An NC Algorithm for Computing a Maximal Independent Set in a Hypergraph of Bounded Valence

Syoudai, Takayosi
Faculty of Engineering, Yamaguchi University

Miyano, Satoru
Research Institute of Fundamental Information Science Kyushu University

出版情報：RIFIS Technical Report. 68, 1993-04-18. Research Institute of Fundamental Information Science, Kyushu University
バージョン：
権利関係：

RIFIS-TR-CS-68

RIFIS Technical Report

An NC Algorithm for Computing a Maximal Independent Set in a Hypergraph of Bounded Valence

Takayosi Shoudai
Satoru Miyano

April 18, 1993

Research Institute of Fundamental Information Science
Kyushu University 33
Fukuoka 812, Japan

E-mail: shoudai@rifis.sci.kyushu-u.ac.jp Phone: 092-641-1101 ex. 4472

An NC Algorithm for Computing a Maximal Independent Set in a Hypergraph of Bounded Valence

Takayoshi Shoudai
Faculty of Engineering
Yamaguchi University, Ube 755, Japan

Satoru Miyano
Research Institute of Fundamental Information Science
Kyushu University 33, Fukuoka 812, Japan

Abstract

Let $H = (V, E)$ be a hypergraph. The valence of a vertex $v \in V$ is the number of hyperedges which contain v. We give an NC algorithm for finding a maximal independent set in a hypergraph of bounded valence.

1 Introduction

A *hypergrah* $H = (V, E)$ 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 and the *valence* of a vertex is the number of hyperedges which contain the vertex. An *independent set* in H is a subset of V which does not contain any hyperedge of E. The hypergraph maximal independent set problem (HMIS) is, given a hypergraph H, to find a maximal independent set in H. The purpose of this paper is to give an NC algorithm for finding a maximal independent set in a hypergraph of bounded valence.

A maximal independent set in a hypergraph is easily computed in polynomial time by a straightforward greedy sequential algorithm. However, the algorithm is hardly parallelizable since it is P-complete [3, 9]. On the other hand, the problem of finding *any* maximal independent set of a graph (or hypergraph of dimension 2) was shown to be in NC [6, 7,

9]. We have studied a way of employing the parallel algorithm for finding any maximal independent set in a graph to solve maximal or minimal problems in NC [10].

Berger et al. [2] showed an NC approximation algorithm for the minimum set cover problem, which is known to be NP-complete [5]. Since the minimum set cover problem is easily reduced from HMIS, their algorithm also computes a maximum independent set approximately. But their algorithm does not necessarily produce a maximal independent set. Recently, Dahlhaus et al. [4, 8] studied the HMIS whose dimension is bounded by constant. They showed that if the dimension is at most 3 then HMIS is computable in NC by using the same technique as Goldberg and Spencer [6]. Moreover, an RNC algorithm is presented for the HMIS when the dimension is greater than 3 but bounded by constant [8]. But it is not known that HMIS can be parallelized efficiently if the dimension is not bounded.

In this paper, we give an HMIS algorithm which runs in $O((\log n)^{s+1}\mathrm{TMIS}(n))$ time on $\mathrm{PMIS}(n)$ processors if the maximum valence is $O((\log n)^s)$ for some fixed $s \geq 0$, where $\mathrm{TMIS}(n)$ and $\mathrm{PMIS}(n)$ are the time and the number of processors needed to find a maximal independent set in a graph with n vertices. Consequently, by using Luby's MIS algorithm [9], our algorithm runs in an $O((\log n)^{s+1}(\log(n+m))^2)$ time on EREW PRAM with n^2m processors.

2 Finding a Maximal Independent Set in a Hypergraph

Let $H = (V, E)$ be a hypergraph where $V = \{1, \cdots, n\}$ and $|E| = m$. We also let $\beta = \max\{|deg(i)| \mid i \in V\}$, where $deg(i) = \{e \in E \mid i \in e\}$.

Lemma 1 *The algorithm HMIS (Figure 1) correctly computes a maximal independent set in a given hypergraph $H_0 = (V_0, E_0)$.*

The variable W gets a maximal independent set. Let I_i, E_i, U_i, W_i and V_i be the contents of the variables I, E, U, W and V just after the ith iteration of the while-loop, respectively. For convenience, let $W_0 = \emptyset$ and $U_0 = \emptyset$. Let $U_i^* = U_0 \cup \cdots \cup U_i$. We also let X_i be the set of edges constructed during lines 7-11. Then from the algorithm we can easily see that V_0, V_{i-1} and W_i are represented as the following disjoint unions:

```
/* A hypergraph H_0 = (V_0, E_0) where H_0 = {1, ..., n} and E_0 = {e_1, ..., e_m} is given. */
/* We assume that V_0 = ⋃_{e∈E_0} e and |e_i| ≥ 2 for i = 1, ..., m. */
1   begin
2       V ← V_0; E ← E_0;
3       W ← ∅; /* W gets a maximal independent set */
4       while V ≠ ∅ do
5           begin
6               X ← ∅;
7               par e ∈ E do
8                   begin
9                       Choose two distinct vertices v, w from e ∩ V;
10                      Add the edge {v, w} to X
11                  end;
12              Find a maximal independent set I of the graph G = (V, X);
13              W ← W ∪ I;
14              V ← V − I;
15              U ← {u ∈ V | e ∩ V ⊆ W ∪ {u} for some e ∈ E};
16              par e ∈ E do if e ∩ U ≠ ∅ then delete e from E;
17              V ← V − U;
18              Y ← V − ⋃_{e∈E} e
19              W ← W ∪ Y
20              V ← V − Y;
21          end
22  end
```

Figure 1: Algorithm *HMIS*

(1) $V_i \cup W_i \cup U_i^* = V_0$.

(2) $V_{i-1} = I_i \cup U_i \cup Y_i \cup V_i$.

(3) $W_i = W_{i-1} \cup I_i \cup Y_i$.

Claim 1 For $e \in E_i$, $e \cap V_i$ contains at least two elements.

Proof. By the assumption on the input, Claim 1 obviously holds for $i = 0$. Assume that the claim holds for i and $V_{i+1} \neq \emptyset$. Let e be in E_i. Then $e \cap U_i = \emptyset$ from line 16 and $e \cap Y_i = \emptyset$ from line 18. Therefore from (2) we see that $e \cap V_i = e \cap (V_{i-1} - I_i)$. If $e \cap V_i = \emptyset$, then $U_i = V_{i-1} - I_i$ from line 15. This yields $V_i = \emptyset$ from line 17. This is a contradiction since V_i is assumed not empty. On the other hand, if $e \cap V_i = \{u\}$, then $e \cap V_i \subseteq W_{i-1} \cup I_i \cup \{u\}$. This means that u is in U_i and, therefore, $e \cap U_i \neq \emptyset$, a contradiction. Thus $|e \cap V_i| \geq 2$. □

Claim 2 W_i is an independent set for E_0.

Proof. We assume that $V_{i-1} \neq \emptyset$. Obviously, $W_0 = \emptyset$ is an independent set for E_0. Assume that W_{i-1} is an independent set for E_0. Let e be in E_0.

Case 1. $e \notin E_i$: e was deleted during the jth iteration for some $1 \leq j \leq i$. Then $e \cap U_j \neq \emptyset$. Hence there is u in $e \cap U_j \subseteq U_i^*$. By (1) u is not in W_i. Therefore we have $e \not\subseteq W_i$.

Case 2. $e \in E_i$: e is also in E_{i-1}. Then by Claim 1 there are v, w in $e \cap E_{i-1}$ with $v \neq w$ and $\{v, w\} \in X_i$. Since I_i is an independent set, $v \notin I_i$ or $w \notin I_i$. Since W_{i-1} is an independent set for E_0, we have $e \not\subseteq W_{i-1}$. Since no element in V_{i-1}, hence no element in I_i, is in W_{i-1}, v or w is not in $W_{i-1} \cup I_i$. Therefore $e \not\subseteq W_{i-1} \cup I_i$. On the other hand, $e \cap Y_i = \emptyset$ by line 18. Therefore $e \not\subseteq W_{i-1} \cup I_i \cup Y_i = W_i$. □

Claim 3 For any $u \in U_i$, there is $e \in E_{i-1}$ such that $e \subseteq W_i$.

Proof. By line 15, for $u \in U_i$ there is $e \in E_{i-1}$ such that $e \cap (V_{i-1} - I_i) \subseteq W_{i-1} \cup I_i \cup \{u\}$. Then $e \cap V_{i-1} \subseteq W_{i-1} \cup I_i \cup \{u\}$. Note that for $e \in E_{i-1}$ we have $e \cap U_{i-1}^* = \emptyset$ by line 16. Then

$$\begin{aligned} e &= e \cap (V_{i-1} \cup W_{i-1} \cup U_{i-1}^*) \quad \text{(by (1))} \\ &= (e \cap V_{i-1}) \cup (e \cap W_{i-1}) \cup (e \cap U_{i-1}^*) \\ &\subseteq W_{i-1} \cup I_i \cup \{u\}. \quad \text{(by } e \cap U_{i-1}^* = \emptyset) \end{aligned}$$

□

Proof of Lemma 1. Let t be the integer such that $V_t = \emptyset$. Then by (1) $V_0 = W_t \cup U_t^*$. From Claim 2, W_t is an independent set. Claim 3 asserts that for any $u \in U_t^*$ there is some e with $e \subseteq W_t \cup \{u\}$. Therefore W_t is a maximal independent set for H_0.

The part of finding a maximal independent set can be implemented on an EREW PRAM in $O((\log(n+m))^2))$ time using n^2m processors [9]. The other steps can also be implemented with at most the same amount of time and processors.□

3 Finding a MIS in a Hypergraph of Bounded Valence is in NC

We show that the algorithm *HMIS* runs in $O((\log n)^{s+1} TMIS(n))$ time on $PMIS(n)$ processors if the valence is $O((\log n)^s)$ for some fixed $s \geq 0$. This analysis is done by using a

classical graph theorem [1].

The alogorithm have a following top-level description:

```
    /* Let H_0 = (V_0, E_0) be a hypergraph. */
 1  begin
 2      H ← H_0; (i.e. E ← E_0; V ← V_0;)
 3      W ← ∅;   /* W gets a maximal independent set. */
 4      while V ≠ ∅ do
 5          begin
 6              G ← MAKE_GRAPH(H);
 7              I ← FINDMIS(G);
 8              V ← V − U;   /* U contains I which is found at line 7. */
 9              W ← W ∪ U';  /* U' contains I which is found at line 7. */
10              H ← MAKE_NEW_HYPERGRAPH(H);
11          end
12  end.
```

It is easy to show that the while-loop of an algorithm with such a top-level structure iterates $O((\log n)^{s+1})$ times if every FINDMIS produces an independent set I such that $|I| = \Omega(\frac{|V|}{(\log |V|)^s})$ for some $s \geq 0$. Thus, if FINDMIS can find such a maximal independent set I, an algorithm with the structure runs in $O((\log n)^{s+1} TMIS(n))$ time. By using Luby's MIS algorithm [9], every application of FINDMIS runs in $O((\log n)^2)$ time. Thus, we can have an $O((\log n)^{s+3})$ time parallel algorithm for HMIS.

Lemma 2 [1] *Let $G = (V, E)$ be an undirected graph with $|V| = n$ and $|E| = m$, and Δ be the maximum degree of G. For any maximal independent set I of G,*

$$|I| \geq \lceil \frac{n}{\Delta + 1} \rceil.$$

Then, if a graph made by MAKE_GRAPH has the maximum valence $\Delta = O((\log |V|)^s)$ then FINDMIS always produces a maximal independent set I with $|I| = \Omega(\frac{|V|}{(\log |V|)^s})$.

Let $H = (V, E)$ be a hypergraph with $V = \{1, \ldots, n\}$ and $|E| = m$ and also let β be the maximum valence of H, i.e., $\beta = \max\{|d_j| \mid j = 1, ..., n\}$, where $d_j = \{e \in E \mid j \in e\}$.

Lemma 3 *Let $G = (U, X)$ be a graph constructed by our version of MAKE_GRAPH (line 7-11 of the algorithm HMIS), that is, $U = V$ and $X = \{\{u_i, v_i\} \mid u_i$ and v_i are arbitralily chosen from $e_i \in E$ for $i = 1, \ldots, m\}$. Let Δ be the maximum valence of G. Then,*

$$\Delta \leq \beta.$$

Proof. It is easy to see that $|\{j \mid \{i,j\} \in X\}| \leq |\{e \in E \mid i \in e\}|$, for $i \in U = V = \{1, \cdots, n\}$.

Therefore,

$$\Delta = \max_i |\{j \mid \{i,j\} \in X\}| \leq \max_i |\{e \in E \mid i \in e\}| = \beta.$$

□

The maximum valence of a graph made by MAKE_GRAPH depends on the valence of each instance for HMIS. The next theorem follows from Lemmas 1, 2 and 3:

Theorem 1 *Let $H = (V, E)$ be a hypergraph with $V = \{1, \cdots, n\}$ and $|E| = m$ and $\beta = \max\{|deg(i)| \mid i \in V\}$, where $deg(i) = \{e \in E \mid i \in e\}$. If $\beta = O((\log n)^s)$ for some fixed $s \geq 0$, then Algorithm HMIS (Figure. 1) computes a maximal independent set in $O((\log n)^{s+1} TMIS(n))$ time on an EREW PRAM with $PMIS(n)$ processors.*

Thus we have the following collorary:

Corollary 1 *The hypergraph maximal independent set problem is in NC^3 for a hypergraph with n vertices of constant valence.*

We have shown that parallel MIS algorithms are useful to compute a maximal independent set in a hypergraph. However, the idea of using MIS does note seem to work unless the valence is bounded. It seems that new approaches are needed for computing a maximal independent set in a hypergraph with no assumptions.

References

[1] C. Berge. *Graphs and Hypergraphs.* North-Holland, Amsterdam, Netherlands, 1973.

[2] B. Berger, J. Rompel, and P. W. Shor. Efficient NC algorithm for set cover with application to learning and geometry. In *Proc. 30th Annual Symposium on Foundations of Computer Science*, pages 54–59, 1989.

[3] S. A. Cook. A taxonomy of problems with fast parallel algorithms. *Inform. Control*, 64:2–22, 1985.

[4] E. Dahlhaus, M. Karpinski and P. Kelsen. An efficient parallel algorithm for computing a maximal independent set in a hypergraph of dimension 3. *Inf. Process. Lett.*, 42:09–313, 1992.

[5] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-completeness.* W.H. Freeman and Company, San Francisco, 1979.

[6] M. Goldberg and T. Spencer. A new parallel algorithm for the maximal independent set problem. *SIAM J. Comput.*, 18:419–427, 1989.

[7] R. M. Karp and A. Wigderson. A fast parallel algorithm for the maximal independent set problem. *J. Assoc. Comput. Mach.*, 32:762–773, 1985.

[8] P. Kelsen. On the parallel complexity of computing a maximal independent set in a hypergraph. In *Proc. 24th ACM Symposium on Theory of Computing*, pages 339–350, 1992.

[9] M. Luby. A simple parallel algorithm for the maximal independent set problem. *SIAM J. Comput.*, 15:1036–1053, 1986.

[10] T. Shoudai and S. Miyano. Using maximal independent sets to solve problems in parallel. In *Proc. 17th Intern. Workshop on Graph-Theoretic Concepts in Computer Science WG91, (Lecture Notes in Computer Science* **570***)*, pages 126–134, 1992.