-1$ edges. If $e_{ij} \in T$, define $s_{ij} \in E_{ij}$ to be that point whose distance to s_i is $(s_i s_j + s_j r - s_i r)/2$ and whose distance to s_j is $(s_i s_j + s_i r - s_j r)/2$. Such a point must exist because E_{ij} is simply an interval between s_i and s_j.

Let S' consist of r together with all s_{ij} for which $e_{ij} \in T$. Equipoise then chooses the minimum cost move of all servers from S to S'.

Implementation in a finite space. We now define the *Virtual Equipoise* algorithm on an arbitrary finite metric space M. The algorithm keeps track of *fictitious* servers in $\mathbf{CH}(M)$, which are moved according to the rules of Equipoise in that space. Each *real* server in M itself is only moved when the corresponding fictitious server actually serves a request.

Name the fictitious servers $f_1, \ldots, f_k$ and the real servers $s_1, \ldots, s_k$. Then, if server f_i "moves" to $r = \chi_r$, then s_i moves (in reality) to r to serve the real request. If f_i does not serve the request in this move, then Equipoise only updates its position in $\mathbf{CH}(M)$, but does not move s_i.

Initially, the fictitious servers match the real ones. Each time we move f_i, we charge ourselves the cost of the fictitious move, but since we do not actually have to pay that charge, we place it as a credit on s_i. By the triangle inequality, and the fact that M embeds in $\mathbf{CH}(M)$, the cost of

each real move of s_i cannot exceed the value of the credits that have accumulated on s_i. The real cost thus never exceeds the fictitious cost, so Virtual Equipoise is c-competitive in M if Equipoise is c-competitive in $\mathbf{CH}(M)$.

Implementation of Virtual Equipoise in an arbitrary space. If the metric space M is infinite, or if we do not know the whole metric space in advance, then we cannot use $\mathbf{CH}(M)$ in the algorithm. Instead, we use $\mathbf{CH}(M^m)$, where M^t is the set of points that are relevant up to time t. (A point in M is *relevant* if it has ever been requested, or if one of our servers started there.) Of course, we cannot know M^m in advance, either. We allow our fictitious servers to reside in $\mathbf{CH}(M^t)$. When a new request r^t is read, we update all fictitious servers by using the embedding $I : \mathbf{CH}(M^{t-1}) \subseteq \mathbf{CH}(M^t)$ given by Lemma 4.4. This embedding has no effect on fictitious cost, so there is no change in the competitiveness results. The same technique was used in [5].

For simplicity, we shall use the name "Equipoise" also to mean Virtual Equipoise in any metric space.

6 The Proof for Two Servers

In this section we present the proof that Equipoise is 2-competitive for two servers. If carefully inspected, the algorithm turns out to be equivalent to the one in [5]. We present the proof here both for the sake of exposition (the argument for $k = 3$ is much harder), and because it is simpler than the proofs from [5, 13].

As it has been explained in Section 5, it is sufficient to present Equipoise in $\mathbf{CH}(M)$. That is, we will show how Equipoise moves its fictitious servers in $\mathbf{CH}(M)$ (for simplicity, we will refer to them simply as "servers").

Lemma 6.1 *For $k = 2$ the adversary pseudo-cost for each move of Equipoise is equal to the distance moved by the one of our servers that moves to the request point, minus the distance moved by our other server.*

Proof: $S = \{x, y\}$ is the current configuration of our servers, r is the new request point, and $S' = \{r, u\}$ is our server configuration after serving r.

The convex hull of $K = \{x, y, r\}$ consists of three "whiskers" meeting at a "branch" point, whose lengths are the three so-called *slack* variables, a, b, c. The point x is at the end of an whisker of length a, the point y is at the end of an whisker of length b, and r is at the end of an whisker of length c, where

$$xy = a + b \qquad xr = a + c \qquad yr = b + c$$

Without loss of generality, $a \leq b$ (see Figure 2).

Equipoise moves one server from x to r, and the other server from y to the point u, on y's whisker, which is distance a from y (see Figure 6).

Figure 6: Equipoise moves servers from $\{x, y\}$ to $\{r, u\}$.

By definition, the adversary pseudo-cost is

$$\min\{xr - yu, yr - xu\} = c,$$

which is also the distance moved by our server from x to r, minus the distance moved by our other server. This completes the proof. □

Theorem 6.1 *Equipoise is 2-competitive for two servers.*

Proof: We define the potential function to be $\Phi = xy$, where x, y are our server positions.

Consider one move. Let a, b, c be the parameters of $\mathbf{CH}(K)$, as before, where $a \leq b$. The adversary pseudo-cost of the move is c by Lemma 6.1, our cost is $2a + c$, and the change in potential is $c - 2a$. Thus

$$cost_{EQ}(S, r) + \Phi(S') - \Phi(S) \;=\; 2 \cdot \nabla(S, r, S').$$

By summing this inequality over the whole sequence of requests (this is the well-known potential argument), we are done. □

7 The Proof for Three Servers

In this section we will prove that Equipoise is 11-competitive for three servers. As explained in Section 5, it is sufficient to present Equipoise in $\mathbf{CH}(M)$.

Before moving to the details, we give first a short overview of the proof:

1. Recall that in Lemma 2.1 we showed that the adversary pseudo-cost is a lower bound for the optimal cost. Therefore in order to prove 11-competitiveness of Equipoise, it is sufficient to

prove that it is 11-competitive against the adversary pseudo-cost (for Equipoise) instead of the optimal cost.

2. In the second step of the proof (Lemma 7.1) we show that the pseudo-cost at every move is equal to the distance moved by the server that serves the request minus the distance moved by the other servers.

3. The 11-competitiveness of Equipoise is proven in Theorem 7.1. We view the problem abstractly as a game between a Blue (Equipoise) and Red (pseudo-cost) players. This game is viewed as a graph whose nodes correspond to configurations of our three servers, and arcs to the moves of Equipoise. Each state of the game is a triple $\{\ell_1, \ell_2, \ell_3\}$ of non-negative real numbers. Each configuration of our servers in $\Lambda^3\mathbf{CH}(M)$ maps to one such game state – this mapping is not one-to-one. On each request we have an arc to another state labeled with two costs: the Blue cost and the Red cost. The Blue cost of the transition between two game states is equal to our cost for the corresponding move, while the Red cost is equal to the adversary pseudo-cost for that move. Our goal is to show that for each path α in this graph, the total Blue cost on α is at most 11 times the total Red cost on α. This is done in two steps:

   (a) First we show that there are four *types* of so-called *prime* edges with the property that each edge of the Blue-Red game can be "factored" into a sequence of prime edges, preserving both the total Blue and Red costs.

   (b) Using the above fact, it is sufficient to consider only the paths consisting of prime edges. We define the potential function Φ on the set of nodes, and show that, for each prime edge $\nu_1 \to \nu_2$, the sum of its Blue cost and the change of the potential $\Delta\Phi = \Phi(\nu_2) - \Phi(\nu_1)$ is at most 11 times the Red cost of this edge. By adding those inequalities for each arc of α, we obtain the theorem.

   An alternative way to prove Theorem 7.1, without using factorization into prime edges, would be to prove that in each move Equipoise's cost plus the potential change is at most 11 times the pseudo-cost of this move. However, because of the way the potential function is defined, this approach would lead to the analysis of a large number of cases and make the proof much more complicated.

Lemma 7.1 *For $k = 3$ the adversary pseudo-cost for each move of Equipoise is equal to the distance moved by the one of our servers that moves to the request point, minus the distance moved by our other servers.*

Proof: We denote by $S = \{x, y, z\}$ the current configuration of our servers, r the new request point, and $S' = \{r, u, v\}$ our server configuration after serving r.

Without loss of generality, we can assume that y matches with r in the maximal matching of

$K = \{x, y, z, r\}$, *i.e.*,

$$yr + xz \;\geq\; xr + yz,\; zr + xy.$$

$\mathbf{CH}(K)$ has the appearance of a rectangle with four whiskers attached to the corners, and the members of K are at the ends of the whiskers. The rectangle has the Manhattan metric, by Theorem 4.1.

Let a, b, c, d, e, f be the six *slack parameters* of $\mathbf{CH}(K)$, which are defined to be non-negative reals satisfying the equations

$$\begin{array}{lll} xy = a+b+e & xz = a+c+e+f & yz = b+c+f \\ xr = a+d+f & yr = b+d+e+f & zr = c+d+e \end{array}$$

The lengths of the four whiskers are a, b, c, d, and the rectangle has dimensions $e \times f$ (See Figure 3). We recognize eight general cases, depending on relations among the slack variables. Taking into account the symmetry which exchanges $x \leftrightarrow z$, and which then also exchanges $a \leftrightarrow c$ and $e \leftrightarrow f$, we reduce these to five cases.

We first compute the edge weights of the graph G:

$$\begin{aligned} \mathit{weight}(x, y) &= a+b+d+e+f, \\ \mathit{weight}(x, z) &= a+c+d+e+f, \\ \mathit{weight}(y, z) &= b+c+d+e+f. \end{aligned}$$

Thus, the minimal spanning tree of G consists of the edges $(x, y), (x, z)$ if $a = \min\{a, b, c\}$ and $(x, y), (y, z)$ if $b = \min\{a, b, c\}$. (The case where c is the minimum is symmetric to the case where a is the minimum.)

Case I : $a \leq b$ and $a + f \leq c$. Then $uy = a$, $vz = a + f$, and both u, v are on whiskers. (See Figure 7).

Case II : $a \leq b$ and $a \leq c \leq a + f$. Then $uy = a$, and u is on a whisker; and $vz = a + f$, $vy = -a + b + c$, and v is located along the side of the rectangle between y and z. (See Figure 8).

Case III : $b \leq a \leq b + e$ and $b \leq c \leq b + f$. Then $ux = b + e$, $uy = a$, and u is located along the side of the rectangle between x and y; and $vz = b + f$, $vy = c$, and v is located along the side of the rectangle between y and z (See Figure 9).

Case IV : $b \leq a \leq b + e$ and $b + f \leq c$. Then $ux = b + e$, $uy = a$, and u is located on the side of the rectangle between x and y; and $vz = b + f$, and v is on a whisker. (See Figure 10).

Case V : $b + e \leq a$ and $b + f \leq c$. Then $ux = b + e$, $vz = b + f$, and both are on whiskers. (See Figure 11).

Figure 7: Equipoise's move in Case I.

Figure 8: Equipoise's move in Case II.

All other possibilities are similar to one of the above five cases.

The adversary pseudo-cost for each case is

$$\nabla(S, r, S') = \min_{X \ni r} \{SX - S'X\},$$

where X may be taken to be one of $\{r, x, y\}$, $\{r, x, z\}$, or $\{r, y, z\}$. For Cases I and II, each of the six quantities (three choices of X for each case) works out to be $d - a$, while for Cases III through V, each of the nine quantities works out to be $d - b$. Thus, the adversary pseudo-cost is always $d - \min\{a, b, c\}$.

Finally, the minimum matching of S with S' determines the motion of our servers. We show that the conclusion of the lemma is true in each of the five cases:

$\diamondsuit$ If x is closest to r, then x moves to r, y moves to u, and z moves to v, and

$$xr - yu - zv = \begin{cases} d - a & \text{in Cases I, II} \\ d - b & \text{in Cases III, IV.} \end{cases}$$

Figure 9: Equipoise's move in Case III.

Figure 10: Equipoise's move in Case IV.

$\diamondsuit$ If y is closest to r, then y moves to r, x moves to u, and z moves to v, and

$$yr - xu - zv = d - b \text{ in Case V.}$$

$\diamondsuit$ If z is closest to r, then z moves to r, x moves to u, and y moves to v, and

$$yr - xu - zv = d - a \text{ in Case II.}$$

All other situations are similar to one of the above. This completes the proof of Lemma 7.1. □

Theorem 7.1 *Equipoise is 11-competitive for three servers.*

Proof: We will reduce the problem of proving that our cost is 11-competitive against the adversary pseudo-cost by reduction to a "Blue–Red game," a problem on a bi-weighted directed graph. We then show that this graph has certain "prime" edges, such that every edge "factors" as a path involving only prime edges. We then define a potential function on the vertices of the graph, and show that,

Figure 11: Equipoise's move in Case V.

for each prime edge, the "Blue" cost plus the increase of the potential does not exceed eleven times the "Red" cost.

The Blue–Red game. We define a *Blue–Red game* to be a directed graph, $\mathcal{G}$, with two costs, the *Blue cost* and the *Red cost*, associated with every edge. We say that $\mathcal{G}$ is *c-competitive* if there exists a function Φ, defined on the vertices of $\mathcal{G}$, such that, for every vertex ν and every path α which starts at ν

$$cost_{Blue}(\alpha) \leq c \cdot cost_{Red}(\alpha) + \Phi(\nu)$$

We let the vertices of $\mathcal{G}$ be all multisets consisting of three non-negative real numbers, not necessarily distinct. (The triple $\{\ell_1, \ell_2, \ell_3\}$ represents the slack parameters of the convex hull of our server positions.) There are five classes of edges of $\mathcal{G}$ (corresponding to the five cases of moves of Equipoise). Let a, b, c, d, e, f be any non-negative real numbers. Then there are edges of $\mathcal{G}$ from $\nu = \{a + e, b, c + f\}$ as follows:

$\diamondsuit$ Class I : If $a \leq b$ and $a + f \leq c$, there is an edge

$$\{a + e, b, c + f\} \to \{d + e, b - a + f, c - a - f\}$$

with Red cost $d - a$ and Blue cost $3a + d + 2f$ (see Fig. 7).

$\diamondsuit$ Class II : If $a \leq b$ and $a \leq c \leq a + f$, there is an edge

$$\{a + e, b, c + f\} \to \{0, b + c - 2a, a - c + d + e + f\}$$

with Red cost $d - a$ and Blue cost $\min\{3a + d + 2f, a + 2c + d + 2e\}$ (see Fig. 8).

$\diamondsuit$ Class III : If $b \leq a \leq b + e$ and $b \leq c \leq b + f$, there is an edge

$$\{a + e, b, c + f\} \to \{a - b, c - b, d - a - c + 2b + e + f\}$$

with Red cost $d - b$ and Blue cost $\min\{2a + b + d + 2f, b + 2c + d + 2e\}$ (see Fig. 9).

$\diamondsuit$ Class IV : If $b \le a \le b + e$ and $b + f \le c$, there is an edge

$$\{a + e, b, c + f\} \to \{f, d - a + b + e, a + c - 2b - f\}$$

with Red cost $d - b$ and Blue cost $2a + b + d + 2f$ (see Fig. 10).

$\diamondsuit$ Class V : If $b + e \le a$ and $b + f \le c$, there is an edge

$$\{a + e, b, c + f\} \to \{d, a - b - e + f, c - b + e - f\}$$

with Red cost $d - b$ and Blue cost $3b + 2d + 2e + f$ (see Fig. 11).

Prime edges. We will identify a set of edges in $\mathcal{G}$, which we call *prime* edges, such that every edge can be *factored* into prime edges. Precisely, a *prime factorization* of an edge $\nu_1 \to \nu_2$ is a path from ν_1 to ν_2, consisting of entirely prime edges, whose total Red (Blue) cost is equal to the Red (Blue) cost of $\nu_1 \to \nu_2$.

We now list the prime edges, of which there are four types. Each prime edge is a special case of one of the five classes of edges listed above.

(A) If $\varepsilon \le \ell_i$ for each i then

$$\{\ell_1, \ell_2, \ell_3\} \stackrel{A}{\to} \{\ell_1 - \varepsilon, \ell_2 - \varepsilon, \ell_3 - \varepsilon\}.$$

The Red cost is $-\varepsilon$ and the Blue cost is 3ε.

(B)

$$\{\ell_1, \ell_2, \ell_3\} \stackrel{B}{\to} \{\ell_1 + \varepsilon, \ell_2, \ell_3\}.$$

The Red cost is ε and the Blue cost is ε.

(C) If $2\varepsilon \le \ell_3$ then

$$\{\ell_1, \ell_2, \ell_3\} \stackrel{C}{\to} \{\ell_1 + \varepsilon, \ell_2, \ell_3 - 2\varepsilon\}.$$

The Red cost is 0 and the Blue cost is 2ε.

(D)

$$\{\ell_1, \ell_2, \ell_3\} \stackrel{D}{\to} \{0, \ell_1 + \ell_2, \ell_3\}.$$

The Red cost is 0 and the Blue cost is $2 \cdot \min\{\ell_1, \ell_2\}$.

Figure 12: Prime edge (A).

Figure 13: Prime Edge (B).

The five classes of edges factor into primes as shown below. The Blue and Red costs of each edge are shown in the two right-hand columns.

(I) Edge I factors into primes as

			Blue	Red
$\{a+e, b, c+f\}$	$\stackrel{A}{\to}$	$\{e, -a+b, -a+c+f\}$	$3a$	a
	$\stackrel{C}{\to}$	$\{e, -a+b+f, -a+c-f\}$	$2f$	0
	$\stackrel{B}{\to}$	$\{d+e, -a+b+f, -a+c-f\}$	d	d

Thus the total Blue cost is $3a+d+2f$ and the Red cost is $d-a$.

(II) Edge II factors into primes as

			Blue	Red
$\{a+e, b, c+f\}$	$\stackrel{A}{\to}$	$\{e, -a+b, -a+c+f\}$	$3a$	$-a$
	$\stackrel{C}{\to}$	$\{e, -2a+b+c, a-c+f\}$	$2(c-a)$	0
	$\stackrel{D}{\to}$	$\{a-c+e+f, -2a+b+c, 0\}$	$2\cdot\min\{e, a-c+f\}$	0
	$\stackrel{B}{\to}$	$\{a-c+d+e+f, -2a+b+c, 0\}$	d	d

Thus the total Blue cost is $\min\{3a+d+2f, a+2c+d+2e\}$ and the Red cost is $d-a$.

Figure 14: Prime edge (C).

Figure 15: Prime edge (D).

(III) Provided $a + f \leq c + e$ (*i.e.*, the server at x moves to r), edge III factors into primes as

			Blue	Red
$\{a+e, b, c+f\}$	$\xrightarrow{A}$	$\{a-b+e, 0, -b+c+f\}$	$3b$	$-b$
	$\xrightarrow{C}$	$\{a-b+e, -b+c, b-c+f\}$	$2(c-b)$	0
	$\xrightarrow{D}$	$\{a-c+e+f, -b+c, 0\}$	$2(b-c+f)$	0
	$\xrightarrow{C}$	$\{-a+2b-c+e+f, -b+c, a-b\}$	$2(a-b)$	0
	$\xrightarrow{B}$	$\{-a+2b-c+d+e+f, -b+c, a-b\}$	d	d

The cost of the D-edge in step three, by definition, is $2 \cdot \min\{a-b+e, b-c+f\}$. But from the assumptions of Case III, and inequality $a + f \leq c + e$ we obtain $b - c + f = (b-a) - c + (a+f) \leq e \leq a - b + e$, proving that this cost is indeed $2(b-c+f)$, as shown above. Thus the total Blue cost is $2a + b + d + 2f = \min\{2a+b+d+2f, b+2c+d+2e\}$ (because $a + f \leq c + e$), and the Red cost is $d - a$.

The case where $a + f \geq c + e$ (*i.e.*, the server at z edges to r) is symmetric.

(IV) Edge IV factors into primes as

			Blue	Red
$\{a+e, b, c+f\}$	$\stackrel{A}{\to}$	$\{a-b+e, 0, -b+c+f\}$	$3b$	$-b$
	$\stackrel{C}{\to}$	$\{a-b+e, f, -b+c-f\}$	$2f$	0
	$\stackrel{C}{\to}$	$\{-a+b+e, f, a-2b+c-f\}$	$2(a-b)$	0
	$\stackrel{B}{\to}$	$\{-a+b+d+e, f, a-2b+c-f\}$	d	d

The total Blue cost is $2a+b+d+2f$ and the Red cost is $d-b$.

(V) Edge V factors into primes as

			Blue	Red
$\{a+e, b, c+f\}$	$\stackrel{A}{\to}$	$\{a-b+e, 0, -b+c+f\}$	$3b$	$-b$
	$\stackrel{C}{\to}$	$\{a-b-e, 0, -b+c+e+f\}$	$2e$	0
	$\stackrel{C}{\to}$	$\{a-b-e+f, 0, -b+c+e-f\}$	$2f$	0
	$\stackrel{B}{\to}$	$\{a-b+d-e+f, 0, -b+c+e-f\}$	d	d

The total Blue cost is $3b+d+2e+2f$ and the Red cost is $b-d$.

Thus we have shown that each edge can be factored into a sequence of prime edges with the same Blue and Red costs.

Lemma 7.2 *The Blue–Red game $\mathcal{G}$ is 11-competitive, that is, if α is any path in $\mathcal{G}$ starting from a vertex ν, then*

$$cost_{Blue}(\alpha) \le 11 \cdot cost_{Red}(\alpha) + \Phi(\nu).$$

Proof: Consider a vertex $\nu = \{\ell_1, \ell_2, \ell_3\}$ of $\mathcal{G}$, where the ℓ_i are ordered so that $\ell_1 \le \ell_2 \le \ell_3$. Then the potential of ν is defined as

$$\Phi(\nu) = 10 \cdot \ell_1 + 8 \cdot \ell_2 + 6 \cdot \ell_3.$$

We claim that, for any edge $\nu_1 \to \nu_2$ of $\mathcal{G}$,

$$cost_{Blue}(\nu_1 \to \nu_2) + \Phi(\nu_2) - \Phi(\nu_1) \;\le\; 11 \cdot cost_{Red}(\nu_1 \to \nu_2). \tag{1}$$

Since every edge can be factored into primes, it suffices to prove (1) only for prime edges. Furthermore, for edges of type (A), (B) or (C), we can restrict our attention to those edges for which ε is so small that the ordering of the ℓ_i does not change. Any prime edge of type (A), (B) or (C) can be decomposed into a sequence of such small edges of the same type with the same total Blue and Red costs.

We consider each of the four prime types, assuming that in cases (A), (B) and (C) the value of ε is sufficiently small, as explained above.

Type A: the Red cost is $-\varepsilon$, the Blue cost is 3ε, and the potential decreases by 24ε.

Type B: the Red cost is ε, the Blue cost is ε, and the potential increases by at most 10ε. (The worst case is that the shortest arm grows.)

Type C: the Red cost is 0, the Blue cost is 2ε, and the potential decreases by at least 2ε.

Type D: the Red cost is 0. Since $\ell_1 \le \ell_2$, the Blue cost is $2\ell_1$. The potential decreases by at least $2\ell_1$. (There are four cases to consider, namely (I) $\ell_1 + \ell_2 \le \ell_3$, (II) $\ell_1 \le \ell_2 \le \ell_3 \le \ell_1 + \ell_2$, (III) $\ell_1 \le \ell_3 \le \ell_2$, and (IV) $\ell_3 \le \ell_1 \le \ell_2$ – simply verify for each separately.)

Thus, (1) is true for all edges, completing the proof. □

We now continue the proof of Theorem 7.1. If $S \in \Lambda^3\mathbf{CH}(M)$ has slack parameters ℓ_1, ℓ_2, ℓ_2, we associate S with $\nu = \{\ell_1, \ell_2, \ell_2\}$, a vertex of the Blue–Red game $\mathcal{G}$. Similarly, we associate S' with $\nu' = \{\ell_1', \ell_2', \ell_2'\}$. By the way the costs of the edges of $\mathcal{G}$ are defined, and by Lemma 7.1, $cost_{Blue}(\nu \to \nu')$ is equal to $cost_{EQ}(S, r) = SS'$, and $cost_{Red}(\nu \to \nu')$ is equal to $\nabla(S, r, S')$. By Lemmas 2.1 and 7.2, we are done. □

7.0.1 A Lower Bound

By actual example, we can show that Lemma 7.2 is tight. Our method is to define a cycle in $\mathcal{G}$ whose Blue cost is at least $(11 - \varepsilon)$ times its Red cost, for any real $\varepsilon > 0$. This does not prove that Equipoise is no better than 11-competitive for 3 servers, but since that algorithm was designed so as to minimize our cost relative to adversary pseudo-cost, it seems to indicate that a better competitiveness for Equipoise will not be easily found.

Lemma 7.3 *For any $\varepsilon > 0$ and any $\ell \ge 2\varepsilon$, there is a path in $\mathcal{G}$ from $\{0, 0, \ell\}$ to $\{\varepsilon, \varepsilon, \ell - 2\varepsilon\}$ whose Blue cost is 5ε and whose Red cost is ε.*

Proof: The path is

$$\{0, 0, \ell\} \stackrel{C}{\to} \{0, \varepsilon, \ell - 2\varepsilon\} \stackrel{B}{\to} \{\varepsilon, \varepsilon, \ell - 2\varepsilon\} \stackrel{D}{\to} \{0, 2\varepsilon, \ell - 2\varepsilon\}$$

□

Lemma 7.4 *For any $\varepsilon > 0$ and any $\ell_1 \ge \varepsilon, \ell_2 \ge 2\varepsilon$, there is a path in $\mathcal{G}$ from $\{0, \ell_1, \ell_2\}$ to $\{0, \ell_1 + 2\varepsilon, \ell_2 - 2\varepsilon\}$ whose Blue cost is 7ε and whose Red cost is ε.*

Proof: The path is

$$\begin{aligned} \{0, \ell_1, \ell_2\} &\stackrel{C}{\to} \{\varepsilon, \ell_1, \ell_2 - 2\varepsilon\} \stackrel{D}{\to} \{0, \ell_1 + \varepsilon, \ell_2 - 2\varepsilon\} \\ &\stackrel{B}{\to} \{\varepsilon, \ell_1 + \varepsilon, \ell_2 - 2\varepsilon\} \stackrel{D}{\to} \{0, \ell_1 + 2\varepsilon, \ell_2 - 2\varepsilon\} \end{aligned}$$

□

Theorem 7.2 *The Blue-Red graph $\mathcal{G}$ is no better than* 11*-competitive.*

Proof: Suppose $\mathcal{G}$ is c'-competitive for some $c' < c$. Let $\varepsilon = \frac{c'-c}{4}$. We construct a cycle, ζ, based at $\{0,0,1\}$, whose Red cost is $\frac{1}{4}$ and whose Blue cost is greater than $\frac{11}{4} - \varepsilon$. Let $n > \frac{1}{2\varepsilon}$ be an integer. Use Lemma 7.3 to construct a path α from $\{0,0,1\}$ to $\{0, \frac{1}{2n}, 1 - \frac{1}{2n}\}$ whose Blue cost is $\frac{5}{4n}$ and whose Red cost is $\frac{1}{4n}$. Then use Lemma 7.4 to construct a path β^i from $\{0, \frac{i-1}{2n}, 1 - \frac{i-1}{2n}\}$ to $\{0, \frac{i}{2n}, 1 - \frac{i}{2n}\}$, for $i = 2, 3, \ldots n$, whose Blue cost is $\frac{7}{4n}$ and whose Red cost is $\frac{1}{4n}$. Finally, let γ be the edge $\{0, \frac{1}{2}, \frac{1}{2}\} \rightarrow \{0,0,1\}$, which has Blue cost 1 and Red cost 0. The cycle $\zeta = \alpha\beta^2\beta^3 \ldots \beta^n\gamma$ has Blue cost $\frac{11}{4} - \frac{1}{2n}$ and Red cost $\frac{1}{4}$. Then

$$\lim_{m \to \infty} \left(cost_{Blue}(\zeta^m) - c' \cdot cost_{Red}(\zeta^m) \right) = \infty$$

(where ζ^m denotes the m-fold concatenation of ζ), contradicting c'-competitiveness. □

8 Final Comments

We conjecture that the forgiveness method, if appropriately refined, can yield a 3-competitive algorithm for three servers, and quite possibly even solve the problem for arbitrary k. The method we suggest is to define some special class $\mathcal{C}$ of work functions, and to forgive the current work function to one in that class after each step. In this paper, the work functions in $\mathcal{C}$ were of the simplest possible form: the characteristic functions. Characteristic functions determine uniquely the adversary position. However, although such forgiveness yields an 11-competitive algorithm, it cannot give a 3-competitive algorithm. It is possible to show that sometimes it is necessary to remember work functions that have two local minima. We conjecture that such functions are in fact sufficient for $k = 3$.

References

[1] S. Ben-Davida, A. Borodin, R. Karp, G. Tardos, and A. Widgerson. On the power of randomization in on-line algorithms. In *Proc. 22nd Symposium on Theory of Algorithms*, pages 379–386, 1990.

[2] P. Berman, H. Karloff, and G. Tardos. A competitive algorithm for three servers. In *Proceedings of the 1st Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 280–290, 1990.

[3] M. Chrobak, H. Karloff, T. Payne, and S. Vishwanathan. New results on server problems. *SIAM Journal on Discrete Mathematics*, 4:172–181, 1991. Also in Proceedings of the 1st Annual ACM-SIAM Symposium on Discrete Algorithms, San Francisco, 1990, pp. 291-300.

[4] M. Chrobak and L. Larmore. Harmonic is three-competitive for two servers. Submitted for publication.

[5] M. Chrobak and L. Larmore. A new approach to the server problem. *SIAM Journal on Discrete Mathematics*, 1991. To appear.

[6] M. Chrobak and L. Larmore. On fast algorithms for two servers. *Journal of Algorithms*, 1991. To appear.

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

[8] M. Chrobak and L. Larmore. The server problem and on-line games. In *Proceedings of DIAMCS Workshop on On-Line Algorithms*, 1991. To appear.

[9] D. Coppersmith, P. G. Doyle, P. Raghavan, and M. Snir. Random walks on weighted graphs and applications to online algorithms. In *Proc. 22nd Annual ACM Symposium on Theory of Computing*, pages 369–378, 1990.

[10] A. Fiat, Y. Rabani, and Y. Ravid. Competitive k-server algorithms. In *Proc. 22nd Symposium on Theory of Algorithms*, 1990.

[11] E. Grove. The harmonic on-line k-server algorithm is competitive. manuscript.

[12] S. Irani and R. Rubinfeld. A competitive 2-server algorithm. manuscript.

[13] M. Manasse, L. A. McGeoch, and D. Sleator. Competitive algorithms for server problems. *Journal of Algorithms*, 11:208–230, 1990. Also in Proc. 20th Annual ACM Symposium on Theory of Computing, 1988, pp. 322-333.

[14] P. Raghavan and M. Snir. Memory versus randomization in online algorithms. In *16th International Colloquium on Automata, Languages, and Programming, Lecture Notes in Computer Science vol. 372*, pages 687–703, Springer-Verlag, 1989.