Lecture Notes on a *Parallel Algorithm for Generating a Maximal Independent Set*

Eric Vigoda
Georgia Institute of Technology

Last updated for 7530 - Randomized Algorithms, Spring 2010.

In this lecture we present a randomized parallel algorithm for generating a maximal independent set. We then show how to derandomize the algorithm using pairwise independence. For an input graph with n vertices, our goal is to devise an algorithm that works in time polynomial in $\log n$ and using polynomial in n processors. See Chapter 12.1 of Motwani and Raghavan [5] for background on parallel models of computation (specifically the CREW PRAM model) and the associated complexity classes NC and RNC.

Here we present the algorithm and proof of Luby [4], see Section 4 for more on the history of this problem. Our goal is to present a parallel algorithm for constructing a maximal independent set of an input graph on n vertices, in time polynomial in $\log n$ and using polynomial in n processors.

1 Maximal Independent Sets

For a graph $G = (V, E)$, an independent set is a set $S \subset V$ which contains no edges of G, i.e., for all $(u, v) \in E$ either $u \notin S$ and/or $v \notin S$. The independent set S is a *maximal independent set* if for all $v \in V$, either $v \in S$ or $N(v) \cap S \neq \emptyset$ where $N(v)$ denotes the neighbors of v.

It's easy to find a maximal independent set. For example, the following algorithm works:

1. $I = \emptyset$, $V' = V$.
2. While ($V' \neq \emptyset$) do
   (a) Choose any $v \in V'$.
   (b) Set $I = I \cup v$.
   (c) Set $V' = V' \setminus (v \cup N(v))$.
3. Output I.

Our focus is finding an independent set using a parallel algorithm. The idea is that in every round we find a set S which is an independent set. Then we add S to our current independent set I, and we remove $S \cup N(S)$ from the current graph V'. If $S \cup N(S)$ is a constant fraction of $|V'|$, then we will only need $O(\log |V|)$ rounds. We will instead ensure that by removing $S \cup N(S)$ from the graph, we remove a constant fraction of the edges.

To choose S in parallel, each vertex v independently adds themselves to S with a well chosen probability $p(v)$. We want to avoid adding adjacent vertices to S. Hence, we will prefer to add low degree vertices. But, if for some edge (u, v), both endpoints were added to S, then we keep the higher degree vertex.

Here's the algorithm:

The Algorithm

Problem : Given a graph find a maximal independent set.

1. $I = \emptyset$, $V' = V$ and $G' = G$.
2. While $(V' \neq \emptyset)$ do:
   (a) Choose a random set of vertices $S \subset V'$ by selecting each vertex v independently with probability $1/(2d_{G'}(v))$ where $d_{G'}(v)$ is the degree of v in the graph G'.
   (b) For every edge $(u, v) \in E(G')$ if both endpoints are in S then remove the vertex of lower degree from S (Break ties arbitrarily). Call this new set S'.
   (c) $I = I \bigcup S'$. Let $V' = V' \setminus (S' \bigcup N_{G'}(S'))$. Finally, let G' be the induced subgraph on V'.
3. Output I

Fig 1: The algorithm

Correctness : We see that at each stage the set S' that is added is an independent set. Moreover since we remove, at each stage, $S' \bigcup N(S')$ the set I remains an independent set. Also note that all the vertices removed from G' at a particular stage are either vertices in I or neighbours of some vertex in I. So the algorithm always outputs a maximal independent set. We also note that it can be easily parallelized on a CREW PRAM.

2 Expected Running Time

In this section we bound the expected running time of the algorithm and in the next section we derandomize it. Let $G_j = (V_j, E_j)$ denote the graph G' after stage j.

Main Lemma: For some $c < 1$,

$$\mathrm{E}\left[|E_j| \mid E_{j-1}\right] < c|E_{j-1}|.$$

Hence, in expectation, only $O(\log m)$ rounds will be required, where $m = |E_0|$.

For graph G_j, we classify the vertices and edges as GOOD and BAD to distinguish those that are likely to be removed. We say vertex v is BAD if more than 2/3 of the neighbors of v are of higher degree than v. We say an edge is BAD if both of its endpoints are bad; otherwise the edge is GOOD.

The key claims are that at least half the edges are GOOD, and each GOOD edge is deleted with a constant probability. The main lemma then follows immediately.

Here are the main lemmas.

Lemma 1. *At least half the edges are GOOD.*

Lemma 2. *If an edge e is GOOD then the probability that it gets deleted is at least α where $\alpha := \frac{1}{2}\left(1 - e^{-1/6}\right)$.*

The constant α is approximately 0.07676. We can now re-state and prove the main lemma:

Main Lemma:

$$\mathrm{E}\left[|E_j| \mid E_{j-1}\right] \le |E_{j-1}|(1 - \alpha/2).$$

Proof of Main Lemma.

$$\begin{aligned}
\mathrm{E}\left[|E_j| \mid E_{j-1}\right] &= \sum_{e \in E_{j-1}} 1 - \Pr\left[e \text{ gets deleted}\right] \\
&\le |E_{j-1}| - \alpha|GOOD \text{ edges}| \\
&\le |E_{j-1}|(1 - \alpha/2).
\end{aligned}$$

$\square$

Thus,

$$\mathrm{E}\left[|E_j|\right] \le |E_0| \left(1 - \frac{\alpha}{2}\right)^j \le m \exp(-j\alpha/2) < 1,$$

for $j > \frac{2}{\alpha} \log m$. Therefore, the expected number of rounds required is $\le 30 \log m = O(\log m)$. Moreover, by Markov's inequality,

$$\Pr\left[E_j \neq \emptyset\right] \le \mathrm{E}\left[|E_j|\right] < m \exp(-j\alpha/2) \le 1/4,$$

for $j = \frac{4}{\alpha} \log m$. Hence, with probability at least 3/4 the number of rounds is $\le 60 \log m$, and therefore we have an RNC algorithm for MIS.

It remains to prove Lemmas 1 and 2. We begin with Lemma 1 which is a cute combinatorial proof.

Proof of Lemma 1. Denote the set of bad edges by E_B. We will define $f : E_B \to \binom{E}{2}$ so that for all $e_1 \neq e_2 \in E_B$, $f(e_1) \cap f(e_2) = \emptyset$. This proves $|E_B| \leq |E|/2$, and we're done.

The function f is defined as follows. For each $(u, v) \in E$, direct it to the higher degree vertex. Break ties as in the algorithm. Now, suppose $e = (u, v) \in E_B$, and is directed towards v. Since e is BAD then v is BAD. Therefore, by the definition of a BAD vertex, at least 2/3 of the edges incident to v are directed away from v, and at most 1/3 of the edges incident to v are directed into v. In other words, v has at least twice as many out-edges as in-edges. Hence for each edge into v we can assign a disjoint pair of edges out of v. This gives our mapping f since each BAD edge directed into v has a disjoint pair of edges directed out of v. $\square$

We now prove Lemma 2. To that end we prove the following lemmas, which say that GOOD vertices are likely to have a neighbor in S, and vertices in S have probability at least 1/2 of being in S'. From these lemmas, Lemma 2 will easily follow since the neighbors of S' are deleted from the graph.

Lemma 3. *If v is GOOD then* $\Pr[N(v) \cap S \neq \emptyset] \geq 2\alpha$, *where* $\alpha := \frac{1}{2}(1 - e^{-1/6})$.

Proof. Define $L(v) := \{w \in N(v) \mid d(w) \leq d(v)\}$.
By definition, $|L(v)| \geq \frac{d(v)}{3}$ if v is a GOOD vertex.

$$\begin{aligned}
\Pr[N(v) \cap S \neq \emptyset] &= 1 - \Pr[N(v) \cap S = \emptyset] \\
&= 1 - \prod_{w \in N(v)} \Pr[w \notin S] \quad \text{using full independence} \\
&\geq 1 - \prod_{w \in L(v)} \Pr[w \notin S] \\
&= 1 - \prod_{w \in L(v)} \left(1 - \frac{1}{2d(w)}\right) \\
&\geq 1 - \prod_{w \in L(v)} \left(1 - \frac{1}{2d(v)}\right) \\
&\geq 1 - \exp(-|L(v)|/2d(v)) \\
&\geq 1 - \exp(-1/6).
\end{aligned}$$

$\square$

Note, the above lemma is using full independence in its proof.

Lemma 4. $\Pr[w \notin S' \mid w \in S] \leq 1/2$.

Proof. Let $H(w) = N(w) \setminus L(w) = \{z \in N(w) : d(z) > d(w)\}$.

$$\begin{aligned}
\Pr[w \notin S' \mid w \in S] &= \Pr[H(w) \cap S \neq \emptyset \mid w \in S] \\
&\leq \sum_{z \in H(w)} \Pr[z \in S \mid w \in S] \\
&\leq \sum_{z \in H(w)} \frac{\Pr[z \in S, w \in S]}{\Pr[w \in S]} \\
&= \sum_{z \in H(w)} \frac{\Pr[z \in S]\Pr[w \in S]}{\Pr[w \in S]} \quad \text{using pairwise independence} \\
&= \sum_{z \in H(w)} \Pr[z \in S] \\
&= \sum_{z \in H(w)} \frac{1}{2d(z)} \\
&\leq \sum_{z \in H(w)} \frac{1}{2d(v)} \\
&\leq \frac{1}{2}.
\end{aligned}$$

□

From Lemmas 3 and 4 we get the following result that GOOD vertices are likely to be deleted.

Lemma 5. *If v is GOOD then* $\Pr[v \in N(S')] \geq \alpha$

Proof. Let V_G denote the GOOD vertices. We have

$$\begin{aligned}
&\Pr[v \in N(S') \mid v \in V_G] \\
&\quad = \Pr[N(v) \cap S' \neq \emptyset \mid v \in V_G] \\
&\quad = \Pr[N(v) \cap S' \neq \emptyset \mid N(v) \cap S \neq \emptyset, v \in V_G]\Pr[N(v) \cap S \neq \emptyset \mid v \in V_G] \\
&\quad \geq \Pr[w \in S' \mid w \in N(v) \cap S, v \in V_G]\Pr[N(v) \cap S \neq \emptyset \mid v \in V_G] \\
&\quad \geq (1/2)(2\alpha) \quad \text{by Lemmas 4 and 3} \\
&\quad = \alpha.
\end{aligned}$$

□

Since vertices in $N(S')$ are deleted, an immediate corollary of Lemma 5 is the following.

Corollary 6. *If v is GOOD then the probability that v gets deleted is at least α.*

Finally, from Corollary 6 we can easily prove Lemma 2, which was our main task remaining in the analysis of the RNC algorithm.

Proof of Lemma 2. Let $e = (u, v)$ and at least one of the endpoints is GOOD, so assume v is GOOD. Therefore, by Lemma 5 we have:

$$\Pr\left[e = (u, v) \in E_{j-1} \setminus E_j\right] \geq \Pr\left[v \text{ gets deleted}\right]. \geq \alpha.$$

□

3 Derandomizing MIS

The only step where we use full independence is in Lemma 3 for lower bounding the probability that a *GOOD* vertex gets picked. The argument we used was essentially the following:

Lemma 7. *Let X_i, $1 \leq i \leq n$, be $\{0, 1\}$ random variables and $p_i := \Pr\left[X_i = 1\right]$. If the X_i are fully independent then*

$$\Pr\left[\sum_{i=1}^{n} X_i > 0\right] \geq 1 - \prod_{i=1}^{n} (1 - p_i)$$

Here is the corresponding bound if the variables are pairwise independent

Lemma 8. *Let X_i, $1 \leq i \leq n$, be $\{0, 1\}$ random variables and $p_i := \Pr\left[X_i = 1\right]$. If the X_i are pairwise independent then*

$$\Pr\left[\sum_{i=1}^{n} X_i > 0\right] \geq \frac{1}{2} \min\left\{\frac{1}{2}, \sum_{i=1}^{n} p_i\right\}$$

Proof. Suppose $\sum_{i=1}^{n} p_i \leq 1$. Then we have the following, (the condition $\sum_i p_i \leq 1$ will only come into some algebra at the end)

$$\begin{aligned}
\Pr\left[\sum_{i=1}^{n} X_i > 0\right] &\geq \Pr\left[\sum_{i=1}^{n} X_i = 1\right] \\
&\geq \sum_i \Pr\left[X_i = 1\right] - \frac{1}{2} \sum_{i \neq j} \Pr\left[X_i = 1, X_j = 1\right] \\
&= \sum_i p_i - \frac{1}{2} \sum_{i \neq j} p_i p_j \\
&\geq \sum_i p_i - \frac{1}{2} \left(\sum_i p_i\right)^2 \\
&= \sum_i p_i \left(1 - \frac{1}{2} \sum_i p_i\right) \\
&\geq \frac{1}{2} \sum_i p_i \quad \text{when} \quad \sum_i p_i \leq 1.
\end{aligned}$$

This proves the lemma when $\sum_i p_i \leq 1$.

If $\sum_i p_i > 1$, then we restrict our index of summation to a set $S \subseteq [n] = \{1, \ldots, n\}$ such that $1/2 \leq \sum_{i \in S} p_i \leq 1$. Note, if $\sum_i p_i > 1$, there always must exist a subset $S \subseteq [n]$ where $1/2 \leq \sum_{i \in S} p_i \leq 1$, since either there is an index j where $1/2 \leq p_j \leq 1$ and we can then choose $S = \{j\}$, or for all i we have $p_i < 1/2$ and it is easy to see then that there is a such a subset S in this case. Given this S, following the above proof we have:

$$\Pr\left[\sum_{i=1}^{n} X_i > 0\right] \geq \Pr\left[\sum_{i \in S} X_i = 1\right] \geq \frac{1}{2}\sum_{i \in S} p_i \geq 1/4,$$

since $\sum_{i \in S} p_i \geq 1/2$, and this proves the conclusion of the lemma in this case. $\square$

Using Lemma 8 in the proof of Lemma 3 we get 2α replaced by $1/12$. Hence, using the construction of pairwise random variables described in an earlier lecture, we can now derandomize the algorithm to get a deterministic algorithm that runs in $O(mn \log n)$ time (this is asymptotically almost as good as the sequential algorithm). The advantage of this method is that it can be easily parallelized to give an NC^2 algorithm (using $O(m)$ processors).

4 History and Open Questions

The $k-$wise independence derandomization approach was developed in [2, 1, 4]. The maximal independence problem (MIS) was first shown to be in NC by Karp and Wigderson [3]. They showed that MIS is in NC^4. Subsequently, improvements and simplifications on their result were found by Alon et al [1] and Luby [4]. The algorithm and proof described in these notes is the result of Luby [4].

The question of whether MIS is in NC^1 is still open.

References

[1] N Alon, L. Babai, A. Itai, A Fast and Simple Randomized Parallel Algorithm for the Maximal Independent Set Problem, *Journal of Algorithms*, 7:567-583, 1986.

[2] B. Chor, O. Goldreich, On the power of two-point sampling, *Journal of Complexity*, 5:96-106, 1989.

[3] R.M. Karp, A. Wigderson, A fast parallel algorithm for the maximal independent set problem, *Proc. 16th ACM Symposium on Theory of Computing* (STOC), 266-272, 1984.

[4] M. Luby, A simple parallel algorithm for the maximal independent set problem, *Proc, 17th ACM Symposium on Theory of Computing* (STOC), 1-10, 1985.

[5] R. Motwani and P. Raghavan, *Randomized Algorithms*, Cambridge University Press, New York, 1995.