An Efficient Parallel Algorithm for the 3MIS Problem [1]

Elias Dahlhaus * Marek Karpinski †

Abstract

The paper considers the problem of computing a maximal independent set in hypergraphs (see [Karp, Ramachandran 88] and [Beame, Luby 89]). We present an efficient deterministic parallel algorithm for the case when the maximal cardinality of any hyperedge is 3. The algorithm works in $O(\log^4 n)$ parallel time with $O(n+m)$ processors on a CREW PRAM and is optimal up to a polylogarithmic factor.

Keywords: Design of algorithms, parallel algorithms, combinatorial algorithms, computational complexity.

*Department of Computer Science, University of Bonn

†Department of Computer Science, University of Bonn, and International Computer Science Institute, Berkeley, California. Supported in part by Leibniz Center for Research in Computer Science, by the DFG Grant KA 673/2-1, and by the SERC Grant GR-E 68297

[1] We were informed recently, that Pierre Kelsen obtained independently the same result [Ke 90].

0 Introduction.

Given a hypergraph **H**$= (V, H)$ with H a collection of subsets of V. The Maximal Independent Set Problem (cf. [KR 88], [BL 89]) in a Hypergraph (*HMIS*) is the problem of finding an inclusion maximal subset $V' \subset V$ such that for no hyperedge $h \in H$, $h \subseteq V'$ (called a *maximal independent set of* H). Generally a set $V' \subset V$ is called *independent* iff for no $h \in H, h \subseteq V'$.

While the efficient NC algorithms for the maximal independent set problem restricted to graphs are known (see [KW 84], [GS 87], [Lu 85]), the fast parallel solution for the HMIS remains open. Here we present an efficient NC algorithm for HMIS restricted to hypergraphs **H**$= (V, H)$ such that each $h \in H$ has cardinality at most 3, in short the *3MIS*.

This algorithm uses similar ideas as [GS 87]. We refer also to [BL 89], where a probabilistic parallel algorithm for 3MIS is also presented.

In the first section we give the necessary terminology of the paper. The second section formulates the main result. The third section describes the global strategy of the algorithm. We describe a coloring of the vertices such that each color forms an independent set. The fourth section explains, how to unify two such colors. The fifth section shows, how to compute a "large enough" independent set. The sixth section gives a correctness analysis of the algorithm and the last section of the paper gives the complexity analysis of the algorithm.

1 Notation.

By a *hypergraph* we mean a pair **H**$= (V, H)$ such that H is a set of subsets of V. V is the set of *vertices* and H is the set of *hyperedges*. The maximal size of a hyperedge $h \in H$ is called the *dimension* of **H**. For example the graphs are hypergraphs of dimension two.

An *independent set* of a hypergraph **H**$= (V, H)$ is defined as a subset V' of V such that no hyperedge $h \in H$ is a subset of V' (cf. [KR 88]). Note that in the case of graphs this notion of an independent set coincides with the usual notion of an independent set.

By a *maximal* independent set we mean an inclusion maximal independent set.

In the whole paper, n will denote the number of vertices $n = \#V$, and m the number of hyperedges, $m = \#H$.

Since we consider only hyperedges of dimension 3, each hyperedge can be described by a data structure of size $O(n + m)$.

The computation model used in the paper is a CREW PRAM ([KR 88]). We assume, that an arithmetic operation on two numbers of length k needs $O(\log k)$ time and $O(k)$ processors. (Since we operate here only on numbers of length $O(\log n)$ the processor exponent of an arithmetic operation is not relevant for a processor analysis of the whole algorithm).

2 The Main Result.

We shall prove the following

Theorem: There exists an algorithm for computing a maximal independent set in hypergraphs of dimension 3 running in $O(\log^4 n)$ parallel time and $O(n + m)$ processors on a CREW-PRAM.

We note that the algorithm is optimal in processor-time product up to a polylogarithmic factor.

3 Global Description.

We shall adopt the technique of [GS 87] to compute iteratively at every step an independent set C such that for $W(C) := \{ x \mid \exists\, h \in H \; h \setminus C = x \}$ the set $C \cup W(C)$ has cardinality at least $c_0 \frac{k}{\log k}$, where k is the number of vertices of the hyperedge in the actual step. Afterwards $W(C) \cup C$ is deleted from the vertex set.

In the whole algorithm we have to repeat this procedure $O(\log^2 n)$ times (until only one vertex remains).

Also to compute such a set C, we proceed similarly as in [GS 87], that we have disjoint independent sets $C_1, \ldots, C_p$ and compute an edge coloring on the complete graph of $\{C_1 \ldots C_p\}$. We select a color with the smallest 'loss' and paste the independent sets C_i, C_j together which are joined by an edge of this color.

4 Pasting two colors C_i, C_j together.

Given a hypergraph **H**$= (V, H)$ and a set $\{C_1 \ldots C_p\}$ of colors.

1. Let $C_j'^{i} := C_j \setminus \{ x \mid \exists\, h \in H \; h \subseteq C_i \cup C_j \;\wedge\; h \cap C_j = \{x\} \}$
(Set of vertices x of C_j, whose addition to C_i make $C_i \cup \{x\}$ dependent)

2. Let $C_i'^{j} := C_i \setminus \{x \mid \exists h \in H \; h \subseteq C_i \cup C_j' \;\wedge\; h \cap C_i = \{x\}\}$

3. Set $C_{ij} := C_i' \cup C_j'$ $\qquad$ (C_{ij} is independent).

Lemma: C_{ij} is independent.

Proof of the Lemma: Let h be some hyperedge of **H** such that $h \subset C_i \cup C_j$. Since h has a cardinality of at most 3, $h \cap C_i$ or $h \cap C_j$ has a size of at most one. In the second case the $x \in h \cap C_j$ does not belong to $C_j'^{i}$. Therefore if $h \subseteq C_i \cup C_j'^{i}$, then $h \cap C_i$ has a size of at most one. But then the $x \in h \cap C_i$ does not belong to $C_i'^{j}$.

End of the proof of the Lemma

The **loss** of i and j, denoted by $l(i,j) = l(C_i, C_j)$ is $\#((C_i \cup C_j) \setminus C_{ij})$.

$$W(C_i) := \{\, x \notin C_i \mid \exists\, h \in H \; h \backslash C_i = \{x\} \,\}$$

is the set of vertices, whose addition to C_i generate a non independent set. It is easily seen that

$$\sum_{i,j} l(i,j) \;\leq\; \sum_i \#W(C_i).$$

The deciding step is computing an independent set such that

$$\#(C \cup W(C)) \;\geq\; c_0 \frac{k}{\log k}$$

where k is the number of vertices.

5 Computing the independent set C.

1. Let $V = \{v_1 \dots v_k\}$.
$C_i := \{v_i\}$ for each $i = 1, \dots, k$; $p = k$.

2. Repeat $\log k$ times:

2.1. Color the edges of $\{[C_i, C_j] \, : \, i,j = 1, \dots, p\}$ minimally such that no adjacent edges have the same color:
If p is odd, color the edge $[C_i, C_j]$ by $i + j \, mod \, p$;
if p is even, color $[C_i, C_j]$ for $i,j = 1, \cdots p-1$ by $i + j \, mod \, p-1$ and $[C_i, C_p]$ by $2i \, mod \, p-1$ (see also [GS 87]).
Let D_l be the set of $[C_i, C_j]$ colored by l. (Note, that in the case, that p is even, we have $q := p-1$ colors and in the case of an odd p we have p colors. It is easily seen that no adjacent edges have the same color.)

2.2. Select a color D_l such that

$$\sum \{ l(C_i, C_j) \, : \, [C_i, C_j] \in D_l \}$$

is minimal;
apply the pasting procedure of two colors for any $[C_i, C_j] \in D_l$

and unify C_{ij} to a new color.
Decolor all vertices in $C_i \cup C_j \setminus C_{ij}$.

3. C is the remaining color.

6 Correctness Analysis.

Assume, for all colors in each step $W(C_i) < \frac{C_0 k}{\log k}$.
Then for each step

$$\begin{aligned}(p-1)\left(\sum\{l(C_i,C_j) : [C_i,C_j] \in D_l\}\right) &\leq \sum_{i,j=1\ldots p} l(C_i,C_j) \\ &\leq \sum_i W(C_i) \\ &< p \cdot \frac{c_0 k}{\log k}\end{aligned}$$

Therefore we can assume

$$\sum\{\, l(C_i,C_j) : [C_i,C_j] \in D_l \,\} < \frac{c_0' k}{\log k}$$

for some constant c_0'. But then at most $\frac{c_0' k}{\log k} \cdot \log k = c_0' k$ vertices are decolored after leaving the repeat loop (Step 2.). Therefore C must contain at least $(1-c_0')k$ vertices. This is a contradiction.

7 Complexity Analysis.

Our computational model is the CREW PRAM.

1. Computing $W(C_i)$ **:**
Let $h = \{x_1, x_2, x_3\}$ or $h = \{x_1, x_2\}$.
For x_i let C^{ih} be the C_j such that $x_i \in C_j$.

If C_j appears $\#h - 1$ times as some C_k^i, then the $x_h \in h \setminus C_j$ is set to be in $W(C_j)$. This can be done by $\mathrm{O}(n+m)$ processors in $\mathrm{O}(\log n)$ time.

2. $C_j'^i$ **for** $i < j$:
For $h \in H$ such that C_j appears once delete $h \cap C_j$ from $C_j'^i$ (which was initialized as C_j).
That can be done by $\mathrm{O}(n+m)$ processors and $\mathrm{O}(\log n)$ time.

3. We get the same analysis for $C_j'^i$ and $j < i$.

4. The computation of the color of an edge $\{[C_i, C_j] : i, j = 1 \ldots p\}$ needs one processor and $\mathrm{O}(\log n)$ time, since we only use an arithmetic operation $+$ on i and j, which are bounded by n (the lengths are bounded by $\log n$). (see [GS 87]).

5. Computing the losses of each coloring needs $\mathrm{O}(n+m)$ processors and $\mathrm{O}(\log n)$ time, since we only have to compute losses and colors of pairs of old colors, where there are hyperedges contained in its union:

Let C_x be the C_i such that $x \in C_i$. Then we have to compute for each hyperedge h and each pair $x, y \in h$ the losses and the colors of $[C_x, C_y]$. The number of such $[C_x, C_y]$ is bounded by $3n$.

6. Selecting the color of smallest loss needs $\mathrm{O}(k)$ processors and $\mathrm{O}(\log n)$ time.

7. The repeat loop needs $\mathrm{O}(\log n)$ time. Therefore computing the independent set C needs $\mathrm{O}(\log^2)$ time.

8. The algorithm for computing C must be repeated $\log^2 n$ times to compute a maximal independent set (compare also [GS 87]).
Therefore computing 3MIS needs $\mathrm{O}(n+m)$ processors and $\mathrm{O}(\log^4 n)$ time.

$\square$

References

[BL 89] Beame, P. and Luby, M. *Parallel Search for Maximal Independence given Minimal Independence*, Technical Report #TR-89-003, International Computer Science Institute, Berkeley, (1989); in Proc. 1^{st} ACM SIAM SODA (1990), pp. 212-218.

[GS 87] Goldberg, M. and Spencer, T. *A New Parallel Algorithm for the Maximal Independent Set Problem*, 28^{th} FOCS (1987), pp. 161-165

[KR 88] Karp, R.M. and Ramachandran, V. *A Survey of Parallel Algorithms for Shared-Memory Machines*, Research Report No. UCB/CSD88/407, University of California, Berkeley (1988); in Handbook of Theoretical Computer Science A, MIT Press (1990), pp. 869-941.

[KW 84] Karp, R. and Widgerson, A. *A Fast Parallel Algorithm for the Maximal Independent Set Problem*, 16^{th} STOC (1984), pp. 266-272

[Ke 90] Kelsen, P., *An Efficient Parallel Algorithm for Finding a Maximal Independent Set in Hypergraphs of Dimension 3*, private communication.

[Lu 85] Luby, M. *A Simple Parallel Algorithm for the Maximal Independent Set Problem* $17^t h$ STOC (1985), pp. 1-10