Probabilistic Analysis of a Parallel Algorithm for Finding the Lexicographically First Depth First Search Tree in a Dense Random Graph

Martin Dyer*
School of Computer Studies, University of Leeds, Leeds, United Kingdom

Alan Frieze†
Department of Mathematics, Carnegie-Mellon University, Pittsburgh, PA 15213

ABSTRACT

We describe an $O((\log n)^2)$ time parallel algorithm, using n processors, for finding the lexicographically first depth first search tree in the random graph $G_{n,p}$, with p fixed. The problem itself is complete for P, and so is unlikely to be efficiently parallelizable always.

1. INTRODUCTION

In this paper we consider the problem of finding the Lexicographically First Depth First Search Tree (LFDFST) in a random graph. The LFDFST of a connected graph is the tree obtained by depth first search starting at some fixed vertex and using the given ordering of the adjacency lists to decide the next vertex to be visited. This tree is clearly computable in polynomial time (actually linear) and so there is the question of whether there exists an NC-algorithm, i.e., a parallel algorithm which uses $O(n^a)$ processors and runs in $O((\log n)^b)$ time for constants $a, b > 0$. This is currently considered to be unlikely since it is known to be complete for P under NC-reductions (Reif [9]), and so if this problem is in NC

* Supported by NATO Grant RG0088/89.
† Supported by NSF Grant CCR-8900112 and NATO Grant RG0088/89.

then NC = P, which is not generally thought to be case. In the case of finding *some* depth first search tree, not ncessarily the lexicographically first tree, the picture is a bit brighter. (A spanning tree T of a graph is a depth first search tree if there is a choice of root r and ordering of the adjacency lists such that depth first search started at r produces T. This definition holds for the directed and undirected cases.) Aggarwal and Anderson [2] constructed an O 9($(\log n)^5$) time randomized algorithm for the undirected case and Aggarwal, Anderson, and Kao [3] constructed an $O((\log n)^7)$ time randomized algorithm for the directed case. Using results of Goldberg, Plotkin, and Vaidya [7] they also construct an $O((\log n)^{11}\sqrt{n})$ time deterministic algorithm for the directed case. In view of the result of Reif, it makes sense to study parallel algorithms for the LFDFST problem which are fast on average. We describe in this paper a parallel algorithm which uses $O(n)$ processors and which computes the LFDFST of a *dense* random graph in $O((\ln n)^2)$ expected time. The results we obtain are therefore analogous to those of Coppersmith, Raghavan, and Tompa [5] and Calkin and Frieze [4] for the problem of finding a lexicographically first maximal independent set in a random graph.

More precisely let $G = G_{n,p}$ denote the random graph with vertex set $[n] = \{1, 2, \ldots, n\}$, in which each of the $\binom{n}{2}$ possible edges is selected independently with probability $p = 1 - q$. We assume throughout this paper that p is a fixed constant. (This is what is meant by the random graph being dense.) The *hidden* constant in the running time depends significantly on p and at this time, extending our results to the case where $p = p(n) \rightarrow 0$ as $n \rightarrow \infty$ is left as an interesting open problem. We also assume that the adjacency lists of each vertex are sorted in ascending order and that the tree is rooted at vertex 1. Another interesting problem is to extend the method here to the case where the adjacency lists are in, say, random order.

Now the LFDFST of a dense random graph is usually very far from being "bushy." In fact we can expect it to consist of a *left-most* path of length $n - O(1)$ plus $O(1)$ vertices "patched on" at the very bottom. So the main problem is to compute this path—the Lexicographically First Maximal Path (LFMP). Once this has been done, it is usually easy to finish off the LFDFST. However, this problem is also complete for P (Anderson and Mayr [1]). Thus the real focus of the paper is on how to compute the LFMP.

2. COMPUTING THE LFMP

From now on we assume that the LFMP is the path $u_1 = 1, u_2, \ldots, u_m$. Let $u_t (1 < t < m)$ be a *splitter* for the LFMP if

$$\{u_1, \ldots, u_{t-1}\} = [t-1] \quad \text{and} \quad u_t = t\,.$$

Thus if u_t is a splitter, then $\{u_t, u_{t+1}, \ldots, u_m\}$ is the LFMP of the subgraph G_t of G induced by $[t, n] = \{t, t+1, \ldots, n\}$. The edge sets of G, G_t will be denoted by $E(G)$, $E(G_t)$, respectively.

Our idea is simply to construct the subpath P_t of the LFMP of G_t from t to the first splitter of the LFMP for each of the graphs G_t, $t = 1, 2, \ldots, n$. We then use these paths to construct the LFMP. We search for the first splitters in a sequential manner relying on the density of the graph to show that it is unlikely that we will need to search very far. (A fast parallel algorithm for finding the first splitter in an arbitrary graph leads easily to a fast parallel algorithm for finding the LFMP, and so is unlikely to exist.) We use the following greedy algorithm to search sequentially for the first splitters: we assume that, prior to the start of **GREEDY**(t), Processor t finds $\sigma_t = \min\{u > t : \{t, u\} \in E(G_t)\}$. This can be found in $O(\ln n)$ time by binary search.

Algorithm GREEDY(t)

{The algorithm halts on output of u_t, the first splitter of G_t (or $n + 1$ if there isn't one), and the corresponding path P_t (given as a list of vertices).};

```
begin
    if$\{t, t+1\} \in E(G_t)$ then output $t+1$ and $t, t+1$ else
    $u := t$; $A := \{t\}$; $B := [t+1, n]$; $P := t$;
    repeat
        $L := \{b \in B : \{u, b\} \in E(G_t)\}$;
        if $L = \emptyset$ then output $n+1$ and $P, n+1$ else
        begin
            $u := \min L$; $A := A \cup \{u\}$; $B := B - \{u\}$; $P := P, u$;
            if $A = [t, u]$ then output $u$ and $P$
        end
    until output
end
```

Given $\sigma_1, \sigma_2, \ldots, \sigma_{n-1}$ it requires $O(u_t - t)$ time to find min L at any stage, and we can easily test for $A = [t, u]$ by keeping track of min A and max A and checking for $|A| = \max A - \min A + 1$. Thus **GREEDY**($t$) can be implemented to run in $O((u_t - t)^2)$ time.

Let $b > 0$, $\beta = \lceil b \ln n \rceil$, and consider the event

$$\mathscr{E} = \{\exists t \in [n - \beta] : u_t > t + \beta\}\ .$$

The subsequent analysis will show that, given any $a > 0$, there exists a $b = b(a, p)$ such that when n is sufficiently large,

$$\Pr(\mathscr{E}) \le n^{-a}\ . \tag{1}$$

Therefore, by (1), the run time of **GREEDY**(t) is $O((\ln n)^2))$ with probability tending to 1 as $n \to \infty$. Our procedue for constructing the LFMP of G is thus:

Algorithm MAKELFMP

```
begin
    for $t = 1$ to $n$ pardo GREEDY($t$)
    Let $\Gamma$ be the digraph with vertex set $[n+1]$ and arcs $(t, u_t)$, $t \in [n]$;
    Construct the (unique) path $v_1, v_2, \ldots, v_k$ from 1 to $n+1$ in $\Gamma$;
    output LFMP = $P_{v_1}, P_{v_2}, \ldots, P_{v_{k-1}}$
end
```

Note that Γ is a tree rooted at $n+1$, so the path from 1 to $n+1$ can be found in $O(\ln n)$ time using parallel tree contraction (Miller and Reif [8]).

Suppose that we now take $a=2$ in the definition of $\mathscr{E}$. Assume that $\mathscr{E}$ does not occur and let S be the set of vertices not on the LFMP at termination of **MAKELFMP.** Note that $S \subseteq [n-\beta+1, n]$. We now use any one of the processors to finish off the depth first search in a sequential manner. This processor can spend $O((\ln n)^2)$ time examining all the edges between vertices of S, and so the only issue is how far down the LFMP the search has to backtrack. If every vertex in $[n-\beta+1, n]$ is adjacent in G to at least one vertex in $[n-2\beta+1, n-\beta]$, then the sequential depth first search will not have to backtrack for more than 2β vertices. The probability that event does not occur is at most βq^{β} which is $o(n^{-2})$ for $b > -3/\ln q$. The definition of $\mathscr{E}$ allows us to inflate the size of b and so we will assume that it exceeds this lower bound. To summarize, conditional on an event of probability $1-O(n^{-2})$ our algorithm computes the LFDFST in $O((\ln n)^2)$ steps. The remaining bad cases can be dealt with in $O(n^2)$ time by sequential depth first search and so our algorithm has expected time complexity of $O((\ln n)^2)$ as claimed. It only remains to prove (1).

3. PROOF OF (1)

We will assume in the following analysis that $t=1$ and that the vertices of G are all the positive natural numbers. We will show that there exists $A=A(p)$ and, for all $b>0$, a $c=c(b,p)$ which grows unboundedly with b such that

$$\mathbf{Pr}(u_1 > \beta = \lceil b \ln n \rceil) \le An^{-c} \tag{2}$$

This is sufficient to prove (1). The only doubt arises from the fact that we have disallowed the possibility that **GREEDY**(t) does not find a splitter, especially for large t. But, so long as $t < n-2\beta$, we only have to add at most βq^{β} to the right-hand side of (2) to account for the possibility that the end vertex of P is not adjacent to any of the vertices not in P. [We also need a surreptitious doubling of b to make it fit together with (1).]

Let k now refer to the number of completed iterations of the **repeat . . . until** loop of **GREEDY**(t). Let A_k refer to the value of A, $m_k = \max A_k$ and v_k denote the end vertex of the path P after the kth iteration.

Let $Z_k = m_k - k - 1$ and $\tau = \min\{k > 1 : Z_k = 0\}$. We show that there exist $C>0$ and $0<\epsilon<1$ such that

$$\mathbf{Pr}(\tau \ge s) \le C\epsilon^{s} \qquad s=1,2,\ldots \tag{3}$$

Note that (3) is equivalent to the assertion that the moment generating function $M_\tau(\lambda)$ is finite for some $\lambda>0$. For, if (3) holds, a simple calculation shows that $M_\tau(\lambda)$ is finite for $\lambda < -\ln \epsilon$. Conversely, if $M_\tau(\lambda)$ is finite for some $\lambda>0$, the Markov inequality gives $\mathbf{Pr}(\tau \ge s) \le e^{-\lambda s} M_\tau(\lambda)$, so (3) holds with $\epsilon = e^{-\lambda}$, $C = M_\tau(\lambda)$.

Inequality (2) follows easily from (3). Suppose $Z_{k_0}=0$. We check whether $\{v_{k_0}, m_{k_0}+1\} \in E(G)$. This will be the first time we have checked for this edge and so it exists with probability p independently of the algorithm's previous history. If the edge is present, the algorithm will halt. However, if it is not, the algorithm will never look again at edges incident with vertices in A_{k_0}. It will select some other edge incident with v_{k_0}, if there is one, and proceed essentially as though A_{k_0} was vertex 1 and there was no edge $\{1,2\}$. (Vertex $m_{k_0}+1$ acts as vertex 2 here.) Thus the duration of **GREEDY**(t) is stochastically dominated by the sum σ of a number of independent random variables with the distribution of τ, where the number in the sum has a geometric distribution with parameter p. But this implies that $M_\sigma(\lambda)=M_\tau(\lambda)p/(1-qM_\tau(\lambda))$, which is finite for small enough $\lambda>0$. Hence an inequality like (3) holds for σ, and so (2) follows.

We continue with the proof of (3). We observe that the process Z_k ($k=1,2,\dots$) is a Markov chain on the non-negative integers with transition probabilities given by

$$\left.\begin{aligned}\mathbf{Pr}(Z_{k+1}=r-1\,|\,Z_k=r)&=1-q^r\\ \mathbf{Pr}(Z_{k+1}=r+s\,|\,Z_k=r)&=pq^{r+s}\end{aligned}\right\}\quad\text{for } r,s\ge 0 \tag{4}$$

This is because the edges joining the end vertex of the current portion of the LFMP to vertices not so far in the LFMP have not yet been examined. Thus one can easily check that, once Z_k gets large, the expected increase in Z_k will be negative. The idea of the proof is therefore first to show that Z_k must be "small" for "many" k, and then to show that this means that (3) must hold. So let

$$r_0=\lceil\log_{1/q}(2/p)\rceil$$

and then let

$$\rho_k=|\{l\le k: Z_l\ge r_0\}|\quad\text{and}\quad \sigma_k=|\{l\le k: Z_l=r_0-1\}|\,.$$

Let $\mathcal{D}_u=\{\exists k:\rho_k\ge u \text{ and } \sigma_k\le\frac{1}{3}u-1\}$ for $u=1,2,\dots$. Our aim now is to prove

Lemma 1.

$$\mathbf{Pr}(\mathcal{D}_u)<e^{-pu/84}\qquad\text{for } u=1,2,\dots$$

Proof. Suppose $\rho_k\ge u$ and let $k_1<k_2<\cdots<k_u$ be the first u values of t for which $Z_t\ge r_0$. Let

$$\delta_i=Z_{k_i+1}-Z_{k_i}\qquad\text{for } i=1,2,\dots,u\,.$$

Observe that

$$\Delta_u=\delta_1+\delta_2+\cdots+\delta_u<-u/3\Rightarrow\sigma_k>\tfrac{1}{3}u-1\,.$$

This follows from the fact that if

$$I=\{t\le k_u: Z_t=r_0 \text{ and } Z_{t+1}=r_0-1\}\,,\qquad l=\max I\,,$$

cancellations in the sum lead to the identity

$$\sum_{k_i \le l} \delta_i = -|I| .$$

We also have

$$\sum_{k_i > l} \delta_i = Z_{k_u+1} - r_0 \ge -1 .$$

Thus to prove the lemma we need only show

$$\mathbf{Pr}(\Delta_u \ge -u/3) \le e^{-pu/84} .$$

Since the δ_i are independent this is not difficult. A straightforward calculation shows that if $r \ge r_0$ and $0 < \lambda < p$, then

$$\begin{aligned}\mathbf{E}(e^{\lambda\delta_i} | Z_{k_i} = r) &= e^{-\lambda}\left(1 + q^r \frac{1 - e^{-\lambda}}{e^{-\lambda} - q}\right) \\ &\le e^{-\lambda}\left(1 + \frac{p\lambda}{2(p-\lambda)}\right) \\ &\le e^{-\lambda} e^{\frac{p\lambda}{2(p-\lambda)}}.\end{aligned}$$

Thus

$$M_{\delta_i}(\lambda) \le e^{-\lambda + \frac{p\lambda}{2(p-\lambda)}} .$$

Hence, using independence, the Markov inequality gives

$$\begin{aligned}\mathbf{Pr}(\Delta_u \ge -u/3) &\le e^{\lambda u/3} \prod_{i=1}^{u} M_{\delta_i}(\lambda) \\ &\le e^{\left(-\frac{2\lambda}{3} + \frac{p\lambda}{2(p-\lambda)}\right)u} \\ &< e^{-pu/84} \qquad \text{for } p/8 < \lambda < p/7 .\end{aligned}$$

∎

Suppose now that $\tau \ge s$ and $\mathcal{D}_{\lceil 3s/4 \rceil}$ does not occur. It follows that either $s - \rho_s \ge \lceil s/4 \rceil - 1$ or $\sigma_s \ge \lceil s/4 \rceil - 1$. In either case $Z_k < r_0$ for at least $\lceil s/4 \rceil - 1$ values of $k \in [s]$. Let us now consider the chain $W_1, W_2, \dots$ with states $\{0, 1, \dots, r_0 - 1\}$ and transition probabilities as in (4), except that

$$\mathbf{Pr}(W_{k+1} = r_0 - 1 | W_k = r) = q^{r_0 - 1} \qquad \text{for } r = 0, 1, \dots, r_0 - 1 .$$

Observe that, in the chain Z_k, states $\{0, 1, \dots, r_0 - 1\}$ can only be reached from higher-numbered states via the transition r_0 to $(r_0 - 1)$. Thus W_k has precisely the realizations that one obtains from realizations of Z_k by deleting all occurrences of states $r \geq r_0$. Therefore the problem now is to show that, for some $K > 0$ and $0 < \gamma < 1$,

$$\Pr(W_i \neq 0, i = 1, 2, \ldots, \lceil s/4 \rceil - 1) \leq K\gamma^s \tag{5}$$

Now (5) simply amounts to an assertion about the recurrence time of state 0 in W_k. Since the chain W_k is clearly finite, aperiodic, and irreducible, this follows directly from elementary Markov chain theory (see, for example, Ref. 6, p. 426, Example 19). Since the number of states and the transition probabilities of W_k depend only on p, we have $K = K(p)$, $b = \gamma(p)$. So finally,

$$\begin{aligned}\Pr(\tau \geq s) &\leq \Pr(\mathcal{D}_{\lceil 3s/4 \rceil}) + \Pr(W_i \neq 0, i = 1, 2, \ldots, \lceil s/4 \rceil - 1)\\ &\leq e^{-ps/112} + K\gamma^s,\end{aligned}$$

and (3) follows.

ACKNOWLEDGMENT

We are grateful to Gary Miller for his expert advice.

REFERENCES

[1] R. J. Anderson and E. Mayr, Parallelism and the maximal path problem, *Inf. Process. Lett.*, **24**, 121–126 (1987).

[2] R. J. Anderson and A. Aggarwal, A random NC algorithm for depth first search, *Combinatorica*, **8**, 1–12 (1988).

[3] R. J. Anderson, A. Aggarwal, and R. J. Kao, A parallel algorithm for depth first search in directed graphs, *SIAM J. Comput.*, **19**, 397–409 (1990).

[4] N. Calkin and A. M. Frieze, Probabilistic analysis of a parallel algorithm for finding maximal independent sets, *Random Struct. Alg.*, **1**, 39–50 (1990).

[5] D. Coppersmith, P. Raghavan, and M. Tompa, Parallel algorithms that are efficient on average, *Inf. Computat.*, **81**, 318–333 (1989).

[6] W. Feller, An Introduction to Probability Theory and Its Applications, Vol. I (3rd ed.), Wiley, New York, 1968.

[7] A. Goldberg, S. Plotkin, and P. Vaidya, Sublinear time parallel algorithms for matching and related problems, *Proceedings of the 29th IEEE Symposium on Foundations of Computing*, 1988.

[8] G. L. Miller and J. H. Reif, Parallel tree contraction Part 1: Fundamentals, in *Randomness and Computation*, S. Micali (Ed.), JAI Press, Greenwich, CT, 1989, pp. 47–72.

[9] J. H. Reif, Depth-first search is inherently sequential, *Inf. Process. Lett.*, 229–234 (1985).

Received February 23, 1990