A note on improving the running time of a class of parallel algorithms using randomization

Carsten Dorgerloh* Jürgen Wirtgen†

December 4, 1996

Abstract

A natural method to avoid memory access conflicts in EREW-PRAM graph algorithms is to compute a large independent set in a constant-degree-bounded conflict graph. Many EREW-PRAM algorithms use results from [CV 86], [GPS 87], which can be used to compute such a set in $\mathcal{O}(\log^* n)$ parallel time. This paper gives an $\mathcal{O}(1)$ time randomized algorithm using $\mathcal{O}(n)$ processors for that problem. Our algorithm improves with high probability the running time of many EREW-PRAM algorithms.

*Institut für Informatik V, Universität Bonn, Römerstr. 164, D-53117 Bonn, Germany, email: carsten@cs.uni-bonn.de

†Institut für Informatik V, Universität Bonn, Römerstr. 164, D-53117 Bonn, Germany, email: wirtgen@cs.uni-bonn.de

1 Introduction

We consider the problem of computing an independent set of size $\Omega(n)$ in a graph where the maximum degree is bounded by a constant on a randomized EREW-PRAM. We present an $\mathcal{O}(1)$ time algorithm for this task which returns an independent set of size $\Omega(n)$. A different $\mathcal{O}(1)$ time algorithm was developed by Dadoun and Kirkpatrick [DK 89]. The relationship between the algorithm presented here and that created by Dadoun et al. will be discussed in section 3. We then show that our algorithm together with some other techniques can be used to improve the running time of many known algorithms. In particular, we prove the following:

- A 6-bounded acyclic orientation of each planar graph can be constructed on a randomized EREW-PRAM in parallel time $\mathcal{O}(\log n)$ with high probability using $\mathcal{O}(n/\log n)$ processors.
- A 5-coloring of the vertices of a planar graph may be computed by a randomized EREW-PRAM in $\mathcal{O}(\log n)$ parallel time with high probability using $\mathcal{O}(n/\log n)$ processors.
- Given a planar graph, the connected component and undirected spanning tree problems can be solved in $\mathcal{O}(\log n)$ time on an randomized EREW-PRAM with $\mathcal{O}(n/\log n)$ processors with high probability.
- A simple cycle of size k in a planar graph, if one exists, may be computed in $\mathcal{O}(\log n)$ expected time by a randomized EREW-PRAM using $\mathcal{O}(n)$ processors.

The paper is organized as follows. Section 2 contains some definitions and notations used throughout the paper. Section 3 describes the core of our unified framework to avoid $\log^* n$-factors in many EREW-PRAM algorithms. We show how to construct a large independent set in constant time. We apply this method in section 4 to the EREW-PRAM algorithms listed above. These ideas do not only work for those problems, but for all EREW-PRAM algorithms which avoid memory access conflicts by computing a large independent set in a constant-degree-bounded conflict graph.

2 Notations and definitions

The terminology used in this paper follows that of Even [Ev 79]. Let $G = (V, E)$ be a graph. For each vertex v, $N(v)$ denotes the set of neighbors of v. As usual, we assume that the vertices of G are represented by positive numbers and that G is presented to the algorithm in the form of a set of *edge-lists* L: The graph is represented as an array of $|V|$ vertices, and each vertex u is equipped with a pointer to its list $L(u)$, a doubly-linked list that contains exactly one entry for each edge that connect u to another vertex in the graph. For implementation reasons it is convenient to assume that there is a fake edge at the end of each edge list. Additionally, each edge (u, v) appearing in the edge-list of u has a pointer to its *twin* edge (v, u) appearing in the edge-list of v. In the literature, pointers of this type are often called *cross links*.
The computational model used is the randomized EREW-PRAM. This model is a synchronized parallel computation model for which simultaneous access to any memory location by different processors is forbidden. Furthermore, each processor has access to a random number generator which returns random numbers of $\log n$ bits in constant time.

3 Finding large independent sets in constant time

In this section we show how to construct randomized an independent set of size $\Omega(n)$ in a constant-degree-bounded graph with $\mathcal{O}(n)$ processors on an EREW-PRAM in constant time. The error-probability will be bounded by a constant.
Our algorithm will be based on randomized 2-colorings. The vertices to be chosen for the independent set will all have the same property:

Definition 1 *Let $G=(V,E)$ be a graph and $c: V \to \{0,1\}$ a coloring of the vertices of G with 2 colors. A vertex $v \in V$ is proper colored, if $c(v)=0$ and all neighbors $w \in N(v)$ are colored with 1.*

Look at the following algorithm:

Algorithm 2 (RandLargeIS)

Input	:	A degree-d-bounded Graph $G=(V,E)$ ($d \in \mathcal{O}(1)$)
Output	:	An independent set LargeIS
Step 1	:	for each $v \in V$ pardo
		$c(v) \in_R \{0,1\}$
Step 2	:	LargeIS $:= \{v \in V \mid v$ is proper colored $\}$

To check that the algorithm runs on a randomized EREW-PRAM in constant time, we have only to analyse step 2. We have to show that we can implement this step in constant time using $O(n)$ processors. Each vertex, edge and twin-edge is assigned to a processor. Note that the degree of each edge is bounded by a constant d. So we have at most $(1+2d)n$ processors. In each edge (u,v) are 2 flags, where the color of the owner u of this edge and the color of the owner v of its twin-edge (v,u) should be copied. With these informations each vertex can check in constant time (at most d steps) whether it is proper colored or not. The copy-process can be implemented in 2 phases:

Phase 1: Each vertex v stores its color in each member of its edge-list $L(v)$.

Phase 2: Each edge stores its color in its twin edge.

The implementation runs in constant time and there are no memory access conflicts.
For the analysis of our algorithm we need the following technical lemma: Suppose you have n 0-1-valued random variables $X_1,\ldots,X_n$ with $\Pr(X_i=1)\geq p$. We define $X := \sum_i X_i$.

Lemma 3 *For $0<\epsilon<p/(1-p)$ and $c(\epsilon) := (1+\epsilon)(1-p)$, is*

$$\Pr(X \leq (1-c(\epsilon))n) \leq 1/(1+\epsilon).$$

PROOF: Let $Y_i := 1-X_i$ and $Y := \sum_i Y_i$. Thus $\mathrm{E}(Y) \leq (1-p)n$.

$$\begin{aligned}
\Pr(X \leq (1-c(\epsilon))n) &= \Pr(n-Y \leq (1-c(\epsilon))n) \\
&= \Pr(Y \geq c(\epsilon)n) \\
&= \Pr(Y \geq (1+\epsilon)\underbrace{(1-p)}_{\geq \Pr(Y_i=1)} n) \\
&\leq \Pr(Y \geq (1+\epsilon)\mathrm{E}(Y)) \\
\text{(Markov inequality)} \quad &\leq \frac{1}{1+\epsilon}
\end{aligned}$$

Note that $1/(1+\epsilon)$ is smaller than 1. ∎

Now let $d \in O(1)$ be the maximum degree of a vertex in G. The probability that a particular vertex v is proper colored is

$$1/2 \frac{1}{2^{\deg(v)}} \geq 1/(2^{(d+1)}) =: p.$$

Now we define $\epsilon := p$. In the following lemma we proof that **RandLargeIS** returns an independent set of size greater than a $(1-c(\epsilon))$ fraction of n with bounded probability.

Lemma 4 *Let $G=(V,E)$ be an input of algorithm* **RandLargeIS**. *Then*

$$\Pr(|\text{LargeIS}| > \frac{1}{2^{2d+2}} n) \geq \frac{1}{2^{d+1}+1}$$

and LargeIS *forms an independent set.*

PROOF: The output is clearly an independent set. Let $X_v = 1$, if v is proper colored. We have $\Pr(X_v = 1) \geq p$ and

$$\epsilon = p = \frac{1}{2^{d+1}} \text{ and } 1/(1+\epsilon) = \frac{2^{d+1}}{2^{d+1}+1}.$$

Now we calculate the fraction of V which should be the independent set.

$$c(\epsilon) = \left(\frac{2^{d+1}+1}{2^{d+1}}\right)\left(\frac{2^{d+1}-1}{2^{d+1}}\right) = \frac{2^{2d+2}-1}{2^{2d+2}}$$

$$(1-c(\epsilon)) = \frac{1}{2^{2d+2}}$$

With lemma 3 we get

$$\Pr(|\text{LargeIS}| \leq \frac{1}{2^{2d+2}} n) \leq \frac{2^{d+1}}{2^{d+1}+1}$$

Thus,

$$\Pr(|\text{LargeIS}| > \frac{1}{2^{2d+2}} n) \geq \frac{1}{2^{d+1}+1}$$

∎

The $\mathcal{O}(1)$ time algorithm proposed in [DK 89] uses a reduction of the problem of finding a large independent set in a constant-bounded-degree graph to the problem of finding such a set in list graphs (digraphs whose vertices have in- and out-degree bounded by 1), instead of using a direct imlementation like here.

4 An Application: Planar Orientations

We start by applying the subroutine presented in section 3 to a parallel EREW-PRAM algorithm of [CE 91] for computing a 6-bounded acyclic orientation of planar graphs, which runs in $\mathcal{O}(\log n \log^* n)$ time with $\mathcal{O}(n/\log n \log^* n)$ processors. Let us call this algorithm **Orient**. The computation of **Orient** is divided in $\mathcal{O}(\log n)$ phases. In phase i we find a set R of vertices of degree at most 6. Now we construct a graph $H=(R,F)$, where $(u,v) \in F$ if either $(u,v) \in E$ or u and v have a common neighbor x such that

the edges (u, x) and (v, x) are consecutive in the adjacency list of x. Observe that the maximum degree in H is $\mathcal{O}(1)$. In the next step a large independent set I in H of size $\Omega(n)$ is computed. Finally, we remove all vertices $v \in I$ from V and orient the incident edges of those vertices in the obvious way.
The time for each phase is dominated by the computation of the independent set I. This can be done e.g. by a deteterministic algorithm of [GPS 87] in $\mathcal{O}(\log^* n)$ time. We substitute this subroutine used in all algorithms listed in section 1 by the randomized algorithm **RandLargeIS** proposed in section 3 and show that the running time of all those algorithms, especially the running time of **Orient**, is in $\mathcal{O}(\log n)$ with high probability.
The proof that the algorithm **Orient** runs in $\mathcal{O}(\log n)$ time uses lemma 4 from section 3 as well as the next lemma (see [LM 86],[Ch 52]), which provides a bound on the tail of a binomial distribution. Consider a set of t independent Bernoulli trials, each with a probability p of success. The next lemma bounds the probability $B(s, t, p)$ that fewer than s successes occur in t trials when $t > 2s$ and $p < 1/2$.

Lemma 5 *For $t > 2s$ and $p < 1/2$, we have*

$$B(s, t, p) \leq \left(\frac{1-p}{1-2p}\right)(1-p)^t \left(\frac{et}{s}\right)^s \quad .$$

We will call an iteration of **Orient** *successful*, if the size of the independent set returned by algorithm **RandLargeIS** (see section 3) is at least $n/2^{2d+2}$. Let LargeIS be the independent set returned by **RandLargeIS**. Then by Lemma 4 we have the following lower bound on the probability that an iteration is successful:

$$\Pr(|\text{LargeIS}| > n/2^{2d+2}) \geq \frac{1}{2^{d+1}+1}$$

We now are ready to show that the running time of all algorithms listed in section 1 may be improved to $\mathcal{O}(\log n)$ even without loosing optimality using the algorithm **RandLargeIS** of section 3. As mentioned before, we start with the algorithm **Orient**. The proofs for the other algorithms are analogously.

Theorem 6 *The EREW-PRAM algorithm* **Orient** *computes a 6-bounded acyclic orientation of planar graphs in $\mathcal{O}(k \log n)$ parallel time with probability 1 - $o(1/n^k)$ for any constant k.*

Proof: During a successful iteration, the number of vertices in the graph is reduced by a constant fraction. Therefore we require at most $\beta \log n$ successful iterations. By Lemma 5, the probability that fewer than $s = \beta \log n$ successful iterations occur in $\alpha k s$ iterations each with probability $p = \frac{1}{2^{d+1}+1}$ of success is

$$\begin{aligned} & B(\beta \log n, \alpha\beta k \log n, 1/(2^{d+1}+1)) \leq \\ \leq \quad & \left(\frac{1-\frac{1}{2^{d+1}+1}}{1-\frac{2}{2^{d+1}+1}}\right)\left(1-\frac{1}{2^{d+1}+1}\right)^{\alpha\beta k \log n}\left(\frac{e\alpha\beta k \log n}{\beta \log n}\right)^{\beta \log n} \\ = \quad & \left(\frac{2^{d+1}}{2^{d+1}-1}\right)\left(\left(\frac{2^{d+1}}{2^{d+1}+1}\right)^{\alpha k} e\alpha k\right)^{\beta \log n} \end{aligned}$$

By some algebraic manipulation and by settting $a = 2^{d+1}$, one may derive the relation

$$\begin{aligned} B(\beta \log n, \alpha\beta k \log n, 1/(2^{d+1}+1)) &\le \frac{a}{a-1} n^{\beta \log(e\alpha k)} \left(\frac{a}{a+1}\right)^{\beta \log n^{\alpha k}} \\ &= \frac{a}{a-1} n^{\beta \log(e\alpha k) + \alpha k \beta \overbrace{\log \frac{a}{a+1}}^{<0}} \end{aligned}$$

We can choose the constant α sufficiently large so that $B(\beta \log n, \alpha\beta k \log n, 1/(2^{d+1}+1))$ is $o(1/n^k)$. ∎

5 Further results

As mentioned before, the techniques used in this paper may also be applied to many other EREW-PRAM algorithms, including:

5-coloring: Compute a coloring of a planar graph G, i.e., a partition of V into disjoint sets $V_1, \ldots, V_5$ such that each V_i is an independent set for G. [CDH 87] gave an EREW-PRAM algorithm for this task which runs in $\mathcal{O}(\log n \log^* n)$ time using $\mathcal{O}(n/\log n \log^* n)$ processors. Their algorithm, though being more complicated, is very similar to the algorithm **Orient** described above.

Connected Components, Undirected Spanning Trees: [Ha 90] presented EREW-PRAM algorithms which solve those basic graph problems for planar graphs in $\mathcal{O}(\log n \log^* n)$ time using $\mathcal{O}(n/\log n \log^* n)$ processors. In fact, his algorithms work for a much broader class of undirected graphs, the so-called linear contractible class of graphs.

Simple Cycles: [Do 96] showed that if a planar graph has a simple cycle of length k, where k is a fixed integer, such a cycle may be computed in $\mathcal{O}(\log n \log^* n)$ expected time by a randomized EREW-PRAM with $\mathcal{O}(n)$ processors.

Acknowledgements

We are grateful to Marek Karpinski for stimulating discussions which were starting points for the present note. We also wish to thank Elias Dahlhaus for helpful comments.

References

[Ch 52] Chernoff, H., *A measure of asymptotic efficiency for tests of hypothesis based on the sum of observations*, Proc. 23^{rd} Annals of Mathematic Statistics (1952), pp. 493–507.

[CDH 87] Chrobak, M., Diks, K., Hagerup, T., *Parallel 5-Colouring of Planar Graphs*, ICALP 87, Lecture Notes in Computer Science 267, pp. 304–313, Springer Verlag Heidelberg, 1987.

[CE 91] Chrobak, M., Eppstein, D., *Planar orientations with low out-degree and compaction of adjacency matrices*, Theoretical Computer Science **86** (1991), pp. 243–266.

[CV 86] Cole, R., Vishkin, U., *Deterministic Coin Tossing with Applications to Optimal Parallel List Ranking*, Information and Control **70** (1986), pp. 32–53.

[DK 89] Dadoun, N., Kirkpatrick, D. G., *Parallel Construction of Subdivision Hierarchies*, Journal of Computing Systems Science **39** (1989), pp. 153–165.

[Do 96] Dorgerloh, C. F., *A Fast Randomized Parallel Algorithm for Finding Simple Cycles in Planar Graphs*, Research Report 85150-CS, Institut für Informatik der Universität Bonn, 1996.

[Ev 79] Even, S., *Graph Algorithms*, Computer Science Press, 1979.

[GPS 87] Goldberg, A. V., Plotkin, S. A., Shannon, G. E., *Parallel Symmetry-Breaking in Sparse Graphs*, Proc. 19^{th} ACM STOC (1987), pp. 315–324.

[Ha 90] Hagerup, T., *Optimal Parallel Algorithms on Planar Graphs*, Information and Computing **84** (1990), pp. 71–96.

[LM 86] Leiserson, C. E., Maggs, B. M., *Communication-efficient parallel graph algorithms*, Proc. Parallel Processing (1986), pp. 861–868.