Closed Form Bounds for Clock Synchronization Under Simple Uncertainty Assumptions

Saâd Biaz[*] Jennifer L. Welch[†]

Key words: Distributed Computing – Clock Synchronization – Optimal Precision – Generalized Hypercube – Lower Bound.

1 Introduction

Many applications in a distributed system rely on processors having synchronized local clocks. Many studies have been dedicated to algorithms and lower bounds for clock synchronization under various network assumptions. We consider the problem of synchronizing the clocks of processors in a failure-free distributed system when the hardware clocks do not drift but there is uncertainty in the message delays. Our goal is to develop closed form expressions for how closely the clocks can be synchronized.

Lundelius and Lynch [5] showed that the local clocks of n processors in a fully connected network with the same uncertainty u on each link cannot be synchronized any more closely than $u \cdot (1 - \frac{1}{n})$. They provide a simple algorithm that achieves this bound. Halpern *et al.* [3] subsequently extended this work to consider arbitrary topologies in which each directed edge may have a different uncertainty. They established a closed form expression for the optimal synchronization in a tree and a triangle. For the tree, the bound is equal to $\frac{1}{2}$*diam*, where *diam* is the diameter of the tree with respect to the uncertainties on the links. For the general case, they show that the optimal synchronization is the solution of an optimization problem using linear programming techniques, but they do not give a closed form expression.

In this paper we address open question 7 in [3]: "...it would be interesting to obtain precise formulas for the imprecision for a number of graphs that arise in practice." For an arbitrary undirected topology with arbitrary symmetric uncertainties, we prove a lower bound of $\frac{1}{2}$*diam* on the closeness of synchronization achievable, where *diam* is the diameter of the graph when the edges are weighted with the uncertainties. Taken together with the upper bound of *radius* (the radius of the graph with respect to the uncertainties) previously shown in [3], our result indicates that the tight bound for any (symmetric) topology is known to within a factor of two, since the radius is at most the diameter.

[*]School of Science and Engineering, Al Akhawayn University, Ifrane, MOROCCO. saadb@cs.tamu.edu. Phone: (212) 5586-2106. Supported by the Fulbright program. On leave from the Ecole Supérieure de Technologie Fès (MOROCCO).

[†]Department of Computer Science, Texas A&M University, College Station, TX 77843. welch@cs.tamu.edu. Phone: (979) 845-5076. Fax: (979) 847-8578. Supported in part by NSF CCR-9396098.

We then consider the class of topologies described as k-ary m-cubes, both with and without wrap-around. The number of nodes, n, equals k^m. We assume that every edge has the same uncertainty, u. For the k-ary m-cube without wrap-around, we show that our lower bound $\frac{1}{2}\mathit{diam} = \frac{1}{2}um(k-1)$ is tight, by analyzing the synchronization achieved by a simple algorithm. Since chains, (square) meshes, and hypercubes are special cases of this topology, we have the following tight bounds: for an n-processor chain ($k = n, m = 1$), $\frac{1}{2}u(n-1)$; for a $\sqrt{n} \times \sqrt{n}$ mesh ($k = \sqrt{n}, m = 2$), $u(\sqrt{n}-1)$; for a $\log n$-dimensional hypercube ($k = 2, m = \log n$), $\frac{1}{2}u \log n$.

For the k-ary m-cube with wrap-around, we show using the same algorithm that our lower bound $\frac{1}{2}\mathit{diam} = \frac{1}{2}um\lfloor\frac{k}{2}\rfloor$ is tight when k is even and is almost tight when k is odd. We restrict attention to the case $k \geq 3$; otherwise wrap-around does not add any additional communication possibilities. When k is even, the tight bound is $\frac{1}{4}umk$. When k is odd, the lower bound is $\frac{1}{4}um(k-1)$ and the upper bound is $\frac{1}{4}um(k-\frac{1}{k})$. This result implies tight or almost tight bounds for rings and tori. In particular, we have the following tight bounds: for n-processor ring with n even ($k = n, m = 1$), $\frac{1}{4}un$; for $\sqrt{n} \times \sqrt{n}$ torus with $\sqrt{n}$ even ($k = \sqrt{n}, m = 2$), $\frac{1}{2}u\sqrt{n}$. The analogous bounds when n is odd are not quite tight. In particular, for the ring the lower bound is $\frac{1}{4}u(n-1)$ and the upper bound is $\frac{1}{4}u(n-\frac{1}{n})$; for the torus the lower bound is $\frac{1}{2}u(\sqrt{n}-1)$ and the upper bound is $\frac{1}{2}u(\sqrt{n}-\frac{1}{\sqrt{n}})$.

Attiya *et al.* [1] take a different approach to the problem. They consider arbitrary topologies under different delay assumptions. Instead of trying to establish the optimal closeness of synchronization achievable in a given network, they consider the problem of establishing the tightest clock synchronization for any given *execution*. In other words, given the information collected by any algorithm strategy, they give the optimal synchronization possible. They show how an external observer who has access to all the views of the processors can compute the optimal adjustment to each clock. Patt-Shamir and Rajsbaum [6] extended this work to provide an on-line algorithm optimal for every execution.

In Section 2, we present our model and review relevant prior results. In Section 3, we establish a lower bound for an arbitrary topology. Section 4 is dedicated to establish closed forms for the upper bounds for k-ary m-cube networks, first for networks without wrap-around and then for networks with wrap-around. We conclude in Section 5 with some open problems.

2 Definitions and Prior Results

We first describe our formal model and problem definition (following [2], which is based on those in [5] and [3]). Then we summarize previous results upon which our results rely, namely the conversion of any communication network to an equivalent clique and the technique of shifting executions.

2.1 Model

We consider a set of n processors p_0 through p_{n-1}, connected via point-to-point links. Each link, say the one between p_i and p_j, reliably delivers messages from p_i to p_j and vice versa subject to some arbitrary delay within a known range. (We assume the range is the same in both directions.) The topology and message delay ranges are represented with an undirected graph $G = (V, E)$, whose nodes represent processors and whose edges represent links, together with two mappings L and H from E to nonnegative reals. $L(i, j)$ is the minimum delay on the edge between p_i and p_j and $H(i, j)$ is the maximum delay on the edge between p_i and p_j. (Thus $H(i, j)$ must be at least as large as $L(i, j)$.) The difference, $H(i, j) - L(i, j)$, is the *uncertainty* on the edge. $G = (V, E, L, H)$ is said to be a *communication network*.

The *events* that can occur at a processor include the arrival of messages from processors that are neighbors in the given topology, as well as internal happenings. Processor p_i has a *hardware clock value* HC_i, which is a function from reals (real time) to reals (clock time) of the form $HC_i(t) = t + c_i$ for some constant c_i. This is a hardware clock with no drift; it runs at the same rate as real time. A *processor* p_i is a state machine with a set of initial states and a transition function. The transition function takes as input the current state, the current value of the hardware clock, and the current event and produces a new state and a set of messages to send to p_i's neighbors. The hardware clock is not part of the processor's state and cannot be modified by the processor.

A *history* of processor p_i is a sequence of alternating states and (event,hardware clock value) pairs for p_i, beginning with p_i's initial state. We require that each subsequent state follows correctly, according to p_i's transition function, from p_i's previous state, event and hardware clock value. A *timed history* of p_i is a history of p_i together with an assignment of real times to each (event,hardware clock value) pair. The real times must be consistent with the hardware clock values, i.e., the time t assigned to (e, T) must satisfy $HC_i(t) = T$. An *execution* (of the entire system) is a set of n timed histories, one per processor. For every pair of processors p_i and p_j, there must be a bijection from messages sent by p_i to messages received by p_j (i.e., every message sent is received, and only messages sent are received). The *delay* of a message is the difference between the real time when it was sent and the real time when it was received.

Two executions α_1 and α_2 are *indistinguishable* if, for each processor p_i, p_i has the same (untimed) history in α_1 as in α_2.

An execution α is *admissible* for communication network $G = (V, E, L, H)$ if every message between any two processors p_i and p_j has delay within the interval $[L(i, j), H(i, j)]$.

2.2 The Clock Synchronization Problem

We now precisely define the problem of synchronizing clocks. We assume that each processor p_i has a state component adj_i which is used to adjust the clock value. We define the processor's *adjusted clock* $AC_i(t)$ to be equal to $HC_i(t) + adj_i(t)$ (the second term is the value of adj_i at real time t). The algorithm has *terminated* in an execution at a point if no processor ever changes its *adj* variable thereafter.

Achieving ε-Synchronized Clocks (in communication network G): *In every execution that is admissible for G, there exists a real time t_f such that the algorithm has terminated by real time t_f, and, for all processors p_i and p_j, and all $t \geq t_f$, $|AC_i(t) - AC_j(t)| \leq \varepsilon$.*

Given a communication network G, let *opt*(G) be the smallest ε such that ε-synchronized clocks can be achieved in G. This quantity is the *optimal clock synchronization* that can be achieved in the network. We are interested in obtaining tight bounds on *opt*(G) for various G's.

2.3 The Equivalent Clique

If two communication networks G and G' satisfy *opt*(G) = *opt*(G'), then we say that they are *equivalent*. Consider any communication network $G = (V, E, L, H)$. Let G' be the communication network (V, E', L', H'), where $E' = V \times V$ (i.e., G' is a clique), and for all i and j, $L'(i,j) = 0$ and $H'(i,j)$ is the length of the shortest path between p_i and p_j in G with respect to the uncertainties. Section 6 of [3] shows:

Theorem 1 *The communication networks G and G' are equivalent.*

We prove our lower bound and analyze our algorithm for cliques, instead of more complicated topologies. Theorem 1 shows that the results obtained on the cliques apply to the original topologies. Note that the uncertainties in G' satisfy the triangle inequality, since H' is defined using shortest path lengths.

2.4 Shifting

Like those in [5] and [3], our lower bound result is obtained by "shifting" executions, that is, by shifting the real times at which events occur. More formally, let $\alpha = (\eta_0, \eta_1, \ldots, \eta_{n-1})$ be an execution and let x_i be a real number, $0 \leq i \leq n-1$. Define *shift*$(\alpha, \langle x_0, \ldots, x_{n-1}\rangle)$ to be $(\eta'_0, \eta'_1, \ldots, \eta'_{n-1})$, where each η'_i is the timed history resulting from η_i by adding x_i to the real time associated with each event in η_i. Shifting an execution changes the hardware clocks and the delays experienced by messages. Lundelius and Lynch [5] quantified the changes as follows:

Lemma 2 *Let α be an execution with hardware clocks HC$_i$ and let x_i be a real number,* $0 \leq i \leq n-1$. *Then shift$(\alpha, \langle x_0, \ldots, x_{n-1}\rangle)$ is an execution that is indistinguishable from α, and in which*

(a) the hardware clock of p_i, HC$'_i(t)$, is equal to HC$_i(t) - x_i$ for all t, and

(b) every message from p_i to p_j has delay $\delta - x_i + x_j$, where δ is the delay of the message in α.

Note that shifting an execution might not result in an admissible execution.

3 Lower Bound for Arbitrary Topology

Let *diam*(G) denote the diameter of communication network G with respect to the uncertainties.

Theorem 3 *For any communication network G,* $\mathrm{opt}(G) \geq \frac{1}{2}\mathrm{diam}(G)$.

Proof Let G' be the clique from Section 2.3 that is equivalent to G. Let u_{ij} denote $H'(i,j)$, the uncertainty on edge (p_i, p_j) in G'. We will show that *opt*(G') is at least $\frac{1}{2}$*diam*(G'). Since Theorem 1 implies that *opt*(G') = *opt*(G) and since the diameter of G is the same as that of G', the result follows.

Let p_a and p_b be two processors in G' such that u_{ab} equals the diameter of G'.

Choose any algorithm for G' that achieves ε-synchronized clocks. Consider the admissible execution α in which the delays are as follows. For any two processors p_i and p_j, where $u_{ai} < u_{aj}$ (i.e., p_i is at least as close to p_a, w.r.t. uncertainties, as p_j is), the delay of every message from p_i to p_j is 0, and the delay of every message from p_j to p_i is u_{ij}. When $u_{ai} = u_{aj}$, we choose the delay of every message from p_i to p_j to be 0, and the delay of every message from p_j to p_i to be u_{ij}.

Since the algorithm must work correctly in α,

$$AC_a \geq AC_b - \varepsilon. \tag{1}$$

Let $\alpha' = \text{shift}(\alpha, \vec{x})$, where $x_i = u_{ai} - u_{ab}$, $0 \leq i \leq n-1$. Note that p_a is shifted by $-u_{ab}$ and p_b is shifted by 0.

By Lemma 2, the delay of every message in α' from p_i to p_j is $0 - (u_{ai} - u_{ab}) + (u_{aj} - u_{ab}) = u_{aj} - u_{ai}$. This expression is at most u_{ij}, by the triangle inequality, so the new delay is not too big. This expression is at least 0, since $u_{ai} \leq u_{aj}$, so the new delay is not too small.

By Lemma 2, the delay of every message in α' from p_j to p_i is $u_{ij} - (u_{aj} - u_{ab}) + (u_{ai} - u_{ab}) = u_{ij} + u_{ai} - u_{aj}$. This expression is at most u_{ij}, since $u_{ai} \leq u_{aj}$, so the new delay is not too big. This expression is at least 0, by the triangle inequality, so the new delay is not too small.

Therefore α' is admissible. Since the algorithm must work correctly in α', we have:

$$AC'_b \geq AC'_a - \varepsilon. \tag{2}$$

Since $AC'_b = AC_b + 0$ and $AC'_a = AC_a + u_{ab}$, inequalities (1) and (2) imply that $\varepsilon \geq \frac{1}{2}u_{ab}$.

Thus for the arbitrary algorithm, ε is at least half the diameter. ■

4 Upper Bounds

4.1 The Algorithm

We consider the averaging algorithm of [5] for a clique, slightly modified to handle the fact that different edges have different uncertainties. (The description of the algorithm and Lemma 4 are modeled after [2].) Each processor p_i begins by sending its current hardware clock value to all the other processors. It has an array *diff* whose j-th entry is set equal to p_i's estimated difference between p_i's hardware clock and p_j's hardware clock. For $j \neq i$, this difference is estimated using the value T contained in the message that p_i receives from p_j and assuming that this message took $\frac{u_{ij}}{2}$ time to arrive. Once p_i has an estimate for

```
initially $\mathit{diff}_i[i] = 0$

at first computation step:
1:  send $HC_i$ (current hardware clock value) to all other processors

upon receiving message $T$ from some $p_j$:
2:  $\mathit{diff}_i[j] := T + \frac{u_{ij}}{2} - HC_i$
3:  if a message has been received from every other processor then
4:      $adj_i := \frac{1}{n}\sum_{k=0}^{n-1} \mathit{diff}_i[k]$
```

Figure 1: Averaging algorithm; code for processor p_i, $0 \leq i \leq n-1$.

every processor, it sets its adjustment to the average of all the estimated differences. (See Figure 1.) We will analyze the clock synchronization achieved by the averaging algorithm for a clique G' with arbitrary symmetric uncertainties (denoted u_{ij}). Then we will show how the general result can be specialized into closed forms for specific cliques that are equivalent to certain topologies. Theorem 1 shows that the upper bounds obtained for these cliques also hold for the other topologies. The applicability of the averaging algorithm to general topologies requires all-pairs shortest-paths computation.

Lemma 4 *Consider any execution of the averaging algorithm that is admissible for G'. For any two processors p_i and p_j and any time t after the algorithm terminates (executes Line 4),*

$$|AC_i(t) - AC_j(t)| \quad \leq \quad \frac{1}{2n}\left(\sum_{l=0,l\neq i}^{n-1} u_{li} + \sum_{l=0,l\neq j}^{n-1} u_{lj}\right).$$

Proof For every time t after p_i sets $\mathit{diff}_i[j]$, $\mathit{diff}_i[j](t) = HC_j(t) - HC_i(t) + err_{ji}$, where err_{ji} is a constant. The definition of the algorithm shows that

$$|AC_i(t) - AC_j(t)| \quad \leq \quad \frac{1}{n}\left(|err_{ij}| + |err_{ji}| + \sum_{l=0,l\neq i,j}^{n-1} |err_{li} - err_{lj}|\right).$$

The bounds on err_{ji} and the rules of absolute value give the result. ■

4.2 Generalized Hypercubes Without Wrap-around

The k-ary m-cube network is a generalization of the hypercube architecture where $k = 2$. In the k-ary m-cube, there are m dimensions and k nodes on each dimension (see [4], p. 86). More formally, each node is represented as a vector $\langle a_0, a_1, \ldots, a_{m-1}\rangle$, where each a_r is an element of $\{0, 1, \ldots, k-1\}$. Thus there are $n = k^m$ nodes. There is an edge between two nodes $\vec{a}$ and $\vec{b}$ if they differ in exactly one component, say the r-th, and $a_r = b_r + 1$. We need the following fact about distances in (unweighted) k-ary m-cubes without wrap-around. Given such a graph G, let $dist_G(\vec{a}, \vec{b})$ be the minimum number of edges in any path from $\vec{a}$ to $\vec{b}$ in G.

Lemma 5 *For any k-ary m-cube $G = (V, E)$ without wrap-around, for any node $\vec{a}$ in V,*

$$\sum_{\vec{b}} dist_G(\vec{a}, \vec{b}) \le \frac{1}{2} m(k^{m+1} - k^m).$$

Proof Consider a particular node $\vec{b}$. The distance between $\vec{a}$ and $\vec{b}$ is the sum, over all dimensions r, of the distance from a_r to b_r. With no wrap-around, this distance is $|b_r - a_r|$. It is obvious that $\sum_{\vec{b}} dist_G(\vec{a}, \vec{b}) \le \sum_{\vec{b}} dist_G(\vec{0}, \vec{b})$ where $\vec{0}$ is the vector of all zeroes.

Let us then compute $\sum_{\vec{b}} dist_G(\vec{0}, \vec{b})$. This sum can be written:

$$\sum_{\vec{b}} \sum_{r=0}^{m-1} b_r. \tag{3}$$

Now let us concentrate on dimension r. For how many vectors $\vec{b}$ is b_r equal to 0? There are k independent choices for each of the $m - 1$ other entries, so the answer is k^{m-1}. The same is true for every value of b_r between 0 and $k - 1$. Thus the contribution to the sum in expression (3) from dimension r is

$$k^{m-1} \sum_{h=0}^{k-1} h.$$

This sum can be written as $\frac{1}{2}(k^{m+1} - k^m)$. We multiply this expression by m, to account for all the dimensions. ■

Theorem 6 *For any k-ary m-cube (without wrap-around) communication network G with uncertainty u on each edge,* opt(G) *is at most $\frac{1}{2}um(k-1)$.*

Proof Let G' be the equivalent clique communication network from Section 2.3. By Theorem 1, it suffices to show that the averaging function on G' achieves ε-synchronized clocks, where $\varepsilon = \frac{1}{2}um(k-1)$.

Consider any execution of the algorithm that is admissible for G'. By Lemma 4, for any two processors p_i and p_j and any time t after the algorithm terminates

$$|AC_i(t) - AC_j(t)| \quad \le \quad \frac{1}{2n}\left(\sum_{l=0, l\neq i}^{n-1} u_{li} + \sum_{l=0, l\neq j}^{n-1} u_{lj}\right).$$

By the definition of G', $u_{lh} = u \cdot dist_G(l, h)$, for $h = i, j$. By Lemma 5, for $h = i, j$,

$$\sum_{l=0, l\neq h}^{n-1} u_{lh} \le um\frac{1}{2}(k^{m+1} - k^m).$$

Since $n = k^m$, the result follows.

■

4.3 Generalized Hypercubes With Wrap-around

The definition of the k-ary m-cube with wrap-around is the same as for the k-ary m-cube without wrap-around, except that we add "mod k" to the condition under which there is an edge between two nodes. That is, there is an edge between two nodes $\vec{a}$ and $\vec{b}$ if they differ in exactly one component, say the r-th, and if $a_r = b_r + 1 \bmod k$.

Given a graph G, let $dist_G(\vec{a}, \vec{b})$ be the minimum number of edges in any path from $\vec{a}$ to $\vec{b}$ in G. We need the following fact about distances in (unweighted) k-ary m-cubes with wrap-around.

Lemma 7 *For any k-ary m-cube $G = (V, E)$ with wrap-around, for any node $\vec{a}$ in V,*

$$\sum_{\vec{b}} dist_G(\vec{a}, \vec{b}) = \begin{cases} \frac{1}{4}mk^{m+1} & when\ k\ is\ even \\ \frac{1}{4}m(k^{m+1} - k^{m-1}) & when\ k\ is\ odd. \end{cases}$$

Proof Consider a particular node $\vec{b}$. The distance between $\vec{a}$ and $\vec{b}$ is the sum, over all dimensions r, of the distance from a_r to b_r. However, we must take into account the wrap-around edges. In this case the distance is $|b_r - a_r|$ if this quantity is at most $\lfloor \frac{k}{2} \rfloor$, otherwise it is $k - |b_r - a_r|$.

Thus the desired sum can be written:

$$\sum_{\vec{b}} \sum_{r=0}^{m-1} \min(|b_r - a_r|, k - |b_r - a_r|). \tag{4}$$

Now let us concentrate on dimension r. For how many vectors $\vec{b}$ is b_r equal to 0? There are k independent choices for each of the $m - 1$ other entries, so the answer is k^{m-1}. The same is true for every value of b_r between 0 and $k - 1$. Thus the contribution to the sum in expression (4) from dimension r is

$$k^{m-1} \sum_{h=0}^{k-1} \min(|h - a_r|, k - |h - a_r|). \tag{5}$$

This sum contains two terms equal to 1 (there are two nodes at distance 1 from $\vec{a}$ along dimension r), two terms equal to 2, etc., up to two terms equal to $\lfloor \frac{k-1}{2} \rfloor$, and, if k is even, one term equal to $\frac{k}{2}$ (there is one node at distance $\frac{k}{2}$ from $\vec{a}$ along dimension r).

First consider the case when k is odd. Then expression (5) becomes

$$2k^{m-1} \sum_{j=1}^{\lfloor \frac{k-1}{2} \rfloor} j.$$

Algebraic manipulations reduce this to $\frac{1}{4}(k^{m+1} - k^{m-1})$. We multiply this expression by m, to account for all the dimensions.

Now consider the case when k is even. Expression (5) now becomes

$$2k^{m-1} \sum_{j=1}^{\frac{k}{2}-1} j + k^{m-1} \cdot \frac{k}{2}.$$

Algebraic manipulations reduce this to $\frac{1}{4}k^{m+1}$. We multiply by m to account for all dimensions. ■

Theorem 8 *For any k-ary m-cube (with wrap-around) communication network G with uncertainty u on each edge,* $\mathrm{opt}(G)$ *is at most $\frac{1}{4}umk$ when k is even and $\frac{1}{4}um(k-\frac{1}{k})$ when k is odd.*

Proof The proof is essentially the same as that for Theorem 6. The differences are that $\varepsilon = \frac{1}{4}umk$ when k is even and $\frac{1}{4}um(k-\frac{1}{k})$ otherwise, and that Lemma 7 is invoked instead of Lemma 5. ■

5 Open Questions

The obvious open question in our work is to tighten the bounds for k-ary m-cubes with wrap-around when k is odd. We believe that the algorithm is optimal and that a more involved shifting argument is needed to strengthen the lower bound. The evidence for this is that a 3-ary 1-cube is a ring with three processors, which is also a clique. Lundelius and Lynch [5] showed in this case that the tight bound is $\frac{2}{3}u$, which is what the algorithm gives, and not $\frac{1}{2}u$ as given by the lower bound of this paper.

It would be interesting to characterize the class of networks and uncertainty assumptions for which the simple algorithm is optimal, and similarly those for which the $\frac{1}{2}$*diam* lower bound is tight.

Further work is needed to obtain closed form expressions for other networks and delay assumptions.

References

[1] H. Attiya, A. Herzberg, and S. Rajsbaum, "Optimal Clock Synchronization Under Different Delay Assumptions," *Proc. of the 12th Annual ACM Symp. on Principles of Distributed Computing,* pp. 109-120, 1993.

[2] H. Attiya and J. L. Welch, *Distributed Computing: Fundamentals, Simulations and Advanced Topics,* (Section 6.3), McGraw-Hill Publishing Company, 1998.

[3] J. Y. Halpern, N. Megiddo, and A. A. Munshi, "Optimal Precision in The Presence of Uncertainty," *Journal of Complexity,* vol. 1, pp. 170-196, 1985.

[4] K. Hwang, *Advanced Computer Architecture,* McGraw-Hill Inc., New York, 1993.

[5] J. Lundelius and N. A. Lynch, "An Upper and Lower Bound for Clock Synchronization," *Information and Control,* vol. 62, pp. 190-204, 1984.

[6] B. Patt-Shamir and S. Rajsbaum, "A Theory of Clock Synchronization," *Proceedings of the 26th Annual ACM Symposium on Theory of Computing,* pp. 810-819, 1994.