数理解析研究所講究録
第 871 巻 1994 年 154-160

Finding Maximal Cycle-Free Sets in Parallel

陳 致中 (Zhi-Zhong Chen)
Dept. of Information Engineering
Mie University
Tsu-shi, Mie 514, Japan
Email: chen@info.mie-u.ac.jp

Xin He
Dept. of Computer Science
State Univ. of New York at Buffalo
Buffalo, NY 14260, U.S.A.
Email: xinhe@cs.buffalo.edu

1 Introduction

Since Karp and Wigderson showed that the maximal independent set (MIS for short) problem is solvable in NC [7], much work has been devoted to the study of maximality problems (see [2] for a precise definition of maximality problems). Most maximality problems are defined on graphs. A typical maximality problem is to find either a maximal vertex-induced subgraph (MVIS for short) or a maximal edge-induced subgraph (MEIS for short) such that certain graph property π is satisfied by the subgraph. We are interested in only those properties π such that checking π is computationally easy (say, in NC) and π is hereditary (i.e., whenever the subgraph induced by a set S of vertices or edges satisfies π, the subgraph induced by any subset of S also satisfies π). For such a property π, the MVIS (or MEIS) problem associated with π can be solved trivially in sequential. However, it is an important open question whether every MVIS (or MEIS) problem associated with such a property π is solvable in NC or RNC. So far, only a few MVIS (or MEIS) problems have been shown to be solvable in NC or RNC. Parallel complexity of many natural and important MVIS (or MEIS) problems remains unknown. Among them is the MVIS problem associated with the property "there is no cycle". We will call this problem the *maximal vertex-induced forest* (MVIF for short) problem. The MVIF problem is natural in the sense that the MEIS problem associated with the property "there is no cycle" is the classical spanning forest problem. It is well known that the spanning forest problem has efficient NC algorithms. This motivates us to study parallel algorithms for the MVIF problem.

A sequential algorithm for the MVIF problem is to greedily add vertices as long as the induced subgraph on the chosen vertices remains acyclic. Using the union-find data structure, the sequential algorithm takes $O((n+m)\alpha(m,n))$ time, where $\alpha(m,n)$ is the functional inverse of Ackerman's function, the smallest integer i such that $A(i,\lfloor m/n \rfloor) > \log n$. However, Miyano's work shows that computing the output of this sequential algorithm is P-complete [8]. Consequently, the sequential algorithm gives no help in designing an NC algorithm for the MVIF problem.

In this paper, we present an NC algorithm for the MVIF problem, establishing that the MVIF problem is in NC. In fact, our NC algorithm is designed for a more general problem, called the *maximal vertex-induced k-jungle* (MVI-k-J for short) problem in this paper. The MVI-k-J problem is the following problem: Given a graph $G = (V, E)$ and a positive integer $k \leq |V|$, find a collection $\{U_1, \cdots, U_k\}$ of k disjoint subsets of V such that the subgraph of G induced by each U_i contains no cycle but for all $v \in V - (\cup_{1 \leq i \leq k} U_i)$ and all $U_i \in \{U_1, \cdots, U_k\}$, the subgraph induced by $U_i \cup \{v\}$ contains a cycle. Since the MVIF problem can be obtained from the MVI-k-J problem by fixing the input integer k to be 1, the latter is a natural generalization of the former. There is another reason for us to consider the MVI-k-J problem. In a very recent paper, Karger and Motwani considered the problem of finding a maximal k-jungle, which can be viewed as the *edge* counterpart of our MVI-k-J problem [6]. In [6], Karger and Motwani show that the problem of finding a maximal k-jungle is in NC. This motivates us to ask whether the MVI-k-J problem is also in NC. Our NC algorithm mentioned above gives an affirmative answer to this question. The algorithm runs in $O(\log^4 n)$ time using $O(kn^{2.376} + k^2n^2)$ processors or in $O(\log^3 n)$ time using $O(k^4n^4)$ processors on an EREW PRAM. In the design and analysis of our algorithm, some idea in Pearson and Vazirani's algorithm for finding a maximal bipartite set [9] is used. We believe that, like NC algorithms for the classical spanning forest problem and for the problem of finding a maximal k-jungle [6], our NC algorithm for the MVIF problem and the MVI-k-J problem may find interesting applications.

Unfortunately, our algorithm mentioned above is far from being optimal. This motivates us to consider in which cases the MVIF (or MVI-k-J) problem admits optimal or nearly optimal NC algorithms. However, the task does not seem to be easy. Even if we restrict the input graph in the MVIF (or MVI-k-J) problem to a constant-degree graph, we are unable to obtain a nearly optimal NC algorithm for the MVIF (or MVI-k-J) problem. Surprisingly, we do get a nearly optimal NC algorithm for the MVIF problem for planar graphs. Given an n-vertex planar graph, this algorithm runs in $O(\log^3 n)$ time with $O(n)$ processors on an EREW PRAM. It is known that an optimal NC algorithm exists for 5-coloring planar graphs [5]. This fact together with our result gives a nearly optimal NC algorithm for the MVI-k-J problem for planar graphs. It is worth mentioning that our algorithm for planar graphs is crucially based on the planarity of the input graph. In the analysis of the algorithm, the following combinatorial lemma plays an important role: If $G = (X, Y, E)$ is a bipartite planar graph such that no vertex of X has degree greater than 6 and no vertex of Y has degree less than 16 in G, then at least a constant fraction of vertices of X have degree at most 2 in G. This lemma may find applications in designing efficient parallel algorithms for other planar graph problems. For example, we have recently used this lemma to show that the maximal f-dependent set problem for planar graphs is in NC [3], which had been an open question [4].

The model of parallel computation we use is the *exclusive read exclusive write parallel random access machine* (EREW PRAM).

2 The algorithm for general graphs

We first introduce some notations. Let $G = (V, E)$ be a graph. For a vertex $v \in V$, $deg_G(v)$ denotes the degree of v in G. For $U \subseteq V$, the *subgraph of G induced by U* is the graph (U, F) with $F = \{\{u, v\} \in E : u, v \in U\}$ and is denoted by $G[U]$. An *independent set* of G is a subset U of V such that $G[U]$ contains no edge. A *maximal independent set* (MIS for short) of G is an independent set that is not properly contained in some other independent set of G. Let k be a positive integer. A *vertex-induced k-jungle* (VI-k-J for short) of G is a collection $\{U_1, \cdots, U_k\}$ of k disjoint subsets of V such that $G[U_i]$, for all $1 \leq i \leq k$, contains no cycle. A *maximal vertex-induced k-jungle* (MVI-k-J for short) of G is a VI-k-J $\{U_1, \cdots, U_k\}$ of G such that for all $v \in V - (\cup_{1 \leq i \leq k} U_i)$ and all $U_i \in \{U_1, \cdots, U_k\}$, $G[U_i \cup \{v\}]$ contains a cycle. A VI-1-J of G is simply called a *vertex-induced forest* (VIF for short) of G. Similarly, an MVI-1-J of G is simply called a *maximal vertex-induced forest* (MVIF for short) of G. The MVI-k-J problem is to find, given $\langle G = (V, E), k \rangle$ with $k \leq |V|$, an MVI-k-J of G. The MVIF problem is to find, given G, an MVIF of G.

In this section, we present an NC algorithm for the MVI-k-J problem (and hence for the MVIF problem). The algorithm is similar to Pearson and Vazirani's algorithm for finding a maximal bipartite set [9].

Algorithm 1:

```
Input: An $n$-vertex graph $G = (V, E)$ and a positive integer $k \leq |V|$.

Output: An MVI-$k$-J $\{U_1, \cdots, U_k\}$ of $G$.

1. For $1 \leq i \leq k$, set $U_i = \emptyset$.

2. While $V \neq \emptyset$, perfom the following steps:

    2.1. Construct a new graph $H = (V_H, E_H)$ as follows. For each vertex $v \in V$ and each
         $1 \leq i \leq k$, if $G[U_i \cup \{v\}]$ contains no cycle, then we introduce a new vertex $v^{(i)}$
         and put it into $V_H$. $V_H$ consists of only these newly introduced vertices. $E_H$ is the
         union of two disjoint sets $E_{H,1}$ and $E_{H,2}$. $E_{H,1}$ consists of all $\{v^{(i)}, v^{(j)}\}$ such that
         $1 \leq i \neq j \leq k$, $v^{(i)} \in V_H$, and $v^{(j)} \in V_H$. $E_{H,2}$ consists of all $\{v^{(i)}, w^{(i)}\}$ such that
         $1 \leq i \leq k$, $v \neq w$, $v^{(i)} \in V_H$, $w^{(i)} \in V_H$, and $v$ is reachable from $w$ in the graph
         $G[U_i \cup \{v, w\}]$.
    2.2. Find an MIS $I$ of $H$.
    2.3. For $1 \leq i \leq k$, compute $W_i = \{v \in V : v^{(i)} \in I\}$.
         (Comment: $W_i \cap W_j = \emptyset$ if $i \neq j$.)
    2.4. For $1 \leq i \leq k$, add all vertices of $W_i$ to $U_i$.
    2.5. Remove all vertices of $\cup_{1 \leq i \leq k} W_i$ from $V$ and also remove those vertices $v$ from $V$
         such that $G[U_i \cup \{v\}]$, for all $i$ with $1 \leq i \leq k$, contains a cycle.

3. Output $\{U_1, \cdots, U_k\}$.
```

We need to introduce several notations in order to prove the correctness and to analyze the complexity of Algorithm 1. Let $d$ be the number of executions of the body of the while-loop. For $1 \leq i \leq k$ and $1 \leq j \leq d$, let $U_i^{(j)}$ ($W_i^{(j)}$) be the content of the variable U_i (resp., W_i) after the jth execution of the body of the while-loop. For convenience, let $U_1^{(0)} = \cdots = U_k^{(0)} = \emptyset$. Obviously, $U_i^{(j)} = U_i^{(j-1)} \cup W_i^{(j)}$ for $1 \leq i \leq k$ and $1 \leq j \leq d$.

Lemma 2.1 Algorithm 1 correctly outputs an MVI-k-J of $G$.

Proof. We first show that the output $\{U_1, \cdots, U_k\}$ of Algorithm 1 is a VI-k-J of $G$. This is done by induction on the number of iterations at step 2. Before the first execution of the body of the while-loop, $\{U_1, \cdots, U_k\} = \{U_1^{(0)}, \cdots, U_k^{(0)}\}$ is clearly a VI-k-J of $G$. Assume that $\{U_1^{(j)}, \cdots, U_k^{(j)}\}$ is a VI-k-J of $G$ for $j \geq 0$. We wish to establish that $\{U_1^{(j+1)}, \cdots, U_k^{(j+1)}\}$ is still a VI-k-J of $G$. The inductive hypothesis together with the comment following step 2.3 implies that no two sets in $\{U_1^{(j+1)}, \cdots, U_k^{(j+1)}\}$ share a vertex. Thus, we need only to show that $G[U_i^{(j+1)}]$ contains no cycle for each $1 \leq i \leq k$. Assume, on the contrary, that some $G[U_i^{(j+1)}]$ contains a cycle $C$. Then, $C$ must contain at least two vertices of $W_i^{(j+1)} = U_i^{(j+1)} - U_i^{(j)}$ by Algorithm 1 and the inductive hypothesis. Let $v$ and $w$ be two vertices of $W_i^{(j+1)}$ such that $v$ is reachable from $w$ in $C$ without using any other vertex of $W_i^{(j+1)}$. By Algorithm 1, the two vertices $v^{(i)}$ and $w^{(i)}$ and the edge $\{v^{(i)}, w^{(i)}\}$ must be contained in the graph $H$ constructed in step 2.1 during the $(j + 1)$st execution of the body of the while-loop. Thus, at most one of $v^{(i)}$ and $w^{(i)}$ is contained in the MIS $I$ of $H$ computed in step 2.2. However, this implies that at most one of $v$ and $w$ is contained in $W_i^{(j+1)}$, giving us a contradiction. Hence, $\{U_1^{(d)}, \cdots, U_k^{(d)}\}$, i.e., the output of Algorithm 1 is really a VI-k-J of $G$. The maximality of the output of Algorithm 1 is obvious, ∎

Lemma 2.2 Suppose that $W_i^{(j)} \neq \emptyset$ for some $1 \leq i \leq k$ and some $2 \leq j \leq d$. Then, for all $v \in W_i^{(j)}$ and all $1 \leq j' \leq j-1$, there is some $w \in W_i^{(j')}$ such that $v$ is reachable from $w$ in the graph $G[U_i^{(j'-1)} \cup \{v, w\}]$.

Lemma 2.3 $d = O(\log n)$.

Proof. For all $1 \leq i \leq k$, all $1 \leq j \leq d$, and all $v \in W_i^{(j)}$, we construct a rooted tree $T_{i,j,v}$ with depth $j$ recursively as follows. The root of $T_{i,j,v}$ is in the $j$th level of $T_{i,j,v}$ and is labeled $v$. If $j = 1$, then $T_{i,j,v}$ consists of only its root. If $j = 2$, then the root of $T_{i,j,v}$ has exactly one son which is put in the first level and is labeled $u$, where $u$ is a vertex in $W_i^{(1)}$ and $u$ is reachable from $v$ in the graph $G[\{v, u\}]$. Note that $u$ must exist by Lemma 2.2. This finishes the base of the construction. Now, suppose that $j \geq 3$. Then, the root of $T_{i,j,v}$ has exactly two sons. One son is put in the $(j-1)$st level and is labeled $u$, and the other is put in the $(j-2)$nd level and is labeled $t$, where $u$ (t) is a vertex in $W_i^{(j-1)}$ (resp., $W_i^{(j-2)}$) and $u$ (resp., $t$) is reachable from $v$ in the graph $G[U_i^{(j-2)} \cup \{v, u\}]$ (resp., $G[U_i^{(j-3)} \cup \{v, t\}]$). By Lemma 2.2, $u$ and $t$ must exist. In $T_{i,j,v}$, the son (of the root) with label $u$ has exactly one child which is put in the $(j-2)$nd level and is labeled $s$, where $s \in W_i^{(j-2)}$ is a vertex reachable from $u$ in the graph $G[U_i^{(j-3)} \cup \{u, s\}]$. By Lemma 2.2, $s$ must exist. The subtree of $T_{i,j,v}$ rooted at the node labeled $t$ (s) is a copy of $T_{i,j-2,t}$ (resp., $T_{i,j-2,s}$). This finishes the constrution of the trees.

Fix an arbitrary vertex $v \in W_i^{(d)}$ to consider. By Algorithm 1, it is easy to see that no two nodes on different levels of $T_{i,d,v}$ have the same label. We claim that no two nodes on the same level of $T_{i,d,v}$ have the same label. Assume, on the contrary, that this is not the case. Consider the highest level (of $T_{i,d,v}$) in which two nodes $\alpha$ and $\beta$ have the same label, say $w$. Let w' be the label of the lowest common ancester of $\alpha$ and $\beta$ in $T_{i,d,v}$. Then, by the construction of $T_{i,d,v}$, there are at least two distinct paths between $w$ and w' in the graph $G[U_i^{(d)}]$. Thus, $G[U_i^{(d)}]$ must contain a cycle, a contradiction. Hence, no two nodes of $T_{i,d,v}$ have the same label. This implies that $T_{i,d,v}$ contains at least $\Omega(2^{d/2})$ vertices. Therefore, the depth $d$ of $T_{i,d,v}$ is $O(\log n)$. ∎

Theorem 2.4 Given an n-vertex graph $G$ and a positive integer $k \leq n$, an MVI-k-J of $G$ can be found in $O(\log^4 n)$ time with $O(kn^{2.376} + k^2n^2)$ processors or in $O(\log^3 n)$ time with $O(k^4n^4)$ processors.

3 Algorithms for planar graphs

To design a nearly optimal $NC$ algorithm for the MVIF problem for planar graphs, we first consider a special case where the input graph is a bipartite planar graph $G = (X, Y, E)$ with $deg_G(x) \leq 6$ for all $x \in X$. The algorithm for general planar graph uses this special case as a subroutine. The following lemma is well-known [1].

Lemma 3.1 Let $G = (X, Y, E)$ be a bipartite planar graph. Then $|E| \leq 2(|X| + |Y|) - 4$.

We next show a combinatorial lemma that will play a key role in our algorithm.

Lemma 3.2 Let $G = (X, Y, E)$ be a bipartite planar graph with $deg_G(x) \leq 6$ for all $x \in X$ and $deg_G(y) \geq 16$ for all $y \in Y$. Then, at least $|X|/6$ vertices of X have degree at most 2 in G.

Proof. Let us assume first that X contains no vertex x with $deg_G(x) = 0$. Partition X into two subsets $X_{\leq 2}$ and $X_{\geq 3}$ as follows:

$$X_{\leq 2} = \{x \in X \ : \ 1 \leq deg_G(x) \leq 2\} \text{ and } X_{\geq 3} = \{x \in X \ : \ 3 \leq deg_G(x) \leq 6\}.$$

Then, we have:

$$|E| = \sum_{x \in X_{\leq 2}} deg_G(x) + \sum_{x \in X_{\geq 3}} deg_G(x) \leq 2|X_{\leq 2}| + 6|X_{\geq 3}|.$$

We also know that $|E| = \sum_{y \in Y} deg_G(y) \geq 16|Y|$. Thus, $16|Y| \leq 2|X_{\leq 2}| + 6|X_{\geq 3}|$ and so $|Y| \leq (|X_{\leq 2}| + 3|X_{\geq 3}|)/8$. On the other hand, by Lemma 3.1, we have:

$$|X_{\leq 2}| + 3|X_{\geq 3}| \leq \sum_{x \in X_{\leq 2}} deg_G(x) + \sum_{x \in X_{\geq 3}} deg_G(x) = |E| < 2(|X_{\leq 2}| + |X_{\geq 3}| + |Y|).$$

Therefore,

$$|X_{\leq 2}| > |X_{\geq 3}| - 2|Y| \geq |X_{\geq 3}| - (|X_{\leq 2}| + 3|X_{\geq 3}|)/4.$$

This implies that $|X_{\geq 3}| \leq 5|X_{\leq 2}|$. Since $|X_{\leq 2}| + |X_{\geq 3}| = |X|$ by our assumption, it follows that $|X_{\leq 2}| \geq |X|/6$ as to be shown.

Now suppose that X contains some vertices x with $deg_G(x) = 0$. Let n_i be the number of vertices $x \in X$ with $deg_G(x) = i$ for $0 \leq i \leq 2$, and let $n_{\geq 1}$ be the number of vertices $x \in X$ with $deg_G(x) \geq 1$. By the arguments above, we know that $n_1 + n_2 \geq n_{\geq 1}/6$. Thus, there are at least $n_0 + n_1 + n_2 \geq |X|/6$ vertices in X with degree at most 2 in G. ∎

We are now ready to show our main lemma of this section.

Lemma 3.3 Let $G = (X, Y, E)$ be a bipartite planar graph with $deg_G(x) \leq 6$ for all $x \in X$. Then, there are two disjoint subsets X', X'' of X such that (i) $|X' \cup X''| \geq c|X|$ for some constant $c > 0$, (ii) $G[Y \cup X']$ contains no cycle, and (iii) $G[Y \cup X' \cup \{x\}]$ contains a cycle for each $x \in X''$. Moreover, X' and X'' can be found in $O(\log n)$ time using $O(n)$ processors, where $n = |X| + |Y|$.

Proof. Consider the following algorithm for finding X' and X'':

Algorithm 2:

1. Compute $Y_{\geq 16} = \{y \in Y \ : \ deg_G(y) \geq 16\}$ and $Y_{\leq 15} = \{y \in Y \ : \ deg_G(y) \leq 15\}$.
2. Let $G' = G[X \cup Y_{\geq 16}]$. Compute $X_{\leq 2} = \{x \in X \ : \ deg_{G'}(x) \leq 2\}$.
3. Construct a graph $K = (X_{\leq 2}, E_K)$ where E_K consists of all $\{x_1, x_2\}$ such that x_1 and x_2 are adjacent to a common $y \in Y_{\leq 15}$ in G.
4. Find an MIS I of the graph K.
5. Compute a spanning forest F of $G'' = G[I \cup Y_{\geq 16}]$.
6. Compute $X'' = \{x \in I \ : \ deg_F(x) = 1 \text{ and } deg_{G''}(x) = 2\}$.
7. Set $X' = I - X''$.

We will show that X' and X'' indeed satisfy the conditions stated in the lemma. We first estimate $|X' \cup X''|$ ($= |I|$). Since $G' = G[X \cup Y_{\geq 16}]$ satisfies the conditions of Lemma 3.2, we know that $|X_{\leq 2}| \geq |X|/6$. The maximum degree of K is obviously no more than $6 \times 15 = 90$. This fact and the maximality of I imply that $|I| \geq |X_{\leq 2}|/91$. Thus, $|I| \geq |X|/546$ and the condition (i) is satisfied.

We next show that the condition (ii) is satisfied. First, note that in the graph G'', each vertex $x \in I \subseteq X_{\leq 2}$ has degree at most 2 by step 2. Let C be a connected component of G'' and let T be the spanning tree of C contained in F. Consider any edge $e = (x, y)$ in C that is not a tree edge of T. Since C, being a subgraph of G'', is bipartite, we must have that $x \in I$ and $y \in Y_{\geq 16}$. Since x has degree at most 2 in C, x must be a leaf vertex of T and is hence contained in X''. Therefore, if we delete all vertices of X'' from G'', we will delete all non-tree edges of the trees contained in F and thus destroy all cycles of G''. So the graph $G[X' \cup Y_{\geq 16}]$ contains no cycle. We claim that the graph $G[X' \cup Y]$ also contains no cycle. Assume, on the contrary, that $G[X' \cup Y]$ contains a cycle. Then this cycle must contain a vertex $y \in Y_{\leq 15}$. Let x_1 and x_2 be the two neighbors of y in the cycle. Obviously, x_1 and x_2 are contained in X' and thus in $X_{\leq 2}$. By step 3, x_1 and x_2 are connected by an edge in K. However, this is impossible because $X' \subseteq I$ and I is an independent set of K. Therefore, the condition (ii) is also satisfied. It is easy to prove that the condition (iii) is satisfied. ∎

Lemma 3.3 suggests the following algorithm for computing an MVIF U of a bipartite planar graph $G = (X, Y, E)$ with $deg_G(x) \leq 6$ for all $x \in X$:

Algorithm 3:

```
1. Initialize $U$ as $Y$.
2. Repeat the following steps until $X = \emptyset$:
    2.1. Compute the connected components, say $T_1, \cdots, T_k$, of $G[U]$. (Note that each $T_i$
         is a tree.)
    2.2. Remove from $X$ all vertices $x$ such that $G[U \cup \{x\}]$ contains a cycle. (Note that
         $G[U \cup \{x\}]$ contains a cycle iff $x$ is adjacent to at least two vertices of $T_i$ for some
         $i$.)
    2.3. Introduce $k$ new vertices $t_1, \cdots, t_k$ and set $Y = \{t_1, \cdots, t_k\}$.
    2.4. Construct a new bipartite planar graph $H = (X, Y, E_H)$, where $E_H$ consists of all
         $\{x, t_i\}$ such that $x$ is adjacent to some vertex of $T_i$ in graph $G$.
    2.5. Find two subsets $X'$, $X''$ of $X$ from $H$ such that $X'$ and $X''$ satisfy the conditions
         of Lemma 3.3 with the graph $G$ in Lemma 3.3 replaced by $H$.
    2.6. Add all vertices of $X'$ to $U$ and remove all vertices of $X' \cup X''$ from $X$.
3. Output $U$.
```

Lemma 3.4 Let $G = (X, Y, E)$ be a bipartite planar graph with $deg_G(x) \leq 6$ for every $x \in X$. Then, an MVIF U of G with $Y \subseteq U$ can be found in $O(\log^2 n)$ time using $O(n)$ processors, where $n = |X| + |Y|$.

We are now ready to show our main result of this section. Let $G = (V, E)$ be a planar graph. Lemma 3.4 suggests the following algorithm for finding an MVIF U of G:

Algorithm 4:

1. Find an independent set X of G such that $|X| \geq |V|/42$ and $deg_G(x) \leq 6$ for each $x \in X$.
2. Recursively call the algorithm to find an MVIF U' of the graph $G' = G[V - X]$.
3. Compute the connected components, say $T_1, \cdots, T_k$, of $G[U']$.
4. Remove from X all vertices x such that $G[U' \cup \{x\}]$ contains a cycle.
5. Introduce k new vertices $t_1, \cdots, t_k$ and set $Y = \{t_1, \cdots, t_k\}$.
6. Construct a bipartite planar graph $H = (X, Y, E_H)$ where E_H consists of all $\{x, t_i\}$ such that x is adjacent to some vertex of T_i in graph G.
7. Find an MVIF U'' of H such that $Y \subseteq U''$.
8. Output $U = U' \cup (U'' - Y)$.

Theorem 3.5 An MVIF of a planar graph G with n vertices can be found in $O(\log^3 n)$ time with $O(n)$ processors.

Corollary 3.6 Given an n-vertex planar graph $G = (V, E)$ and a positive integer $k \leq |V|$, an MVI-k-J $\{U_1, \cdots, U_k\}$ of G can be found in $O(\log^3 n)$ time with $O(n)$ processors.

References

[1] J. A. Bondy and U. S. R. Murty, "Graph Theory with Applications," North-Holland, New York, 1980.

[2] Z.-Z. Chen and S. Toda, The Complexity of Selecting Maximal Solutions, *in* "Proceedings, 8th IEEE Conf. on Structure in Complexity Theory 1993," pp. 313-325; Also to appear in *Information and Computation.*

[3] Z.-Z. Chen, The Maximal f-Dependent Set Problem for Planar Graphs Is in *NC*, *In preparation.*

[4] K. Diks, O. Garrido, and A. Lingas, Parallel Algorithm for Finding maximal k-Dependnet Sets and Maximal f-Matchings, *Proc. 2nd International Symposium on Algorithms*, Lecture Notes in Computer Science **557** (1991) 385-395.

[5] T. Hagerup, M. Chrobak, and K. Diks, Optimal Parallel 5-Colouring of Planar Graphs, *SIAM J. Comput.* **18** (1989), 288-300.

[6] D.R. Karger and R. Motwani, Derandomization through Approximation: An *NC* Algorithm for Minimum Cuts, to appear *in* "Proceedings, 26th ACM Sympos. on Theory of Comput. 1994."

[7] R. M. Karp and A. Wigderson, A Fast Parallel Algorithm for the Maximal Independent Set Problem, *Journal of ACM* **32** (1985), 762-773.

[8] S. Miyano, The Lexicographically First Maximal Subgraph Problems: P-Completeness and NC Algorithms, *Math. Systems Theory* **22** (1989), 47-73.

[9] D. Pearson and V. V. Vazirani, Efficient Sequential and Parallel Algorithms for Maximal Bipartite Sets, *Journal of Algorithms* **14** (1993), 171-179.