A Simple Parallel Algorithm for the Maximal Independent Set Problem

Michael Luby
Department of Computer Science
University of Toronto
Toronto, Canada M5S 1A4

ABSTRACT

Simple parallel algorithms for the maximal independent set (MIS) problem are presented. The first algorithm is a Monte Carlo algorithm with a very local property. The local property of this algorithm may make it a useful protocol design tool in distributed computing environments and artificial intelligence. One of the main contributions of this paper is the development of powerful and general techniques for converting Monte Carlo algorithms into deterministic algorithms. These techniques are used to convert the Monte Carlo algorithm for the MIS problem into a simple deterministic algorithm with the same parallel running time.

0. 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. The first algorithm (section 3) is a Monte Carlo algorithm with a very local property: each vertex is randomly chosen to be added to the independent set based only on information about adjacent vertices in the graph. The local properties of this algorithm may make it a useful protocol design tool in distributed computing environments and artificial intelligence (sections 8 and 9).

One of the main contributions of this paper is the development of powerful and general techniques for converting Monte Carlo algorithms into deterministic algorithms. The random choices made in the first algorithm are mutually independent. A more sophisticated analysis of the algorithm shows that the random choices need only be *pairwise independent* [Fr] (section 4). A general technique is developed which converts a probability space with a large sample space where events are mutually independent to a new probability space with a very small sample space where events are pairwise independent (section 5). The new sample space contains $O(n^2)$ points. The second algorithm, which randomly samples from the new sample space, uses a very small number of random bits to choose each sample point (section 6). The third algorithm in parallel samples all points in the sample space and uses the best sample point at each step (section 6). This algorithm is deterministic.

For purposes of analysis the EREW P-RAM parallel computer is used, in which concurrent reads or writes to the same memory location are disallowed. The third algorithm can be modified to a uniform circuit family, where the circuit which accepts inputs of length n has depth $O((\log n)^2)$ and polynomial in n gates. This establishes that the MIS problem is in NC^2 (see [Co] for a discussion of the complexity class NC). The following table summarizes the features of each algorithm, where $EO(p)$ is used to denote *the expected value is* $O(p)$.

Algorithm	Processors	Time	Random Bits
1	$O(m)$	$EO((\log n)^2)$	$EO(n\log n)$
2	$O(m)$	$EO((\log n)^2)$	$EO((\log n)^2)$
3	$O(n^2 \cdot m)$	$O((\log n)^2)$	none

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 $v = 1, ..., n$, if vertex v is not adjacent to any vertex in I then add vertex v 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.

Surprisingly, Karp and Wigderson [KW] did develop a fast parallel algorithm for the MIS problem. They present 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 $O\left(\frac{n^3}{(log n)^3}\right)$ 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 shortly after this work was completed which is very similar to the Monte Carlo algorithm presented in section 3. The running time of their algorithm is the same as the running time of the Monte Carlo algorithm presented in section 3 on a EREW P-RAM, but on the more powerful CRCW P-RAM they have an implementation of their algorithm with a running time of $EO(log n)$.

2. Applications of the MIS Algorithm

A growing number of parallel algorithms use the MIS algorithm as a subroutine. Karp and Wigderson [KW] gave NC^1 reductions from the *Maximal Set Packing* and the *Maximal Matching* problems to the MIS problem, and an NC^2 reduction from the *2-Satisfiability* problem to the MIS problem. In this paper, it is shown that there is a NC^1 reduction from the *Maximal Coloring* problem (section 7) 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], had 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). Israeli and Shiloach [IS] have an algorithm for *Maximal Matching*, using a more powerful CRCW P-RAM, where the running time is $O((log n)^3)$. Israeli and Itai [II] have a Monte Carlo algorithm for *Maximal Matching*, using the more powerful CRCW P-RAM, where the expected running time is $O(log n)$. The results in this paper improve upon all of these results.

Recently, Karloff [Kf1] used 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 small, where Δ is the maximum degree of any vertex in the graph.

3. Monte Carlo MIS Algorithm

3.1 Algorithm Description

In this section a simple Monte Carlo algorithm is presented for the MIS problem. The input to the MIS algorithm is an undirected graph $G=(V,E)$. The output is a maximal independent set $I \subseteq V$. For all $W \subseteq V$, let $N(W)=\{w \in V : \exists v \in W, (v,w) \in E\}$. Let $d(v)$ be the degree of vertex v with respect to the graph G' in the description below.

It is easy to show that I is a maximal independent set in G at the termination of the algorithm. 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 $O(n)$. In the next section the expected number of executions of the **while** loop before termination of the algorithm is proven to be $O(log n)$. 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 $O(n log n)$.

```
begin
    I ← ∅
    G′=(V′,E′) ← G=(V,E)
    while V′ ≠ ∅ do
    begin
        X ← ∅
        In parallel, for all v ∈ V′,
(*)         randomly choose to add v to X with prob. 1/(2·d(v))
            { If d(v)=0 then always add v to X }
        I′ ← X
        In parallel, for all v ∈ X, w ∈ X,
            if (v,w) ∈ E′ then
                if d(v) ≤ d(w) then I′ ← I′ − {v}
                else I′ ← I′ − {w}
        { Note that I′ is an independent set in G′ }
        I ← I ∪ I′
        Y ← I′ ∪ N(I′)
        G′=(V′,E′) is the induced subgraph on V′ − Y.
    end
end
```

3.2 Algorithm Analysis

Let m_1 be the number of edges in G' before the execution of the body of the **while** loop. Let m_2 be the random variable which is the number of edges in G' after the execution of the body of the **while** loop. Let m_3 be the random variable which is the number of edges eliminated from G' due to one execution of the **while** loop. Then, $m_2 = m_1 - m_3$. Theorem 1 establishes that $E[m_3] \geq \frac{1}{8} \cdot m_1$. Thus, $E[m_2] \leq \frac{7}{8} \cdot m_1$. From this it is easily seen that the expected number of executions of the **while** loop is $O(\log n)$ with very high probability. These details are omitted from this paper.

For all $v \in V$ let E_v be the event that v is chosen in the (*) step of the algorithm (these events are mutually independent), let $p_v = Pr[E_v] = \frac{1}{2 \cdot d(v)}$ and let $sum_v = \sum_{w \in adj(v)} p_w$. The following lemma will be useful for the proof of theorem 1.

Lemma 1 : $Pr[v \in N(I')] \geq \frac{1}{4} \cdot \min(sum_v, 1)$.

Proof : WLOG let $\{1, \cdots, d(v)\}$ be the vertices in $adj(v)$. Let E_1' be the event E_1, and for $2 \leq i \leq d(v)$, let

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

Let

$$A_i = \bigcap_{\substack{z \in adj(i) \\ d(z) \geq d(i)}} \neg E_z .$$

Then,

$$Pr[v \in N(I')] \geq \sum_{i=1}^{d(v)} Pr[E_i'] \cdot Pr[A_i \mid E_i'] .$$

But

$$Pr[A_i \mid E_i'] \geq Pr[A_i] \geq 1 - \sum_{\substack{z \in adj(i) \\ d(z) \geq d(i)}} p_z \geq \frac{1}{2} .$$

In the next section, lemma 2, it is shown that

$$\sum_{i=1}^{d(v)} Pr[E_i'] = Pr\left[\bigcup_{i=1}^{d(v)} E_i\right] \geq \frac{1}{2} \cdot \min(sum_v, 1)$$

even when the events $\{E_w\}$ are only pairwise independent. Thus, $Pr[v \in N(I')] \geq \frac{1}{4} \cdot \min(sum_v, 1)$ □

Theorem 1 : $E[m_3] \geq \frac{1}{8} \cdot m_1$

Proof : For all $W \subseteq V$, the set of edges touching W is

$$ET(W) = \{ (v,w) \in E \mid v \in W \ or \ w \in W \} .$$

The edges eliminated due to one execution of the **while** loop are the edges in the set $ET[I' \cup N(I')]$. Each edge (v,w) is in $ET[I' \cup N(I')]$ either because $v \in I' \cup N(I')$ or because $w \in I' \cup N(I')$. Thus,

$$E[m_3] \geq \frac{1}{2} \cdot \sum_{v \in V'} d(v) \cdot Pr[v \in I' \cup N(I')]$$

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

From lemma 1, $Pr[v \in N(I')] \geq \frac{1}{4} \cdot \min(sum_v, 1)$. Thus,

$$E[m_3] \geq \frac{1}{8} \cdot \left(\sum_{\substack{v \in V' \\ sum_v \leq 1}} d(v) \cdot sum_v + \sum_{\substack{v \in V' \\ sum_v > 1}} d(v) \right) \geq$$

$$\frac{1}{8} \cdot \left(\sum_{\substack{v \in V' \\ sum_v \leq 1}} \sum_{w \in adj(v)} \frac{d(v)}{2 \cdot d(w)} + \sum_{\substack{v \in V' \\ sum_v > 1}} \sum_{w \in adj(v)} 1 \right) \geq$$

$$\frac{1}{8} \cdot \sum_{\substack{(v,w) \in E' \\ sum_v \leq 1 \\ \wedge sum_w \leq 1}} \frac{1}{2} \cdot \left(\frac{d(v)}{d(w)} + \frac{d(w)}{d(v)} \right)$$

$$+ \frac{1}{8} \cdot \sum_{\substack{(v,w) \in E' \\ sum_v \leq 1 \\ \wedge sum_w > 1}} \left(\frac{d(v)}{2 \cdot d(w)} + 1 \right) + \frac{1}{8} \sum_{\substack{(v,w) \in E' \\ sum_v > 1 \\ \wedge sum_w > 1}} 2$$

$$\geq \frac{1}{8} \cdot |E'| = \frac{1}{8} \cdot m_1 \quad \square$$

4. Pairwise Independent Events

The algorithm described in the previous section assumed that the events $\{E_v\}$ were mutually independent. In this section, the events $\{E_v\}$ are only assumed to be *pairwise independent* [Fr]. The analysis shows that under this weaker assumption the algorithm still has the property that the expected number of iterations of the body of the while loop is $O(\log n)$. This surprisingly leads to a deterministic implementation of the first algorithm.

The following lemma is used in the proof of lemma 1 and completes the analysis of the first algorithm. It may also be useful as a tool in other applications.

Lemma 2 : Let $E_1, \cdots, E_n$ be events such that for $1 \le i \le n$, $Pr[E_i] = p_i$, and for $1 \le i \ne j \le n$, $Pr[E_i \cap E_j] = p_i \cdot p_j$. Let $sum = \sum_i p_i$. Then $Pr[\bigcup_i E_i] \ge \frac{1}{2} \cdot min(sum, 1)$.

Proof : WLOG that $p_1 \ge p_2 \ge \cdots p_n$. Let E_k' be the event $\bigcup_{i=1}^{k} E_i$ and let $\alpha_k = \sum_{i=1}^{k} p_i$. Then, for all $1 \le k \le n$, $Pr[E_n'] \ge Pr[E_k']$. Since the events $\{E_i\}$ are pairwise independent,

$$Pr[E_k'] \ge \alpha_k - \sum_{1 \le i < j \le k} p_i \cdot p_j \ge \alpha_k \left[1 - \frac{\alpha_k (k-1)}{2k}\right].$$

(This last inequality follows by observing that LHS of the inequality is minimum when all p_i are equal to $\frac{\alpha_k}{k}$). If $\alpha_n \le 1$ then $Pr[E_n'] \ge \frac{\alpha_n}{2} = \frac{sum}{2}$. If $\alpha_n > 1$ then let $l = \min\{k : \alpha_k \ge 1\}$. If $l = 1$ then $Pr[E_1'] \ge 1$. If $l \ge 2$ then $\alpha_{l-1} < 1 \le \alpha_l \le \frac{l}{l-1}$ (This last inequality follows because $p_1 \ge \cdots \ge p_n$). Then,

$$Pr[E_l'] \ge \alpha_l \cdot \left[1 - \frac{\alpha_l (l-1)}{2l}\right] \ge \frac{1}{2} \quad \square$$

One of the basic steps for converting Monte Carlo algorithms into deterministic algorithms using the techniques developed in this paper is the development of stronger lemmas of this type. Bonferroni and Galambos [Ga] inequalities are useful tools for deriving upper and lower bounds on the probability of an event that is described in terms of elementary events that are d-wise independent for some fixed value of d. The work of Galambos [Ga] can be used towards this end. In fact, a stronger version of lemma 2 is implicit in the work of Galambos.

The following lemma is a stronger version of lemma 1. The proof of this lemma is similar to the proof of lemma 2, only more complicated.

Lemma 3 : Consider the events $\{E_v\}$ defined above lemma 1, only now it is assumed that the events are only pairwise independent. Then

$$Pr[v \in N(I')] \ge \frac{1}{8} \cdot min(sum_v, 1)$$

Proof : The notation introduced in the proof of lemma 1 is retained. Assume WLOG that $p_1 \ge \cdots \ge p_{d(v)}$. Let $\alpha_0 = 0$ and, for $1 \le i \le d(v)$, let $\alpha_i = \sum_{j=1}^{i} p_j$ (note that $\alpha_{d(v)} = sum_v$). Then,

$$Pr[v \in N(I')] \ge \sum_{i=1}^{d(v)} Pr[E_i'] Pr[A_i \mid E_i'] .$$

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

$$Pr[\neg A_i \mid E_i'] \le \sum_{\substack{z \in adj(i) \\ d(z) \ge d(i)}} Pr[E_z \mid E_i']$$

and

$$Pr[E_z \mid E_i'] = \frac{Pr[E_z \cap \neg E_1 \cap \cdots \cap \neg E_{i-1} \mid E_i]}{Pr[\neg E_1 \cap \cdots \cap \neg E_{i-1} \mid E_i]} .$$

The numerator is less than or equal to $Pr[E_z \mid E_i] = p_z$. The denominator is equal to

$$1 - Pr[\bigcup_{j=1}^{i-1} E_j \mid E_i] \ge 1 - \sum_{j=1}^{i-1} Pr[E_j \mid E_i] = 1 - \alpha_{i-1} .$$

Thus, $Pr[E_z \mid E_i'] \le \frac{p_z}{1-\alpha_{i-1}}$. Consequently,

$$Pr[\neg A_i \mid E_i'] \le \sum_{\substack{z \in adj(i) \\ d(z) \ge d(i)}} \frac{p_z}{1-\alpha_{i-1}} \le \frac{1}{2(1-\alpha_{i-1})} .$$

Thus,

$$Pr[A_i \mid E_i'] \ge 1 - \frac{1}{2(1-\alpha_{i-1})} = \frac{1-2\alpha_{i-1}}{2(1-\alpha_{i-1})} .$$

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

$$Pr[E_i'] = Pr[E_i] Pr[\neg E_1 \cap \cdots \cap \neg E_{i-1} \mid E_i] =$$

$$p_i \left(1 - Pr\left[\bigcup_{j=1}^{i-1} E_j \mid E_i\right]\right) \ge p_i (1 - \alpha_{i-1}) .$$

Thus, for $1 \le l \le d(v)$,

$$Pr[v \in N(I')] \ge \sum_{i=1}^{l} \frac{p_i(1-2\alpha_{i-1})}{2}$$

$$= \frac{\alpha_l}{2} - \sum_{1 \le j < i \le l} p_j p_i \ge \frac{\alpha_l}{2}\left[1 - \alpha_l \frac{(l-1)}{l}\right].$$

(This last inequality is similar to an inequality in the proof of lemma 2.) If $sum_v \le \frac{1}{2}$ then $Pr[v \in N(I')] \ge \frac{sum_v}{4}$.

If $sum_v \ge \frac{1}{2}$ then let $l = \min\{k : \alpha_k \ge \frac{1}{2}\}$. If $l = 1$ then $Pr[v \in N(I')] \ge \frac{1}{4}$. If $l \ge 2$ then $\alpha_{l-1} < \frac{1}{2} \le \alpha_l \le \frac{l}{2(l-1)}$. Then, $Pr[v \in N(I')] \ge \frac{\alpha_l}{4} \ge \frac{1}{8}$ □

Consider the first MIS algorithm modified so that the events $\{E_v\}$ are assumed to be only pairwise independent. Let m_1, m_2 and m_3 be as defined in the discussion preceeding theorem 1 with respect to this modified algorithm. Theorem 2, the analog to Theorem 1, states that this modified algorithm will work well on the average.

Theorem 2 : $E[m_3] \ge \frac{1}{16} m_1$.

Proof : Use lemma 3 in place of lemma 1 in the proof of theorem 1 □

5. Generating Pairwise Independent Events

Consider a probability space where the sample space is the set of all binary vectors of length n. Let E_i be an event that occurs for all binary vectors $(b_0, \cdots, b_{n-1})$ such that $b_i = 1$. Let the probability of $(b_0, \cdots, b_{n-1})$ be $\prod_{i=0}^{n-1} p_i^{b_i}(1-p_i)^{(1-b_i)}$. The events $E_0, \cdots, E_{n-1}$ are mutually independent and $Pr[E_i] = p_i$. Note that $\Omega(n)$ random bits are needed to randomly choose a binary vector. In this section, a new probability space is defined where the sample space is a small subset of all binary vectors of length n. A probability distribution is defined on this new sample space such that the events $E_1, \cdots, E_n$ are pairwise independent.

Let q be a prime number between n and $2n$. The new probability distribution has the property that $Pr[E_i] = p_i' = \frac{\lfloor p_i \cdot q \rfloor}{q}$. Thus, $|p_i - p_i'| \le \frac{1}{q} \le \frac{1}{n}$ (This is a sufficient approximation to p_i for the applications considered in this paper). Consider a n by q matrix A. Intuitively, row i of A corresponds to E_i (In the following discussion, x, y, j, j_1 and j_2 are integers between 0 and $q-1$, inclusive, and i, i_1, and i_2 are integers between 0 and $n-1$, inclusive). Let

$$A_{i,j} = \begin{cases} 1 \text{ if } 0 \le j \le p_i' \cdot q - 1 \\ 0 \text{ otherwise} \end{cases}$$

The new sample space is the collection of q^2 binary vectors

$$b_{x,y} = (b_{x,y}{}^0, \cdots, b_{x,y}{}^{n-1}),$$

where $b_{x,y}{}^i = A_{i,(x+y\cdot i) mod q}$. The new probability distribution assigns a probability of $\frac{1}{q^2}$ to each such $b_{x,y}$.

Lemma 4 : $Pr[E_i] = p_i'$.

Proof : For fixed l, there are exactly q pairs of x,y such that $(x+y\cdot i) mod q = l$. Event E_i occurs in all $b_{x,y}$ such that $(x+y\cdot i) mod q$ is between 0 and $p_i' \cdot q - 1$, inclusive, i.e. for exactly $p_i' \cdot q^2$ boolean vectors □

Lemma 5 : $Pr[E_{i_1} \cap E_{i_2}] = p_{i_1}' \cdot p_{i_2}'$.

Proof : For fixed l_1 and l_2, there is exactly one x,y pair such that $(x+y\cdot i_1) mod q = l_1$ and $(x+y\cdot i_2) mod q = l_2$ simultaneously. Events E_{i_1} and E_{i_2} both occur for $p_{i_1}' \cdot p_{i_2}' \cdot q^2$ pairs of l_1, l_2. Thus, E_{i_1} and E_{i_2} both occur for $p_{i_1}' \cdot p_{i_2}' \cdot q^2$ boolean vectors $b_{x,y}$ □

These two lemma together show that the events $\{E_i\}$ are pairwise independent in the new probability space. Since the new sample space only has q^2 binary vectors, it is possible to sample all vectors quickly in parallel. This feature is exploited in the next section to design a deterministic algorithm for the MIS problem.

This construction may be a useful tool to make other Monte Carlo algorithms deterministic. The only requirement is that the algorithm only requires pairwise independent events. This construction can be extended to generate pairwise independent random variables in the following way. The matrix A can be filled with any set of computable rational numbers.

Each row of A corresponds to a random variable which takes on each of the q entries in that row of A with equal probability. Both Paul Beame and Noga Alon have found a natural extension of this technique to generate d-wise independent random variables using n^d sample points.

6. Deterministic Algorithm for the MIS Problem

The algorithms for the MIS problem described in this section are almost immediate consequences of the results in sections 3, 4 and 5. In section 3, a Monte Carlo algorithm for the MIS problem is presented. Under the assumption that the events $\{E_v\}$ (defined above lemma 1) are mutually independent, the analysis shows that the expected running time of the algorithm is small. In section 4, it is shown that the same result holds when the events $\{E_v\}$ are assumed to be pairwise independent. In section 5, it is shown how to construct a probability distribution such that the events $\{E_v\}$ are pairwise independent and such that the number of points in the sample space is $O(n^2)$, thus they they can all be sampled in parallel. Putting these results together yields a deterministic algorithm with a small running time.

The only difficulty is that the events $\{E_v\}$ do not have exactly the same probability in the new probability space as they did in the original. More precisely, in the original probability space, $Pr[E_v] = \frac{1}{2d(v)}$, whereas in the new probability space,

$$Pr[E_v] = p_v' = \frac{\lfloor p_v \cdot q \rfloor}{q} = \frac{\lfloor \frac{q}{2d(v)} \rfloor}{q} .$$

This difficulty is overcome with the following modification to the algorithm. If there is a vertex $v \in V$ such that $d(v) \geq \frac{n}{16}$ then v is immediately added to the independent set I and $\{v\} \bigcup N(\{v\})$ is deleted from the graph. In this case, at least $\frac{1}{16}$ of the vertices are eliminated from the graph. If no such vertex exists, then for all $v \in V$, $d(v) < \frac{n}{16}$, which implies $\frac{q}{2d(v)} \geq \frac{n}{2d(v)} > 8$ and consequently $p_v' \geq \frac{8}{q}$. But this implies that $\frac{\lfloor p_v \cdot q \rfloor}{8} \geq 1$. Thus, since $\frac{\lfloor p_v \cdot q \rfloor + 1}{q} \geq p_v$, $\frac{8}{9} p_v \leq p_v' \leq p_v$.

Lemma 6 : Let the events $\{E_v\}$ be pairwise independent such that $Pr[E_v] = p_v'$ and such that for all $v \in V$, $d(v) < \frac{n}{16}$. Then $Pr[v \in N(I')] \geq \frac{1}{9} min(sum_v, 1)$.

Proof : The same proof as for lemma 3, except $\frac{8}{9} p_v$ is used as a lower bound and p_v is used as an upper bound on $Pr[E_v]$ □

Theorem 3 : Consider the first MIS algorithm as modified above so that the events $\{E_v\}$ are pairwise independent and $Pr[E_v] = p_v'$ and for all $v \in V$, $d(v) < \frac{n}{16}$. Let m_1, m_2, and m_3 be as defined in the discussion preceeding theorem 1 with respect to this modified algorithm. Then $E[m_3] \geq \frac{1}{18} m_1$.

Proof : Use lemma 6 in place of lemma 1 in the proof of theorem 1 □

The description of the second algorithm is given on the following page. 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 $O(logn)$ time on a EREW P-RAM using $O(m)$ processors, where the expected number of random bits used is $O(logn)$. The expected number of executions of the **while** loop before termination of the algorithm is $O(logn)$. Thus, the expected running time of the entire algorithm on a EREW P-RAM is $O((logn)^2)$ using $O(m)$ processors, where the expected number of random bits used is $O((logn)^2)$.

The algorithm can be easily modified to remove the random choices made in the (*) step of the algorithm. The idea is to let the algorithm try all q^2 possible pairs x,y in parallel, creating q^2 sets $I_{x,y}'$. The set which maximizes the number of edges eliminated from G' is used to form the new 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 $O((logn)^2)$.

```
begin
    I ← Ø
    G′=(V′,E′) ← G=(V,E)
    while V′ ≠ Ø do
    begin
        In parallel, for all v ∈ V′, compute d(v)
        In parallel, for all v ∈ V′
            if d(v)=0 then add v to I and delete v from V′.
        compute n′ = |V′|
        find v ∈ V′ such that d(v) is maximum
        if d(v) ≥ n′/16 then add v to I and let G′ be the
            graph induced on the vertices V′−({v} ∪ N({v}))
        else (for all v ∈ V′, d(v) < n′/16)
        begin
            compute a prime q such that n′ ≤ q ≤ 2n′
(*)         randomly choose x and y such that 0 ≤ x,y ≤ q−1
            X ← Ø
            In parallel, for all v ∈ V′,
            begin
                compute n(v) = ⌊ q / 2d(v) ⌋
                compute l(v) = (i + v j) mod q
                if l(v) ≤ n(v) then add v to X
            end
            I′ ← X
            In parallel, for all v ∈ X, w ∈ X,
                if (v,w) ∈ E′ then
                    if d(v) ≤ d(w) then I′ ← I′−{v}
                    else I′ ← I′−{w}
            I ← I ∪ I′
            Y ← I′ ∪ N(I′)
            G′=(V′,E′) is the induced subgraph on V′−Y.
        end
    end
end
```

7. The Maximal Coloring Problem

The Maximal Coloring problem generalizes two problems of interest. The input 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 $C_v=\{red\}$ for each vertex $v \in V$. The set of vertices colored *red* in any maximal coloring are a MIS.

Another problem which is a special case of the Maximal Coloring problem is the Δ+1VC problem. The input to the Δ+1VC 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 $C=\{c_1,\ldots,c_{\Delta'}\}$ 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 Δ+1VC 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 Δ+1VC problem follows: For $v=1,\ldots,n$, 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 Δ+1VC problem by emulating the sequential algorithm. However, this is unlikely since

Lemma 7 : 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 section 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 NC^1 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. Brook 's 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.

8. 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]. Informally, they model the brain cells by a large collection of computational units with a finite number of states and with very limited computing power. They model the synaptic connections by interconnections between the units through which they can pass information about their current state. They describe various models for representing knowledge, and suggest massively parallel asynchronous algorithms for computing, i.e. for acquiring new information, recalling previously acquired information and concepts, deducing new concepts from old, etc. They are faced with two basic questions: how to represent knowledge so that the representation has the desired semantic properties and how to implement parallel asynchronous algorithms to compute quickly given a particular representation. The hope is that the development of problems which seem to model a problem typically encountered by the brain, together with a study of the inherent complexity of algorithms for these problems, will give some insight into the low level workings of the brain.

One particular model of the brain is a binary coherent system [Ho], [Hi]. The interconnections of the system can be represented by an undirected graph $G=(V,E)$. Each edge $e \in E$ has a real-valued weight w_e. Each vertex $v \in V$ has a real-valued threshold t_v. Each vertex v has a state s_v which can be either -1 or 1. The state of the system is a tuple $(s_1,...,s_{|V|})$. The energy of vertex v in a system state is

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

Hopfield [Ho] addresses the following problem, which is hereafter called the Binary Coherent System (BCS) problem. The input is an undirected graph $G=(V,E)$, and weights and thresholds for all edges and vertices in G, respectively. 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, the algorithm described in section 6 shows that at least a special case of the BCS problem is in NC, and Baruch Awerbuch has observed that the algorithm given in section 3 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. Karloff [Ka2] 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 $\square$

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.

9. 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. There are a number of problems for which these techniques provide an easy deterministic algorithm.

2. Develop probabilistic bounds on events that are $d-wise$ independent. As mentioned previously, the work of [Ga] implicitly addresses this type of question. This seems to be one of the necessary steps for proving that a Monte Carlo algorithm can be converted into a deterministic algorithm using the techniques developed in this paper. In a different setting, [ACGS] used the fact that Chebychev's Inequality holds for pairwise independent events to prove the security of RSA/Rabin Bits. Their basic idea was the same, you need less random bits to generate pairwise independent events than you do to generate mutually independent events.

3. The Monte Carlo algorithm presented in section 3 has a very local property. This property seems particularly well-suited to distributed computing networks where the processors can only communicate with neighboring processors. As pointed out in section 8, this feature also seems desirable in some Artificial Intelligence applications. Find applications for the algorithms presented in this paper in these areas.

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.

10. Acknowledgements

I thank Paul Beame for extensive discussions which helped simplify the analysis of the algorithms, and especially for conversations which contributed significantly to the results in sections 4 and 5. I thank Stephen Cook for many discussion 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 section 3.1. I thank Allan Borodin for introducing me to the work on connectionist models.

11. References

[ACGS]
Alexi, W., Chor, B., Goldreich, O. and Schnorr, C. *RSA/Rabin Bits are* $\frac{1}{2}+\frac{1}{poly(logN)}$ *Secure*, 25^{th} FOCS, October 1984

[ABI]Alon, N., Babai, L., and Itai, A.,*A Fast and Simple Randomized Parallel Algorithm for the Maximal Independent Set Problem*, private communication

[Br] Brooks, R.L., *On colouring the nodes of a network*, Proc. Cambridge Philos. Soc., 37, 194-197, 1941

[Co] Cook, S. *A Taxonomy of Problems with Fast Parallel Algorithms*, to appear in Information and Control

[CL]Cook, S. and Luby, M. *A Fast Parallel Algorithm for Finding a Truth Assignment to a 2-CNF Formula*, paper in preparation

[Fe] Feldman, J. A. *A Connectionist Model of Visual Memory*, Parallel Models of Associative Memory, G.E. Hinton and J.A. Anderson editors, Hillsdale, N.J., Lawrence Erlbaum Associates, publishers, 1981

[Fr] Feller, W. *An Introduction to Probability Theory and Its Applications*, vol. 1,3^{rd} edition, 1968, John Wiley and Sons, publishers

[Ga]Galambos, J., *Bonferroni Inequalities*, The Annals of Probability, 1977, vol. 5, no. 4, 577-581

[Go]Goldschlager, L.M. *A Computational Theory of Higher Brain Function*, Technical Report, Stanford University, April 1984

[Ho]Hopfield, J. J., *Neural networks and physical system with emergent collective computational abilities*, Proceedings National Academy of Sciences, vol. 79, pp. 2554-2558, April 1982

[Hi] Hinton, G.E., Sejnowski, T. and Ackley, D., *Boltzmann Machines: Constraint Satisfaction Networks that Learn*, technical report CMU-CS-84-119

[II] Israeli, A. and Itai, A. *A Fast and Simple Randomized Parallel Algorithm for Maximal Matching*, Computer Science Dept., Technion, Haifa Israel, 1984

[IS] Israeli, A. and Shiloach, Y. *An Improved Maximal Matching Parallel Algorithm*, Tech. Rep. 333, Computer Science Dept., Technion, Haifa Israel, 1984

[Kf1] Karloff, H., *Randomized Parallel Algorithm for the Odd-Set Cover Problem*, preprint

[Kf2] Karloff, H., private communication

[KSS] Karloff, H., Shmoys, D. and Soroker, D., *Efficient Parallel Algorithms for Graph Coloring and Partitioning Problems*, preprint

[KW] Karp, R.M. and Wigderson, A. *A Fast Parallel Algorithm for the Maximal Independent Set Problem*, Proceedings 16^{th} *ACM STOC (1984), pp. 266-272*

[KUW]

Karp, R.M., Upfal, E. and Wigderson, A., *Constructing a Perfect Matching is in Random NC*, this conference

[Le] Lev G. *Size Bounds and Parallel Algorithms for Networks*, Report CST-8-80, Department of Computer Science, University of Edinburgh (1980)

[Ts] Tsotsos, J. *Representational Axes and Temporal Cooperative Processes*, Technical Report RBCV-84-2, University of Toronto, April 1984

[Va] Valiant, L. G. *Parallel Computation*, Proceedings 7^{th} IBM Symposium on Mathematical Foundations of Computer Science (1982)