SIAM J. COMPUT.
Vol. 28, No. 4, pp. 1232–1246

# DESIGN OF ON-LINE ALGORITHMS USING HITTING TIMES[*]

PRASAD TETALI[†]

**Abstract.** Random walks are well known for playing a crucial role in the design of randomized off-line as well as on-line algorithms. In this work we prove some basic identities for ergodic Markov chains (e.g., an interesting characterization of *reversibility* in Markov chains is obtained in terms of first passage times). Besides providing new insight into random walks on weighted graphs, we show how these identities give us a way of designing competitive randomized on-line algorithms for certain well-known problems.

**Key words.** random walk, graph, Markov chain, reversibility, competitive ratio, first passage time, M-matrices

**AMS subject classifications.** 68Q25, 60C05, 60J10, 60J15, 60J20, 15A51

**PII.** S0097539798335511

## 1. Introduction.

In a recent paper, Coppersmith et al. [8] made clever use of results from synthesis of electrical networks to design reversible random walks useful for certain randomized on-line algorithms.

At the heart of their methods lies the following problem on random walks. We are given a weighted (undirected) graph with $n$ vertices, and with weight $C_{ij}$ on edge $\{i, j\}$, for $1 \le i, j \le n$. Assume that the weights are symmetric and that they satisfy the triangle inequality. Every traversal of an edge $\{i, j\}$ costs $C_{ij}$.

Recall that a random walk on an undirected graph is the Markov chain whose state space is the vertex set of the graph, whose behavior is given by the rule that when the chain is at any given vertex the next transition is along an edge incident to that vertex that is chosen at random, according to some probability distribution. (The probability distribution could depend on the vertex, and the case of uniform distribution over the incident edges is often called a *simple* random walk.) We say that a random walk on $G$ has *stretch* $s$ (with respect to the cost matrix $C$) if, for any sequence of vertices $v_0, \ldots, v_k$, the expected cost of the random walk to traverse these nodes in the prescribed order is at most $s$ times the optimal cost, up to an additive constant. Let $e_{uv}$ denote the expected cost of the walk to go from vertex $u$ to vertex $v$. If for every $v_0, \ldots, v_k$, $\sum_{i=1}^{k} e_{v_{i-1}v_i} \le s \sum_{i=1}^{k} C_{v_{i-1}v_i} + a$, where $a \ge 0$, then the walk is said to have stretch $s$. Given the cost matrix $C$, the problem is to design a random walk with as low a stretch as possible.

Coppersmith et al. proved the following tight result for all symmetric cost matrices: Any random walk on a weighted (undirected) graph with $n$ vertices has stretch $\ge (n-1)$, and every weighted (undirected) graph has a random walk with stretch $\le (n-1)$.

Coppersmith et al. justified the relevance of this problem by providing bounds for the *cat and mouse* game, which they showed was central to the analysis of on-line

[*]Received by the editors March 10, 1998; accepted for publication March 20, 1998; published electronically March 22, 1999. A preliminary version of this paper appeared in *Proc. 5th Annual ACM-SIAM Symposium on Discrete Algorithms*, SIAM, Philadelphia, PA, 1994, pp. 402–411. The work of this author was done while at AT&T Bell Labs, Murray Hill, NJ 07974.
http://www.siam.org/journals/sicomp/28-4/33551.html

[†]School of Mathematics, Georgia Institute of Technology, Atlanta, GA 30332-0160 (tetali@math.gatech.edu).

algorithms for well-known problems such as the metrical task system problem and the $k$-server problem. (We shall define these shortly for the nonspecialist.)

Symmetry of the costs is crucial to the basic technique used in [8] in designing the appropriate random walk, and thus an interesting question is left open on stretch under asymmetric costs—equivalently, the stretch of random walks on directed graphs. In this work we prove close-to-optimal lower and upper bounds on a stretch for a class of matrices, without the symmetry assumption. Moreover, under symmetry our results imply those of [8].

In a nutshell, Coppersmith et al. interpret a given cost matrix as an effective resistance matrix of a resistive network, and then they synthesize an actual network that has the desired properties. Classical analogues between resistive networks and reversible Markov chains yield a corresponding random walk that achieves the optimal stretch. We use a different way of synthesizing a random walk, the advantage being that we do not need reversibility of the walk (i.e., symmetry of the costs) for our techniques to work. We interpret the cost matrix as the hitting time (first passage time) matrix of an ergodic (not necessarily reversible) Markov chain and then describe how to find the unique chain that yields the desired hitting times, i.e., design the transition probabilities which yield the desired hitting times. While such a synthesis of an ergodic chain from a valid hitting time matrix is unique and efficient, it is not true that an arbitrary cost matrix is always a hitting time matrix.[1] (The consolation, however, is that we can check for the latter condition with essentially one matrix inversion.) We call a cost matrix *ergodic* if it can be interpreted as a certain hitting time matrix, and we call the corresponding random walk an *ergodic* walk. We show that this walk achieves optimal stretch when the costs are symmetric and is close to optimal when the costs are asymmetric. We show that our techniques also work when the cost matrix is essentially an effective resistance matrix, thus extending several results proved in [8].

Given a weighted (directed) graph with a weight (or *cost*) matrix $C = \{C_{ij}\}$, define the *cycle offset ratio* $\Psi(C)$ as follows. $\Psi(C)$ is the maximum over all sequences

$$v_0, \ldots, v_k = v_0 \text{ of } \frac{\sum_{i=1}^{k} C_{v_{i-1},v_i}}{\sum_{i=1}^{k} C_{v_i,v_{i-1}}}.$$

(Note that $1 \leq \Psi(C) \leq (n-1)$, since the costs satisfy the triangle inequality.)

We prove the following result.

Any random walk on a weighted graph with $n$ vertices has stretch $\geq (n-1)/\Psi(C)$, and the ergodic walk has stretch $\leq (n-1)$, with equality under a symmetric $C$.

While we show examples that achieve equality in the lower bound, any tightening of the upper bound seems quite hard. However, it is interesting that the stretch of random walks on directed graphs can be brought down below $n-1$, while the counterpart on undirected graphs has an optimal bound of $n-1$.

The first application, for the *cat and mouse* game (see section 3), follows immediately from the above. It was mentioned in [8] that the cat and mouse game is at the core of several on-line algorithms. We show that for any $n \times n$ cost matrix $C$ and any "blind" cat strategy (i.e., a random walk strategy), there is a mouse strategy that forces the competitiveness of the cat to be at least $(n-1)/\Psi(C)$, and the ergodic walk by the cat achieves a competitive ratio $\leq (n-1)$, on ergodic $C$.

[1]Similarly, an arbitrary cost matrix is not necessarily an effective resistance matrix for the techniques of [8] to work.

The second and more interesting application is for the notoriously hard $k$-server problem (see [11], [16], [20], [17]). The $k$-server problem (defined in [20]) is as follows. There are $k$ mobile servers located on $k$ vertices of a graph $G$ with positive, real costs on the edges. (The costs can be thought of as distances between the positions.) An on-line algorithm manages the servers in such a way as to satisfy an on-line sequence of requests for service at vertices $v_i$, $i = 1, 2, \ldots$; i.e., servicing a request corresponds to moving a server to the requested vertex whenever a server isn't there. The algorithm pays a cost equal to the cost on the edge traversed by the server. The competitiveness of the algorithm is measured with respect to the cost an adversary pays, wherein the adversary moves the servers but also gets to choose the request sequence.

Due to the hardness of the $k$-server problem, it is significant to prove competitive ratios even for special cases such as special classes of graphs (e.g., [4], [8], [6]). Coppersmith et al. [8] provide one such example class. They use random walks to design optimal randomized $k$-competitive server algorithms when the cost matrix has a resistive inverse. We extend this class by allowing asymmetric costs, or equivalently, weighted *directed* graphs, with ergodic cost matrices.

Define *edge offset ratio* $\Psi'(C)$ to be $\max_{ij} \frac{C_{ij}}{C_{ji}}$. (Note that $\Psi(C) \leq \Psi'(C)$, and that $\Psi(C) = \Psi'(C) = 1$, for symmetric $C$.) We prove the following result for the *asymmetric* $k$-server problem. (This implies the result of [8].)

Let $C$ be a cost matrix on $n$ nodes. If every submatrix on $k+1$-nodes is ergodic, then we have a randomized $k\Psi'(C)$-competitive strategy for the $k$-server problem on $C$.

The final application is to the task system problem, defined as follows. We have a task system $(S, C)$ for processing sequences of tasks wherein $S$ is a set of states, and $C$ is a cost matrix, describing the cost of changing from state $i$ to state $j$. We assume that the costs satisfy the triangle inequality and that there is no cost of staying in the same state ($C_{ii} = 0$). Furthermore, when the costs are symmetric we refer to the task system as a *metrical* task system (MTS). Each task $T$ has a cost vector $v_T$, where $v_T(i)$ is the cost of processing $T$ in state $i$. A schedule for a given sequence of tasks $T_1, \ldots, T_k$ is a sequence of states $s_1, \ldots, s_k$, where $s_i$ is the state in which $T_i$ is processed. The *task system problem* is to design an *on-line* schedule (choose $s_i$ only knowing $T_1, \ldots, T_i$) so that the algorithm is $w$-competitive—on any input sequence of tasks, the cost of the on-line algorithm is, barring an additive constant, at most $w$ times that of the optimal off-line algorithm.

Borodin et al. [5] designed a deterministic algorithm with a competitive ratio of at most $(2n-1)\Psi(C)$ for the task system problem with asymmetric costs. If the costs are symmetric (i.e., an MTS), they prove a matching lower bound of $(2n-1)$. It is straightforward to extend their lower bound proof for the asymmetric case to get a lower bound of $(2n-1)/\Psi(C)$. Coppersmith et al. provided a simpler, *memoryless*, $(2n-1)$-competitive, randomized algorithm for any MTS, and also showed that no randomized algorithm can do better against an adaptive on-line adversary. We extend the results of [8] to the task system with ergodic cost matrices. In particular, we prove a lower bound of $(2n-1)/\Psi(C)$ for any randomized on-line scheduler. We also provide a randomized on-line scheduler with a competitive ratio of at most $(2n-1)$. While $(2n-1)$ is the best possible for symmetric cost matrices, our randomized scheduler achieves a competitive ratio of strictly less than $(2n-1)$, whenever the (ergodic) cost matrix is asymmetric. This shows that it is possible to design random walks with lower stretch on directed graphs than on undirected graphs. This also suggests that the deterministic algorithm of [5] (or a variant thereof) may have a competitive ratio of at most $(2n-1)$ even under asymmetric costs.

Thus the novel technique of using the synthesis of random walks from hitting times for the design of on-line algorithms, while yielding the results of [8] for undirected graphs in a natural and simpler way, also yields results for directed graphs. We conclude this work with an interesting question on "approximating" an ergodic Markov chain, which will have useful implications in terms of extending our results to all cost matrices.

## 2. Results on ergodic Markov chains.

In this section we prove two identities involving the first passage times and the transition probabilities. These results are crucial (later) to the analysis of our on-line algorithms. Lemma 2.1 and Theorem 2.2 below appeared in an earlier paper of this author [23]; however, the proofs are included here to keep the presentation self-contained.

Consider an electrical network on $n$ nodes with resistors $r_{ij}$ between nodes $i$ and $j$. Let $R_{ij}$ denote the *effective resistance* between the nodes. Then Foster's theorem asserts that

$$\sum_{i\sim j}\frac{R_{ij}}{r_{ij}} = n-1,$$

where $i\sim j$ denotes that $i$ and $j$ are connected by a finite $r_{ij}$. The proof appears in [13]. Also, [22] shows an alternative way (using random walks) of proving the same. In this section we prove an elementary identity for ergodic Markov chains which yields Foster's theorem when the chain is time-reversible.

Let $P$ denote the transition probability matrix (size $n\times n$) of an ergodic Markov chain with stationary distribution $\pi$. Let $P_{ii}=0\ \forall i$. Furthermore, let $H$ denote the expected first-passage matrix (also size $n\times n$) of the above chain; i.e., $H_{ij}$ denotes the expected time to reach state $j$ starting from state $i$. We call these the *hitting times.* Then we have the following lemma.

LEMMA 2.1. $\sum_{i,j}\pi_j P_{ji} H_{ij} = n-1$.

*Proof.*

$$\begin{aligned}\sum_{i,j}\pi_j P_{ji} H_{ij} &= \sum_j \pi_j\left(\sum_i P_{ji}H_{ij}\right) = \sum_j \pi_j[H_{jj}-1]\\ &= \sum_j \pi_j[1/\pi_j - 1] = n-1,\end{aligned}$$

since $H_{jj}=1/\pi_j$ . ☐

We prove a stronger statement than Lemma 2.1 in the form of Corollary 2.7 below. Both Lemma 2.1 and Corollary 2.7 were formulated while attempting to interpret Foster's theorem in terms of ergodic Markov chains. For a proof that Lemma 2.1 implies Foster's theorem, see [23].

### 2.1. Synthesis of an ergodic walk.

In the following we describe the construction (whenever one such exists) of an ergodic walk given an all-pairs hitting times matrix $H$. Given $P$ as above, we define $\bar{P}$ to be the following $(n-1)\times(n-1)$ matrix. Let

$$\bar{P}_{ii} = \pi_i\left(=\sum_{\substack{j=1\\ j\neq i}}^{n}\pi_i P_{ij}\right),$$

and $\bar{P}_{ij} = -\pi_i P_{ij}$, for $1 \leq i, j \leq n-1$.

Furthermore, let $\bar{H}_{jj} = H_{jn} + H_{nj}$, and $\bar{H}_{jk} = H_{jn} + H_{nk} - H_{jk}$, for $1 \leq j, k \leq n-1$. We use the standard notation of $I_n$ for an identity matrix of size $n \times n$, and $\delta_{ij}$ to denote the entries of $I$. The following theorem is a generalization of the *resistive inverse identity* (well known in electrical network theory) used in [8].

THEOREM 2.2.

$$\bar{P}\bar{H} = I_{n-1}.$$

*Proof.* The basic identity we use is the triangle inequality for the hitting times. Using a "renewal type" theorem (see section 2.3 of [2] or Proposition 9-58 of [19]) one can show the following:

$$H_{xz} + H_{zy} - H_{xy} = \frac{N_y^{xz}}{\pi_y}, \tag{2.1}$$

$$H_{xz} + H_{zx} = \frac{N_x^{xz}}{\pi_x}. \tag{2.2}$$

(Recall that $N_y^{xz}$ denotes the expected number of visits to $y$ in a random walk from $x$ to $z$.) From (2.1) and (2.2) we have

$$\bar{H}_{jk} = \frac{N_k^{jn}}{\pi_k} \quad \forall j, k.$$

Consider

$$\begin{aligned}
\sum_{j=1}^{n-1} \bar{P}_{ij}\bar{H}_{jk} &= \bar{P}_{ii}\bar{H}_{ik} + \sum_{\substack{j=1 \\ j\neq i}}^{n-1} \bar{P}_{ij}\bar{H}_{jk} \\
&= \pi_i \frac{N_k^{in}}{\pi_k} - \sum_{\substack{j=1 \\ j\neq i}}^{n-1} \pi_i P_{ij} \frac{N_k^{jn}}{\pi_k} \\
&= \frac{\pi_i}{\pi_k}\left[N_k^{in} - \sum_{\substack{j=1 \\ j\neq i}}^{n-1} P_{ij} N_k^{jn}\right] \\
&= \frac{\pi_i}{\pi_k}[\delta_{ik}] \qquad \text{(taking conditional means, given the first outcome)} \\
&= \delta_{ik}. \qquad \square
\end{aligned}$$

REMARK 1. *We have defined $\bar{P}$ and $\bar{H}$ by treating $n$ as a special state of the chain. Clearly, we could have chosen any other state $j$ and carried out a similar analysis.*

REMARK 2. *For reversible chains, we have $\bar{H}_{jk} = \bar{H}_{kj}$. This is because*

$$H_{jn} + H_{nk} + H_{kj} = H_{jk} + H_{kn} + H_{nj} \qquad \forall i, j \quad (^{**}).$$

The proof of this can be found in [9], or it can be verified directly by using the formula for the hitting times in terms of either resistances (see [22]) or the *fundamental* *matrix* (see [18]). Thus the proof of Theorem 2.2 becomes simpler for the reversible case. (In particular, we do not need to use (2.1) and (2.2).)

An interesting consequence of Theorem 2.2 is that the property in (**) is sufficient (not only necessary) to imply reversibility. For (**) implies that $\bar{H}$ is symmetric which in turn implies that $\bar{P}$ is symmetric, i.e., $\pi_i P_{ij} = \pi_j P_{ji}\ \forall i, j$. This alternative characterization of reversibility is interesting for yet another reason: Interpreted in the electrical world, it can be shown (see [23]) to be equivalent to what is known as the *reciprocity theorem* (see [22]).

COROLLARY 2.3. *Given the hitting times, the chain can be tested for reversibility in $O(n^2)$ time.*

*Proof.* First we designate an arbitrary state as state $n$ and then verify (**) for all pairs of vertices in $O(n^2)$ time. (Here we abuse notation by using $n$ for both the number of states and the name of a particular state.) $\square$

COROLLARY 2.4. *Given $P$ and $\pi$, the hitting times $(H_{ij})$ can be computed with a single matrix inversion, and conversely, given the hitting times, $P$ and $\pi$ can be computed with a single matrix inversion.*

*Proof.* In view of Theorem 2.2, we need to show only (a) how to compute $H$ from $\bar{H}$, and (b) how to compute $P$ from $\bar{P}$.

(a) For $1 \le i, j \le n-1$, we have

$$H_{in} = \sum_k N_k^{in} = \sum_k \pi_k \bar{H}_{ik},$$

$$H_{ni} = \bar{H}_{ii} - H_{in},$$

$$H_{ij} = H_{in} + H_{nj} - \bar{H}_{ij}.$$

Thus we can first compute $H_{in}$ and $H_{ni}$ $\forall i < n$, and then compute $H_{ij}$ for $1 \le i, j \le n-1$.

(b) We need to compute $\pi_n$ and $P_{ni}$, since the rest of the information is available in $\bar{P}$. Since $\pi$ is stochastic, and $\pi P = \pi$, we have

$$\pi_n = 1 - \sum_{i<n} \pi_i = 1 - \sum_{i<n} \bar{P}_{ii},$$

$$\pi_n P_{ni} = \pi_i - \sum_{j \ne i,n} \pi_j P_{ji} = \sum_{j \ne n} \bar{P}_{ji}. \qquad \square$$

We observe that Theorem 4.4.12 of [18] gives an alternative way of computing the chain, given all-pairs hitting times. However, the method outlined above seems simpler, since the solution can be written in essentially one equation; see Theorem 2.2.

**2.2. A trace inequality.** Based on some empirical results and Lemma 2.1 above, we conjectured that $\sum_{i,j} \pi_i P_{ij} H_{ij} \le n-1$, with equality under reversibility of the chain. This plays a crucial role in all our applications below, besides having an intrinsic importance. Recently, Aldous [1] proved this conjecture using a result due to Fiedler et al. [12]. We provide a slightly different proof using Theorem 2.2 and the main theorem in [12].

DEFINITION 2.5. *An $M$-matrix is an $n \times n$ matrix $A$ of the form $A = \alpha I - P$ in which $P$ is nonnegative and $\alpha$ is at least as big as the largest eigenvalue of $P$.*

*An alternative characterization of nonsingular $M$-matrices (see [21]) is that a nonsingular matrix $A$ with nonpositive off-diagonal entries is an $M$-matrix iff $A^{-1} \geq 0$, meaning that all the nonzero entries are positive. From this, it is clear that the matrix $\bar{P}$ defined above is an $M$-matrix. The following theorem of Fiedler et al. is an interesting trace-inequality.*

THEOREM 2.6 (Fiedler et al. [12]). *For a nonsingular $M$-matrix $A$ (size $n \times n$), $\mathrm{tr}(A^{-1}A^T) \leq n$, with equality holding iff $A$ is symmetric.*

Now we are ready to state and prove the generalization of Lemma 2.1.

COROLLARY 2.7. *$\sum_{i,j=1}^{n} \pi_i P_{ij} H_{ij} \leq n-1$, with equality holding iff $P(,)$ is a reversible chain.*

*Proof.* Using Theorem 2.6 with $\bar{P}$ in place of $A$, we have $\mathrm{tr}(\bar{H}\bar{P}^T) \leq n-1$. We are done by noticing that

$$\begin{aligned}
&\mathrm{tr}(\bar{H}\bar{P}^T)\\
&= \sum_{i=1}^{n-1}\sum_{j=1}^{n-1} \bar{H}_{ij}\bar{P}_{ij}\\
&= \sum_i \bar{H}_{ii}\pi_i - \sum_{i \neq j} \bar{H}_{ij}\pi_i P_{ij}\\
&= \sum_i [H_{in} + H_{ni}]\pi_i - \sum_{i \neq j}[H_{in} + H_{nj} - H_{ij}]\pi_i P_{ij}\\
&= \sum_i [H_{in} + H_{ni}]\pi_i - \sum_i H_{in}\pi_i(1 - P_{in})\\
&\quad - \sum_j H_{nj}(\pi_j - \pi_n P_{nj}) + \sum_{i,j=1}^{n-1} \pi_i P_{ij} H_{ij}\\
&= \sum_{i,j=1}^{n} \pi_i P_{ij} H_{ij}. \qquad \square
\end{aligned}$$

## 3. Lower and upper bounds on stretch.

We recall the definition of stretch of a random walk from the introduction: a random walk is said to have stretch $s$ if there exists an $a > 0$ such that, for every $v_0, \ldots, v_k$, $\sum_{i=1}^{k} e_{v_{i-1}v_i} \leq s \sum_{i=1}^{k} C_{v_{i-1}v_i} + a$. The following facts follow easily from the definition of stretch.

FACT 1. *If a random walk has stretch $s$ on cost matrix $C = \{C_{ij}\}$, then the walk has the same stretch on $C' = \{\beta C_{ij}\}$, where $\beta$ is any positive constant.*

FACT 2. *In computing the stretch of a random walk, it suffices to consider sequences of vertices, $v_0, v_1, \ldots, v_k = v_0$, that form simple cycles in the graph $G$.*

Note that if a random walk has a stretch of $c$ on simple cycles, then since any cycle can be decomposed as the union of disjoint simple cycles, the random walk will have stretch $c$ on arbitrary closed paths. Now, as shown on page 426 of [8], Fact 2 follows from the fact that we gave ourselves room by allowing for an additive constant in the definition of stretch.

Let $C = \{C_{ij}\}$ be the given cost matrix of size $n \times n$.

DEFINITION 3.1. *Let $\Psi(C)$ be defined as the maximum over all cycles $(v_0, \ldots, v_k = v_0)$ of the ratio*

$$\frac{\sum_{i=0}^{k-1} C_{v_i,v_{i+1}}}{\sum_{i=0}^{k-1} C_{v_{i+1},v_i}}.$$

*If we assume that the costs satisfy the triangle inequality, then it is easy to see that* $\Psi(C) \leq n-1$. *(Note that* $\Psi$ *is defined in* [5] *and is termed the cycle offset ratio.)*

THEOREM 3.2. *Any random walk over a directed weighted graph has stretch at least* $(n-1)/\Psi(C)$, *where* $C = \{C_{ij}\}$ *is an* $n \times n$ *matrix specifying weight* $C_{ij}$ *on edge* $(i, j)$.

*Proof.* Note that the lower bound is $n-1$, when $C$ is symmetric, since $\Psi(C) = 1$. In fact, the proof is identical to Theorem 1 of [8], wherein the symmetric case was dealt with. □

We now introduce the notion of *ergodic* cost matrices, which subsumes the class of resistive cost matrices, introduced in [8]. First, we describe two types of cost matrices for which there exist random walks with stretch at most $n-1$.

**Type I**. Let $C_{ij} = H_{ij}\ \forall i, j,\ i \neq j$, where $H_{ij}$ denote the hitting times of an $n$-state ergodic Markov chain.

CLAIM 1. *Every cost matrix of Type* I *has a random walk with stretch at most* $n-1$.

*Proof.* Note that, in view of Fact 2, it suffices to bound stretch over all simple cycles; this can then be extended to all paths, with an additive constant such as $\max_{i,j} C_{ij}$. □

Consider the walk with $H_{ij}$ as the hitting times. The expected cost per move is

$$E = \sum_{i,j} \pi_i P_{ij} C_{ij} = \sum_{i,j} \pi_i P_{ij} H_{ij} \leq (n-1)$$

by Corollary 2.7. The claim now follows by noticing that the expected cost of a traversal over any sequence $v_0, \ldots, v_k$ of vertices equals $E \times \sum_{i=0}^{k-1} H_{v_i v_{i+1}} = E \times \sum_{i=0}^{k-1} C_{v_i v_{i+1}}$.

**Type II**. Let $C_{ij} = \frac{1}{2}[H_{ij} + H_{ji}]\ \forall i, j,\ i \neq j$, where $H_{ij}$ now denote the hitting times of any *reversible* Markov chain.

CLAIM 2. *Every cost matrix of Type* II *has a random walk with stretch at most* $n-1$.

*Proof.* As in the proof of Claim 1, without loss of generality, it suffices to bound stretch over all simple cycles. □

The expected cost per move is

$$E = \sum_{i,j} \pi_i P_{ij} C_{ij} = \frac{1}{2} \sum_{i,j} \pi_i P_{ij} [H_{ij} + H_{ji}] = n-1$$

by Lemma 2.1 and reversibility of $P$. Moreover, the expected cost of a traversal over any (cyclic) sequence $v_0, \ldots, v_k = v_0$ of vertices is, as before, equal to

$$\begin{aligned}
&E \times \sum_{i=0}^{k-1} H_{v_i v_{i+1}} \\
&= E \times \frac{1}{2} \left[ \sum_{i=0}^{k-1} H_{v_i v_{i+1}} + \sum_{i=0}^{k-1} H_{v_{i+1} v_i} \right] \quad \text{by (**)} \\
&= (n-1) \sum_{i=0}^{k-1} \frac{1}{2} [H_{v_i v_{i+1}} + H_{v_{i+1} v_i}] \\
&= (n-1) \times \sum_{i=0}^{k-1} C_{v_i v_{i+1}}.
\end{aligned}$$

Hence we have the claim.

Remark 3. *Any quantitative sharpening of the trace inequality (Theorem* 2.6*) immediately gives an improved upper bound on stretch for cost matrices of Type* I.

Definition 3.3. *A cost matrix is ergodic if it is either the hitting time matrix of an ergodic chain (Type* I*) or the commute time matrix of a reversible chain (Type* II*).*

*Note that Theorem* 2.2 *guarantees that we can test if a matrix is ergodic or not with essentially a single matrix inversion.*

Theorem 3.4. *Every graph with an ergodic cost matrix has a random walk with stretch $\leq n-1$. Moreover, this walk (termed ergodic walk) can be designed with a single matrix inversion.*

*Proof.* From Claims 1 and 2, the first part of the theorem follows. We now show how Theorem 2.2 can be used to design the desired random walk. Let $G$ be a graph with the ergodic cost matrix $C$. In view of (**) it is easy to see that $C_{in}+C_{nj}-C_{ij} = H_{in}+H_{nj}-H_{ij}$, regardless of whether $C$ is of Type I or II. Define $\bar{H}_{ij} = C_{in}+C_{nj}-C_{ij}$, for $1 \leq i,j \leq n-1$. The rest should be obvious: We construct $P$, the transition probability matrix of the desired walk, by first computing $\bar{P}$ using Theorem 2.2. ∎

Remark 4. *Recall that a cost matrix is resistive if the $C_{ij}$ can be interpreted as effective resistance $R_{ij}$ $\forall i,j$. Note that any resistive cost matrix is of Type* II *(modulo a constant factor), since effective resistance $R_{ij}$ is essentially the commute time $H_{ij}+H_{ji}$ (modulo the same constant factor) of a reversible chain. This, together with Fact* 1*, shows that our results imply those of* [8].

The lower and upper bounds are obviously tight under symmetry, since $\Psi(C) = 1$. The following example shows that the lower bound is, in general, tight.

*Example.* Consider a directed cycle $1, 2, \ldots, n, 1$ with cost 1 on each directed edge $(i, i+1)$. Now put in all other edges to make a directed $K_n$ and assign the distance along the original cycle to be the cost of each edge. Thus the cost matrix has $\Psi = n-1$. The optimal random walk (with stretch 1) is, in fact, the deterministic walk always going around the cycle. ∎

With each undirected cycle, we associate the following notion of a (strongly connected) "bicycle." A bicycle is a sequence of nodes $v_0, v_1, \ldots, v_{k-1}, v_0, v_{k-1}, \ldots, v_1, v_0$, i.e., the undirected cycle traversed once in either direction. The following asserts that our random walk is optimal over traversals of 1.

Corollary 3.5. *The stretch over any bicycle of any random walk is $\geq (n-1)$, and the ergodic walk achieves the equality.*

*Proof.* The equality is obvious in view of the preceding theorem. The proof of the lower bound is essentially the proof of Theorem 1 of [8]. ∎

**The cat and mouse game**. As mentioned in the introduction this game is a convenient tool in analyzing more complicated on-line strategies. For completeness, we describe the game here, but we refer the reader to [8] for further details and related interesting references. This is a game played for a fixed number of rounds between a cat and a mouse on a graph. Each round begins with both the cat and the mouse on the same vertex; the mouse moves once (and just once) at the beginning of the round to some (carefully chosen) vertex unknown to the cat. The rest of the round consists of the cat's moves (which could be deterministic or randomized) on the edges of the graph until the cat reaches the vertex that the mouse is at. Each move of the mouse can use information about all previous moves by the cat. A strategy for the cat is $c$-competitive if there exists an (additive) constant $a \geq 0$ such that for any number of rounds and any strategy of the mouse, the cat's expected cost is at most $c$ times the mouse's cost $+a$.

THEOREM 3.6. *For any $n \times n$ ergodic cost matrix $C$ and for any random walk strategy by the cat, there is a mouse strategy that forces the competitiveness of the cat to be at least $(n-1)/\Psi(C)$, and the ergodic walk by the cat achieves a competitive ratio $\leq (n-1)$.*

*Proof.* It was pointed out in [8] that a random walk with stretch $c$ defines a *memoryless* $c$-competitive strategy for the cat: in each round the cat, without recourse to its previous moves, executes a random walk with stretch $c$. Thus the upper bound is immediate from Theorem 3.4 above. For the lower bound, we use a standard argument (used, e.g., in [8] and [20]). Consider $(n-1)$ mice, one on each node except where the cat is. Whenever a cat moves from $i$ to $j$, the mouse on $j$ moves to $i$. Thus the mice together pay a cost of $\sum_{i=0}^{k-1} C_{v_{i+1},v_i}$, whenever the cat takes a walk $v_0, \ldots, v_k$ incurring a cost of $\sum_{i=0}^{k-1} C_{v_i,v_{i+1}}$. The single mouse strategy is going to be (just as in [8]) that we choose one of the $(n-1)$ strategies uniformly at random. By the definition of $\Psi(C)$, the mouse can always make the competitive ratio to be $\geq (n-1)/\Psi(C)$. □

## 4. *k*-servers with asymmetric costs.

Consider the usual $k$-server problem with the triangle inequality constraint on the costs. An *adaptive on-line* adversary (provably different from the *oblivious* and the *adaptive off-line* ones) chooses the next request at each step, knowing the current position of the on-line algorithm and, if 1, moves one of its servers to satisfy the request. We describe here a randomized on-line algorithm that works well against such an adaptive on-line adversary. (The significance of results proved against such an adversary is as follows. It was shown in [3] that a $c$-competitive randomized on-line algorithm against an adaptive on-line adversary implies the existence of a $c^2$-competitive deterministic algorithm.) Let us assume that all the costs are positive and bounded. Let us, however, *not* make the assumption that the cost $C_{ij}$ of moving a server from position $i$ to $j$ be the same as that of moving a server from $j$ to $i$, $C_{ji}$. We also make the strong assumption that every $k \times k$ submatrix is *ergodic* in the sense defined above.

DEFINITION 4.1. *Let the edge offset ratio $\Psi'(C)$ be $\max_{i,j}(C_{ij}/C_{ji})$.*

*We may simply write $\Psi'$ to denote $\Psi'(C)$ as long as there is no confusion as to the underlying $C$. Note that $\Psi(C) \leq \Psi'(C)$, and when $C$ is symmetric $\Psi(C) = \Psi'(C) = 1$.*

We are now ready to state the main theorem of this section.

THEOREM 4.2. *Let $C$ be a cost matrix on $n$ nodes. If every submatrix on $k+1$-nodes is ergodic, then we have a randomized $k\Psi'(C)$ competitive strategy (against an adaptive on-line adversary) for the $k$-server problem on $C$.*

*Proof.* For convenience, we refer to the $k$-servers under our randomized on-line strategy as *randomized servers*. The strategy is as follows. Suppose the randomized servers are on positions 1 through $k$. Let the current request be on position $x$. We find the (unique) ergodic walk that corresponds to these $k+1$ vertices by interpreting the costs as the hitting times. (Note that this computation, for all choices of $k+1$ vertices, can be done once and for all at the beginning.) Let $p_{ij}$ denote the transition probabilities of this walk. Then the strategy is to move the (randomized) server at $j$ (for $j = 1, \ldots, k$) to $x$ with probability $(p_{xj}/\sum_j p_{xj})$. (The probabilities clearly sum to 1, over all $j$.)

We can model the situation as a game between the randomized server and an adversary. The adversary controls $k$ "off-line" servers. In each round of the game, the adversary picks the next request (vertex) and moves one of his servers to that vertex.

Then the randomized server uses her strategy to move one of her servers to the request. We want to show that the expected cost of the randomized server is within a constant factor of $k$ times the cost incurred by the off-line server. Toward this, let us denote the positions of the randomized servers and the adversary's servers by $\mathbf{a} = \{a_1, \dots, a_k\}$ and $\mathbf{b} = \{b_1, \dots, b_k\}$, respectively. We define the following "potential function" $\Phi$:

$$\Phi(\mathbf{a}, \mathbf{b}) = \sum_{i,j=1}^{n} C_{ij} - \sum_{x} \sum_{j} C_{x a_j} + k \min_{\sigma} \sum_{i} C_{b_i a_{\sigma(i)}},$$

where $x$ ranges over nodes where there is currently no randomized server, and $\sigma$ ranges over (directed) matchings between the adversary's positions and the randomized servers' positions. (The first term in the definition of $\Phi$ is introduced simply to make $\Phi$ positive and is thus not essential to the proof.) Let $Cost_R$ and $Cost_A$ denote the accumulated costs of the randomized servers and the adversary's servers, respectively.

Furthermore, let

$$\Delta = \Phi + Cost_R - k\psi' \times Cost_A.$$

We would like to show that $\Delta$ is always nonincreasing with every move of either the randomized servers or the adversary.

**Adversary's move**. Consider a move by the adversary's server from node $b_j$ to $b_{j'}$. The adversary clearly pays cost $C_{b_j b_{j'}}$. We want to show that $\Delta(new) - \Delta(old) = \Phi(new) - \Phi(old) - k\psi' C_{b_j b_{j'}}$ is $\leq 0$. Let the minimum matching in $\Phi(old)$ be $b_i \to a_i \forall i$. Then define a new matching as follows. Match $b_i$ with $a_i$ for $i \neq j'$, and match $b_{j'}$ with $a_j$. Clearly,

$$\begin{aligned}
&\Phi(new) - \Phi(old) - k\psi' C_{b_j b_{j'}} \\
&\leq kC_{b_{j'} a_j} - kC_{b_j a_j} - k\psi' C_{b_j b_{j'}} \\
&\leq kC_{b_{j'} a_j} - kC_{b_j a_j} - kC_{b_{j'} b_j} \qquad \text{(by the definition of } \psi'\text{)} \\
&\leq 0, \qquad \text{since} \quad C_{b_{j'} b_j} + C_{b_j a_j} \geq C_{b_{j'} a_j}.
\end{aligned}$$

**Randomized servers' move**. For this half of the proof, it was shown in [8] that (restrictive as it may sound) it suffices to prove the case when the randomized servers and the adversary agree on (i.e., share the same vertices) $k-1$ positions and differ in only one position. The same justification applies in our situation as well, and we omit the straightforward proof.

Thus let us assume without loss of generality that the adversary occupies positions 1 through $m-1=k$, and the randomized servers occupy positions 2 through $m$. Let the request be on 1. So, the randomized server moves from $j$ to 1 with probability $p_{1j}/(\sum_j p_{1j})$, paying a cost of $C_{j1}$. Here and in the claim below, $\sum_j$ represents $\sum_{j=2}^{m}$. Note that $\sum_j p_{1j}$ is equal to 1, if $p_{11} = 0$, but in general, $\sum_j p_{1j} = 1 - p_{11}$.

Claim.

$$\sum_j \frac{p_{1j}}{\sum_j p_{1j}} [\Phi(new) - \Phi(old) + C_{j1}] = 0.$$

*Proof.* Clearly, the minimum matching before the move has cost $C_{1m}$, since the minimium matching consists of the (directed) edge $(1, m)$. Similarly, after the move (of the server from $j$ to 1), the minimum matching has cost $C_{jm}$ (match $j$ with $m$), since the adversary and the randomized servers are identical everywhere else. Thus

$$\begin{aligned}
&[\Phi(new) - \Phi(old) + C_{j1}] \\
&= -\sum_{i=1}^{m} C_{ji} + \sum_{i=2}^{m} C_{1i} + kC_{jm} - kC_{1m} + C_{j1} \\
&= -\sum_{i=2}^{m} C_{ji} + \sum_{i=2}^{m} C_{1i} + kC_{jm} - kC_{1m} \\
&= -\sum_{i=2}^{m} H_{ji} + \sum_{i=2}^{m} H_{1i} + kH_{jm} - kH_{1m}.
\end{aligned}$$

Thus

$$\begin{aligned}
&\sum_j \frac{p_{1j}}{\sum_j p_{1j}}[\Phi(new) - \Phi(old) + C_{j1}] \\
&= \frac{1}{\sum_j p_{1j}}\left[-\sum_{i=2}^{m}\sum_j p_{1j}H_{ji} + \sum_{i=2}^{m} H_{1i}\sum_j p_{1j}\right] + \frac{1}{\sum_j p_{1j}}\left[k\sum_j p_{1j}H_{jm} - kH_{1m}\sum_j p_{1j}\right] \\
&= \frac{1}{\sum_j p_{1j}}\left[\sum_{i=2}^{m}\left(-\sum_j p_{1j}H_{ji} + H_{1i}(1-p_{11})\right)\right] \\
&+ \frac{1}{\sum_j p_{1j}}\left[k\left(\sum_j p_{1j}H_{jm} - H_{1m}(1-p_{11})\right)\right] \\
&= \frac{1}{\sum_j p_{1j}}\left[\left(\sum_{i=2}^{m} 1\right) - k\right] \qquad \left(\text{since } H_{1i} = 1 + p_{11}H_{1i} + \textstyle\sum_{j=2}^{m} p_{1j}H_{ji}\right) \\
&= 0.
\end{aligned}$$

We showed that $\Delta$ is always nonincreasing. This concludes the proof that the randomized strategy is $k\Psi'$-competitive against an adaptive on-line adversary. □

## 5. Task systems.

Recall the description of a task system from the introduction: We have a task system $(S, C)$ for processing sequences of tasks wherein $S$ is a set of states, and $C$ is a cost matrix, describing the cost of changing from state $i$ to state $j$. The *task system problem* is to design an *on-line* schedule (choose $s_i$ only knowing $T_1, \ldots, T_i$) so that the algorithm is $w$-competitive—on any input sequence of tasks, the cost of the on-line algorithm is, barring an additive constant, at most $w$ times that of the optimal off-line algorithm. In [5] it was shown that $w(S,C) = 2|S| - 1$ for every *metrical* task system (MTS), and $w(S,C) \leq (2|S|-1)\Psi(C) = O(|S|^2)$, for every task system. Subsequently, [8] gave a $(2|S|-1)$-competitive *randomized* on-line algorithm for every MTS. It is to be noted that although this is a weaker result in light of the deterministic algorithm of [5], the randomized algorithm is conceptually and otherwise much simpler and is, moreover, *memoryless*.

Our contribution is as follows. We prove the analogous simplification for the (nonmetrical) task systems using randomization. We prove a lower bound of $(2|S| - 1)/\Psi(C)$ on the competitive ratio of any randomized on-line algorithm and provide a randomized on-line scheduler with $w(S,C) \leq (2|S|-1)$. Note that our results imply those of [8] in the case of metrical task systems. In the asymmetric case, we provide improved bounds to those of [5] with simpler memoryless randomized schedulers. However, we do have the restriction that the cost matrix should essentially be a hitting time matrix of an ergodic chain. Thus the random walk we refer to below is the ergodic walk designed by interpreting the cost matrix as the hitting time matrix as explained in sections 2 and 3.

**5.1. Lower and upper bounds.** The following lower bound on the competitiveness of any deterministic or randomized algorithm for the task system problem is straightforward to prove from the proofs in [5] and [8] for the lower bounds of the MTS problem.

THEOREM 5.1. *Any deterministic or randomized on-line algorithm for the task system problem has a competitive ratio of at least* $(2n-1)/\Psi(C)$ *against an adaptive on-line adversary.*

*Proof.* The proof is similar to the proofs for the symmetric case. For the deterministic part, follow the proof of Theorem 2.2 of [5]. For the randomized part, the proof is essentially that of Theorem 11 of [8]. ☐

For the upper bound we use the basic traversal algorithm first described in [5] and also used in [8] in the following modified form:

(i) The positions are visited in a sequence, that is, prescribed by a random walk, but independent of the input task sequence $T_1, T_2, \ldots$.

(ii) There is a sequence of positive *threshold* costs $\beta_1, \beta_2, \ldots$ such that the transition from $s_i$ to $s_{i+1}$ occurs when the total task processing cost incurred since entering $s_i$ reaches $\beta_i$.

Please refer to both [5] and [8] for a full account on such traversal algorithms, especially for the technical difficulties involved. The only originality on our part is in choosing an appropriate value for $\beta_i$. We simply choose $\beta_i$ to be the return time $H_{s_i s_i}$; i.e., when in state $j$, the random scheduler leaves state $j$ once the task processing cost incurred since reaching $j$ equals the expected time for a random walk beginning from $j$ to return to $j$ for the first time. (This has an intuitive appeal!) Once again the analysis in our case turns out to be quite simple.

THEOREM 5.2. *There exists a randomized traversal algorithm for task systems (with ergodic cost matrices) which is* $(2n-1)$*-competitive.*

*Proof.* By *total cost* we mean the sum of the task processing costs and the moving costs. We can assume without loss of generality (see [8]) that the adversary is a "cruel taskmaster," i.e., changes position only at the time the randomized on-line algorithm reaches its current position. We further distinguish *moving phases*, where the adversary changes positions, and *staying phases*, where the adversary stays in the same position but the randomized server moves. Let the current position be $i$. We first consider the cost incurred by the (randomized) on-line algorithm. Recall that we set $\beta_i = H_{ii}$. Also, from our results in section 3, recall that the expected cost per move is $E \leq (n-1)$. The expected task processing cost per move is $\sum_i \pi_i \beta_i = \sum_i \pi_i H_{ii} = n$, since $\pi_i$ is the steady state probability of being at $i$. So, the ratio of the expected total cost to expected cost per move (of the on-line algorithm) is $[n+E]/E$. Note that it suffices to show that the expected moving cost of the on-line algorithm is at most $E$ times the cost of the adversary—we would then have a competitive ratio of $[n+E] \leq (2n-1)$. We show this in the following two phases, depending on when the adversary is moving (the moving phase) and when the adversary is staying (the staying phase):

1. *The moving phase.* The average cost in this phase can be analyzed as a cat

and mouse game, with the adversary playing the mouse's role, yielding a competitive ratio of $E$.

2. *The staying phase.* The cost of the adversary starting (and ending) at node $i$ is $\beta_i$, whereas the expected moving cost of the on-line algorithm in that phase is $E \times H_{ii} = E\beta_i$. ∎

**5.2. Memoryless counterpart.** The above algorithm needs to store a current virtual position and a counter for the accumulated task processing cost at that position. However, this can also be made *memoryless* in the spirit of [8], without sacrificing the competitive ratio with the idea of using a *probabilistic counter*. We omit further discussion since the details are the same as those of [8], modulo the following aspect. We need to bound the stretch of a random walk while allowing for positive costs on self-loops. We merely state the requisite lemma and outline the proof idea.

LEMMA 5.3. *The ergodic walk has a stretch of at most* $(2n-1)$*, on a graph with cost matrix* $C$*, where* $C_{ii}$ *are not necessarily zero.*

*Proof idea.* We adopt the procedure that was described in detail in [8]. We merely suggest the solution and omit the proof, since it is similar to that of Theorem 7 of [8]. Intuitively, the idea is to place a special vertex on each self-loop with the appropriate transition probabilities and appeal to the case of no self-loops. Since the number of vertices is doubled (in the worst case), the stretch is at most $2n-1$ rather than $n-1$. ∎

## 6. Loose ends.

The most important issue here is in extending our results to *all* cost matrices (say, those that satisfy the triangle inequality). In particular, tighter upper and lower bounds on a stretch of random walks on directed graphs would constitute significant progress. Coppersmith et al. [8] extend their results from *resistive* cost matrices to all cost matrices (at least existentially, if not with an efficient construction); i.e., they show that given any symmetric cost matrix $\{C_{ij}\}$, satisfying the triangle inequality, there exists a resistive (approximation) network (with conductances $c_{ij}$) such that the effective resistance $R_{ij} \le C_{ij}$ whenever $c_{ij} \ge 0$, with $R_{ij} = C_{ij}$ whenever $c_{ij} > 0$. Much in the same spirit we would like to aim for an ergodic Markov chain with the property that $H_{ij} \le C_{ij}$, with equality whenever $p_{ij} > 0$. The existence of such an "approximate chain" is clearly implied by the result of [8] when we are seeking a reversible chain. The main hurdle in mimicking the approach taken in [8] for the nonreversible case is the following. Consider the space of all $n-1 \times n-1$ matrices, $\mathcal{P} = \{\bar{P}\}$, where $\bar{P}$ corresponds (as in Theorem 2.2) to an ergodic chain on $n$ states. It is easy to see that for $0 \le \alpha \le 1$,

$$\bar{P}_1, \bar{P}_2 \in \mathcal{P} \quad \Rightarrow \quad \alpha\bar{P}_1 + (1-\alpha)\bar{P}_2 \in \mathcal{P}.$$

Let's even assume that $\mathcal{P}$ is a space of positive definite matrices. Now the function, log of the determinant, is concave over the space of positive definite *symmetric* (or Hermitian) matrices, and symmetry is crucial here. We have examples of asymmetric $\bar{P}$ (i.e., nonreversible chains) which show that the log of the determinant is neither concave nor convex over $\mathcal{P}$. This makes the analogous result for the nonreversible case much harder. On the other hand, Coppersmith et al. benefit from convexity as follows. They formulate the approximation as an appropriate convex programming problem; the existence of the approximate chain (resistive network) is then guaranteed by simply appealing to the "Kuhn–Tucker" (necessary and sufficient) conditions arising in the solution of the convex programming problem. It is still conceivable that a somewhat different approach works for the nonreversible case.

**Acknowledgments.** The author thanks Steve Phillips, Prabhakar Raghavan, Nick Reingold, and Emre Telatar for many useful discussions. The author would also like to thank Mike Saks (the no-longer-anonymous referee) for a careful reading and constructive criticism of the manuscript.

## REFERENCES

[1] D. Aldous, *Personal communication,* 1993.

[2] D. Aldous and J. Fill, *Reversible Markov Chains and Random Walks on Graphs*, draft of book in preparation.

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

[4] P. Berman, H. J. Karloff, and G. Tárdos, *A competitive* 3*-server algorithm*, in Proc. 1st Annual ACM-SIAM Symp. on Discrete Algorithms, SIAM, Philadelphia, PA, 1990, pp. 280–290.

[5] A. Borodin, N. Linial, and M. Saks, *An Optimal on-line algorithm for metrical task system*, J. Amer. Math. Soc., 39 (1992), pp. 745–763.

[6] M. Chrobak and L. L. Larmore, *An optimal on-line algorithm for $k$-servers on trees*, SIAM J. Comput., 20 (1991), pp. 144–148.

[7] 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*, Comput. Complexity, 6 (1996/97), pp. 312–340.

[8] D. Coppersmith, P. Doyle, P. Raghavan, and M. Snir, *Random walks on weighted graphs, and applications to on-line algorithms*, J. Amer. Math. Soc., 40 (1993), pp. 421–453.

[9] D. Coppersmith, P. Tetali, and P. Winkler, *Collisions among random walks on a graph*, SIAM J. Discrete Math., 6 (1993), pp. 363–374.

[10] P. G. Doyle and J. L. Snell, *Random Walks and Electric Networks*, Carus Mathematical Monographs 22, Mathematical Association of America, Washington, DC, 1984.

[11] A. Fiat, Y. Rabani, and Y. Ravid, *Competitive $k$-server algorithms*, J. Comput. System Sci., 48 (1994), pp. 410–428.

[12] M. Fiedler, C. R. Johnson, T. L. Markham, and M. Neumann, *A trace inequality for $M$-matrices and the symmetrizability of a real matrix by a positive diagonal matrix*, Linear Algebra Appl., 71 (1985), pp. 81–94.

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

[14] R. M. Foster, *An extension of a network theorem*, IRE Trans. Circuit Theory, 8 (1961), pp. 75–76.

[15] F. Gobel and A. A. Jagers, *Random walks on graphs*, Stochastic Process. Appl., 2 (1974), pp. 311–336.

[16] E. Grove, *The harmonic online $K$-server algorithm is competitive*, in On-Line Algorithms, DIMACS Ser. Discrete Math. Theoret. Comput. Sci., 7, AMS, Providence, RI, 1992, pp. 65–75.

[17] E. Koutsoupias and C. H. Papadimitriou, *On the $k$-server conjecture*, J. Amer. Math. Soc., 42 (1995), pp. 971–983.

[18] J. G. Kemeny and J. L. Snell, *Finite Markov Chains*, Springer-Verlag, New York, 1983.

[19] J. G. Kemeny, J. L. Snell, and A. W. Knapp, *Denumerable Markov Chains*, Springer-Verlag, New York, 1976.

[20] M. S. Manasse, L. A. McGeoch, and D. D. Sleator, *Competitive algorithms for server problems*, J. Algorithms, 11 (1990), pp. 208–230.

[21] H. Minc, *Nonnegative Matrices*, Wiley-Interscience, New York, 1988.

[22] P. Tetali, *Random walks and the effective resistance of networks*, J. Theoret. Probab., 4 (1991), pp. 101–109.

[23] P. Tetali, *An extension of Foster's network theorem*, Combin. Probab. Comput. (special issue in honor of Paul Erdős's 80th birthday), 3 (1994), pp. 421–427.