SIAM J. COMPUT.
Vol. 15, No. 4, November 1986

012

A SIMPLE PARALLEL ALGORITHM FOR THE MAXIMAL INDEPENDENT SET PROBLEM*

MICHAEL LUBY†

Abstract. Two basic design strategies are used to develop a very simple and fast parallel algorithms for the maximal independent set (MIS) problem. The first strategy consists of assigning identical copies of a simple algorithm to small local portions of the problem input. The algorithm is designed so that when the copies are executed in parallel the correct problem output is produced very quickly. A very simple Monte Carlo algorithm for the MIS problem is presented which is based upon this strategy. The second strategy is a general and powerful technique for removing randomization from algorithms. This strategy is used to convert the Monte Carlo algorithm for this MIS problem into a simple deterministic algorithm with the same parallel running time.

Key words. parallel computations, NC, maximal independent set, randomizing algorithms, pairwise independences

AMS(MOS) subject classifications. 05B99, 05C99, 62E25, 62K99, 68O0, 68E10, 68G99

Introduction. A maximal independent set (MIS) in an undirected graph is a maximal collection of vertices I subject to the restriction that no pair of vertices in I are adjacent. The MIS problem is to find a MIS. In this paper, fast parallel algorithms are presented for the MIS problem. All of the algorithms are especially noteworthy for their simplicity.

One of the key properties behind any successful parallel algorithm design is that the algorithm can be productively subdivided into a number of almost identical simple algorithms, which when executed in parallel produce a correct problem output very quickly. Monte Carlo Algorithm A for the MIS problem (§ 3.2) has an even stronger property: the subdivision into almost identical simple algorithms respects the inherent subdivision in the problem input. More specifically, the problem input is a local description of the graph in terms of vertices and edges. Algorithm A is described in terms of two algorithm templates called *ALGVERTEX* and *ALGEDGE*. A copy of *ALGVERTEX* is assigned to each vertex in the graph and a copy of *ALGEDGE* is assigned to each edge in the graph. The algorithm runs in phases. During each phase all copies of *ALGVERTEX* are executed in parallel followed by the execution of all copies of *ALGEDGE* in parallel. The algorithm has the property that after a very small number of phases the output is a MIS. This property of the Monte Carlo algorithm may make it a useful protocol design tool in distributed computation.

One of the main contributions of this paper is the development of a powerful and general technique for converting parallel Monte Carlo algorithms into deterministic algorithms (§ 4.1). Monte Carlo Algorithm B (§ 3.3) is very similar to, but slightly more complicated than, Algorithm A. The random variables in Algorithm B are mutually independent. The general technique is used to convert Algorithm B into a deterministic algorithm with the same running time. The first major step in the conversion process is a more sophisticated analysis of the algorithm (§ 4.3), which shows that if the random variables are only *pairwise independent* [Fr] then the algorithm has essentially the same expected running time as when the random variables are mutually independent. The second major step is a method for generating a probability space over n random variables containing $O(n^2)$ sample points, where the n random variables are pairwise

* Received by the editors July 1, 1985, and in final form September 18, 1985.
† Department of Computer Science, University of Toronto, Toronto, Ontario, Canada M5S 1A4.

independent (§ 4.2). Algorithm C, which is almost exactly the same as Algorithm B, chooses values for the random variables by randomly choosing one of the sample points in this probability space (§ 4.4). The number of random bits needed to choose a random sample point is $O(\log n)$. Algorithm D tests in parallel all of the sample points and uses the best (§ 4.4). This algorithm is deterministic.

For purposes of analysis the P-RAM parallel computer is used [FW]. Two models of a P-RAM are considered: the CRCW P-RAM, in which concurrent reads and writes to the same memory location are allowed; and the less powerful but perhaps more realistic EREW P-RAM, in which concurrent reads and writes to the same memory location are disallowed. Algorithm A is the simplest Monte Carlo algorithm and has the best running time on a CRCW P-RAM. Algorithm B is slightly more complicated, but it is presented because it is the basis for Algorithms C and D. Algorithm D can be implemented by a logspace-uniform circuit family, where the circuit which accepts inputs of length k has depth $O((\log k)^2)$ and polynomial in k gates. (All logs are base 2 in this paper.) This establishes that the MIS problem is in NC^2 (see [Co] for a discussion of the complexity class NC). Let n be the number of vertices and m be the number of edges in the graph. Let $EO(k)$ denote "the expected values is $O(k)$." Table 1 summarizes the features of each algorithm. The analysis for Algorithm A assuming implementation on a EREW P-RAM is the same as for Algorithm B. The column labelled "Random bits" indicates the number of unbiased random bits consumed during the execution of the algorithm.

1. History of the MIS problem. The obvious sequential algorithm for the MIS problem can be simply stated as: Initialize I to the empty set; for $i = 1, \cdots, n$, if vertex i is not adjacent to any vertex in I then add vertex i to I. The MIS output by this algorithm is called the lexicographically first maximal independent set (LFMIS). Valiant [Va] noted that the MIS problem, which has such an easy sequential algorithm, may be one of the problems for which there is no fast parallel algorithm. Cook [Co] strengthened this belief by proving that outputting the LFMIS is NC^1-complete for P. This gave strong evidence that there is no NC algorithm which outputs the LFMIS. Thus, it became clear that either there was no fast parallel algorithm for the MIS problem or else the fast parallel algorithm had to have a completely different design than the sequential algorithm.

TABLE 1

Algorithm	P-RAM Type	Processors	Time	Random bits
A	CRCW	$O(m)$	$EO(\log n)$	$EO(n(\log n)^2$
B	EREW	$O(m)$	$EO((\log n)^2)$	$EO(n \log n)$
C	EREW	$O(m)$	$EO((\log n)^2)$	$EO((\log n)^2)$
D	EREW	$O(n^2 \cdot m)$	$O((\log n)^2)$	none

Surprisingly, Karp and Wigderson [KW] did develop a fast parallel algorithm for the MIS problem. They presented a randomized algorithm with expected running time $O((\log n)^4)$ using $O(n^2)$ processors, and a deterministic algorithm with running time $O((\log n)^4)$ using $O(n^3/(\log n)^3)$ processors on a EREW P-RAM, also establishing the result that the MIS problem is in NC^4. This paper describes algorithms which are substantially simpler than their algorithm, and establishes that the MIS problem is in NC^2.

Alon, Babai and Itai [ABI] independently found a Monte Carlo algorithm for the MIS problem, which is similar to Algorithm B, shortly after the present author found Algorithms B, C and D [Lu]. Their algorithm can be implemented on a EREW P-RAM with the same efficiency as is shown for Algorithm B in Table 1. They have an implementation of their algorithm on a CRCW P-RAM where the running time is $EO(\log n)$ using $O(\Delta m)$ processors, where Δ is the maximum degree of any vertex in the graph. Algorithm A was developed after seeing a preliminary version of [ABI], inspired by the CRCW P-RAM parallel computation model they consider. Algorithm A is the simplest Monte Carlo algorithm for the MIS problem. It has a more processor efficient implementation then the algorithm described in [ABI], using $O(m)$ processors versus $O(\Delta m)$.

2. Applications of the MIS algorithm. A growing number of parallel algorithms use the MIS algorithm as a subroutine. Karp and Wigderson [KW] gave logspace reductions from the *Maximal Set Packing* and the *Maximal Matching* problems to the MIS problem, and a nondeterministic logspace reduction from the 2-*Satisfiability* problem to the MIS problem. In this paper, it is shown that there is a logspace reduction from the *Maximal Coloring* problem (§ 6.1) to the MIS problem. Thus, using the results of this paper, all of these problems are now known to be in NC^2. However, Cook and Luby [CL] previously showed that 2-*Satisfiability* is in NC^2.

Lev [Le], previous to [KW], designed an algorithm for the *Maximal Matching* problem with running time $O((\log n)^4)$ on a P-RAM (and also established that the problem is in NC^5). Subsequently, Israeli and Shiloach [IS] found an algorithm for *Maximal Matching*, implemented on a CRCW P-RAM, where the running time is $O((\log n)^3)$. More recently and independently of this paper, Israeli and Itai [II] found a Monte Carlo algorithm for the *Maximal Matching* problem. The running time of their algorithm implemented on a EREW P-RAM is $EO((\log n)^2)$ and implemented on a CRCW P-RAM is $EO(\log n)$. The reduction of [KW] from the *Maximal Matching* problem to the MIS problem together with the results in this paper establishes the stronger results that there is a deterministic algorithm for *Maximal Matching* which can be implemented on a EREW P-RAM with running time $O((\log n)^2)$, and that the *Maximal Matching* problem is in NC^2.

Karloff [Kf1] uses the MIS algorithm as a subroutine for the *Odd Set Cover* problem. This algorithm can be used to convert the Monte Carlo algorithm for *Maximum Matching* [KUW] into a Las Vegas algorithm. Also, [KSS] use both the MIS algorithm and the Maximal Coloring algorithm to find a Δ vertex coloring of a graph when Δ is $O((\log n)^c)$ for some constant c, where Δ is the maximum degree of any vertex in the graph.

3. Monte Carlo MIS algorithms.

3.1. A high level description of the algorithm. All of the MIS algorithms in this paper adhere to the outline described below. The input to the MIS algorithm is an undirected graph $G=(V, E)$. The output is a maximal independent set $I \subseteq V$. Let $G'=(V', E')$ be a subgraph of G. For all $W \subseteq V'$, define the neighborhood of W to be $N(W)=\{i \in V': \exists j \in W, (i, j) \in E'\}$.

begin
$I \leftarrow \varnothing$
$G'=(V', E') \leftarrow G=(V, E)$
while $G' \neq \varnothing$ **do**
  begin
    select a set $I' \subseteq V'$ which is independent in G'

$I \leftarrow I \cup I'$
$Y \leftarrow I' \cup N(I')$
$G' = (V', E')$ is the induced subgraph on $V' - Y$.
end
end

It is easy to show that I is a maximal independent set in G at the termination of the algorithm. The crux of the algorithm is the design of the **select step**, which must satisfy two properties:

1. The **select step** can be implemented on a P-RAM so that its execution time is very small.

2. The number of executions of the body of the **while** loop before G' is empty is very small.

For each MIS algorithm presented in this paper only the description of the **select step** will be given.

The analysis of the algorithms assumes that $n = |V|$ and $m = |E|$ are stored in the first two common memory locations of the P-RAM, and that the edge descriptions follow in consecutive memory locations. The body of the **while** loop, excluding the **select step**, can be implemented on a CRCW P-RAM using $O(m)$ processors, where each execution takes time $O(1)$. The same portion of the loop can be implemented on a EREW P-RAM using $O(m)$ processors, where each execution takes time $O(\log n)$. Here and throughout the rest of this paper the low level implementation details are omitted.

3.2. Monte Carlo Algorithm A description. The simplest Monte Carlo algorithm for the MIS problem is described in this section. Without loss of generality, assume that $V' = \{1, \cdots, n'\}$. For all $i \in V'$, define adj $(i) = \{j \in V' | (i, j) \in E'\}$. A very high level description of the **select step** is:

1. Choose a random reordering (permutation) π of V',
2. $I' \leftarrow \{i \in V' | \pi(i) < \min \{\pi(j) | j \in \text{adj } (i)\}\}$.

Here is a more detailed description of the **select step.** Define an algorithm to be executed by each vertex $i \in V'$ as follows.

ALGVERTEX (i)
begin
$\pi(i) \rightarrow$ a number randomly chosen from $\{1, \cdots, n^4\}$.
end

Define an algorithm to be executed by each edge $(i, j) \in E'$ as follows.

ALGEDGE (i, j)
begin
if $\pi(i) \geqq \pi(j)$ **then** $I' \leftarrow I' - \{i\}$
else $I' \leftarrow I' - \{j\}$
end

The **select step** is:

begin
In **parallel**, $\forall i \in V'$, **execute** *ALGVERTEX* (i)
$I' \leftarrow V'$
In **parallel**, $\forall (i, j) \in E'$, **execute** *ALGEDGE* (i, j)
end

The random choices of the values of $\pi(i)$ in *ALGVERTEX* (i) are mutually independent. This is not literally an implementation of the high level description of the algorithm because there is some chance that for some $(i,j) \in E'$, $\pi(i) = \pi(j)$. However, since each pair of vertices receives the same π value with probability $1/n^4$, and there are at most $\binom{n}{2}$ pairs of vertices from V', π is a random reordering of the vertices with probability at least $1 - 1/2n^2$.

The **select step** can be implemented on a CRCW P-RAM using $O(m)$ processors, where each execution takes time $O(1)$ and consumes $O(n \log n)$ random bits. An implementation on a EREW P-RAM uses $O(m)$ processors, where each execution takes time $O(\log n)$. In § 3.4, the number of executions of the **while** loop before termination of the algorithm is proven to be $EO(\log n)$. Thus, an implementation of the algorithm on a CRCW (EREW) P-RAM uses $O(m)$ processors, where the total execution time is $EO(\log n)$ $(EO((\log n)^2))$.

3.3. Monte Carlo Algorithm B description. In this section a Monte Carlo algorithm for the MIS problem is described which is slightly more complicated than Algorithm A. Algorithm B is the basis for the deterministic algorithm presented in the following sections.

The notation of § 3.2 is retained. For each $i \in V'$, define $d(i)$, the degree of i, to be $|\text{adj}(i)|$. Define a collection of mutually independent $\{0, 1\}$ valued random variables $\{\text{coin}(i) \mid i \in V'\}$ such that if $d(i) \geqq 1$ then coin (i) takes on value 1 with probability $1/2d(i)$ and if $d(i) = 0$ then coin (i) is always 1. The **select step** is:

begin
  In parallel, $\forall i \in V'$, **compute** $d(i)$
  $X \leftarrow \varnothing$
  In parallel, $\forall i \in V'$ {**choice step**}
    randomly choose a value for coin (i)
    if coin $(i) = 1$ **then** $X \leftarrow X \cup \{i\}$
  In parallel, $\forall (i,j) \in E'$
    if $i \in X$ and $j \in X$ **then**
      if $d(i) \leqq d(j)$ **then** $I' \leftarrow I' - \{i\}$
      else $I' \leftarrow I' - \{j\}$
end

The **select step** can be implemented on a EREW P-RAM using $O(m)$ processors, where each execution takes time $O(\log n)$ and consumes $EO(n)$ random bits. In § 3.4, the number of executions of the **while** loop before termination of the algorithm is proven to be $EO(\log n)$. Thus, an implementation of the algorithm on a EREW P-RAM uses $O(m)$ processors, where the total execution time is $O((\log n)^2)$.

3.4. Analysis of Algorithms A and B. Let t_A and t_B be the number of executions of the body of the **while** loop before $G' = \varnothing$ for Algorithms A and B, respectively. In this section it is shown that $E(t_A) = O(\log n)$ and $E(t_B) = O(\log n)$. The proof is very similar for both algorithms. It is shown that on the average each execution of the body of the **while** loop eliminates a constant fraction of the edges in E'.

Let Y_k^A and Y_k^B be the number of edges in E' before the kth execution of the body of the **while** loop for Algorithms A and B, respectively. The number of edges eliminated from E' due to the kth execution of the body of the **while** loop for A and B is $Y_k^A - Y_{k+1}^A$ and $Y_k^B - Y_{k+1}^B$, respectively.

THEOREM 1.
(1) $E[Y_k^A - Y_{k+1}^A] \geqq \frac{1}{8} \cdot Y_k^A - \frac{1}{16}$.
(2) $E[Y_k^B - Y_{k+1}^B] \geqq \frac{1}{8} \cdot Y_k^B$.

From Theorem 1 it is easily shown that $E(t_A) = O(\log n)$ and that $E(t_B) = O(\log n)$. One can make even stronger claims. For example,

$$\Pr[Y_{k+1}^B \leqq \tfrac{15}{16} Y_k^B] \geqq \tfrac{1}{15}.$$

Thus, $\Pr[t_B \geqq 700 \log n] \leqq 2n^{-2000}$. This can be shown using an inequality due to Bernstein which is cited in Rényi [R, p. 387]. From this it is easy to see that the bounds on the running time and on the number of random bits used by Algorithms A and B hold with high probability. These details are omitted from this paper.

For all $i \in V'$ such that $d(i) \geqq 1$, let

$$\text{sum}(i) = \sum_{j \in \text{adj}(i)} \frac{1}{d(j)}.$$

Proof of Theorem 1. Let $G' = (V', E')$ be the graph before the kth execution of the body of the **while** loop. The edges eliminated due to the kth execution of the body of the **while** loop are the edges with at least one endpoint in the set $I' \cup N(I')$, i.e., each edge (i, j) is eliminated either because $i \in I' \cup N(I')$ or because $j \in I' \cup N(I')$. Thus,

$$E[Y_k^B - Y_{k+1}^B] \geqq \frac{1}{2} \cdot \sum_{i \in V'} d(i) \cdot \Pr[i \in I' \cup N(I')]$$

$$\geqq \frac{1}{2} \cdot \sum_{i \in V'} d(i) \cdot \Pr[i \in N(I')].$$

The remaining portion of the proofs for part (1) and part (2) of Theorem 1 are based upon Lemmas A and B, respectively. Here, only the proof for part (2) is given. (The proof for part (1) is a consequence since Lemma A is strictly stronger than Lemma B except for the $1 - 1/2n^2$ multiplicative factor.) Lemma B states that $\Pr[i \in N(I')] \geqq \frac{1}{4} \cdot \min\{\text{sum}(i)/2, 1\}$. Thus,

$$E[Y_k^B - Y_{k+1}^B] \geqq \frac{1}{8} \cdot \left(\frac{1}{2} \cdot \sum_{\substack{i \in V' \\ \text{sum}(i) \leqq 2}} d(i) \cdot \text{sum}(i) + \sum_{\substack{i \in V' \\ \text{sum}(i) > 2}} d(i) \right)$$

$$\geqq \frac{1}{8} \cdot \left(\sum_{\substack{i \in V' \\ \text{sum}(i) \leqq 2}} \sum_{j \in \text{adj}(i)} \frac{d(i)}{2 \cdot d(j)} + \sum_{\substack{i \in V' \\ \text{sum}(i) > 2}} \sum_{j \in \text{adj}(i)} 1 \right)$$

$$\geqq \frac{1}{8} \cdot \left(\sum_{\substack{(i,j) \in E' \\ \text{sum}(i) \leqq 2 \\ \text{sum}(j) \leqq 2}} \frac{1}{2} \cdot \left(\frac{d(i)}{d(j)} + \frac{d(j)}{d(i)} \right) + \sum_{\substack{(i,j) \in E' \\ \text{sum}(i) \leqq 2 \\ \text{sum}(j) > 2}} \left(\frac{d(i)}{2 \cdot d(j)} + 1 \right) + \sum_{\substack{(i,j) \in E' \\ \text{sum}(i) > 2 \\ \text{sum}(j) > 2}} 2 \right)$$

$$\geqq \frac{1}{8} \cdot |E'| = \frac{1}{8} \cdot Y_k^B. \qquad \square$$

Lemmas A and B are crucial to the proof of Theorem 1. Lemma A, due to Paul Beame, was proved after Lemma B and uses some of the same ideas as Lemma B. For expository reasons, it is presented first.

LEMMA A (Beame). *For Algorithm* A, $\forall i \in V'$ *such that* $d(i) \geqq 1$,

$$\Pr[i \in N(I')] \geqq [\tfrac{1}{4} \cdot \min\{\text{sum}(i), 1\}] \cdot \left(1 - \frac{1}{2n^2}\right).$$

Proof. With probability at least $1-1/2n^2$, π is a random reordering of V'. Assume that π is a random reordering of V'. $\forall j \in V'$, let E_j be the event that

$$\pi(j) < \min\{\pi(k) \mid k \in \text{adj}(j)\}$$

and let

$$p_j = \Pr[E_j] = \frac{1}{d(j)+1}.$$

Without loss of generality let

$$\text{adj}(i) = \{1, \cdots, d(i)\}$$

and let

$$p_1 \geqq \cdots \geqq p_{d(i)}.$$

Then, by the principle of inclusion-exclusion, for $1 \leqq l \leqq d(i)$,

$$\Pr[i \in N(I')] \geqq \Pr\left[\bigcup_{j=1}^{l} E_j\right] \geqq \sum_{j=1}^{l} p_j - \sum_{j=1}^{l} \sum_{k=j+1}^{l} \Pr[E_j \cap E_k].$$

For fixed j, k such that $1 \leqq j < k \leqq l$, let E_j' be the event that

$$\pi(j) < \min\{\pi(v) \mid v \in \text{adj}(j) \cup \text{adj}(k)\}$$

and let E_k' be the event that

$$\pi(k) < \min\{\pi(v) \mid v \in \text{adj}(j) \cup \text{adj}(k)\}.$$

Let

$$d(j,k) = |\text{adj}(j) \cup \text{adj}(k)|.$$

Then,

$$\Pr[E_j \cap E_k] \leqq \Pr[E_j'] \cdot \Pr[E_k \mid E_j'] + \Pr[E_k'] \cdot \Pr[E_j \mid E_k']$$

$$\leqq \frac{1}{d(j,k)+1} \cdot \left(\frac{1}{d(k)+1} + \frac{1}{d(j)+1}\right) \leqq 2 \cdot p_j \cdot p_k.$$

Let $\alpha = \sum_{j=1}^{d(i)} p_j$. By the technical lemma which follows,

$$\Pr[i \in N(I')] \geqq \tfrac{1}{2} \cdot \min\{\alpha, \tfrac{1}{2}\} \geqq \tfrac{1}{4} \cdot \min\{\text{sum}(i), 1\}$$

when π is a random reordering of V'. □

LEMMA B. *For Algorithm* B, $\forall i \in V'$ *such that* $d(i) \geqq 1$,

$$Pr[i \in N(I')] \geqq \tfrac{1}{4} \cdot \min\left\{\frac{\text{sum}(i)}{2}, 1\right\}.$$

Proof. $\forall j \in V'$, let E_j be the event that $\text{coin}(j) = 1$ and let

$$p_j = \Pr[E_j] = \frac{1}{2 \cdot d(j)}.$$

Without loss of generality let

$$\text{adj}(i) = \{1, \cdots, d(i)\}$$

and let

$$p_1 \geqq \cdots \geqq p_{d(i)}.$$

Let E_1' be the event E_1, and for $2 \leqq j \leqq d(i)$ let

$$E_j' = \left(\bigcap_{k=1}^{j-1} \neg E_k \right) \cap E_j.$$

Let

$$A_j = \bigcap_{\substack{v \in \text{adj}(j) \\ d(v) \geqq d(j)}} \neg E_v.$$

Then,

$$\Pr[i \in N(I')] \geqq \sum_{j=1}^{d(i)} \Pr[E_j'] \cdot \Pr[A_j \mid E_j'].$$

But

$$\Pr[A_j \mid E_j'] \geqq \Pr[A_j] \geqq 1 - \sum_{\substack{v \in \text{adj}(j) \\ d(v) \geqq d(j)}} p_v \geqq \frac{1}{2}$$

and

$$\sum_{j=1}^{d(i)} \Pr[E_j'] = \Pr\left[\bigcup_{j=1}^{d(i)} E_j \right].$$

For $k \neq j$, $\Pr[E_j \cap E_k] = p_j \cdot p_k$. Thus, by the principle of inclusion-exclusion, for $1 \leqq l \leqq d(i)$,

$$\Pr\left[\bigcup_{j=1}^{d(i)} E_j \right] \geqq \Pr\left[\bigcup_{j=1}^{l} E_j \right] \geqq \sum_{j=1}^{l} p_j - \sum_{j=1}^{l} \sum_{k=j+1}^{l} p_j \cdot p_k.$$

Let $\alpha = \sum_{j=1}^{d(i)} p_j$. The technical lemma which follows implies that

$$\Pr\left[\bigcup_{j=1}^{d(i)} E_j \right] \geqq \frac{1}{2} \cdot \min\{\alpha, 1\} \geqq \frac{1}{2} \cdot \min\left\{ \frac{\text{sum}(i)}{2}, 1 \right\}.$$

It follows that $\Pr[i \in N(I')] \geqq \frac{1}{4} \cdot \min\{\text{sum}(i)/2, 1\}$. □

TECHNICAL LEMMA. *Let $p_1 \geqq \cdots \geqq p_n \geqq 0$ be real-valued variables. For $1 \leqq l \leqq n$,* let

$$\alpha_l = \sum_{j=1}^{l} p_j, \quad \beta_l = \sum_{j=1}^{l} \sum_{k=j+1}^{l} p_j \cdot p_k, \quad \gamma_l = \alpha_l - c \cdot \beta_l,$$

where $c > 0$ is a constant. Then

$$\max\{\gamma_l \mid 1 \leqq l \leqq n\} \geqq \tfrac{1}{2} \cdot \min\left\{ \alpha_n, \frac{1}{c} \right\}.$$

Proof. It can be shown by induction that β_l is maximized when $p_1 = \cdots = p_n = \alpha_l / l$, and consequently $\beta_l \leqq \alpha_l^2 \cdot (l-1)/2l$. Thus,

$$\gamma_l \geqq \alpha_l \cdot \left(1 - c \cdot \alpha_l \cdot \frac{(l-1)}{2l} \right).$$

If $\alpha_n \leqq 1/c$ then $\gamma_n \geqq \alpha_n/2$. If $\alpha_1 \geqq 1/c$ then $\gamma_1 \geqq 1/c$. Otherwise, $\exists l$, $1 < l < n$ such that $\alpha_{l-1} \leqq 1/c \leqq \alpha_l \leqq 1/c \cdot l/(l-1)$. The last inequality follows because $p_1 \geqq \cdots \geqq p_n$. Then, $\gamma_l \geqq 1/2c$. □

4. Removing randomization from parallel algorithms.

4.1. Overview. This section outlines a general strategy for converting fast parallel Monte Carlo algorithms into fast parallel deterministic algorithms. Algorithm B is

converted into a very simple and fast deterministic algorithm using this strategy. The general strategy contains several ideas that were discovered and used by previous authors. The relationship of the general strategy to other work is discussed in § 5.

To use the general strategy to convert a specific parallel Monte Carlo algorithm into a deterministic algorithm, it must be possible to describe the Monte Carlo algorithm in the following way. All of the randomization is incorporated in one step of the algorithm called the **choice step** (which may be executed more than once during the course of the algorithm). In the **choice step,** values for random variables $X_0, \cdots, X_{n-1}$ are chosen mutually independently, such that on the average a set of values for these random variables is **good.** The algorithm can determine very quickly whether or not a set of values is **good** after the execution of the **choice step.** The analysis of the algorithm shows that if at each execution of the **choice step** a **good** set of values are chosen, then the algorithm outputs a correct output within a specified time bound.

To be able to convert a Monte Carlo algorithm which fits the above description into a deterministic algorithm, the following additional criteria are sufficient.

1. Let r be a positive integer and let $q \geqq n$ be a prime number, such that both r and q are bounded by a polynomial in n. The set of random variables $X_0, \cdots, X_{n-1}$ can be modified so that the range of random variable X_i is $R = \{R_1, \cdots, R_r\}$, such that X_i takes on value R_j with probability n_{ij}/q, where n_{ij} is an integer greater than or equal to zero and $\sum_{j=1}^{r} n_{ij} = q$.

2. The analysis of the algorithm can be modified to show that if $X_0, \cdots, X_{n-1}$ are only *pairwise independent* [Fr] then with positive probability a random set of values for $X_0, \cdots, X_{n-1}$ is **good.**

The deterministic algorithm is the same as the Monte Carlo algorithm except that the **choice step** is simulated by the following: Construct the probability space described in § 4.2 with q^2 sample points, where each sample point corresponds to a set of values of $X_0, \cdots, X_{n-1}$. In parallel, spawn q^2 copies of the algorithm, one for each sample point, and test to see which sample points are **good.** Use the set of values for $X_0, \cdots, X_{n-1}$ corresponding to a **good** sample point as the output from the **choice step.**

Since the analysis shows that with positive probability a random sample point is **good,** at least one of the sample points must be **good.** Since at each **choice step** a **good** set of values for $X_0, \cdots, X_{n-1}$ is used, the algorithm is deterministic and is guaranteed to run within the specified time bound.

This strategy for converting a Monte Carlo algorithm into a deterministic algorithm can be generalized by relaxing criterion 2 above to allow d-wise independence for any integer constant $d \geqq 1$.

4.2. Generating pairwise independent random variables. Let $X_0, \cdots, X_{n-1}$ be a set of random variables satisfying the first criterion stated in § 4.1. In this section a probability space of q^2 sample points is generated, where the random variables are pairwise independent. In contrast, the number of sample points in a probability space where the random variables are mutually independent is $\Omega(2^n)$, assuming each random variable takes on at least two values with a nonzero probability.

Consider an n by q matrix A. The values in row i of A correspond to the possible values for random variable X_i. Row i of A contains exactly n_{ij} entries equal to R_j. Let $0 \leqq x, y \leqq q-1$. The sample space is the collection of q^2 sample points

$$b^{x,y} = (b_0, \cdots, b_{n-1}),$$

where $b_i = A_{i,(x+y \cdot i) \bmod q}$ is the value of X_i at sample point $b^{x,y}$. The probability assigned to each sample point is $1/q^2$.

LEMMA 1. $\Pr[X_i = R_j] = n_{ij}/q$.

Proof. For fixed l, there are exactly q pairs of x, y such that $(x+y\cdot i) \bmod q = l$. There are n_{ij} values of l such that $A_{i,l} = R_j$. □

LEMMA 2. $\Pr[X_i = R_j \text{ and } X_{i'} = R_{j'}] = (n_{ij}\cdot n_{i'j'})/q^2$.

Proof. For fixed l and l', there is exactly one x, y pair such that $(x+y\cdot i) \bmod q = l$ and $(x+y\cdot i') \bmod q = l'$ simultaneously. □

Lemma 2 shows that the random variables $X_0, \cdots, X_{n-1}$ are pairwise independent in the probability space.

4.3. Reanalysis of Algorithm B assuming pairwise independence. In this section Algorithm B is analyzed assuming events are only pairwise independent. More specifically, assume that the collection of random variables $\{\text{coin}(i) \mid i \in V'\}$ are only pairwise independent. $\forall i \in V'$, let E_i be the event that coin $(i) = 1$ and let

$$p_i = \Pr[E_i] = \frac{1}{2d(i)}.$$

The analogue of Lemma B is the following.

LEMMA C. $\Pr[i \in N(I')] \geqq \frac{1}{8}\cdot \min\{\text{sum}(i), 1\}$.

Proof. The notation introduced in the proof of Lemma B is retained. Let $\alpha_0 = 0$ and for $1 \leqq l \leqq d(i)$, let $\alpha_l = \sum_{j=1}^{l} p_j$. As in the proof of Lemma B,

$$\Pr[i \in N(I')] \geqq \sum_{j=1}^{d(i)} \Pr[E_j'] \Pr[A_j \mid E_j'].$$

A lower bound is first derived on $\Pr[A_j \mid E_j']$. $\Pr[A_j \mid E_j'] = 1 - \Pr[\neg A_j \mid E_j']$. But,

$$\Pr[\neg A_j \mid E_j'] \leqq \sum_{\substack{v \in \text{adj}(j) \\ d(v) \geqq d(j)}} \Pr[E_v \mid E_j']$$

and

$$\Pr[E_v \mid E_j'] = \frac{\Pr[E_v \cap \neg E_1 \cap \cdots \cap \neg E_{j-1} \mid E_j]}{\Pr[\neg E_1 \cap \cdots \cap \neg E_{j-1} \mid E_j]}.$$

The numerator is less than or equal to $\Pr[E_v \mid E_j] = p_v$. The denominator is equal to

$$1 - \Pr\left[\bigcup_{l=1}^{j-1} E_l \mid E_j\right] \geqq 1 - \sum_{l=1}^{j-1} \Pr[E_l \mid E_j] = 1 - \alpha_{j-1}.$$

Thus, $\Pr[E_v \mid E_j'] \leqq p_v/(1-\alpha_{j-1})$. Consequently,

$$\Pr[\neg A_j \mid E_j'] \leqq \sum_{\substack{v \in \text{adj}(j) \\ d(v) \geqq d(j)}} \frac{p_v}{1-\alpha_{j-1}} \leqq \frac{1}{2(1-\alpha_{j-1})}.$$

Thus,

$$\Pr[A_j \mid E_j'] \geqq 1 - \frac{1}{1(1-\alpha_{j-1})} = \frac{1-2\alpha_{j-1}}{2(1-\alpha_{j-1})}.$$

Now, a lower bound is derived on $\Pr[E_j']$.

$$\begin{aligned}\Pr[E_j'] &= \Pr[E_j] \Pr[\neg E_1 \cap \cdots \cap \neg E_{j-1} \mid E_j] \\ &= p_j\left(1 - \Pr\left[\bigcup_{l=1}^{j-1} E_l \mid E_j\right]\right) \geqq p_j(1-\alpha_{j-1}).\end{aligned}$$

Thus, for $1 \leqq l \leqq d(v)$ and $\alpha_l < \frac{1}{2}$,

$$\Pr[i \in N(I')] \geqq \sum_{j=1}^{l} \frac{p_j(1-2\alpha_{j-1})}{2} = \frac{1}{2} \cdot \left(\sum_{j=1}^{l} p_j - 2 \cdot \sum_{j=1}^{l} \sum_{k=j+1}^{l} p_j \cdot p_k \right).$$

By the technical lemma,

$$\Pr[i \in N(i')] \geqq \tfrac{1}{4} \cdot \min\{\alpha_{d(i)}, \tfrac{1}{2}\} \geqq \tfrac{1}{8} \cdot \min\{\text{sum}(i), 1\}. \qquad \square$$

The analogue to Theorem 1 for Algorithm B when the random variables are only pairwise independent is:

THEOREM 2. *$E[Y_k^B - Y_{k+1}^B] \geqq \frac{1}{16} Y_k^B$ when the random variables $\{\text{coin}(i) \mid i \in V'\}$ are only pairwise independent.*

Proof. Use Lemma C in place of Lemma B in the proof of Theorem 1. $\square$

This analysis shows that Algorithm B almost fulfills the criteria stated in § 4.1. The range of the set of random variables $\{\text{coin}(i) \mid i \in V'\}$ is $\{0, 1\}$. A **good** set of values for $\{\text{coin}(i) \mid i \in V'\}$ is a set of values such that when they are used in the **choice step** at least $\frac{1}{16}$ of the edges in E' are eliminated. Theorem 2 implies that in any probability space where the random variables $\{\text{coin}(i) \mid i \in V'\}$ are pairwise independent there is at least one sample point which is **good**. To determine whether or not a set of values for $\{\text{coin}(i) \mid i \in V'\}$ is **good**, the steps in the body of the **while** loop are executed using these values and the number of edges eliminated is computed. The only requirement that is not fulfilled is criterion 1.

4.4. Algorithms C and D for the MIS problem. In this section, the deterministic implementation of Algorithm B is presented. The only missing requirement is that the set of random variables $\{\text{coin}(i) \mid i \in V'\}$ as defined in § 3.3 does not fulfill criterion 1 of § 4.1. To fulfill this criterion there are two changes: (1) the probabilities assigned to the random variables $\{\text{coin}(i) \mid i \in V'\}$ are modified slightly, and (2) the algorithm is modified slightly.

The probabilities are modified as follows. Let q be a prime number between n and $2n$. The probability that $\text{coin}(i) = 1$ is $p_i' = \lfloor p_i \cdot q \rfloor / q$, where $p_i = 1/2d(i)$ is the previous probability.

Here is a description of the **choice step** for Algorithm C.

Case 1. There is a vertex $i \in V'$ such that $d(i) \geqq n/16$. The algorithm sets I' to $\{i\}$.

Case 2. $\forall i \in V'$, $d(i) < n/16$. The algorithm constructs the probability space described in § 4.2 and randomly chooses a sample point. The algorithm uses the values for $\{\text{coin}(i) \mid i \in V'\}$ corresponding to the sample point to choose I' as before.

Here is the analysis for Algorithm C. Case 1 can occur at most 16 times, because each time it occurs at least $\frac{1}{16}$ of the vertices in the original graph G are eliminated. In Case 2, $\forall i \in V'$, $d(i) < n/16$, which implies $q/2d(i) \geqq n/2d(i) > 8$ and consequently $p_i' \geqq 8/q$. But this implies that $\lfloor p_i \cdot q \rfloor / 8 \geqq 1$. Thus, since $(\lfloor p_i \cdot q \rfloor + 1)/q \geqq p_i$, $\frac{8}{9} p_i \leqq p_i' \leqq p_i$.

LEMMA D. *Let the set of random variables $\{\text{coin}(i) \mid i \in V'\}$ be pairwise independent such that $\Pr[\text{coin}(i) = 1] = p_i'$ and such that $\forall i \in V'$, $d(i) < n/16$. Then $\Pr[i \in N(I')] \geqq \frac{1}{9} \min\{\text{sum}(i), 1\}$.*

Proof. The same proof as for Lemma C, except $\frac{8}{9} p_i$ is used as a lower bound and p_i is used as an upper bound on $\Pr[\text{coin}(i) = 1]$. $\square$

THEOREM 3. *Consider Algorithm* B, *where the random variables $\{\text{coin}(i) \mid i \in V'\}$ are as described in this section and where $\forall i \in V'$, $d(i) < n/16$. Then $E[Y_k^B - Y_{k+1}^B] \geqq \frac{1}{18} Y_k^B$.*

Proof. Use Lemma D in place of Lemma B in the proof of Theorem 1. □

Thus, if Case 2 is true then Algorithm C eliminates at least $\frac{1}{18}$ of the edges in E' on the average. From this it is easy to see that the bounds on the running time and on the number of random bits used by Algorithm C hold with high probability.

The code for Algorithm C follows. This algorithm is very practical because it is simple, fast, uses a small number of processors and a small number of random bits. Each execution of the body of the **while** loop can be implemented in $O(\log n)$ time on a EREW P-RAM using $O(m)$ processors, where the expected number of random bits used is $O(\log n)$. The expected number of executions of the **while** loop before termination of the algorithm is $O(\log n)$. Thus, the expected running time of the entire algorithm on a EREW P-RAM is $O((\log n)^2)$ using $O(m)$ processors, where the expected number of random bits used is $O((\log n)^2)$.

Algorithm D is the same as Algorithm C, except that the **choice step** is executed by testing in parallel all q^2 sets of values for $\{\text{coin}(i) \mid i \in V'\}$ and using the set of values which maximizes the number of edges eliminated from G'. Theorem 3 shows that the best set will eliminate at least $\frac{1}{18}$ of the edges in the graph G'. This algorithm can be implemented on a EREW P-RAM with $O(mn^2)$ processors with running time $O((\log n)^2)$. The number of executions of the **while** loop is guaranteed to be at most

$$\frac{\log(n^2)}{\log(18/17)} + 16 \leqq 25 \cdot \log n + 16.$$

begin
  $I \leftarrow \varnothing$
  compute $n = |V|$
  compute a prime q such that $n \leqq q \leqq 2n$
  $G' = (V', E') \leftarrow G = (V, E)$
  while $G' \neq \varnothing$ **do**
  begin
    In parallel, $\forall i \in V'$, compute $d(i)$
    In parallel, $\forall i \in V'$
      if $d(i) = 0$ **then** add i to I and delete i from V'.
    find $i \in V'$ such that $d(i)$ is maximum
    if $d(i) \geqq n/16$ **then** add i to I and let G' be the
      graph induced on the vertices $V' - (\{i\} \cup N(\{i\}))$
    else $(\forall i \in V', d(i) < n/16)$
    begin
(choice) **randomly choose** x and y such that $0 \leqq x, y \leqq q-1$
      $X \leftarrow \varnothing$
      In parallel, $\forall i \in V'$,
      begin
        compute $n(i) = \lfloor q/2d(i) \rfloor$
        compute $l(i) = (x + y \cdot i) \bmod q$
        if $l(i) \leqq n(i)$ **then** add i to X
      end
      $I' \leftarrow X$
      In parallel, $\forall i \in X, j \in X$,
        if $(i, j) \in E'$ **then**
          if $d(i) \leqq d(j)$ **then** $I' \leftarrow I' - \{i\}$
          else $I' \leftarrow I' - \{j\}$

$I \leftarrow I \cup I'$
$Y \leftarrow I' \cup N(I')$
$G' = (V', E')$ is the induced subgraph on $V' - Y$.
end
end
end

5. A history of the ideas used in the general strategy. Some of the ideas outlined in § 4 have been developed in other contexts. The purpose of this section is threefold:

1. Give proper credit to previous researchers.
2. Indicate the innovations in § 4 by contrasting this work with previous work.
3. Highlight techniques common to a large body of seemingly unrelated work.

The last point may be the most important: fragments of these techniques have been used successfully many times in different contexts by many authors. By making the connections between these techniques explicit, a more unified approach to problem solving using these techniques may evolve.

There are many randomizing algorithms in the computer science literature. These algorithms generally use random input bits to generate in a straightforward way a number of mutually independent random variables. One of the general strategies used in this paper consists of two parts:

(1) Prove that the necessary probability theorems still hold when the random variables are pairwise independent instead of mutually independent (the proof itself is not always straightforward, e.g. the proof in this paper).
(2) Construct a sample space with special structural properties where the random variables are pairwise independent (constructing the same space with the necessary structural properties is usually quite easy, given the plethora of previous research in this area which is discussed in §§ 5.1 and 5.2).

The reason for doing this is that the special structural properties of the sample space can be used to great advantage. Previous papers which exploit this strategy are [ACGS], [CW], [Si], [Sh]. In this paper, the special structural property of the sample space is that all of the sample points can be constructed efficiently in parallel. This property is used to deterministically find a good sample point quickly in parallel.

5.1. Generating *d*-wise independent random variables. The construction given in § 4.2 was subsequently generalized to a probability distribution where the random variables $X_0, \cdots, X_{n-1}$ are d-wise independent, where there are q^d sample points for any integer constant $d \geqq 1$. Credit for this generalization goes to Noga Alon [Al], Richard Anderson [An] and Paul Beame [Be], each of whom found it independently.

Let $0 \leqq x_0, \cdots, x_{d-1} \leqq q-1$. The sample space is the collection of q^d sample points

$$b^{x_0, \cdots, x_{d-1}} = (b_0, \cdots, b_{n-1}),$$

where b_i is the $(i, (\sum_{j=0}^{d-1} x_j \cdot i^j) \bmod q)$ entry in matrix A. The probability assigned to each sample point is $1/q^d$. It is possible to show that the random variables $X_0, \cdots, X_{n-1}$ are d-wise independent and that $\Pr[X_i = R_j] = n_{ij}/q$. [CFGHRS] proves that any probability space with n d-wise independent random variables must contain at least $n^{d/2}$ sample points.

5.2. History of *d*-wise independence constructions. The first person to give an example of random variables which are pairwise independent but not mutually independent is Bernstein in 1945 (see [Fr, p. 126]). His example consists of three $\{0, 1\}$ valued random variables. Lancaster [La] generalizes this example to $n-1$ pairwise independent

variables on n sample points, giving constructions involving the Hadamard matrix and Latin squares. Joffe [Jo1], [Jo2] gives a construction for generating d-wise independent random variables which is exactly the same as the constructions given in §§ 4.2 and 5.1, except as noted below. O'Brien [OB] discusses generating pairwise independent random variables with additional properties. Incomplete Block Designs (see [Ha]) are another way to generate pairwise independent random variables.

All of the work mentioned in the preceding paragraph concentrates solely on constructions of pairwise (d-wise) independent random variables such that each value of each random variable is equally likely. The constructions given in §§ 4.2 and 5.1 generalizes this slightly: the random variables are allowed to take on different values with different probabilities.The freedom to specify nonequal probabilities for different values of the random variables is essential to the general strategy described in § 4.

5.3. Techniques for analyzing d-wise independent random variables. One of the important features of the general strategy given in § 4 is the separation of the analysis of d-wise independent random variables from the construction of the probability space. The modularity of this approach allows the researcher to study the properties of d-wise independent random variables without worrying about the details of the probability space construction. A short list of techniques for analyzing d-wise independent random variables is given here.

1. Markov's inequality holds for random variables assuming no independence.
2. Chebyshev's inequality holds for pairwise independent random variables.
3. Bonferroni bounds and stronger similar bounds [CE], [Ga] are useful. These types of bounds are used extensively in the proof of the technical lemma, which is similar to a lemma contained in [CE].
4. Use of the d-wise independence in a natural way to divide a complicated expression into simpler expressions. For example, $E[A|B]=E[A]$ if A and B are pairwise independent random variables.
5. The pigeon-hole principle. For example, if it can be proved that $E[X]>c$ assuming d-wise independent random variables, where X is a real-valued random variable which is possibly a function of several of the original random variables and c is a constant, then there must be a sample point where $X>c$.

The proofs of Lemmas A, B, C and D use a rich mixture of these techniques.

5.4. Other applications of d-wise independent random variables. In this section, some of the uses of d-wise independent random variables are highlighted. A complete history is beyond the scope of this paper.

Carter and Wegman [CW] use constructions for pairwise independent random variables to generate hashing functions with certain desirable properties. One of the constructions they use is very similar to that discussed in [Jo2], but not quite as general as the construction given in § 5.1. They are perhaps the first to separate the analysis of the algorithm from the construction used to generate the probability space. The techniques they use to analyze the algorithm include techniques 1, 3 and 4 from the previous section. The work of [CW] is used by Sipser to simplify the proof in [Si] that BPP is at the second level of the polynomial time hierarchy. Similar ideas appear in [St].

Alexi, Chor, Goldreich and Schnorr [ACGS] prove that RSA/Rabin bits are secure. They use the same construction as given in [Jo2] to generate pairwise independent random variables and they use Chebyshev's inequality in their analysis. Chor and Goldreich [CG] have a simple solution to a problem originally introduced and solved in a more complicated way by Karp and Pippenger [KP]. The solution of [CG] uses the same construction as given in [Jo2] to generate pairwise independent random

variables and Chebyshev's inequality in the analysis. Shamir [Sh] uses the construction given in [Jo2] to distribute a secret among n people in such a way that no subset of d people can determine anything about the secret, but any subset of $d+1$ people can completely determine the secret.

Karp and Wigderson [KW] are the first to introduce a special case of the strategy given in § 4. They use their strategy to convert their Monte Carlo MIS algorithm into a deterministic algorithm. They use an incomplete block design to construct a probability space with $\{0, 1\}$ valued pairwise independent random variables, where each random variable takes on value 1 with probability $\frac{1}{2}$. Their analysis of the algorithm, which is heavily dependent upon their particular construction of the probability space, is quite complicated. There is a simpler analysis of their algorithm, which uses the techniques in § 5.3, which shows that the algorithm still works well on the average if the random variables are only pairwise independent. This analysis, together with the construction given in § 4.2, where random variables can take on different values with different probabilities, gives a much simpler and faster deterministic algorithm than the one they give. However, the algorithm is still not as simple nor as fast as those presented in this paper.

6. Generalizations of the MIS problem.

6.1. The Maximal Coloring problem. The Maximal Coloring problem generalizes the MIS problem. The input to the Maximal Coloring problem is an undirected graph $G=(V, E)$ and a set of colors C_v for each vertex $v \in V$. The output is a maximal coloring. In a maximal coloring, each vertex v is either assigned a color from C_v or is not assigned a color, subject to the restrictions that no two adjacent vertices are assigned the same color and that if v is not assigned a color then each color in C_v must be assigned to at least one neighbor of v. The MIS problem is a special case of the Maximal Coloring problem where $C_v = \{\text{red}\}$ for each vertex $v \in V$. The set of vertices colored red in any maximal coloring are a MIS.

Another problem which the Maximal Coloring problem generalizes is the $\Delta+1$VC problem. The input to the $\Delta+1$VC problem is an undirected graph $G=(V, E)$. Let Δ be the maximum degree of any vertex in V, let $\Delta' = \Delta+1$ and let $C = \{c_1, \cdots, c_{\Delta'}\}$ be a set of distinct colors. The output is an assignment of a color from C to each vertex such that no two adjacent vertices are assigned the same color. The $\Delta+1$VC problem is the special case of the Maximal Coloring problem where for each vertex $v \in V$, $C_v = C$. In any Maximal Coloring each vertex will be assigned some color from C because Δ' is larger than $d(v)$. The obvious sequential algorithm for the $\Delta+1$VC problem follows: For $v = 1, \cdots, n$, vertex v is assigned the smallest indexed color from C which is not assigned to a smaller indexed adjacent vertex. One might hope to devise a fast parallel algorithm for the $\Delta+1$VC problem by emulating the sequential algorithm. However, this is unlikely since

LEMMA 3. *The problem of deciding what color vertex n is assigned by the above sequential algorithm is* NC^1 *complete for P.*

Proof. There is an easy reduction from the LFMIS problem (see § 1) to this problem. □

Thus, as was the case for the MIS problem, the coloring algorithm cannot simply emulate the sequential algorithm.

There is a logspace reduction from the Maximal Coloring problem to the MIS problem. Given a Maximal Coloring problem with input $G=(V, E)$ and color sets $\{C_v\}$, a new graph G' is formed. The vertices in G' are $V' = \{(v, c): v \in V \text{ and } c \in C_v\}$.

The edges in G' are

$$E' = \{((v, c_1), (v, c_2)) : v \in V \text{ and } c_1, c_2 \in C_v\}$$
$$\cup \{((v, c), (w, c)) : (v, w) \in E \text{ and } c \in C_v \cap C_w\}.$$

There is a one-to-one correspondence between maximal colorings in G and maximal independent sets in G'. This reduction together with the MIS algorithm shows that the Maximal Coloring problem is in NC^2.

The ΔVC problem is to color all the vertices using only Δ distinct colors. Brooks' theorem [Br] proves that all but very special graphs can be colored with Δ colors, and implicitly gives a polynomial time sequential algorithm. Karloff, Shmoys and Soroker [KSS] have found a NC parallel algorithm for the ΔVC problem when Δ is polylog in the number of vertices. Their algorithm uses the algorithm for the $\Delta+1$VC problem as a subroutine. The classification of the ΔVC problem with respect to parallel computation is still open for unrestricted Δ.

6.2. Binary coherent systems. Recently, researchers in Artificial Intelligence have been actively investigating various connectionist models of the brain [Fe], [Go], [Hi], [Ho]. Some of the basic features of the connectionist model are shared by knowledge representation schemas [Ts].

One particular model of the brain is a binary coherent system [Ho], [Hi]. The binary coherent system (BCS) problem, which is studied by Hopfield [Ho], can be formally stated as follows. The input is an undirected graph $G = (V, E)$ together with a real-valued weight w_e for each edge and a real-valued threshold t_v for each vertex. Each vertex v has a state s_v which can be either -1 or 1. The state of the system is a tuple $(s_1, \cdots, s_{|V|})$. The energy of vertex v in a system state is

$$s_v \cdot \left(t_v + \sum_{e=(v,z)\in E} w_e \cdot s_z \right).$$

The output is a system state where all vertices have energy greater than or equal to zero. The BCS problem has a polynomial time sequential algorithm if all of the weights and thresholds are input in unary. The algorithm repeats the following step until all vertices have energy greater than or equal to zero: find a vertex with negative energy and flip its state. The running time of this algorithm is slow if the system is large. Hopfield suggests a simple asynchronous parallel algorithm for this problem, but provides no formal analysis of its running time, although he does give some empirical evidence that it is fast. An open question is whether or not the BCS problem has an NC algorithm.

The MIS problem is a special case of the BCS problem, where all edge weights are -1 and for each $v \in V$, $t_v = -d(v) + 1$. Thus, Algorithm D shows that at least a special case of the BCS problem is in NC, and Baruch Awerbuch has observed that Algorithm A can be easily modified to run asynchronously in parallel.

Another natural problem which is a special case of the BCS problem is the Different Than Majority Labelling (DTML) problem. The input to this problem is an undirected graph $G = (V, E)$. The output is a label of -1 or 1 for each vertex v such that at least half of the neighbors of v have the opposite label. The DTML problem is a BCS problem where all thresholds are zero and all edge weights are -1. The DTML problem may also be viewed as a graph partition problem: partition the vertices of an undirected graph into two sets such that for each vertex v at least half of the edges out of v cross the partition. (Using the techniques developed in this paper, there is a fast parallel algorithm for an easier graph partition problem which can be stated as follows: partition

the vertices of an undirected graph into two sets such that at least half of the edges in the graph cross the partition.) Karloff [Kf2] has found a NC algorithm for this problem when the input is a cubic graph, but the general problem is still open. However, there is evidence that a different type algorithm than the MIS algorithm will have to be found.

THEOREM 4. *The problem of deciding whether there is a* DTML *for a graph such that two specified vertices receive the same label is* NP-complete. □

This theorem gives evidence that no fast algorithm for the DTML problem can permanently decide the labels of vertices in a local manner in the same way as is the case for the MIS algorithm.

7. Open problems and further work.

1. Find other Monte Carlo algorithms for which the techniques developed in this paper are applicable for converting the algorithm into a deterministic algorithm.

2. Develop probabilistic bounds on random variables that are d-wise independent.

3. Algorithm A has a local property which seems particularly well suited to distributed computing networks where the processors can only communicate with neighboring processors. Find applications for this algorithm in distributed computing protocols.

4. There is no known lower bound on the parallel complexity of the MIS problem. Either find a problem which is complete in some complexity class (like NL) and reduce it to the MIS problem, or else find a faster MIS algorithm.

Acknowledgments. I thank Paul Beame for extensive discussions which significantly simplified the analysis of the algorithms. Paul also made numerous suggestions which substantially streamlined the presentation. I thank Stephen Cook for many discussions about the MIS problem and related problems, and for his unswerving belief that there must be a faster algorithm for the MIS problem. Thanks go to both Charlie Rackoff and Gary Miller for helpful discussions about the MIS problem, and more specifically for suggesting that an analysis be done on an algorithm very similar to the algorithm described in § 3.2. I thank Allan Borodin for introducing me to the work on connectionist models. I thank both Paul Beame and Richard Anderson for pointing out some of the references discussed in § 5.

REFERENCES

[Al] N. ALON, private communication.

[An] R. ANDERSON, private communication.

[ACGS] W. ALEXI, B. CHOR, O. GOLDREICH AND C. SCHNORR, *RSA/Rabin bits are* $\frac{1}{2}+1/\text{poly}(\log N)$ *secure*, Preliminary version in Proc. 25th IEEE Symposium on Foundations of Computer Science, October 1984; this Journal, to appear.

[ABI] N. ALON, L. BABAI AND A. ITAI, *A fast and simple randomized parallel algorithm for the maximal independent set problem*, private communication.

[Be] P. BEAME, private communication.

[Br] R. L. BROOKS, *On colouring the nodes of a network*, Proc. Cambridge Philos. Soc., 37 (1941), pp. 194–197.

[CE] K. CHUNG AND P. ERDOS, *On the application of the Borel–Cantelli lemma*, Amer. Math. Soc., 72 (1952), pp. 179–186.

[CFGHRS] B. CHOR, G. FRIEDMAN, O. GOLDREICH, J. HASTAAD, S. RUDICH, AND R. SMOLANSKI, *The bit extraction problem*, preprint, accepted to Proc. 26th IEEE Symposium on Foundations of Computer Science.

[CG] B. CHOR AND O. GOLDREICH, *On the power of two-points based sampling*, technical report, MIT Laboratory for Computer Science, Cambridge, MA.

[Co] S. COOK, *A taxonomy of problems with fast parallel algorithms*, Information and Control, Vol. 64, Nos. 1–3, January/February/March 1985, Academic Press, New York.

[CL] S. COOK AND M. LUBY, *A fast parallel algorithm for finding a truth assignment to a* 2-CNF *formula*, in preparation.

[CW] J. CARTER AND M. WEGMAN, *Universal classes of hash functions*, J. Comput. System Sci., 18 (1979), pp. 143–154.

[Fe] J. A. FELDMAN, *A connectionist model of visual memory*, Parallel Models of Associative Memory, G. E. Hinton and J. A. Anderson, eds., Lawrence Erlbaum Associates, Hillsdale, NJ, 1981.

[Fr] W. FELLER, *An Introduction to Probability Theory and Its Applications*, Vol. 1, 3rd edition, John Wiley, New York, 1968.

[FW] S. FORTUNE AND J. WYLLIE, *Parallelism in random access machines*, Proc. Tenth ACM Symposium on Theory of Computing, 1978, pp. 114–118.

[Ga] J. GALAMBOS, *Bonferroni inequalities*, Ann. Probab., 5 (1977), pp. 577–581.

[Go] L. M. GOLDSCHLAGER, *A computational theory of higher brain function*, Technical Report, Stanford Univ. Stanford, CA, April 1984.

[Ha] M. HALL, Jr., *Combinatorial Theory*, Blaisdell, Waltham, MA, 1967.

[Ho] J. J. HOPFIELD, *Neural networks and physical system with emergent collective computational abilities*, Proc. National Acad. Sci., 79 (1982), pp. 2554–2558.

[Hi] G. E. HINTON, T. SEJNOWSKI AND D. ACKLEY, *Boltzmann machines: constraint satisfaction networks that learn*, Technical Report CMU-CS-84-119, Carnegie-Mellon Univ., Pittsburgh, PA.

[II] A. ISRAELI AND A. ITAI, *A fast and simple randomized parallel algorithm for maximal matching*, Computer Science Dept., Technion, Haifa, Israel, 1984.

[IS] A. ISRAELI AND Y. SHILOACH, *An improved maximal matching parallel algorithm*, Technical Report 333, Computer Science Dept., Technion, Haifa, Israel, 1984.

[Jo1] A. JOFFE, *On a sequence of almost deterministic pairwise independent random variables*, Proc. Amer. Math. Soc., 29 (1971), pp. 381–382.

[Jo2] ———, *On a set of almost deterministic k-independent random variables*, Ann. Probab., 2 (1974), pp. 161–162.

[Kf1] H. KARLOFF, *Randomized parallel algorithm for the odd-set cover problem*, preprint.

[Kf2] ———, private communication.

[KP] R. KARP AND N. PIPPENGER, *A time-randomness tradeoff*, presented at the AMS Conference on Probabilistic Computational Complexity, Durham, NH, 1982.

[KSS] H. KARLOFF, D. SHMOYS AND D. SOROKER, *Efficient parallel algorithms for graph coloring and partitioning problems*, preprint.

[KW] R. M. KARP AND A. WIGDERSON, *A fast parallel algorithm for the maximal independent set problem*, Proc. 16th ACM Symposium on Theory of Computing, 1984, pp. 266–272.

[KUW] R. M. KARP, E. UPFAL AND A. WIGDERSON, *Constructing a perfect matching is in random* NC, Proc. 17th ACM Symposium on Theory of Computing, 1985, pp. 22–32.

[La] H. LANCASTER, *Pairwise statistical independence*, Ann. Math. Statist., 36 (1965), pp. 1313–1317.

[Le] G. LEV, *Size bounds and parallel algorithms for networks*, Report CST-8-80, Dept. Computer Science, Univ. Edinburgh, 1980.

[Lu] M. LUBY, *A simple parallel algorithm for the maximal independent set problem*, Proc. 17th ACM Symposium on Theory of Computing, 1985, pp. 1–10.

[OB] G. O'BRIEN, *Pairwise independent random variables*, Ann. Probab., 8 (1980), pp. 170–175.

[R] A. RÉNYI, *Probability Theory*, North-Holland, Amsterdam, 1970.

[Si] M. SIPSER, *A complexity theoretic approach to randomness*, Proc. 15th ACM Symposium on Theory of Computing, 1983, pp. 330–335.

[Sh] A. SHAMIR, *How to share a secret*, Comm. ACM, 22 (1979), pp. 612–613.

[St] L. STOCKMEYER, *The complexity of approximate counting*, Proc. 15th ACM Symposium on Theory of Computing, 1983, pp. 118–126.

[Ts] J. TSOTSOS, *Representational axes and temporal cooperative processes*, Technical Report RBCV-84-2, Univ. Toronto, Toronto, Ontario, April 1984.

[Va] L. G. VALIANT, *Parallel computation*, Proc. 7th IBM Symposium on Mathematical Foundations of Computer Science, 1982.