Random Walks on Weighted Graphs, and Applications to On-line Algorithms (Extended Abstract)

Don Coppersmith
Peter Doyle
Prabhakar Raghavan
Marc Snir

March 1990
Version 1.0A1 dated 15 September 1994

Abstract

We study the design and analysis of randomized on-line algorithms. We show that this problem is closely related to the synthesis of random walks on graphs with positive real costs on their edges.

1 Introduction

Let G be a weighted undirected graph with n nodes $\{1, \ldots, n\}$; $c_{ij} = c_{ji} > 0$ is the *cost* of the edge connecting nodes i and j, $c_{ii} = 0$. Consider a random walk on the graph G, executed according to a transition probability matrix $P = (p_{ij})$; p_{ij} is the probability that the walk moves from node i to node j, and the walk pays a cost c_{ij} in the process. Let e_{ij} (not in general $= e_{ji}$) be the expected cost of a random walk starting at node i and ending at node j (e_{ii} is the expected cost of a round trip from i). We say that the random walk has *stretch* c if there exists a constant a such that, for any sequence $i_0, i_1, \ldots, i_\ell$ of nodes $\sum_{j=1}^{\ell} e_{i_{j-1}i_j} \leq c \cdot \sum_{j=1}^{\ell} c_{i_{j-1}i_j} + a$. We prove the following tight result:

Any random walk on a weighted graph with n nodes has stretch at least $n-1$, and any weighted graph with n nodes has a random walk with stretch $n-1$.

The upper bound proof is constructive, and shows how to compute the transition probability matrix P from the cost matrix $C = (c_{ij})$. The proof uses new connections between random walks and effective resistances in networks of resistors, together with results from electric network theory. Consider a network of resistors with n nodes, and *conductance* σ_{ij} between nodes i and j (nodes i and j are connected by a resistor with *branch resistance* $1/\sigma_{ij}$). Let R_{ij} be the *effective resistance* between nodes i and j (i.e., $1/R_{ij}$ is the current that would flow from i to j if one volt were applied between i and j; it is known that $1/R_{ij} \geq \sigma_{ij}$). Let the *resistive random walk* be defined by the probabilities $p_{ij} = \sigma_{ij}/\sum_k \sigma_{ik}$. In Section 3 we show that this random walk has stretch $n-1$ in the graph with costs $c_{ij} = R_{ij}$. Thus, a random walk with optimal stretch is obtained by computing the *resistive inverse* (σ_{ij}) of the cost matrix (c_{ij}): a network of branch conductances $(\sigma_{ij} \geq 0)$, so that, for any i, j, c_{ij} is the effective (not branch) resistance between i and j. Unfortunately, not all cost matrices have resistive inverses (with positive conductances). However, every matrix (c_{ij}) has a *generalized resistive inverse*: a network of non-negative branch conductances σ_{ij} with associated effective resistances R_{ij}, such that either $R_{ij} = c_{ij}$, or $R_{ij} < c_{ij}$ and $\sigma_{ij} = 0$. In Section 4 we show that the resistive random walk has stretch $n-1$ for the graph with costs R_{ij}, and consequently for the graph with costs c_{ij}, since it never traverses those edges whose costs it underestimates.

Chandra *et al.* [6] use electric networks to *analyze* a particular random walk, in which $p_{ij} = (1/c_{ij})/(\sum_k 1/c_{ik})$. Traditionally, this is how electric networks have been used in studying random walks: to *analyze* a given random walk (cf. Doyle and Snell [9]). Here we instead use electric networks to *synthesize* a (different, in general) random walk with optimal stretch.

Next, we outline the relevance of this random walk synthesis problem to the design of on-line algorithms. Consider the following game played between a *cat* and a *mouse* on the graph G. Round r starts with both cat and mouse on the same node i_{r-1}. The mouse moves to a new node i_r not known to the cat; the cat then walks on the graph until it reaches the mouse at i_j, at which point round $j+1$ starts. A strategy for the cat is *c-competitive* if there exists a constant a such that for any sequence $i_0, i_1, \ldots i_k$ of nodes the cat's expected cost is $\leq c \cdot$ (the mouse's cost) $+a$. The *competitiveness coefficient* of the

cat-and-mouse game is the infimum of c such that a c-competitive strategy exists. A random walk with stretch c defines a strategy for the cat that is c-competitive: in each round, the cat executes a random walk according to P until it finds the mouse. This strategy is very simple, and *memoryless*: the cat need not remember its previous moves, and the next cat move depends only on its current position.

We show that this cat-and-mouse game is at the core of many other on-line algorithms that have evoked tremendous interest of late [2, 3, 4, 7, 8, 10, 14, 16, 17, 18]. We consider two settings. The first is the *k-server problem*, defined in [14]. An on-line algorithm manages k mobile servers located at the nodes of a graph G whose edges have positive real lengths; it has to satisfy a sequence of requests for service at node v_i, $i = 1, 2, \ldots$, by moving a server to v_i unless it already has a server there. Each time it moves a server, it pays a cost equal to the distance moved by that server. We compare the cost of such algorithm, to the cost of an adversary that, in addition to moving its servers, also generates the sequence of requests. The competitiveness of an on-line algorithm is defined with respect to these costs (Section 5) [2, 17]. It was conjectured in [14] that for every cost matrix there exists a k-competitive algorithm for this problem. Repeated attempts to prove this conjecture have succeeded only in a few special cases [7, 8, 17]. We use our optimal random walk to derive a k-competitive server algorithm in two situations: (1) when the graph G has a resistive inverse, and (2) when the graph G has $k + 1$ nodes. This includes all previously known cases where the conjecture was proven true, as well as many new cases. We do so with a single unified theory — that of resistive inverses. The algorithm is very simple, and memoryless.

The other setting is the *metrical task system (MTS)*, defined in [4]. A MTS consists of a weighted graph (the nodes of the graph are states, and edge weights are the costs of moving between states). The algorithm occupies one state at any time. A *task* is represented by a vector $(c_1, \ldots, c_n)$, where c_i is the cost of processing the task in state i. The algorithm is presented a sequence of tasks $\mathcal{T} = T_1, T_2, \ldots$ and can move before processing each task. The cost incurred by the algorithm is the sum of the costs of moving and processing tasks. A $(2n - 1)$-competitive on-line algorithm for MTS is presented in [4], and shown to be optimal. The algorithm is deterministic, but somewhat complex. In Section 6 we present a simple, memoryless randomized algorithm for MTS that is $(2n - 1)$-competitive.

2 Lower bound on Stretch

Theorem 2.1 *For any $n \times n$ cost matrix C and any transition probability matrix P, the stretch of the random walk defined by P on the graph with weights given by C is $\geq n-1$.*

Proof: We can assume w.l.o.g. that P is irreducible (the underlying directed graph is strongly connected). Let ϕ_i be the ith component of the left eigenvector of P for the eigenvalue 1 (when P is aperiodic, this is the stationary probability of node i), so that $\phi_j = \sum_i \phi_i p_{ij}$. Let $e_i = \sum_j p_{ij} c_{ij}$ denote the expected cost of the first move out of node i, and let $E = \sum_i \phi_i e_i = \sum_{ij} \phi_i p_{ij} c_{ij}$ be the average cost per move. We have

$$\begin{aligned} \sum_{i,j} (\phi_i p_{ij}) e_{ji} &= \sum_i \phi_i \left(\sum_j p_{ij} e_{ji} \right) = \\ \sum_i \phi_i (e_{ii} - e_i) &= \sum_i \phi_i (E/\phi_i - e_i) = (n-1)E \end{aligned}$$

while $\sum_{i,j} (\phi_i p_{ij}) c_{ji} = \sum_{i,j} (\phi_i p_{ij}) c_{ij} = E$. Thus, $\sum_{i,j} (\phi_i p_{ij}) e_{ji} = (n-1) \sum_{i,j} (\phi_i p_{ij}) c_{ji}$.

Notice that, if each directed edge (ji) (note the order!) is counted with multiplicity proportional to $\phi_i p_{ij}$, then a flow condition is satisfied: the total multiplicity of edges leading out of i is equal to that of those leading into i. Thus, the above equation represents a convex combination of cycles so that there is some cycle $(i_1, i_2, \ldots i_\ell, i_{\ell+1} = i_1)$ with stretch at least $n-1$; thus,

$$\sum_{j=1}^{\ell} e_{i_j i_{j+1}} \geq (n-1) \sum_{j=1}^{\ell} c_{i_j i_{j+1}}.$$

□

The symmetry of the cost matrix C is necessary for the theorem.

3 Upper bound: resistive case

We next consider the complementary upper bound problem: given C, to synthesize a matrix P that achieves a stretch of $n-1$ on C. In this section

we will describe a construction and proof for a class of matrices C known as *resistive matrices*.

Let (σ_{ij}) be a non-negative symmetric real matrix with zero diagonal. Build the support graph (V, E), with vertex set $V = \{1, 2, ..., n\}$ and edge set $E = \{(i\ j) \mid \sigma_{ij} > 0\}$, and let (V, E) be connected. Consider a network of resistors based on (V, E), such that the resistor between vertices i and j has *branch conductance* σ_{ij}, or *branch resistance* $1/\sigma_{ij}$.

Let c_{ij} be the *effective resistance* between vertices i and j. (A unit voltage between i and j in this network of resistors results in a current of $1/c_{ij}$.) We require that the support graph be connected so that the effective resistances will be finite.

Definition 1 *A cost matrix* (c_{ij}) *is* resistive *if it is the matrix of effective resistances obtained from a connected non-negative symmetric real matrix* (σ_{ij}) *of conductances. The matrix* (σ_{ij}) *is the* resistive inverse of C. □

The following facts are not difficult to prove, and follow from standard electric network theory [19]. Resistive cost matrices are symmetric, finite, positive off the diagonal, zero on the diagonal, and satisfy the triangle inequality: $c_{ij} + c_{jk} \geq c_{ik}$. A principal submatrix of a resistive cost matrix is resistive.

Define two $(n-1) \times (n-1)$ matrices $\bar{\sigma}, \bar{C}$ by

$$\begin{aligned} \bar{\sigma}_{ii} &= \sum_{j \leq n, j \neq i} \sigma_{ij}, \quad 1 \leq i \leq n-1, \\ \bar{\sigma}_{ij} &= -\sigma_{ij}, \quad i \neq j, \quad 1 \leq i, j \leq n-1, \\ \bar{c}_{ij} &= [c_{in} + c_{jn} - c_{ij}]/2, \quad 1 \leq i, j \leq n-1. \end{aligned}$$

Then $\bar{\sigma}$ is the inverse of $\bar{C}$:

$$\sum_{j=1}^{n-1} \bar{\sigma}_{ij} \bar{c}_{jk} = \delta_{ik}.$$

It can happen that a given cost matrix $C = (c_{ij})$ gives rise to a putative resistive inverse with some negative conductances:

$$\exists i, j : \quad \sigma_{ij} < 0$$

and in this case there is no resistive inverse for C.

Examples of resistive cost matrices include:
(1) Any three points with the distances satisfying the triangle inequality.
(2) Points on a line: vertex i is at a real number r_i, with $c_{ij} = \mid r_i - r_j \mid$.
(3) The uniform cost matrix $c_{ij} = d$, if $i \neq j$.
(4) Tree closure: given a spanning tree on n vertices and positive costs for the tree edges, the distance between any pair of points equals the distance between them on the tree.
(5) A cost matrix C given by a graph with $m + n$ vertices $x_1, x_2, \ldots, x_m$, $y_1, y_2, \ldots, y_n$, $m, n > 1$, where $c_{x_i,x_j} = 2m$, $c_{y_i,y_j} = 2n$, and $c_{x_i,y_j} = m + n - 1$. The associated resistive inverse is a complete bipartite graph $K_{m,n}$ with resistors of resistance mn on each edge. This example cannot be expressed as a tree closure (Example (4) above): for if C were a tree closure, then the midpoint of the tree path joining x_1 and x_2 would be at distance $n - 1$ from both y_1 and y_2, contradicting $c_{y_1,y_2} = 2n > 2(n-1)$.

If C is a resistive cost matrix, its resistive inverse (σ_{ij}) provides a way of synthesizing an optimal random walk P achieving a stretch of $n - 1$. In fact, in determining the stretch of a random walk, it suffices to consider sequences of nodes $i_1, i_2, \ldots i_\ell, i_{\ell+1}$ that form a cycle in G.

Theorem 3.1 *Let $C = (c_{ij})$ be a resistive cost matrix and (σ_{ij}) its resistive inverse. Let the transition probabilities be $p_{ij} = \sigma_{ij}/(\sum_{\ell \neq i} \sigma_{i\ell})$. Then every cycle $(v_1, v_2, ..., v_\ell, v_{\ell+1} = v_1)$ has stretch $n - 1$:*

$$\sum_{i=1}^{\ell} e_{v_i v_{i+1}} = (n-1) \cdot \sum_{i=1}^{\ell} c_{v_i v_{i+1}}.$$

Proof: Following Doyle and Snell [9] we define the *escape probability* $P_{esc}(ij)$ to be the probability that a random walk, starting at vertex i, will reach vertex j before returning to vertex i. Doyle and Snell [9] show that

$$P_{esc}(ij) = \frac{1/c_{ij}}{\sum_k \sigma_{ik}}.$$

On average, out of each $\sum_{gh} \sigma_{gh}$ steps, the random walk visits vertex i with frequency $\sum_k \sigma_{ik}$, and the number of traversals of the ordered edge (ij) is σ_{ij}. The average cost of $\sum_{gh} \sigma_{gh}$ steps is

$$\sum_{gh} \sigma_{gh} c_{gh} = 2(n-1),$$

from Foster's Theorem [11, 12]. Of the $\sum_k \sigma_{ik}$ round trips to vertex i, the number visiting vertex j is

$$P_{esc}(ij) \sum_k \sigma_{ik} = 1/c_{ij}.$$

So the average cost of a round trip from vertex i to j and back to i is

$$\frac{\sum_{gh} \sigma_{gh} c_{gh}}{1/c_{ij}} = 2(n-1) \cdot c_{ij} = (n-1) \cdot [c_{ij} + c_{ji}].$$

This cost is also, by definition, $e_{ij} + e_{ji}$, so that

$$e_{ij} + e_{ji} = (n-1) \cdot [c_{ij} + c_{ji}].$$

So the stretch of any two-cycle is $n-1$.

We need a bound on the stretch of any cycle, not just two-cycles. The stationary probability of traversing the directed edge (ij) is $\sigma_{ij} / \sum_{gh} \sigma_{gh}$, which is symmetric because σ is symmetric. Thus our random walk is a *reversible* Markov chain [13]. For any cycle $(v_1, v_2, ..., v_\ell, v_{\ell+1} = v_1)$, the expected number of forward traversals of the cycle (not necessarily consecutive) is the same as the expected number of backward traversals of the cycle, and the expected cost per forward traversal is the same as the expected cost per backward traversal. Thus

$$\begin{aligned}
\sum_{i=1}^{\ell} e_{v_i v_{i+1}} &= \sum_{i=1}^{\ell} e_{v_{i+1} v_i} \\
&= \frac{1}{2} \left[\sum_{i=1}^{\ell} e_{v_i v_{i+1}} + \sum_{i=1}^{\ell} e_{v_{i+1} v_i} \right] \\
&= \frac{1}{2} \sum_{i=1}^{\ell} \left[e_{v_i v_{i+1}} + e_{v_{i+1} v_i} \right] \\
&= \frac{1}{2} \sum_{i=1}^{\ell} (n-1) \left[c_{v_i v_{i+1}} + c_{v_{i+1} v_i} \right] \\
&= \sum_{i=1}^{\ell} (n-1) c_{v_i v_{i+1}}.
\end{aligned}$$

So every cycle has stretch $n-1$. □

4 Upper bound: non-resistive case

In this section we prove the existence of a generalized resistive inverse. The generalized resistive inverse turns out to be the solution to a convex variational problem, and we present a simple iterative algorithm for finding it. From the generalized resistive inverse we get an $n-1$-competitive strategy for the cat-and-mouse game with an arbitrary positive symmetric cost matrix.

Theorem 4.1 *Let C be any positive symmetric cost matrix. Then there is a unique resistive cost matrix $\hat{C}$ with associated conductance matrix σ, such that $\hat{c}_{ij} \leq c_{ij}$, $\sigma_{ij} \geq 0$ and $\hat{c}_{ij} = c_{ij}$ if $\sigma_{ij} \neq 0$.*

Thus σ is the generalized resistive inverse of C.

Proof: For simplicity, we will limit the discussion to the case of the triangle graph, with assigned costs $R_0 = c_{1,2}$, $S_0 = c_{1,3}$, $T_0 = c_{2,3}$, and with edge conductances $a = \sigma_{1,2}$, $b = \sigma_{1,3}$, $c = \sigma_{2,3}$ and corresponding effective resistances $R = R_{1,2}$, $S = R_{1,3}$, $T = R_{2,3}$. This case will exhibit all the features of the general case, and yet allow us to get by without cumbersome subscripts. Please note, however, that for a triangle graph a cost matrix is resistive if and only if it satisfies the triangle inequality, while for a general graph the triangle inequality is necessary but by no means sufficient. Needless to say, we will make no use of this condition for resistivity in our analysis of the triangle graph.

We begin by recalling the relevant electrical theory (cf. Weinberg [19] and Bott and Duffin [5]). The admittance matrix of our network is

$$K = \begin{pmatrix} a+b & -a & -b \\ -a & a+c & -c \\ -b & -c & b+c \end{pmatrix}.$$

If you hook the network up to the world outside so as to establish node voltages v_1, v_2, v_3, the currents I_1, I_2, I_3 flowing into the network at the three nodes are given by

$$\begin{pmatrix} I_1 \\ I_2 \\ I_3 \end{pmatrix} = K \begin{pmatrix} v_1 \\ v_2 \\ v_3 \end{pmatrix}.$$

The power being dissipated by the network is

$$(I_1v_1 + I_2v_2 + I_3v_3) = \begin{pmatrix} v_1 & v_2 & v_3 \end{pmatrix} K \begin{pmatrix} v_1 \\ v_2 \\ v_3 \end{pmatrix}$$

which is ≥ 0. The matrix K is non-negative definite, with 0-eigenvector $(1,1,1)$. Label its eigenvalues

$$0 = \lambda_0 \leq \lambda_1 \leq \lambda_2.$$

On the orthogonal complement $P = \{v_1 + v_2 + v_3 = 0\}$ of $(1,1,1)$, K has eigenvalues λ_1, λ_2, and the determinant of $K|_P$ — that is, the product of the non-zero eigenvalues of K — is given by the next-to-lowest order coefficient of the characteristic polynomial of K, which can be expressed using Kirchhoff's tree formula:

$$\begin{aligned} \det K|_P &= \lambda_1\lambda_2 \\ &= \lambda_0\lambda_1 + \lambda_0\lambda_2 + \lambda_1\lambda_2 \\ &= \begin{vmatrix} a+b & -a \\ -a & a+c \end{vmatrix} + \begin{vmatrix} a+b & -b \\ -b & b+c \end{vmatrix} \\ &\quad + \begin{vmatrix} a+c & -c \\ -c & b+c \end{vmatrix} \\ &= (ab+ac+bc) + (ab+ac+bc) \\ &\quad + (ab+ac+bc) \\ &= 3D. \end{aligned}$$

Here the *discriminant* $D = ab + ac + bc$ is obtained by summing over the spanning trees of the network the product of the conductivities of the edges making up the tree (cf. Bott and Duffin [5]). The effective resistances are obtained by taking the gradient of $\log D$ in edge-conductance space:

$$\begin{aligned} (R,S,T) &= \left(\frac{\partial}{\partial a}\log D, \frac{\partial}{\partial b}\log D, \frac{\partial}{\partial c}\log D\right) \\ &= \nabla_{(a,b,c)} \log D. \end{aligned}$$

That was all ostensibly review. Now then, on the non-negative orthant $\bar{\Pi} = \{a,b,c \geq 0\}$ in edge-conductance space the function

$$\log D = \log \det K|_P - \log 3$$

is concave; as Gil Strang has pointed out to us, this follows from the fact that on the set of positive definite matrices the function $\log D$ is concave (see [15]).

Since $\log D$ is concave and the effective resistances are its partial derivatives, if conductances a_0, b_0, c_0 induce finite effective resistances R_0, S_0, T_0 then

$$(a_0, b_0, c_0) = \arg \max_{(a,b,c)\in\bar{\Pi}} \log D - (R_0 a + S_0 b + T_0 c).$$

Thus if a resistive inverse exists, it is given as the solution to a convex programming problem. Now for any $R_0, S_0, T_0 > 0$ this extremal problem still has a unique solution, i.e., the equation above uniquely determines a point $(a_0, b_0, c_0) \in \bar{\Pi}$. The Kuhn-Tucker conditions identify this point as the unique point where $R \leq R_0$ with $R = R_0$ if $a_0 > 0$, etc. Thus when (a_0, b_0, c_0) lies in the interior of $\bar{\Pi}$ we have a genuine resistive inverse; when it lies on the boundary we have a generalized inverse (or a true resistive inverse with some zero conductances). So we're all set.

This proof applies as well to the case where we demand that $\sigma_{ij} = 0$ for certain selected edges (ij), and place no upper bounds on the corresponding $\hat{c}_{ij}$ (i.e. set $c_{ij} = \infty$). □

If $C = (c_{ij})$ is resistive, the matrix inversion of Section 3 will find the associated conductance matrix σ, with $\hat{c}_{ij} = c_{ij}$. If C is not resistive — or even if it is — there is an iterative algorithm that converges to the generalized resistive inverse whose existence is guaranteed by Theorem 4.1. In presenting this algorithm we will once again limit the discussion to the case where the graph is a triangle, and use the same notation as above.

By Foster's theorem $aR + bS + cT = 2$, (the 2 here being one less than the number of nodes in the graph), and hence $a_0 R_0 + b_0 S_0 + c_0 T_0 = 2$. Thus

$$(a_0, b_0, c_0) = \arg \max_{(a,b,c)\in\bar{\Sigma}} D,$$

where $\bar{\Sigma}$ is the closure of the open simplex

$$\Sigma = \{a, b, c > 0;\ aR_0 + bS_0 + cT_0 = 2\}.$$

To locate the maximum we can use the *knee-jerk algorithm*, according to which we iterate the mapping

$$T(a, b, c) = \left(a\frac{R}{R_0}, b\frac{S}{S_0}, c\frac{T}{T_0}\right).$$

This algorithm is a particular instance of a general method known as the *Baum algorithm*. The mapping T takes Σ to itself, and strictly increases the objective function D for any (a, b, c) (other than (a_0, b_0, c_0)) in Σ. (See Baum and Eagon [1].) It follows from this that for any starting guess $(a, b, c) \in \Sigma$ the sequence $T^n(a, b, c)$ of iterates converges to the generalized resistive inverse (a_0, b_0, c_0).

Now let's return to the cat-and-mouse game.

Corollary 4.2 *Let G be any weighted graph with n nodes. The cat has an $(n-1)$-competitive strategy for the cat-and-mouse game on G.*

5 The k-Server Problem

We consider here the *k-server problem* of Manasse *et al.* [14] defined in Section 1. We compare the performance of an on-line k-server algorithm to the performance of an adversary with k servers. The adversary chooses the next request at each step, knowing the current state of the on-line algorithm, and moves one of its servers to satisfy the request (if necessary). The on-line algorithm then moves one of its servers if necessary, without knowing the state of the adversary. The algorithm is *c-competitive* if there exists a constant a such that, for any adversary and any request sequence, E[cost on-line algorithm] $\leq c \cdot$ [cost adversary] $+ a$. Such an adversary is termed *adaptive on-line* [2, 17]. One can weaken the adversary by requiring it to choose the sequence of requests in advance, so that it does not know of the actual random choices made by the on-line algorithm in servicing the request sequence; this is an *oblivious* adversary. Alternatively, one can strengthen the adversary by allowing it to postpone its decision on its server moves until the entire sequence of requests has been generated; this is an *adaptive off-line* adversary. These three types of adversaries for randomized algorithms are provably different [2, 10, 17]. However, they all coincide when the on-line algorithm is deterministic. Furthermore, if there is a randomized algorithm that is c-competitive against adaptive on-line adversaries, then there is a c^2-competitive deterministic algorithm [2].

Theorem 5.1 *Let C be a resistive cost matrix. Then we have a randomized k-competitive strategy for the k-server problem against an adaptive on-line*

adversary. More generally, if every $(k+1)$-node subgraph of C is resistive, we have a k-competitive strategy for the k-server problem on C.

Proof: We exhibit a k-competitive on-line algorithm for the more general case; we call this algorithm RWALK. If a request arrives at one of the k vertices that RWALK's servers cover (let us denote these vertices by $a_1, a_2, ..., a_k$), it does nothing. Suppose a request arrives at a vertex a_{k+1} it fails to cover. Consider the $(k+1)$-vertex subgraph C' determined by $a_1, a_2, ..., a_k, a_{k+1}$. By hypothesis, C' is resistive. Let σ' denote its resistive inverse. With probability

$$p'_i = \frac{\sigma'_{i,k+1}}{\sum_{j=1}^{k} \sigma'_{j,k+1}}$$

it selects the server at vertex a_i to move to the request at vertex a_{k+1}. Since C' is finite, σ' is connected, and the denominator $\sum_{j=1}^{k} \sigma'_{j,k+1}$ is nonzero, the probabilities are well defined and sum to 1.

We need to prove that the RWALK is k-competitive. To this end, we define a *potential* Φ. (This is not to be confused with an electrical potential.) Say the RWALK's servers are presently at vertices $\mathbf{a} = \{a_1, a_2, ..., a_k\}$, and the adversary's servers are presently at vertices $\mathbf{b} = \{b_1, b_2, ..., b_k\}$, where $\mathbf{a}$ and $\mathbf{b}$ may overlap. We define $\Phi(\mathbf{a}, \mathbf{b})$ as the sum of the costs of all the edges between vertices currently occupied by RWALK's servers, plus k times the cost of a minimum-weight matching between vertices occupied by RWALK's servers and the adversary's servers. That is,

$$\Phi(\mathbf{a}, \mathbf{b}) = \sum_{1 \le i < j \le k} c_{a_i, a_j} + \min_{\pi} k \cdot \sum_{i=1}^{k} c_{a_i, b_{\pi(i)}},$$

where π ranges over the permutations on $\{1, 2, ..., k\}$. We also define a quantity Δ depending on the present position and the past history:

$$\begin{aligned} \Delta(\mathbf{a}, \mathbf{b}, \text{ History}) &= \Phi(\mathbf{a}, \mathbf{b}) \\ &+(\text{RWALK's Cost}) \ - k \cdot (\text{Adversary's Cost}), \end{aligned}$$

where both "Cost"s are cumulative. We will show that the expected value of Δ is a non-increasing function of time, and then show how this will imply the theorem.

Let us consider the changes in Δ due to (i) a move by the adversary (which could increase Φ), and (ii) a move by RWALK, which (hopefully) tends to decrease Φ. By showing that in both cases, the expected change in Δ is ≤ 0, we will argue that over any sequence of requests the expected cost of RWALK is at most k times the adversary's cost plus an additive term independent of the number of requests.

If the adversary moves one of its servers from b_j to b'_j, its cumulative cost is increased by c_{b_j,b'_j}. The potential Φ can increase by at most k times that quantity, since the minimum-weight matching can increase in weight by at most c_{b_j,b'_j}. (Obtain a new matching π' from the old one by matching $a_{\pi^{-1}(j)}$ to b'_j instead of b_j, and note that the weight of this new matching is no more than c_{b_j,b'_j} plus the weight of the old one; the new minimum-weight matching will be no heavier than this constructed matching.) So in this case Δ does not increase.

Next, we consider a move made by RWALK, and compare its cost to the expected change in Φ. First, we suppose that $\mathbf{a}$ and $\mathbf{b}$ overlap in $k-1$ places (later we remove this assumption):

$$a_i = b_i, \quad i = 2, 3, ..., k; \quad a_1 \neq b_1.$$

Define $b_{k+1} = a_1$. For convenience, set $m = k+1$, and let c_{ij}, σ_{ij}, for $i, j = 1, 2, ..., m$ be defined by $c_{ij} = c_{b_i,b_j}$. Recall the equations relating σ and C, specialized to the entries of interest:

$$\begin{aligned}
\bar{\sigma}_{11} &= \sum_{j=2}^{k+1} \sigma_{1j} \\
\bar{\sigma}_{1j} &= -\sigma_{1j}, \ 2 \leq j \leq k \\
\bar{c}_{ji} &= [c_{jm} + c_{im} - c_{ji}]/2 \\
\sum_{j=1}^{k} \bar{\sigma}_{1j}\bar{c}_{ji} &= \delta_{1i}, i \leq k
\end{aligned}$$

Multiply this last equation by 2 and sum over $i = 2, 3, ..., k$, noticing that in this range $\delta_{1i} = 0$. We obtain:

$$0 = 2\sum_{i=2}^{k}\sum_{j=1}^{k} \bar{\sigma}_{1j}\bar{c}_{ji}$$

$$
\begin{aligned}
&= 2\sum_{i=2}^{k}\left(\bar{\sigma}_{11}\bar{c}_{1i}+\sum_{j=2}^{k}\bar{\sigma}_{1j}\bar{c}_{ji}\right)\\
&= \sum_{i=2}^{k}\left\{\sum_{j=2}^{k+1}\sigma_{1j}\left[c_{1m}+c_{im}-c_{i1}\right]\right.\\
&\qquad \left.-\sum_{j=2}^{k}\sigma_{1j}\left[c_{jm}+c_{im}-c_{ji}\right]\right\}
\end{aligned}
$$

For $j = m = k+1$ the latter bracketed expression $[c_{jm}+c_{im}-c_{ji}]$ is zero, so we can include it in the sum, extending the limits of summation to $k+1$:

$$
\begin{aligned}
0 &= \sum_{i=2}^{k}\left\{\sum_{j=2}^{k+1}\sigma_{1j}[c_{1m}+c_{im}-c_{i1}]\right.\\
&\qquad \left.-\sum_{j=2}^{k+1}\sigma_{1j}[c_{jm}+c_{im}-c_{ji}]\right\}\\
&= \sum_{j=2}^{k+1}\sigma_{1j}\left[(k-1)c_{1m}+\sum_{i=2}^{k}c_{im}-\sum_{i=2}^{k}c_{i1}\right.\\
&\qquad \left.-(k-1)c_{jm}-\sum_{i=2}^{k}c_{im}+\sum_{i=2}^{k}c_{ji}\right]\\
&= \sum_{j=2}^{k+1}\sigma_{1}j\left[kc_{1m}-\sum_{i=2}^{m}c_{i1}-kc_{jm}+\sum_{i=2}^{m}c_{ji}\right]\\
&= \sum_{j=2}^{k+1}\sigma_{1}j\left[kc_{1m}-\sum_{i=2}^{m}c_{i1}-kc_{jm}\right.\\
&\qquad \left.+\sum_{1\le i\le m,\ i\neq j}c_{ji}-c_{j1}\right]
\end{aligned}
$$

Defining

$$
\begin{aligned}
\tau_\ell &= kc_{\ell m}+\sum_{1\le i<j\le m,\ i,j\neq\ell}c_{ij}\\
&= kc_{\ell m}+\sum_{1\le i<j\le m}c_{ij}-\sum_{i=1}^{m}c_{i\ell}
\end{aligned}
$$

we discover

$$\sum_{j=2}^{k+1} \sigma_{1j}[\tau_1 - \tau_j - c_{j1}] = 0.$$

Now it is straightforward to verify that the expected change in Δ, as RWALK makes its random move with probability $(\sigma_{1j})/(\sum_{i=2}^{k+1} \sigma_{1i})$, is

$$\frac{1}{\sum_{i=2}^{m} \sigma_{1i}} \times \sum_{j=2}^{k+1} \sigma_{1j}[\tau_1 - \tau_j - c_{j1}] = 0.$$

Thus the expected change in Δ is zero on RWALK's move.

Finally we verify the case in which $\mathbf{a}$ and $\mathbf{b}$ overlap in fewer than $k-1$ vertices, and RWALK makes a move. Suppose the request is at vertex b_1. Suppose the current minimum-weight matching pairs a_i with b_i, $i = 1, 2, ..., k$. Perform the previous analysis as if the adversary's other servers $b_2, ..., b_k$ were presently at the same vertices as our $a_2, ..., a_k$. Obtain again

$$\frac{1}{\sum_{i=2}^{m} \sigma_{1i}} \times \sum_{j=2}^{m} \sigma_{1j}[\tau_1 - \tau_j - c_{j1}] = 0.$$

The true potential Φ differs from that of the previous case only in the weight of the minimum-weight matching. Consider a new matching, not necessarily of minimum weight, after our current move from a_j to b_1, obtained from the old matching by matching a_1 to b_j, a_j to b_1, and a_i to b_i for $i \neq 1, j$. This new matching differs from the old one by

$$c_{a_1,b_j} - c_{a_1,b_1} - c_{a_j,b_j} \leq c_{a_1,a_j} - c_{a_1,b_1}$$

by the triangle inequality. But the previous analysis guaranteed that the expected change in Δ was zero, and for that calculation we used a value of

$$c_{a_1,a_j} - c_{a_1,b_1}$$

as the change in Φ. The true change in Φ is less than that, and even less when we allow the new matching to be of minimum weight, so that again the expected change in Δ is non-positive.

So the expected value of $\Delta(\mathbf{a}, \mathbf{b}, \text{History}) = \Phi(\mathbf{a}, \mathbf{b}) + (\text{RWALK's Cost}) - k \cdot (\text{Adversary's Cost})$ is nonincreasing at every step. Since Φ is positive, we find that

$$(\text{RWALK's Cost}) - k \cdot (\text{Adversary's Cost})$$

remains bounded, in expectation, by the initial value of Δ. So the competitiveness is k. $\Box$

The last result is valid even if the graph is infinite; one only requires that the cost of a simple path be bounded and every $k+1$-node subgraph be resistive. The potential Φ we developed to prove the last result seems to be very natural and useful for the server problem. It has been subsequently used by several authors [7, 8].

As corollaries of Theorem 5.1, we have k-competitive algorithms for the server problem for $k = 2$ in any metric space [14], for points on a line [7], for the weighted cache problem [7, 17], for the uniform cost (caching) case [18] and for points on a tree [8]. These algorithms are extremely simple, and memoryless. Berman *et al.* [3] give an algorithm for 3 servers that achieves a finite competitiveness in any metric space. With the sole exception of this result, every special case of the server problem for which *any* finite competitiveness is known is in a resistive metric space. Certainly, all known cases where we know of k-competitive on-line algorithms are in (special cases of) resistive metric spaces. Thus our theory based on resistive random walks both unifies and generalizes our current picture of the k-server conjecture, and implies k^2-competitive deterministic algorithms in resistive spaces [2].

Theorem 5.1 can be used to derive competitive k-server algorithms for non-resistive spaces as well, when these can be approximated by resistive spaces. A cost matrix C' is a λ-*approximation* for the matrix C if, for all ij, $c'_{ij} \leq c_{ij} \leq \lambda c'_{ij}$. If a server algorithm is c-competitive for the matrix C', then it is λc-competitive for the matrix C. Using this observation, we can derive a $2k$-competitive algorithm for k servers when the nodes are on a circle, with distances being measured along the circumference. Consider points on a circle, with the cost c_{ij} between two points i, j given as the distance along the smaller arc joining them. We can construct a 2-approximation C' to this cost C. Each arc of the circle becomes a resistor with resistance equal to the arc-length. If the smaller and larger arc distances joining two points are α, β respectively, then the effective resistance c' is $\alpha\beta/(\alpha+\beta)$ while $c = \alpha < \beta$. Then easily $c' \leq c \leq 2c'$. In conjunction with results in [2], this implies that there is a $4k^2$-competitive deterministic algorithm for k servers on the circle. No finitely competitive deterministic algorithm was known before for this problem.

On the other hand, it is not possible to finitely approximate arbitrary distance matrices derived from the Euclidean plane (proof omitted in this

version). Thus, this approximation technique does not solve the server problem in the plane.

We now turn to the case $k = n - 1$.

Theorem 5.2 *Let C be any cost matrix. If there are n nodes and $k = n - 1$ servers, we have an $(n-1)$-competitive strategy.*

The significance of Theorem 5.2 is that it holds even when the c_{ij} do not satisfy the triangle inequality, a case for which no prior result exists [14].

Proof outline: We can assume that servers always occupy distinct nodes. Both the on-line algorithm and the adversary have one unoccupied node which we consider, respectively, to be "cat" and "mouse". Whenever a server moves from i to j the cat (resp. the mouse) moves from j to i, at cost $c_{ij} = c_{ji}$. We can assume that the adversary always requests the unique node (cat's position) which is not occupied by the on-line algorithm. It has to move one of its own servers to satisfy this request only when the positions of the cat and of the mouse coincide. This situation corresponds exactly to the cat-and-mouse game, and the result follows from Corollary 4.2. □

6 Metrical Task Systems

We now consider Metrical Task Systems, as defined by Borodin *et al.* [4]. Definitions are omitted here for brevity; the reader is referred to [4].

We compare the performance of an on-line algorithm to the performance of an adversary. At each step, the adversary chooses the next task, knowing the current state of the on-line algorithm, and chooses its next position. An on-line algorithm is c-competitive if there is a constant a such that for any n and any adversary E[cost of on-line algorithm] $\leq c \cdot$ [cost of adversary] $+ a$ (where cost includes the task processing cost and the cost of moves).

Borodin *et al.* [4] define an on-line algorithm for metrical task systems to be a *traversal algorithm* if:
(1) the states are visited in a fixed sequence $s_1, s_2, \cdots$ independent of the input task sequence; and,
(2) there is a sequence of positive *threshold costs* $c_1, c_2, \cdots$ such that the transition from s_j to s_{j+1} occurs when the total task processing cost incurred since entering s_j reaches c_j. In fact, they set $c_j = c_{s_j, s_{j+1}}$.

We extend this definition to *randomized traversal algorithms*. Condition (1) is replaced by (1′): the states are visited by a Markov process that is independent of the input task sequence.

Borodin *et al.* [4] give a $8(n-1)$-competitive deterministic traversal algorithm, and a more complex $(2n-1)$-competitive deterministic algorithm, which is optimal. We give here a $(2n-1)$-competitive randomized traversal algorithm. The algorithm is very simple, and memoryless. It is based on the random walks developed in Sections 3 and 4.

Let (c_{ij}) be the cost matrix for a metrical task system on a graph with n nodes. Let (σ_{ij}) be the generalized resistive inverse of (c_{ij}), and let p_{ij} be the transition probabilities for the resistive random walk. The on-line algorithm makes a transition out of current state i when the expected total task processing cost since entering state i exceeds a threshold β_i (to this end, Borodin *et al.* describe a continuous-time view of the process in which a state-transition can be made at any point in time rather than at discrete steps; details on how this is done omitted in this version); it then randomly chooses the next state, where state j is chosen with probability p_{ij}.

Theorem 6.1 *The on-line algorithm is* $(2n-1)$*-competitive against an adaptive on-line adversary, for the choice of thresholds* $\beta_i = 2\sum_j p_{ij}c_{ij}/(\sum_j \sigma_{ij}c_{ij})$.

Proof outline: One can show that this algorithm corresponds to a cat-and-mouse game, with the following two modifications: (1) the cat pays β_i whenever it reaches node i; (2) if the mouse is caught at node i by the cat, then the mouse can either move to a new node j and pay c_{ij}, or it can stay put at node i until the cat catches it again, and pay β_i. Using some additional properties we prove about resistive walks, we show that the expected total task-processing cost of the cat in the extended game is $n/(n-1)$ times the expected total cost of edges traversed by cat. Each non-trivial loop in the random walk of the cat has a stretch $\leq n-1$. We also show that $e_{ii} \leq (n-1)\cdot\beta_i$. It follows that the expected move cost of the cat is at most $n-1$ times the mouse cost, and the expected total cat cost is $\leq (n-1)\cdot(1+n/(n-1)) = 2n-1$ times the mouse cost. □

7 Open Problems

In this section we list several open problems raised by our work.

We do not know what stretch can be achieved by random walks when the cost matrix C is not symmetric.

It would be interesting to study the cat-and-mouse game under a wider class of strategies. For instance, on the circumference of a circle, it is easy to give a deterministic algorithm for the cat that achieves a constant competitiveness. Moreover, one can consider randomized algorithms other than those based on random walks. In fact, a simple (though not memoryless) randomized algorithm achieves a competitiveness of $n/2$ when the graph is the complete graph on n nodes with the same cost on every edge.

We have no results for the k server problem in general metric spaces. We would like to prove that the resistive random walk yields a server algorithm that achieves a competitiveness that is a function of k alone, in any metric space (against an adaptive on-line adversary). This would yield [2] a deterministic algorithm having finite competitiveness in an arbitrary metric space. We can prove that the resistive server algorithm is $(2k-1)$-competitive against a *lazy* adaptive on-line adversary that moves only when it must: whenever there is a node occupied by an adversary server that is not occupied by an on-line algorithm's server, the adversary requests such node. The *lazy adversary conjecture* is that the resistive on-line algorithm achieves its worst performance against a lazy adversary. A proof of this conjecture would show that the resistive algorithm is $(2k-1)$-competitive in every metric space.

References

[1] L.E. Baum and J.A. Eagon. An inequality with applications to statistical estimation for probabilistic functions of Markov processes and to a model for ecology. *Bull. Amer. Math. Soc.*, 73:363–363, 1967.

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

[3] P. Berman, H.J. Karloff, and G. Tardos. A competitive 3-server algorithm. In *Proceedings 1st ACM-SIAM Symposium on Discrete Algorithms*, pages 280–290, 1990.

[4] A. Borodin, N. Linial, and M. Saks. An optimal online algorithm for metrical task systems. *Journal of the ACM*, 39:745–763, 1992.

[5] R. Bott and R. J. Duffin. On the algebra of networks. *Trans. Amer. Math. Soc.*, 74:99–109, 1953.

[6] A. K. Chandra, P. Raghavan, W.L. Ruzzo, R. Smolensky, and P. Tiwari. The electrical resistance of a graph captures its commute and cover times. In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing*, pages 574–586, Seattle, May 1989.

[7] M. Chrobak, H.J. Karloff, T. Payne, and S. Vishwanathan. New results on server problems. In *Proceedings of the 1st ACM-SIAM Symposium on Discrete Algorithms*, pages 291–300, 1990.

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

[9] P.G. Doyle and J.L. Snell. *Random Walks and Electric Networks*. The Mathematical Association of America, 1984.

[10] A. Fiat, R. M. Karp, M. Luby, L. A. McGeoch, D. D. Sleator, and N. Young. Competitive paging algorithms. *Journal of Algorithms*, 12:685–699, 1991.

[11] R. M. Foster. The average impedance of an electrical network. In *Contributions to Applied Mechanics (Reissner Anniversary Volume)*, pages 333–340. Edwards Bros., Ann Arbor, Mich., 1949.

[12] R. M. Foster. An extension of a network theorem. *IRE Trans. Circuit Theory*, 8:75–76, 1961.

[13] J.G. Kemeny, J. L. Snell, and A.W. Knapp. *Denumerable Markov Chains*. The University Series in Higher Mathematics. Van Nostrand, Princeton, NJ, 1966.

[14] M.S. Manasse, L.A. McGeoch, and D.D. Sleator. Competitive algorithms for server problems. *Journal of Algorithms*, 11:208–230, 1990.

[15] A. W. Marshall and I. Olkin. *Inequalities: Theory of Majorization and Its Applications*. Academic Press, New York, 1979.

[16] C.H. Papadimitriou and M. Yanakakis. Shortest paths without a map. *Theoretical Computer Science*, 84:127–150, 1991.

[17] P. Raghavan and M. Snir. Memory versus randomization in on-line algorithms. In *16th International Colloquium on Automata, Languages, and Programming*, volume 372 of *Lecture Notes in Computer Science*, pages 687–703. Springer-Verlag, July 1989. Revised version available as IBM Research Report RC15840, June 1990.

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

[19] L. Weinberg. *Network Analysis and Synthesis*. McGraw-Hill, New York, 1962.