An Efficient Parallel Algorithm for Computing a Maximal Independent Set in a Hypergraph of Dimension 3

Elias Dahlhaus
Dept. of Computer Science
University of Bonn
5300 Bonn 1

Marek Karpinski *
Dept. of Computer Science
University of Bonn
5300 Bonn 1
and
International Computer Science Institute
Berkeley, California

Pierre Kelsen †
Dept. of Computer Science
University of Texas
Austin, TX 78712

Abstract

The paper considers the problem of computing a maximal independent set in a hypergraph (see [3] and [7]). We present an efficient deterministic NC algorithm for finding a maximal independent set in a hypergraph of dimension 3: the algorithm runs in time $O(\log^4 n)$ time on $n+m$ processors of an EREW PRAM and is optimal up to a polylogarithmic factor. Our algorithm adapts the technique of Goldberg and Spencer ([5]) for finding a maximal independent set in a graph (or hypergraph of dimension 2). It is the first NC algorithm for finding a maximal independent set in a hypergraph of dimension greater than 2.

Key words. Design of algorithms, parallel algorithms, combinatorial problems, computational complexity.

*Supported in part by the Leibniz Center for Research in Computer Science, by the DFG Grant KA 673/4–1 and by the SERC Grant GR–E 68297

†Supported by NSF Grant CCR89-10707

1 Introduction

A *hypergraph* $H = (V, E)$ consists of a set V of *vertices* and a collection E of subsets of V called *hyperedges*. The *dimension* of H is the maximum size of a hyperedge in E. An *independent set* in H is a subset of V that does not contain any hyperedge of E; an independent set is *maximal* if it is not properly contained in another independent set.

Although several efficient parallel algorithms ([5], [6], [1], [9]) are known for computing a maximal independent set in ordinary graphs (i.e., hypergraphs of dimension 2), the question of whether there is an NC algorithm for arbitrary hypergraphs is still open ([7]). In this paper we present an efficient NC algorithm for the special case where the size of each hyperedge is at most 3. The algorithm is optimal up to a polylogarithmic factor.

The only nontrivial known parallel algorithm for this problem is a randomized algorithm by Beame and Luby ([3]): they claim that their algorithm is in randomized NC if the dimension of the hypergraph is bounded by some fixed constant; their analysis, however, is incomplete ([2]).

For the definition of the EREW PRAM model and complexity classes such as NC and RNC the reader is referred to [7].

The algorithm in the present paper was independently discovered by Dahlhaus & Karpinski ([4]) and Kelsen ([8]).

2 The Algorithm and its Analysis

In this section we prove the following result.

Theorem 1 *Let H be a hypergraph of dimension 3 with n vertices and m edges. A maximal independent set in H can be computed in $O(\log^4 n)$ time on $n + m$ processors of an EREW PRAM.*

Let $H = (V, E)$ be an arbitrary hypergraph of dimension 3, i.e., the hyperedges in E are subsets of V of size at most 3. For a subset $C \subseteq V$ define

$$N_1(C) = \{v \in V - C : \exists e \in E \; e - C = \{v\}\}.$$

Thus, $N_1(C)$ contains those vertices in V forming a hyperedge with nodes from C (assuming there are no hyperedges of size 1 in H). Note that an independent set C in H is a maximal independent set in H if and only if $C \cup N_1(C) = V$.

We present an algorithm $FINDMIS$ that accepts as input a hypergraph H of dimension 3 and returns as output a maximal independent set in H. Throughout this paper n and m are used to denote the number of vertices and hyperedges, respectively, of the input graph H. The high-level structure of $FINDMIS$ is similar to that of Goldberg and Spencer's algorithm ([5]): initially, $H' = H$ and $I = \emptyset$ (I is an independent set). As long as the vertex set of H' is nonempty,

$FINDMIS$ finds a large independent set C in H', adds C to I, and updates H'. To update H', $FINDMIS$ removes the vertices of C from H', removes any hyperedge properly containing another hyperedge, and deletes all vertices that form a hyperedge of size 1. Upon termination the set I is a maximal independent set in H.

The key part of $FINDMIS$ is the procedure that finds a large independent set C in H'. Let $H' = (V', E')$ with $\mid V' \mid = p$ and $\mid E' \mid = q$. We describe a procedure, called $FINDIS$, that finds an independent set C in H' of size at least $c_0 p / \log p$ in $O(\log^2 n)$ time on $n+m$ processors; here c_0 denotes a constant whose value will be determined later. It is not hard to see that the number of calls to $FINDIS$ is $O(\log^2 n)$. Therefore, algorithm $FINDMIS$ runs in time $O(\log^4 n)$ on $n+m$ EREW processors.

At a high level $FINDIS$ is similar to procedure $FINDSET$ described in [5] (which finds a large independent set in a graph): FINDIS maintains a collection of mutually disjoint independent sets $C_1, \ldots, C_r$ in H'. We view each independent set as a *color class* with the vertices in C_i having *color* i. Initially, each vertex of H' forms a color class by itself. $FINDIS$ iteratively checks whether there is a color class C with

$$\mid C \cup N_1(C) \mid \geq c_0 \cdot \frac{p}{\log p}.$$

If there is such a color class, $FINDIS$ outputs it and stops. If no such color class exists, $FINDIS$ merges certain pairs of color classes. This is repeated until either a color class C is found with $\mid C \cup N_1(C) \mid \geq c_0 \cdot \frac{p}{\log p}$ or a single color class is left. There are two main differences between our procedure $FINDIS$ and procedure $FINDSET$ described in [5], namely how the pairs of color classes are selected for merging and how they are actually merged.

We first describe how $FINDIS$ merges color classes. Since the union of two color classes C and C' is not necessarily an independent set in H', we shall decolor some vertices in $C \cup C'$ (i.e, remove them from $C \cup C'$) when merging C and C' in order to make the resulting set independent. For graphs (or hypergraphs of dimension 2) it suffices to decolor vertices in either C or C' that are adjacent to vertices in the other class (see [5]). The decoloring for hypergraphs of dimension 3 is based on a simple fact: if a hyperedge of size 3 is contained in $C \cup C'$, then it intersects exactly one of C and C' in a single vertex. Let $D(C, C')$ denote the set

$$(N_1(C) \cap C') \cup (N_1(C') \cap C).$$

$FINDMIS$ merges C and C' by taking their union and decoloring (or removing) the vertices in $D(C, C')$.

The goal in selecting the pairs of color classes that are to be merged is to keep the total number of decolored vertices small. $FINDMIS$ accomplishes this with the help of an auxiliary graph $B(\phi, h)$, where ϕ is the current collection $C_1, \ldots, C_r$ of color classes in H' and $h > 0$. The vertices of $B(\phi, h)$ are the color classes of ϕ; C_i is adjacent to C_j in $B(\phi, h)$ iff $\mid D(C_i, C_j) \mid > h$.

Intuitively, the following lemma states that if there is no color class C for which $C \cup N_1(C)$ is large then there is a merging strategy which does not decolor too many vertices.

Lemma 1 *Let $\phi = (C_1, \ldots, C_r)$ be the current collection of color classes and let $h = \frac{c_1 p}{r \log p}$ (where c_1 is a constant whose value will be determined later). If $\mid C \cup N_1(C) \mid < c_0 p / \log p$ for all color classes C in Φ, then $B(\Phi, h)$ has less than $2c_0/c_1 \cdot r^2$ hyperedges.*

Proof. Assume for a contradiction that $B(\Phi, h)$ has at least $2c_0/c_1 \cdot r^2$ hyperedges. Define digraph B' as follows: B' has the same vertices as $B(\phi, h)$; for each hyperedge $\{C, C'\}$ of $B(\phi, h)$, exactly one of (C, C') and (C', C) is a hyperedge of B': if $\mid N_1(C) \cap C' \mid > h/2$, make (C, C') a hyperedge of B'; otherwise, (C', C) will be a hyperedge of B'. The graphs $B(\phi, h)$ and B' have the same number of hyperedges . Therefore B' contains a vertex C with outdegree at least $2c_0/c_1 \cdot r$. From the definition of B' it follows that

$$\mid N_1(C) \mid > 2c_0/c_1 \cdot r \cdot h/2 = c_0 p / \log p.$$

This clearly contradicts the assumption of the lemma.[]

Assume that for each color class C maintained by $FINDIS$ we have

$$C \cup N_1(C) < c_0 p / \log p.$$

Let $h = \frac{c_1 p}{r \log p}$. By lemma 1 the number of hyperedges in the complement $\overline{B}(\phi, h)$ of $B(\phi, h)$ is at least

$$r(r-1)/2 - 2c_0/c_1 \cdot r^2 \geq (1/4 - 2c_0/c_1) \cdot r^2.$$

Procedure $FINDIS$ invokes the procedure $MATCH$ of [5] to find a matching M in $\overline{B}(\phi, h)$ of size

$$\lceil (1/4 - 2c_0/c_1) \cdot r \rceil < (1/4 - 2c_0/c_1) \cdot r + 1.$$

$FINDIS$ merges all pairs of color classes (C, C') where (C, C') is a hyperedge of M (thereby decoloring the vertices in $D(C, C')$). Note that this merging decolors at most

$$\mid M \mid \cdot h < (c_1/4 + c_1/r - 2c_0) \cdot p / \log p < (3c_1/4 - 2c_0) \cdot p / \log p$$

vertices and reduces the number of color classes to at most

$$r - (1/4 - 2c_0/c_1) \cdot r = (3/4 + 2c_0/c_1) \cdot r = a \cdot r$$

where $a = 3/4 + 2c_0/c_1$.

As mentioned above $FINDIS$ repeats merging color classes according to the strategy we have just outlined until either $C \cup N_1(C) \geq c_0 p / \log p$ for some color class C or a single color class C_0

is left. In the former case we are done. The number of iterations in the latter case is at most $-\log p/\log a$. Thus the total number of vertices that are decolored is at most

$$-\frac{\log p}{\log a}\cdot(3c_1/4-2c_0)\cdot p/\log p<\frac{2c_0-3c_1/4}{\log a}\cdot p.$$

Hence, the size of the unique color class C_0 is at least

$$p-\frac{2c_0-3c_1/4}{\log a}\cdot p=p\cdot\left(1-\frac{2c_0-3c_1/4}{\log a}\right).$$

For $c_1=0.25$ and $c_0=0.01$ we have $1-\frac{2c_0-c_1/4}{\log a}>c_0/\log p$, i.e., C_0 is a large enough independent set for this particular choice of constants.

Procedure $FINDIS$ can be implemented to run in parallel time $O(\log^2 n)$ time on $n+m$ EREW processors using standard techniques. These techniques are described in [5] and carry over to our problem with only minor modifications. We leave this as a straightforward exercise.

3 Concluding Remarks

In this paper we have presented an efficient parallel algorithm for finding a maximal independent set in a hypergraph of dimension 3. The algorithm is similar to an algorithm of Goldberg and Spencer for finding a maximal independent set in an ordinary graph. It does not seem that the type of approach common to both algorithms is likely to lead to a fast parallel algorithm for higher dimensions. Rather it appears that an entirely new approach is needed for hypergraphs of dimension greater than 3. This will be the subject of further research.

References

[1] N. Alon, L. Babai, A. Itai, *A fast randomized parallel algorithm for the maximal independent set problem*, J. Algorithms 7, pp. 567-583, 1986.

[2] P. Beame, *Personal communication*, May 1991.

[3] P. Beame, M. Luby, *Parallel Search for Maximal Independence Given Minimal Dependence*, Technical Report TR-89-003, International Computer Science Institute, February 1989; in Proc. First ACM-SIAM SODA (1990), pp. 212-218.

[4] E. Dahlhaus, M. Karpinski, *An efficient parallel algorithm for the 3MIS problem,* Technical Report TR-89-052, September 1989, International Computer Science Institute, Berkeley, CA.

[5] M. Goldberg, T. Spencer, *A new parallel algorithm for the maximal independent set problem,* SIAM J. Computing, vol. 18, 1989, pp.419-427.

[6] M. Goldberg, T. Spencer, *Constructing a Maximal Independent Set in Parallel*, SIAM J. Disc. Math., vol. 2, 1989, pp. 322-328.

[7] R. Karp, V. Ramachandran, *Parallel algorithms for shared memory machines*, in *Handbook of Theoretical Computer Science*, J. Van Leeuwen, ed., North Holland, 1990, pp. 869-941.

[8] P. Kelsen, *An efficient parallel algorithm for finding a maximal independent set in hypergraphs of dimension 3,* manuscript, Department of Computer Sciences, University of Texas, Austin, TX, January 1990.

[9] M. Luby, *A simple parallel algorithm for the maximal independent set problem*, SIAM J. Computing, vol. 15, 1986, pp. 1036-1053.