Randomized K-Server on Hierarchical Binary Trees

Aaron Coté
Computer Science Department
UCLA
4732 Boelter Hall
Los Angeles, CA 90095
acote@ucla.edu

Adam Meyerson
Computer Science Department
UCLA
4732 Boelter Hall
Los Angeles, CA 90095
awm@cs.ucla.edu

Laura Poplawski
College of Computer and Information Science
Northeastern University
360 Huntington Avenue
Boston, MA 02115
ljp@ccs.neu.edu

ABSTRACT

We design a randomized online algorithm for k-server on binary trees with hierarchical edge lengths, with expected competitive ratio $O(\log \Delta)$, where Δ is the diameter of the metric. This is one of the first k-server algorithms with competitive ratio poly-logarithmic in the natural problem parameters, and represents substantial progress on the randomized k-server conjecture. Extending the algorithm to trees of higher degree would give a competitive ratio of $O(\log^2 \Delta \log n)$ for the k-server problem on general metrics with n points and diameter Δ.

Categories and Subject Descriptors

F.2.2 [**Theory of Computation**]: Non-numerical Algorithms and Problems—*Computations on Discrete Structures*

General Terms

Algorithms, Theory

Keywords

Online Competitive Analysis, K-Server

1. INTRODUCTION

The metric k-server problem is among the oldest and most-studied problems in online algorithms. We are given a set of k initial server locations (an initial configuration) in some underlying metric space. Requests for service arrive at various nodes in this space, and as each request arrives we must move one of our k servers to that location (thus obtaining a new configuration which includes the new request as one of the server nodes). The goal is to minimize the total distance traveled by the k servers; in the offline setting this can be solved optimally using flow techniques, but many of the natural applications are *online*, where we are unable to see the future requests before determining which server to move.

STOC'08, May 17–20, 2008, Victoria, British Columbia, Canada.

Our main result is a randomized online algorithm for k-server on hierarchically structured binary trees, which guarantees an expected cost of at most $O(\log \Delta)$ times optimum, where Δ is the diameter of the metric space. This is one of the first sub-linear competitive ratios for any non-uniform metric with $n > k + O(1)$ (n is the number of nodes), and is particularly interesting because of a potential extension to non-binary trees, which would give a sub-linear competitive ratio for general graphs by applying the metric embedding techniques of Bartal [1, 2] or Fakcharoenphol et al [14].

Once we are considering a binary tree, we model the problem as a set of many online decision makers, one residing at each node i of the tree. Each decision-maker sees the request stream in an online fashion, and also sees a dynamic number of servers available in the subtree rooted at node i over time. The job of this decision-maker is to partition its servers between the subtrees rooted at its children, thus passing a similar problem down to the next layer. We bound the cost thus passed down, as well as the cost paid to transfer servers between child subtrees, summing over all levels to get a general bound of $O(L)$, where L is the number of levels of the tree. Because of the hierarchical nature of the distances, the number of levels must be bounded by $O(\log \Delta)$, allowing us to obtain the overall bound claimed.

We view our results as making progress towards the *randomized k-server conjecture*, that there exists an $O(\log k)$-competitive randomized online algorithm for the problem. Of course, our results depend heavily on the underlying graph, but currently the only case where any similar ratio is known for general graphs comes from algorithms for metrical task systems [5] which apply only when $n = k + O(1)$.

1.1 Previous Work

Online k-server has been studied at least since 1990, when Manasse, McGeoch and Sleator presented an $(n-1)$-competitive algorithm for $(n-1)$ servers, a 2-competitve algorithm for 2 servers, and a deterministic lower bound of k for the competitive ratio. They also introduced the "k-server conjecture" - the conjecture that there exists a deterministic k-competitive solution [20]. The same year, Fiat, Rabani and Ravid showed that a competitive algorithm exists for any metric space and any k [17]. The best known general deterministic algorithm is the *work function algorithm* by Koutsoupias and Papadimitriou [19], which achieves a competitive ratio of $2k-1$ on any metric.

While no known deterministic algorithm has a competitive ratio better than $2k-1$ for arbitrary metrics, there are k-competitive deterministic algorithms for many special

classes of metric. In addition to the case where $k = n - 1$ and where $k = 2$. Chrobak, Karloff, Payne, and Vishwanathan gave a k-competitive algorithm on the line and the star graph [8]. Chrobak and Larmore gave a k-competitive algorithm on trees [11]. The *work function algorithm* is k-competitive on the line, the weighted star, and metrics with $k + 2$ points [6].

There is also a great deal of interest in randomized algorithms for k-server, since randomized algorithms often obtain better performance for online problems (against an adversary which is oblivious to the random choices). The "randomized k-server conjecture" states that there exists an $O(\log k)$-competitive randomized algorithm. So far, no randomized $O(\text{polylog}(n, k))$ algorithm has been published for the general k-server problem on arbitrary metrics. However, there are known randomized algorithms for a number of special cases. Perhaps the best-known example is the *marking algorithm* of Fiat et al for the paging problem (k-server on a uniform metric) [15]. Chrobak and Sgall presented the HARMONIC algorithm, which is 3-competitive when k=2 [10, 12]. Csaba and Lodha gave a $O(n^{2/3} \log n)$-competitive randomized algorithm for equally spaced points on a line [13]. Seiden proved the existence $O(\text{polylog}(k))$ randomized algorithms for metrics consisting of some number (polylogarithmic in k) of widely separated subspaces [22]. Seiden also provided an algorithm that is $O(h \log n \log \log n)$-competitive on any metric with h-HSTs (Hierarchically Separated Subtrees), as long as $h = \Theta > 2k$. Bartal and Mendel gave an algorithm for graphs with bounded growth-rate that is $O(\Delta^{1-(1/(\rho+2))}\text{polylog}(n))$-competitive, where ρ is the growth-rate and Δ is the diameter of the graph [7].

Blum, Karloff, Rabani, and Saks proved a lower bound of $\Omega(\sqrt{\log k / \log \log k})$ for any randomized algorithm on *any* metric where $k < n$ [5]. Bartal, Bollobás, and Mendel improved this to $\Omega(\log k / \log^2 \log k)$ [3].

Throughout our algorithm and analysis, we use the concept of the *work function*, which was initially discussed by Chrobak and Larmore in 1992 [9] and was used to deterministically solve the k-server problem by Koutsoupias and Papadimitriou [19]. The work function value is the optimum offline cost to handle all requests up through time t and end with servers in a particular configuration C. The work function algorithm of Koutsoupias and Papadimitriou [19] handles each request by moving the servers to the configuration which minimizes the sum of the work function value and the distance from the prior configuration.

2. PROBLEM STATEMENT

A k-server instance $(V, d, C[0], \rho, k)$ is defined by a metric space (V, d), assumed to satisfy symmetry and the triangle inequality, along with an initial configuration $C[0] \subseteq V$ with $|C[0]| = k$ and requests $\rho = \{\rho[1], \rho[2], ..., \rho[m]\} \in V^m$.

A solution to this instance is a sequence of configurations $\{C[1], C[2], ..., C[m]\}$ where $C[t] \subseteq V$ and $|C[t]| = k$, with the requirement that $\rho[t] \in C[t]$ for each t. The cost of the solution is $\Sigma_{t=1}^{m} d(C[t-1], C[t])$ where the distance between two configurations is defined to be the cost of the minimum-cost matching between them on a complete bipartite graph weighted by the distance function d.

We observe that the k-server problem can be solved to optimality in polynomial time via minimum-cost flow on a time-dilated graph with flow requirements on the vertices. Typically we will be interested in the *online* version of the problem, where the request sequence ρ arrives one at a time in order, and we must select $C[t]$ immediately after the arrival of $\rho[t]$, without knowledge of the yet-to-arrive $\rho[u]$ for $u > t$. The competitive ratio for an online algorithm for k-server is the maximum over all instances of the ratio of the cost of the solution constructed by the online algorithm to the cost of the minimum-cost (offline) solution. For a randomized online algorithm, replace the cost of the solution constructed with the *expected* cost of the solution constructed. It is known that the online version of the problem has a $2k - 1$ competitive deterministic algorithm and that no deterministic algorithm can have competitive ratio better than k. Algorithms with competitive ratio k are known for some special metrics (for example uniform metric and trees). Randomized algorithms might be able to improve upon the competitive ratio; it is known that no randomized algorithm can have competitive ratio better than $\log k$, but results better than $2k - 1$ are only known for a few special cases of the problem.

3. QUASI-CONVEXITY

We state a general version of the quasi-convexity lemma first proved by Koutsoupias and Papadimitriou [19]. These will be used in sections 5 and 6.

THEOREM 1. **Quasi-Convexity***: Let $(V, d, C[0], \rho, k)$ be a k-server instance. For any configuration X, define $c(X)$ to be the cost of the best solution to this instance which ends in configuration X (we can suppose $X = C[m]$ in the solution, or more generally allow $\rho[m] \notin X$ by adding a null $\rho[m+1]$ to the request sequence and setting $X = C[m+1]$). For any pair of configurations A, B there exists a matching $\mu : A \to B$ which guarantees that for all partitions $A_1 \subseteq A$ and $A_2 = A - A_1$, we have $c(A_1 \cup \mu(A_2)) + c(\mu(A_1) \cup A_2) \leq c(A) + c(B)$. Further, for any vertex $x \in A \cap B$, we have $\mu(x) = x$.*

PROOF. We can solve k-server offline by constructing a time-dilated graph G. The vertices of this graph come from V^{m+1}, with a directed edge $((u, i), (v, i + 1))$ for every vertices $u, v \in V$ and every $0 \leq i \leq m$. The cost of this edge will be $d(u, v)$. We add a source node with capacity one, cost zero edges to each vertex $(v, 0)$ with $v \in C[0]$. We add a sink node with a capacity one, cost zero edge from each vertex (v, m). A k-server solution corresponds to a flow on this graph with the added condition that for each vertex $\rho[t]$ we have at least one unit of flow through $(\rho[t], t)$. The cost of the k-server solution is equal to the cost of the flow.

For each of the configurations A, B there are corresponding flows f_A, f_B in this graph. Consider constructing the *residual graph* G_A corresponding to flow f_A. This graph has some reverse direction edges where $f_A(e) > 0$, and these edges have potentially negative cost. Flow $f_B - f_A$ is feasible in this residual graph and has cost $c(B) - c(A)$. Note that $f_B - f_A$ does not place any flow on edges from the source (these edges have equal flow in both f_A and f_B); this flow travels from the sink to the sink, exiting via reverse edges into the nodes of A and returning to the sink via edges from the nodes of B. Like any integral flow, we can decompose $f_B - f_A$ into paths each of which carries one unit of flow. Call these paths $P_1, P_2, ..., P_k$ and let P_i connect a_i to b_i. We define $\mu(a_i) = b_i$.

Now consider any partition A_1, A_2 of configuration A. One possible solution ending in configuration $A_1 \cup \mu(A_2)$ corresponds to the flow which starts from f_A and adds paths P_i

for $a_i \in A_2$. This flow has cost $c(A)+\Sigma_{a_i \in A_2} c(P_i)$. Similarly we can end in configuration $\mu(A_1) \cup A_2$ by adding paths P_i for $a_i \in A_1$, finding a flow with cost $c(A) + \Sigma_{a_i \in A_1} c(P_i)$. Of course, there might be a better (cheaper) flow corresponding to the same configuration, so we conclude that $c(A_1 \cup \mu(A_2)) \leq c(A)+\Sigma_{a_i \in A_2} c(P_i)$ and that $c(\mu(A_1) \cup A_2) \leq c(A) + \Sigma_{a_i \in A_1} c(P_i)$. Combining these two inequalities, and noticing that $c(A) + \Sigma_{a_i \in A} c(P_i) = c(B)$ gives the desired inequality.

For the further point, any $x \in A \cap B$ has $f_A(x,t) = f_B(x,t) = 1$ so there is no flow on (x,t) in $f_B - f_A$. Thus no path starts or ends at x (or equivalently there is a path which both starts and ends at x without any nodes in between), so $\mu(x) = x$. □

We can use this theorem to prove two useful corollaries.

COROLLARY 1. *Let ρr be the request sequence ρ followed by one additional request r. Then for any k, we have:*

$$c(V,d,C[0],\rho r,k+1) - c(V,d,C[0],\rho,k+1) \leq c(V,d,C[0],\rho r,k) - c(V,d,C[0],\rho,k)$$

PROOF. Let A be the final configuration in the optimum solution to $(V,d,C[0],\rho,k+1)$ and B be the final configuration for the optimum solution to $(V,d,C[0],\rho r,k)$. We observe that $r \in B$. We can consider B to instead be a solution to $(V,d,C[0],\rho,k)$; it will not necessarily be an optimum such solution. We now observe that there are flows f_A and f_B through the time-dilated graph corresponding to the solutions of $(V,d,C[0],\rho,k+1)$ and $(V,d,C[0],\rho,k)$ ending in A and B respectively, and that the costs of these flows are equal to $c(V,d,C[0],\rho,k+1)$ and $c(V,d,C[0],\rho r,k)$ respectively. Despite the fact that the *amount* of flow differs ($k+1$ for f_A and k for f_B), we can still apply theorem 1 to find a partial mapping $\mu : A \to B$; however since the values of the flows differ, there will be one path in the residual graph G_A that leads from the sink to the source rather than from sink to sink (this corresponds to the "extra" unit of flow in f_A). Recall that $r \in B$ and let $q \in A$ be such that $\mu(q) = r$. We now apply theorem 1 with $A_1 = \{q\}$ and $A_2 = A - \{q\}$, and note that $\mu(A_2) = B - \{r\}$ to get $c(\{q\} \cup B - \{r\}) + c(\{r\} \cup A - \{q\}) \leq c(A) + c(B)$. We observe that $|\{q\} \cup B - \{r\}| = k$. Thus this is a feasible solution to $(V,d,C[0],\rho,k)$, and the optimum solution costs only less. Similarly, we have $|\{r\} \cup A - \{q\}| = k+1$ and this is a feasible solution to $(V,d,C[0],\rho,k+1)$. Further, this solution includes a server at r, meaning it is also a solution to $(V,d,C[0],\rho r,k+1)$ with the same cost. We conclude that $c(V,d,C[0],\rho,k) + c(V,d,C[0],\rho r,k+1) \leq c(\{q\} \cup B - \{r\}) + c(\{r\} \cup A - \{q\})$. □

In other words, the difference in optimal cost when request r is tacked on cannot be larger when an additional server is available.

COROLLARY 2. *Let $c(V,d,C[0],\rho,x,X)$ represent the cost of the cheapest solution to $(V,d,C[0],\rho,x)$ which also has final configuration $C[m] = X$ for the given $X \subseteq V$. For any X and $x = |X|$, and any integer y, there exists Y such that $|Y| = y$, $|X \cap Y| = \min(x,y)$, and:*

$$c(V,d,C[0],\rho,y,Y) \leq c(V,d,C[0],\rho,x,X) + c(V,d,C[0],\rho,y) - c(V,d,C[0],\rho,x)$$

PROOF. Let $A = X$. Let B represent the final configuration in the optimum solution for $(V,d,C[0],\rho,y)$; thus $c(V,d,C[0],\rho,y,B) = c(V,d,C[0],\rho,y)$. We now apply theorem 1, constructing flows f_A and f_B and considering the flow through residual graph G_A to transform f_A to f_B. This will apply despite the fact that flows f_A and f_B have different values (x and y respectively). We will produce a partial matching $\mu : A \to B$. First, suppose that $x > y$. Then let $Y \subseteq A$ be those members of A which are matched. We have $|X \cap Y| = y$. The partial matching guarantees that $c(A - Y \cap B) + c(Y) \leq c(A) + c(B)$. Since $|A - Y \cap B| = x$, it is a feasible final configuration for $(V,d,C[0],\rho,x)$ and thus has cost at least the optimum cost for that problem. It follows that $c(V,d,C[0],\rho,x) + c(V,d,C[0],\rho,y,Y) \leq c(V,d,C[0],\rho,x,X) + c(V,d,C[0],\rho,y)$ which completes the proof. We also need the case $x < y$. Here, let $W \subseteq B$ be the members of B which remain unmatched. Let $Y = W \cup A$. Now we again apply theorem 1 to guarantee that $c(B-W) + c(Y) \leq c(A) + c(B)$. We observe that $|B - W| = x$ and thus it is a feasible solution for $(V,d,C[0],\rho,x)$ and has cost at least the optimum cost for that problem. We conclude that $c(V,d,C[0],\rho,x) + c(V,d,C[0],\rho,y,Y) \leq c(A) + c(B)$ which again completes the proof. □

This essentially says that for any y, there is some final configuration of y servers that overlaps X as much as possible, such that the additional optimal cost to end in this configuration is at most the difference between the optimal cost to end with any x server placements and the optimal cost to end with any y server placements.

4. METRIC EMBEDDING AND HSTS

A sequence of results on metric embedding allows us to transform the arbitrary metric space (V,d) given in the k-server definition to a tree with hierarchical properties. The following theorem is the main result of Fakcharoenphol et al [14]:

THEOREM 2. *For any metric space (V,d) and any $\alpha > 1$, we can randomly construct a rooted, weighted tree (T,d_T) along with a function $f : V \to T$. We write $d_T(x,y)$ for vertices $x,y \in T$ for the length of the path through T (according to weighting function d_T) between x and y. The construction guarantees that:*

1. *If $c_1(x)$ and $c_2(x)$ are both children of x in T, then $d_T(x,c_1(x)) = d_T(x,c_2(x))$.*
2. *For any node $x \in T$ not the root or a leaf, let $p(x)$ and $c(x)$ be a parent and a child of x respectively. Then $d_T(x,p(x)) = \alpha d_T(x,c(x))$.*
3. *All leaves of the tree are exactly $\log_\alpha \Delta$ hops from the root (in other words, the depth of the tree is $\log_\alpha \Delta$) where Δ is the diameter of the metric space (V,d).*
4. *For all nodes $u \in V$, $f(u)$ is a leaf of the tree T.*
5. *For any $u,v \in V$, $d_T(f(u),f(v)) \geq d(u,v)$.*
6. *For any $u,v \in V$, $E[d_T(f(u),f(v))] \leq (\alpha \log |V|) d(u,v)$.*

Suppose that we could solve k-server as long as the metric were a tree with the properties described. This would enable us to solve k-server on any metric space, with a somewhat worse competitive ratio.

THEOREM 3. *Suppose we could solve k-server online, with some expected competitive ratio γ, provided that the metric space given was a tree as described in theorem 2 with all requests at the leaves. Then we have an expected $O(\gamma\alpha \log |V|)$-competitive randomized online algorithm for k-server on any metric space.*

5. A HIERARCHICAL ALGORITHM

We consider the following more general version of the k-server problem. We are given a rooted tree with the properties described in theorem 2 *which we additionally assume to be a binary tree.* We also have an initial configuration and a vector κ telling us how many servers will be in this subtree at each time, along with a sequence of requests. Thus we are given $(T, d_T, C[0], \rho, \kappa)$, where $|C[0]| = \kappa[0]$ and both $C[0]$ and all elements of ρ come from the leaves of T. Let δ represent the distance from the root of T to any leaf. The goal is to select a sequence of configurations $C[1], C[2], ..., C[m]$ such that $\rho[t] \in C[t]$ for each t and $|C[t]| = \kappa[t]$ for each t. We define $d_T(C[t], C[t+1])$ to be the minimum cost of a matching between these two configurations; the matching should be maximal, but since $|C[t]|$ may not equal $|C[t+1]|$, there can be unmatched vertices on one side or the other (but not both). The cost of the solution is $\Sigma_{t=1}^{m} d_T(C[t], C[t+1]) + \delta g(\kappa)$ where $g(\kappa) = \Sigma_t |\kappa[t] - \kappa[t+1]|$. Effectively this means that we must pay to move servers which are leaving T to the root, or to move new incoming servers from the root to their locations.

Note that solving this more general version of k-server in an online scenario would allow us to solve k-server on a binary tree (the more general version reduces to k-server if we set $\kappa[t] = k$ for all t).

Instead of actually solving this problem directly, we will solve it in a hierarchical manner. We will construct (online) vectors κ_1, κ_2 corresponding to the children of the root of T, representing the number of servers in that subtree at each time. We will then recursively do the same for the instance $(T_i, d_T, C[0] \cap T_i, \rho \cap T_i, \kappa_i)$ (here T_i is the subtree rooted at the i'th child of the root, and $\rho \cap T_i$ represents the subset of requests that are made to nodes in subtree T_i). We let $c(T, d_T, C[0], \rho, \kappa)$ represent the optimum (minimum) cost for any set of configurations acting as a solution to this instance. Supposing that $C[t]$ is the optimum configuration for request t, we observe that we can set $\kappa_i[t] = |C[t] \cap T_i|$. We can then use $C[t] \cap T_i$ as the configuration for instance $(T_i, d_T, C[0] \cap T_i, \rho \cap T_i, \kappa_i)$ at time t, implying that:

$$c(T, d_T, C[0], \rho, \kappa) = \Sigma_{i=1}^{2} c(T_i, d_T, C[0] \cap T_i, \rho \cap T_i, \kappa_i) + (\delta - \delta')\Sigma_{i=1}^{2} g(\kappa_i)$$

Where δ' is the distance from the root of any child subtree T_i to one of its leaves.

Ideally, we would like to be able to select the κ_i online in order to minimize the right-hand side of this equation. This would correspond to selecting the optimum such vectors, and we could prove (by induction) that we would thereby have an optimum solution to k-server. The problem is that these vectors κ_i must be constructed online and will therefore not be optimum. We notice that there are two pieces to the right-hand side of the equation in question: the first part represents the cost which is "passed down" to the next level of the hierarchical process and the second part represents the cost of moving servers between subtrees. These parts should be considered separately, because multiplying the second part (cost of moving servers) by a competitive factor will lose us only the same competitive factor for our overall algorithm, whereas multiplying the first part (cost passed down) by a factor will lead to a competitive ratio exponential in the number of levels. We will define two functions which approximate these two costs and solve the problem online. This is essentially an instance of metrical task systems [4]; however, this instance has special properties which enable us to design a novel algorithm with much improved competitive ratio.

DEFINITION 1. *The* **Move Cost** *of vectors κ_i for instance $(T, d_T, C[0], \rho, \kappa)$ is*

$$MC = \delta\Sigma_{i=1}^{2} g(\kappa_i)$$

DEFINITION 2. *For ease of notation, we define ρ^t as the sequence of requests $\{\rho[0], \rho[1], ..., \rho[t]\}$.*

The **Hit Cost** *of a set of vectors κ_i for instance $(T, d_T, C[0], \rho, \kappa)$ is the sum over child trees T_i of the sum over all times t of:*

$$c(T_i, d_T, C[0] \cap T_i, \rho^t \cap T_i, \kappa_i[t]) - c(T_i, d_T, C[0] \cap T_i, \rho^{t-1} \cap T_i, \kappa_i[t])$$

Our main result will follow from hierarchically applying an algorithm to construct vectors κ_i to minimize the move cost and hit cost. There are three main parts to this proof: first, we need to show that there in fact exists a solution with low move cost and hit cost, second, that we can find such a solution, and third, that such solutions imply that the cost passed down to the next level of the hierarchy is bounded.

LEMMA 1. *$\delta' \leq \frac{1}{\alpha}\delta$ (where α is the hierarchical separation factor, as described in theorem 2).*

We define $HC(\kappa_i)$ as the hit cost on subtree T_i resulting from κ_i, so that the total hit cost is $HC = \Sigma_{i=1}^{2} HC(\kappa_i)$. We define $TC(\kappa_i) = c(T_i, d_T, C[0] \cap T_i, \rho \cap T_i, \kappa_i)$, so the total cost is $TC = \Sigma_{i=1}^{2} TC(\kappa_i)$.

THEOREM 4. *There exists a set of vectors κ_i with total move cost plus hit cost $MC^* + HC^* \leq \frac{\alpha+2}{\alpha-1} c(T, d_T, C[0], \rho, \kappa)$.*

PROOF. If $C[1], C[2], ..., C[m]$ are the optimum configurations, we will use vectors κ_i defined by $\kappa_i[t] = |C[t] \cap T_i|$. These vectors guarantee $c(T, d_T, C[0], \rho, \kappa) = \Sigma_{i=1}^{2} TC(\kappa_i) + (\delta - \delta')\Sigma_{i=1}^{2} g(\kappa_i)$. This means we need to show that $MC^* + HC^*$ is at most:

$$\frac{\alpha+2}{\alpha-1}(TC(\kappa_1) + TC(\kappa_2) + (\delta - \delta')(g(\kappa_1) + g(\kappa_2)))$$

Subtracting the move cost from both sides, it will be sufficient to show that $HC(\kappa_i) \leq TC(\kappa_i) + \frac{\alpha+2}{\alpha-1}(\delta - \delta')g(\kappa_i) - \delta g(\kappa_i)$. We observe based upon lemma 1 that this implies it will be sufficient to show that $HC(\kappa_i) \leq TC(\kappa_i) + 2\delta' g(\kappa_i)$.

We will prove this by induction on the value of $g(\kappa_i)$. If this value is zero, then $\kappa_i[t]$ is the same for all t. In this case the hit cost summation telescopes and the two sides of the equation will be equal. This is the base case; we continue to the inductive case. Here, we can assume that $g(\kappa_i) > 0$

so there exists some first time τ where $\kappa_i[\tau] \neq \kappa_i[\tau+1]$. We define a new vector κ'_i by letting $\kappa'_i[x] = \kappa_i[x]$ for $x > \tau$. For $x \leq \tau$, we set $\kappa'_i[x] = \kappa_i[\tau+1]$. We observe that $g(\kappa'_i) = g(\kappa_i) - |\kappa_i[\tau] - \kappa_i[\tau+1]|$. By applying the inductive hypothesis, the desired inequality will hold for κ'_i:

$$HC(\kappa'_i) \leq TC(\kappa'_i) + 2\delta' g(\kappa'_i)$$

We observe that most terms in the hit cost summation for κ_i, κ'_i are the same, and the terms which are different telescope in each sum. This yields:

$$\begin{aligned} HC(\kappa_i) - HC(\kappa'_i) = \\ c(T_i, d_T, C[0] \cap T_i, \rho^\tau \cap T_i, \kappa_i[\tau]) \\ -c(T_i, d_T, C[0] \cap T_i, \rho^\tau \cap T_i, \kappa'_i[\tau]) \end{aligned}$$

There is some set of configurations for κ_i. Let these be $C[0], C[1], ..., C[m]$. These configurations have the sum of their matching costs equal to $TC(\kappa_i)$. We will modify these configurations to get a sequence of configurations for κ'_i. In particular, we set $C'[x] = C[x]$ for $x > \tau$. For $x = \tau$, we make use of corollary 2 to find a configuration $C'[\tau]$ such that:

$$c(C'[\tau], C'[\tau+1]) \leq 2\delta' |\kappa_i[\tau] - \kappa_i[\tau+1]| + c(C[\tau], C[\tau+1])$$

$$\begin{aligned} c(T_i, d_T, C[0] \cap T_i, \rho^\tau \cap T_i, \kappa'_i[\tau], C'[\tau]) \leq \\ c(T_i, d_T, C[0] \cap T_i, \rho^\tau \cap T_i, \kappa_i[\tau], C[\tau]) - HC(\kappa_i) + HC(\kappa'_i) \end{aligned}$$

Of course, $TC(\kappa'_i)$ cannot exceed the cost of this set of configurations, so we have:

$$TC(\kappa'_i) \leq TC(\kappa_i) + HC(\kappa'_i) - HC(\kappa_i) + 2\delta'(g(\kappa_i) - g(\kappa'_i))$$

Substituting gives us the desired inequality. □

THEOREM 5. *There is a randomized online algorithm which can construct a set of vectors κ_i such that, if there exists a set of vectors with move cost MC^* and hit cost HC^*, we guarantee $E[MC] \leq MC^* + HC^*$ and $E[HC] \leq MC^* + HC^*$.*

The proof of theorem 5 is our main algorithmic result, and will be the subject of the next section. For now, we will assume this theorem in order to see how it leads to a result for the k-server problem in general.

THEOREM 6. *Suppose we compute a set of vectors κ_i with hit cost HC and move cost MC. Then these vectors guarantee $\Sigma_{i=1}^{2} c(T_i, d_T, C[0] \cap T_i, \rho \cap T_i, \kappa_i) \leq HC + \frac{1}{\alpha} MC$.*

PROOF. Note that the left and right sides of the desired inequality look very much like the sums over i of $TC(\kappa_i)$ and $HC(\kappa_i)$ respectively, with the addition of the move cost to the right side. It will be sufficient to show that:

$$TC(\kappa_i) \leq HC(\kappa_i) + \delta' g(\kappa_i)$$

We will show this by induction on the value of $g(\kappa_i)$. For the base case, if this value is zero then the hit cost summation telescopes and the two sides of the inequality will be equal. Otherwise, we let τ be the minimum such that $\kappa_i[\tau] \neq \kappa_i[\tau+1]$. We define κ'_i by $\kappa'_i[x] = \kappa_i[x]$ for $x > \tau$ and $\kappa'_i[x] = \kappa_i[\tau+1]$ for $x \leq \tau$. We observe that $g(\kappa'_i) = g(\kappa_i) - |\kappa_i[\tau] - \kappa_i[\tau+1]|$, and that the inductive hypothesis can thus be applied to κ'_i. Because most terms in the hit cost summation are identical, we have:

$$\begin{aligned} HC(\kappa_i) - HC(\kappa'_i) = \\ c(T_i, d_T, C[0], \rho^\tau, \kappa_i[\tau]) - c(T_i, d_T, C[0], \rho^\tau, \kappa'_i[\tau]) \end{aligned}$$

There is some set of configurations which comprise the cheapest solution for κ'_i; let these configurations be $C'[0], C'[1], ...$ where their pairwise matching costs sum to $TC(\kappa'_i)$. We will construct a set of configurations for κ_i, then argue that $TC(\kappa_i)$ is at most the sum of pairwise matching costs for these configurations. For $x > \tau$ we let $C[x] = C'[x]$. We set $C[\tau]$ according to corollary 2, guaranteeing that $C[\tau]$ overlaps $C'[\tau]$ as much as possible and the cost to end in configuration $C[\tau]$ is at most:

$$\begin{aligned} c(T_i, d_T, C[0], \rho^\tau, \kappa_i[\tau], C[\tau]) \leq \\ c(T_i, d_T, C[0], \rho^\tau, \kappa'_i[\tau], C'[\tau]) + HC(\kappa_i) - HC(\kappa'_i) \end{aligned}$$

The cost represented by $TC(\kappa_i)$ is at most the sum of the matching costs of the $C[x]$. For x up to τ we have cost $c(T_i, d_T, C_0, \rho^\tau, \kappa_i[\tau], C[\tau])$. The matching cost between $C[\tau]$ and $C[\tau+1]$ will be very similar to the cost of matching $C'[\tau]$ and $C'[\tau+1]$, except that we have $g(\kappa_i) - g(\kappa'_i)$ fewer servers to match. For each server we don't have to match (representing a difference in total number of servers between configurations), we must pay δ'. For matching costs $C[\tau+1]$ onward, we pay the same as we paid for κ'_i, so we have:

$$TC(\kappa_i) \leq TC(\kappa'_i) + HC(\kappa_i) - HC(\kappa'_i) + \delta'(g(\kappa_i) - g(\kappa'_i))$$

Substituting from the various equations, summing this over all i, and recalling that $\delta' \leq \frac{1}{\alpha}\delta$ gives the desired inequality. □

THEOREM 7. *We can solve k-server on a binary tree with the properties described in theorem 2 with an expected competitive ratio $\Theta(\log \Delta)$ provided $\alpha = \Omega(\log \Delta)$.*

PROOF. We solve the problem hierarchically as described. The actual cost paid to move servers will be bounded by the total move cost. For level j, let z_j represent the sum over subtrees rooted at level j of $c(T, d_T, C[0] \cap T, \rho \cap T, \kappa_T)$. Let $L \leq \log \Delta$ be the number of levels. At the top level, we have $z_L = OPT$. By theorem 4 there exists a set of solutions at level j with total hit cost plus move cost bounded by $\frac{\alpha+2}{\alpha-1} z_j$. We then find solutions with expected hit cost and move cost both bounded by this value according to theorem 5. It follows by theorem 6 that the next level has $E[z_{j-1}] \leq \frac{(\alpha+2)(\alpha+1)}{\alpha(\alpha-1)} z_j$. The expected move cost is also bounded by z_j, so summing this up we have a total move cost of at most: $\Sigma_j z_j$ and using martingales to bound the expected cost we get a geometric sum. If we let $A = \frac{(\alpha+2)(\alpha+1)}{\alpha(\alpha-1)}$ then the summation will look like $\frac{A^L - 1}{A-1}$. If we set $A \leq 1 + \frac{1}{L}$ then this will give us an $O(L)$ bound as desired. Making use of the definition of A and solving for α shows that for $\alpha = \Omega(L)$ we can satisfy this inequality. □

6. ONLINE ALGORITHM

It remains to describe our local algorithm for determining how to partition servers between subtrees, and to prove theorem 5. This is essentially an instance of metrical task systems, where the possible states are equally spaced points along a line and refer to partitions of servers between the left and right subtrees. Each state is 2δ away from its neighbors, representing the cost to move a server between subtrees.

We first observe that the costs applied to the various states at each time t look like the change in hit cost. If the request $\rho[t] \in T_1$ then we have:

$$c_x[t] = c(T_1, d_T, C[0] \cap T_1, \rho^t \cap T_1, x) - c(T_1, d_T, C[0] \cap T_1, \rho^{t-1} \cap T_1, x)$$

Otherwise if request $\rho[t] \in T_2$ then we have:

$$c_x[t] = c(T_2, d_T, C[0] \cap T_2, \rho^t \cap T_2, \kappa[t] - x) - c(T_2, d_T, C[0] \cap T_2, \rho^{t-1} \cap T_2, \kappa[t] - x)$$

By applying corollary 1, we see that this will be either non-decreasing or non-increasing as we traverse the possible states.

We consider these cost vectors to arrive one at a time online, as does the current number of servers $\kappa[t]$. Our goal is to minimize the total cost (essentially to solve this metrical task system instance). We define the *work function value* for a state to be the minimum cost way to end in that state; thus for state x we have $w_x[t] = \min\{w_x[t-1] + c_x[t], w_{x-1}[t] + 2\delta, w_{x+1}[t] + 2\delta\}$. We will essentially try to stay in the state with minimum $w_x[t]$ at each time, but we must add some randomization to the process to prevent the adversary from causing us to move between states (thus move servers between subtrees) with too much frequency. The algorithm is described below.

Algorithm 1 SUBPROBLEM(distance 2δ, initial state s_i): outputs a sequence of states as a solution to the above problem

1: generate a random number r between -1 and 1.
2: output initial state $x[0]$ corresponding to $C[0]$
3: **for** each timestep t **do**
4: current number of servers, κ_t arrives
5: cost vector $c_x[t]$=$(c_0[t], c_1[t], ..., c_{\kappa_t}[t])$ arrives
6: calculate the work function value for each state $w_x[t]$
7: find the state $x[t]$ that minimizes $X_x[t] = w_x[t] - xr(2\delta)$.
8: output the current state $x[t]$

THEOREM 8. *Algorithm 1 pays expected hit cost ≤ the total cost of the optimum offline solution to the metrical task system instance.*

PROOF. We claim that over time 0 through t our algorithm pays hit cost at most $X_{x[t]}[t] - X_{x[0]}[0]$ where $X_x[t] = w_x[t] - 2rx\delta$. We will prove this by induction.

For the base case, t=0, there is no hit cost yet, and $X_{x[t]}[t] - X_{x[0]}[0] = 0$. Inductively, we will consider two cases. If $x[t] = x[t-1]$ then the hit cost paid at time t will be exactly $c_{x[t]}[t]$. We observe that if $w_x[t] = w_{x+1}[t] + 2\delta$ or $w_x[t] = w_{x-1}[t] + 2\delta$, then because r is between positive and negative one, the probability of selecting $x[t] = x$ will be zero. It follows that for whatever state x we return, we must have $w_x[t] = w_x[t-1] + c_x[t]$. From here we conclude that $X_{x[t]}[t] = X_{x[t-1]}[t-1] + c_{x[t]}[t]$ and that our algorithm pays hit cost $c_{x[t]}[t]$, completing the induction. The second case has $x[t] \neq x[t-1]$. However, we can consider cost to arrive in ϵ sized increments without changing the behavior of the algorithm, and observe that at the moment we transition from one state to another, the values of $w_{x[t-1]}[t] - 2rx[t-1]\delta = w_{x[t]}[t] - 2rx[t]\delta$. So at this moment we pay no hit cost (although we do pay to move servers) and the value of $X_{x[t]}$ remains unchanged.

The overall hit cost paid by our algorithm is therefore bounded by $X_{x[t]}[t] - X_{x[0]}[0]$. Since the initial work function value at $x[0]$ is zero, this is at most:

$$w_{x[t]}[t] - 2r\delta x[t] + 2r\delta x[0]$$

Since we select $x[t]$ to *minimize* the value of $w_{x[t]}[t] - 2r\delta x[t]$, we can conclude that if the *optimum* algorithm would have final partition p, then $w_{x[t]}[t] - r2\delta x[t] \leq w_p[t] - 2r\delta p$. Of course, the value of $w_p[t]$ is just the total cost of the optimum offline solution, so we have our total hit cost bounded by:

$$OPT + 2r\delta(x[0] - x[p])$$

Obviously this depends upon r, but in expectation $r = 0$ so the expected hit cost paid by our algorithm looks like the optimum total cost for the offline solution. □

THEOREM 9. *Algorithm 1 pays expected move cost ≤ the total cost of the optimum offline solution to the metrical task system instance.*

PROOF. Let $M[t]$ be the total move cost paid by our algorithm up until time t. Define a potential function:

$$\phi[t] = 2k\delta - \sum_{x=0}^{k-1} |w_{x+1}[t] - w_x[t]|$$

We will show that at any timestep t, $E[M[t]] + \frac{1}{2}\phi[t] \leq OPT[t]$, where $OPT[t]$ is the total cost paid by the optimum solution so far.

By application of corollary 1, we will assume that, $c_0[t] \leq c_1[t] \leq \ldots \leq c_k[t]$ (the other case where there is a request on the opposite child subtree will be symmetrc). We can view timestep t as a series of smaller timesteps: $\frac{c_0}{\epsilon}$ steps in which ϵ cost is put on all states, followed by $\frac{c_1 - c_0}{\epsilon}$ steps in which ϵ work is put on all states except c_0, etc. Notice that the cost of the optimum offline solution cannot get worse because of this change, while no online algorithm can achieve a lower cost.

We will show that $E[M[t]] + \frac{1}{2}\phi[t] \leq OPT[t]$ by induction on t. The base case is $t = 0$. Here, $E[M[t]] = OPT[t] = 0$, since no requests have arrived yet. Since we are given an initial state, $w_x[0] = 0$ for exactly one state. For all other states, $w_x[0] = 2\delta|x - x[0]|$ (the cost to move from our initial state to state x). So

$$\phi[0] = 2k\delta - \sum_{x=0}^{k-1} |w_{x+1}[0] - w_x[0]| = 2k\delta - \sum_{i=0}^{k-1} 2\delta = 0$$

Now, assume for timestep $t-1$, $E[M[t-1] + \frac{1}{2}\phi[t-1] \leq OPT[t-1]$. Consider how each term will change at timestep t.

By definition of the work function, $OPT[t] = min_x(w_x[t])$.

Our expected move cost is just the probability that we move times the distance that we would move. Suppose we are in state x. By definition of the algorithm, there is only one state, y, to which we might move (this is the cheapest state to which no cost was added). Additionally, since we are considering ϵ size cost, there is only one state x from which we might move; also note that $x > y$ because of our assumption about the costs. If there are two valid states to move from, we will just shrink the timesteps until there is only one valid state to move from. Now, the probability that we move in this step is exactly the probability that our random number r is high enough that we would move during this timestep but low enough that we would not have moved given the previous work function values. This requires both $X_x[t-1] \leq X_y[t-1]$ and $X_y[t] \leq X_x[t]$. The first inequality implies that $w_x[t-1] - w_y[t-1] \leq 2r(x-y)\delta$ and the second implies that $w_x[t] - w_y[t] \geq 2r(x-y)\delta$. Since no cost was added to y and ϵ cost was added to x, we have:

$$w_x[t-1] - w_y[t-1] \leq 2r(x-y)\delta \leq w_x[t-1] - w_y[t-1] + \epsilon$$

This means that r must fall within a range of $\frac{\epsilon}{2(x-y)\delta}$ and since r is chosen uniformly between plus and minus one we have:

$$Pr[\text{move } x \text{ to } y] \leq \frac{\epsilon}{4(x-y)\delta}$$

If we do move from x to y, we pay exactly $2\delta(x-y)$. This gives us an expected move cost $= \frac{\epsilon}{2}$.

Finally, we examine $\phi[t]$. For most pairs of states $w_{x+1}[t] - w_x[t] = w_{x+1}[t-1] - w_x[t-1]$, with the exception of one pair of states where they differ by ϵ. Depending on which was larger, we will have $\phi[t] = \phi[t-1] - \epsilon$ or $\phi[t] = \phi[t-1] + \epsilon$. The first case occurs when the minimum work value did not actually increase, and thus when $OPT[t] = OPT[t-1]$, whereas the second occurs when $OPT[t] = OPT[t-1] + \epsilon$.

This shows that, at each timestep t, $E[M[t]] + \frac{1}{2}\phi[t] \leq OPT[t]$. Notice that $\phi[t] \geq 0$ for all t, since $|w_{x+1}[t] - w_x[t]| \leq 2\delta$ by definition of the work function (we could handle request t and end in state $x+1$ by ending in state x and then paying 2δ to move to $x+1$). Therefore, after the last request has arrived, we can say that $E[M] \leq OPT$. □

7. REFERENCES

[1] Y. Bartal. Probabilistic approximation of metric spaces and its algorithmic applications. *37th IEEE Symposium on Foundations of Computer Science*, 1996.

[2] Y. Bartal. On approximating arbitrary metrics by tree metrics. *30th ACM Symposium on Theory of Computing*, 1998.

[3] Y. Bartal, B. Bollobás, and M. Mendel. A Ramsey-type Theorem for Metric Spaces and its Applications for Metrical Task Systems and Related Problems. *42nd IEEE Symposium on Foundations of Computer Science*, 2001.

[4] Y. Bartal, A. Blum, C. Burch, and A. Tomkins. A polylog(n)-competitive algorithm for metrical task systems. *ACM Symposium on Theory of Computing*, 1997.

[5] A. Blum, H. Karloff, Y. Rabani, M. Saks. A Decomposition Theorem for Task Systems and Bounds for Randomized Server Algorithms. *33rd IEEE Symposium on Foundations of Computer Science*, 1992.

[6] Y. Bartal, E. Koutsoupias. On the competitive ratio of the work function algorithm for the k-server problem. *17th Symposium on Theoretical Aspects of Computer Science*, 2000.

[7] Y. Bartal, M. Mendel. Randomized k-Server Algorithms for Growth-Rate Bounded Graphs. *Symposium on Discrete Algorithms*, 2004.

[8] M. Chrobak, H. Karloff, T. Payne, S. Vishwanathan. New Results on Server Problems. *Symposium on Discrete Algorithms*, 1990.

[9] M. Chrobak and 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. Larmore. HARMONIC is three-Competitive for two servers. *Theoretical Computer Science, 98:339-346*, 1992.

[11] M. Chrobak and L. Larmore. An Optimal On-line Algorithm for k-Server on Trees. *SIAM Journal on Computing, 20(1)*, 1991.

[12] M. Chrobak and J. Sgall. A simple analysis of the harmonic algorithm for two servers. *Information Processing Letters, 75:75-77*, 2000.

[13] B. Csaba and S. Lodha. A randomized on-line algorithm for the k-server problem on a line. *Technical Report 2001-34, DIMACS*, 2001.

[14] J. Fakcharoenphol, S. Rao, and K. Talwar. A tight bound on approximating arbitrary metrics by tree metrics. *ACM Symposium on Theory of Computing*, 2003.

[15] A. Fiat, R. Karp, M. Luby, L. McGeoch, D. Sleator, and N. Young. Competitive Paging Algorithms. *Journal of Algorithms, 12:685-699*, 1991.

[16] A. Fiat and M. Mendel. Better Algorithms for Unfair Metrical Task Systems and Applications. *SIAM Journal on Computing, 32:1403-1422*, 2003.

[17] A. Fiat, Y. Rabani, Y. Ravid. Competitve k-Server Algorithms. *31st IEEE Symposium on Foundations of Computer Science*, 1990.

[18] E. Koutsoupias. Weak adverseries for the k-server problem. *40th IEEE Symposium on Foundations of Computer Science*, 1999.

[19] E. Koutsoupias and C. Papadimitriou. On the k-server conjecture. *Journal of the ACM 42(5)*, 1995.

[20] M. Manasse, L. McGeoch, and D. Sleator. Competitive Algorithms for Server Problems. *Journal of Algorithms (11, 208-230)*, 1990.

[21] S. Seiden. Unfair Problems and Randomized Algorithms for Metrical Task Systems. *Information and Computation, 148:219-240*, 1999.

[22] S. Seiden. A General Decomposition Theorem for the k-Server Problem. *European Symposium on Algorithms*, 2001.