PROCEEDINGS OF THE INTERNATIONAL CONFERENCE
ON FIFTH GENERATION COMPUTER SYSTEMS 1988,
edited by ICOT.

PARALLEL COMPLEXITY AND P-COMPLETE PROBLEMS

Satoru Miyano

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

ABSTRACT

The class **NC** consists of problems solvable by parallel algorithms running in time $O((\log n)^k)$ using polynomial number of processors for some $k \geq 0$. It is strongly believed that **P**≠**NC**. If **P**≠**NC** is assumed, the P-completeness of a problem implies that no efficient parallel algorithm exists for the problem. This paper presents very general P-completeness theorems which yield a new series of P-complete problems including the lexicographically first maximal independent set problem (Cook 1983). We also give a method of finding parallelism in some kind of sequential algorithms.

1 INTRODUCTION

An important role of the parallel complexity theory is to gain insights into *inherent* parallelism in various types of computing problems. It is intended to provide knowledges for answering the following basic questions:

1. What kind of problems allow fast efficient parallel algorithms?
2. What kind of problems are inherently sequential?

Namely, it aims at understanding of the range of problems which allow fast efficient parallel algorithms. Simultaneously, it tries to capture theoretical limits of parallel computations.

It has been observed that there are some problems which can be solved by easy polynomial time sequential algorithms but do not seem to allow any fast parallel algorithms. Recent researches show that these problems are P-complete. On the other hand, the class **NC**, identified by Pippenger (1979), is understood as the class of problems which allow fast efficient parallel algorithms. The class **NC** is a subclass of **P** but, unfortunately, nobody has succeeded in proving **P**≠**NC** although it is strongly believed that **P**≠**NC** like **NP**≠**P** question. If **P**≠**NC** is approved, a proof that a problem is P-complete will be a *social* proof of nonparallelizability.

This paper gives very general P-complete theorems concerning graph algorithms. These theorems yield a new series of P-complete problems arising from graph optimization problems including the lexicographically first maximal independent set problem (Cook 1983). Potentially, infinitely many nontrivial P-complete problems can be derived.

This paper is also concerned with a method of finding parallelism in problems. Parallelism can be considered in two ways. One is parallelism in existing sequential algorithms (Kuck 1977). The other is inherent parallelism in problems themselves that requires more mathematical analysis. A method we present in this paper locates between these two parallelisms. This method is very helpful to show that some problems are in **NC**. We exemplify the method through some applications.

2 EFFICIENT PARALLEL ALGORITHMS AND NC

NC is a mnemonics for Nick's Class (Nick Pippenger 1979) named by S.A. Cook in recognition of his contribution. The purpose of this section is to convince that **NC** is a reasonable class which provides a yardstick measuring parallel complexity of problems.

2.1 Efficient Parallel Algorithms

Many formal parallel computation models have been considered (Cook 1981). A broadly accepted model is the parallel RAM (PRAM) model in which a number of processors work together synchronously and commu-

nicate with a common random access memory. The PRAM model is further classified according to read and write access abilities as CREW PRAM (Concurrent Read Exclusive Write), EREW PRAM and CRCW PRAM.

Problems for which we consider parallel complexity are formulated as follows:

Definition A *problem* (or *search problem*) S with a size parameter $h(n)$ is a family $\{S_n\}_{n\geq 1}$ of relations $S_n \subseteq \{0,1\}^n \times \{0,1\}^{h(n)}$ for $n \geq 1$. For $n \geq 1$, $x \in \{0,1\}^n$ is called an *instance* and an object $y \in \{0,1\}^{h(n)}$ satisfying $S_n(x,y)$, if any, is called a *solution* for x.

Example 2.1 A *maximal independent set* (MIS) of an undirected graph is a maximal set U of vertices such that no two vertices in U are adjacent. The problem of finding a MIS is formulated in the following way: A graph with n vertices is represented by an $n \times n$-adjacency matrix and a subset of vertices is represented by an n-bit vector. Then MIS=$\{\text{MIS}_n\}$ is defined only for integers of the form n^2 as

$$\text{MIS}_{n^2} \subseteq \{0,1\}^{n^2} \times \{0,1\}^n,$$

where for $(x,y) \in \text{MIS}_{n^2}$, x is a symmetric matrix representing an undirected graph with n vertices and y a bit vector representing a MIS in the graph.

Definition A parallel algorithm on a PRAM solving a problem is *efficient* if, given an input of size n, it runs

(1) in time $O((\log n)^k)$ for some constant $k \geq 0$,

(2) with a polynomial number of processors.

This definition is based on the observation that the time $O((\log n)^k)$ is very fast and a polynomial number of processors is feasible.

The PRAM model may not seem very realistic in the practical sense. A more realistic model is, for example, a network model. But there are some works showing that PRAM is a good model. For example, Alt et al. (1987) showed that the shared memory of any EREW PRAM with n processors and m cells of shared memory can be emulated by an n-processor module parallel computer with a slow down of $O(\log m)$. Ranade (1987) proved that one step of an n-processor CRCW PRAM can be emulated on an n-processor FFT network in time $O(\log n)$ with high probability, using an FIFO queue of size $O(1)$ at each node. Hence the gap is only $O(\log n)$ while keeping the number of processors.

2.2 Uniform Circuits and Parallel RAM

The class **NC** is defined by the uniform circuit model that is suited for classifying problems precisely.

Definition A *circuit* α with n inputs and m outputs is a finite labelled directed acyclic graph satisfying the followings: There are exactly n nodes of indegree 0 called *inputs* which are labelled with $x_1, ..., x_n$, respectively. The nodes of indegree 2 (resp. indegree 1) are labelled with either $\vee$ (or) or $\wedge$ (and) (resp. $\neg$ (not)). Exactly m nodes labelled with $y_1, ..., y_m$ are specified as *outputs*. We denote by $size(\alpha)$ (resp. $depth(\alpha)$) the number of nodes in α (resp. the length of the longest path from some input to some output).

Definition We say that a circuit family $\{\alpha_n\}_{n\geq 1}$ with output size $h(n)$ computes a function $f = \{f_n\}_{n\geq 1}$, if each α_n computes f_n, where α_n is a circuit with n inputs and $h(n)$ outputs and $f_n : \{0,1\}^n \to \{0,1\}^{h(n)}$. For a search problem $S = \{S_n\}_{n\geq 1}$, we say that $\{\alpha_n\}_{n\geq 1}$ *solves* S if the function $\{f_n\}_{n\geq 1}$ computed by $\{\alpha_n\}_{n\geq 1}$ satisfies $S_n(x, f_n(x))$ for all $n \geq 1$.

Several kinds of uniformities of circuits have been proposed (Ruzzo 1981, Cook 1981, 1983, 1985). One of

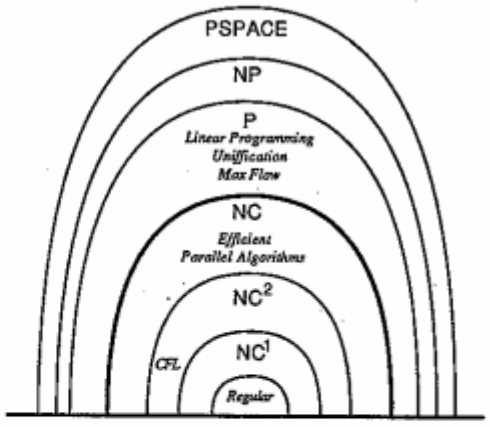

Fig. 2.1 Overview of Complexity Classes

them defines that a circuit family $\{\alpha_n\}_{n\geq 1}$ is "uniform" (called *log uniform*) if, given n in unary, the description of the nth circuit α_n is log space computable[1]. But **NC** defined below does not change by the choice of uniformity (Ruzzo 1981). The location of **NC** is shown in Fig. 2.1 together with its neighbors.

[1] A function f is log space computable if it is computed by a deterministic off-line Turing machine using $O(\log n)$ worktape space.

Definition

(1) $\mathbf{NC}^k = \{S \mid S$ is solvable by a uniform circuit family $\{\alpha_n\}_{n\geq 1}$ such that $size(\alpha_n)$ is bounded by some polynomial and simultaneously $depth(\alpha_n) = O((\log n)^k)\}$.

(2) $\mathbf{NC} = \bigcup_{k\geq 0} \mathbf{NC}^k$.

From the above arguments and the following result, we see that **NC** is a reasonable class.

Theorem 2.1 (Stockmeyer and Vishkin 1984)

$$\mathbf{NC}^k \subseteq \text{CRCW-PRAM}(n^{O(1)}, O((\log n)^k)) \subseteq \mathbf{NC}^{k+1},$$

where CRCW-PRAM$(n^{O(1)}, O((\log n)^{O(1)}))$ is the class of problems solvable on a CRCW PRAM with polynomial number of processors in time $O((\log n)^k)$ for some constant $k \geq 0$.

In the arguments above, we have ignored constants appearing in O-notation and the degrees of polynomials. In parallel computations, constants affect very seriously and we should be more careful about these things. For example, the Batcher's sorting network is of depth $1/2 \log n(1 + \log n)$. On the other hand, the AKS sorting network improved by Cole and Ó'Dúnlaing (1986) is still of depth about $200 \cdot \log n$. It should be noted that $200 > \log n$ even if n is the total number of atoms in the solar system.

However, the class **NC** and the hierarchy within it grasp the parallel complexity of problems and deepen the understanding of the problems. As was experienced for sequential computations, the notion of complexity directs people to design faster and more efficient algorithms.

3 CONVINCING THE HARDNESS OF PARALLELIZATION

No mathematical proof has been given showing that **P**≠**NC**. Even a proof separating NP from $\mathbf{NC}^1$ has not been known. The best result known is that $\mathbf{NC}^0 \neq \mathbf{NC}^1$ (Furst et al. 1981). There is, however, a strong belief that **P**≠**NC**.

Assuming that **P**≠**NC**, we can prove that no P-complete problem allows any parallel algorithm running in time $O((\log n)^{O(1)})$ with $n^{O(1)}$ processors. Thus the P-completeness plays a very important role to convince the hardness of parallelization.

The knowledge that a problem is P-complete provides algorithm designers valuable information about the approaches they should choose. It would relieve wasting efforts for devising drastically fast parallel algorithms and, instead, direct toward ways which lead to useful algorithms. Proofs of P-completeness may also tell us which parts of problems are hard to parallelize.

The P-completeness due to Cook (1985) adopted the $\mathbf{NC}^1$-reducibility that uses $\mathbf{NC}^1$-computable $O(\log n)$ depth uniform circuits with oracle gates. However, we use an extended version of the many-one log space reducibility (Hopcroft and Ullman 1979) instead of the $\mathbf{NC}^1$-reducibility because of the following reasons. It is easier to find required log space computable functions which are also computable in $\mathbf{NC}^2$. Above all, all known P-completeness of natural problems can also be shown via log space reductions.

Definition Let S and T be problems of size parameters $g(n)$ and $h(n)$, respectively. We say that S is *log space reducible* to T, denoted $S \leq^{\log} T$, if there are log space computable functions $f = \{f_n\}_{n\geq 1}$ and $g = \{g_n\}_{n\geq 1}$ with $f_n : \{0,1\}^n \to \{0,1\}^{p(n)}$ and $g_n : \{0,1\}^n \to \{0,1\}^{q(n)}$ such that for every $x \in \{0,1\}^n$ the following statement holds:

$$\forall z \in \{0,1\}^{h(p(n))}\ [S_n(x, g_{h(p(n))}(z)) \Longleftrightarrow T_{p(n)}(f_n(x), z)]$$

If $S \leq^{\log} T$ and an algorithm solving T is given, then S can be solved as follows: For an instance x of S, first compute $f(x)$, then find a solution z for $f(x)$ by the algorithm solving T. Finally, compute $g(z)$, which is guaranteed to be a solution for x.

The above definition is also a natural extension of the many-one reducibility between sets and a modified version of the $\mathbf{NC}^1$-reducibility.

Proposition 3.1

(1) *The relation* $\leq^{\log}$ *is transitive.*

(2) *If* $S \leq^{\log} T$ *and* $T \in$**NC**, *then* $S \in$**NC**.

Definition A problem $\hat{S}$ is said to be *P-complete* if the following conditions are satisfied:

(1) $\hat{S}$ is in **P**.

(2) For each problem S in **P**, $S \leq^{\log} \hat{S}$.

Proposition 3.1 implies easily the following fact.

Proposition 3.2

No P-complete problem is in NC *if* P≠NC.

After Cook (1974), some amount of P-complete problems were reported in Jones and Laaser (1977), Ladner (1977), Goldschlager (1979). Then some important problems have been shown P-complete; the linear programming (Dobkin et al. 1979), the maximum flow problem (Goldschlager et al. 1982), and the unifiability (Dwork et al. 1984, Yasuura 1984). It should be emphasized that proving the P-completeness of a problem is just the start of work on that problem. Even if a problem is shown P-complete, it just tells us that no drastic speed up may be expected theoretically. However, parallelism may help speeding up constant times or reducing the degree of polynomial of the time complexity. For example, the maximum flow problem that is P-complete allows an $O(n^2 \log n)$ parallel algorithm using n processors (Shiloach and Vishkin 1982). The best sequential algorithm runs in $O(n^3)$ time.

4 NEW SERIES OF P-COMPLETE PROBLEMS

It is rumored that, at present, several thousands of natural NP-complete problems have been found. Compared with NP-complete problems, the number of known P-complete problems is small (Miyano et al. 1988) although the first P-complete problem was found by Cook (1974) only a few years after his paper on NP-complete problems. One of the reasons why P-completeness did not attract very much is that it was concerned only with space complexity as was done in Cook (1974) and NC was not identified until Pippenger (1979) while NP was heating up. However, the P-completeness is now very important in evaluating parallel complexity of problems.

An important contribution was made in the field of NP-complete problems by Lewis and Yannakakis (1980), Yannakakis (1981), Watanabe et al. (1981), Asano and Hirata (1982). They unified NP-completeness proofs for graph optimization problems and established very general NP-completeness theorems.

In this section we prove very general P-completeness theorems that cover many problems solvable by polynomial time greedy algorithms. These are the first results in this direction.

4.1 Greedy Algorithms and P-Completeness

Greedy algorithms are of very sequential nature (Anderson and Mayr 1984). For example, a straightforward sequential greedy algorithm finds a MIS called the *lexicographically first maximal independent set* (LFMIS) in polynomial time.

Valiant (1982) noted that LFMIS does not seem to allow any efficient parallel algorithm. This was confirmed by the P-completeness of LFMIS due to Cook (1983). Then problems similar to LFMIS have been investigated to some extent (Anderson and Mayr 1984, 1987, Luby 1985, Miyano 1987, 1988a, 1988b, Reif 1985). Reif (1985) showed the computing the lexicographically first depth-first search tree is P-complete. Further, the lexicographically first maximal path problems is also shown P-complete (Anderson and Mayr 1987).

The problem we consider is the *lexicographically first maximal subgraph problem for a graph property* π (abbreviated LFMSP(π)) that involves LFMIS as a special case.

LFMSP(π)

Instance: A graph (directed graph) $G = (V, E)$ with $V = \{1, ..., n\}$.

Problem: Compute the lexicographically first maximal (abbreviated lfm) subset U of V such that the vertex-induced subgraph $G[U]$ of U satisfies the property π.

4.2 P-Completeness Theorems

A graph property π is said to be *nontrivial* on a graph family D if infinitely many graphs in D satisfy π and some graph in D violates π. The property π is said to be *hereditary* on induced subgraphs if, whenever a graph G satisfies π, all subgraphs of G also satisfy π.

Examples of nontrivial hereditary properties are *independent set, planar, bipartite, outerplanar, forest, edge graph, chordal, interval graph, maximum degree k, comparability graph, without cycles of length k, complete bipartite, transitive, symmetric, uniconnected,* etc.

For each such property π, LFMSP(π) can be computed in polynomial time by the greedy algorithm in Algorithm 4.1 since π is polynomial time testable.

Our main results are the following theorems.

```
begin
  U ← ∅;
  for i ← 1 to n
    if G[U ∪ {i}] satisfies π then U ← U ∪ {i}
  end for
end
```

Algorithm 4.1 Algorithm for LFMSP(π)

Theorem 4.1 *Let π be a polynomial time testable nontrivial property on* **undirected graphs** *which is hereditary on induced subgraphs. Then LFMSP(π) is P-complete.*

Theorem 4.2 *Let π be a polynomial time testable nontrivial property on* **directed graphs** *which is hereditary on induced subgraphs. Then LFMSP(π) is P-complete.*

Theorem 4.3 *Let π be a hereditary polynomial time testable property which is nontrivial on undirected* **planar bipartite graphs** *and satisfied by all independent edges. Then LFMSP(π) restricted to planar bipartite graphs is P-complete.*

4.3 Reducing LFMIS

As a basis of reduction we use the following lemma (Miyano 1987).

Lemma 4.4 *The following problems are P-complete.*

(1) *LFMIS restricted to planar graphs of degree 3.*

(2) *LFMIS restricted to bipartite graphs of degree 3.*

(3) *The lfm subgraph of maximum degree one problem restricted to planar bipartite graphs of degree 3.*

Let H be a connected undirected graph. A vertex c is called a *cutpoint* of H if deletion of c from H separates the graph into at least two connected components. A subgraph consisting of a resulting connected component together with c and the edges joining c and the component is called a *component relative to c*. A connected graph without any cutpoint is called *biconnected*.

For a connected graph H, we define the *α-sequence* α_H of H in the following way. If H is not biconnected, let c be any cutpoint of H and let $H_1, ..., H_{j(c)}$ be connected components relative to c. Then $\alpha_{c,H} = (|H_1|, ..., |H_{j(c)}|)$, where $|H_i|$ represents the number of vertices in H_i and we assume $|H_1| \geq \cdots \geq |H_{j(c)}|$. Then α_H is defined by $\alpha_H = \min\{\alpha_{c,H} \mid c \text{ is a cutpoint of } H\}$, where min is the minimum with respect to the lexicographic order on sorted lists of positive integers. Let c_H be any cutpoint with $\alpha_H = \alpha_{c_H,H}$. If H is biconnected, we defined $\alpha_H = (|H|)$ and c_H be any vertex. For a graph G with connected components $G_1, ..., G_t$, the *β-sequence* β_G of G is $(\alpha_{G_1}, ..., \alpha_{G_t})$, where $\alpha_{G_1} \geq \cdots \geq \alpha_{G_t}$. It should be noticed that any set of β-sequences has a minimum since lists are sorted.

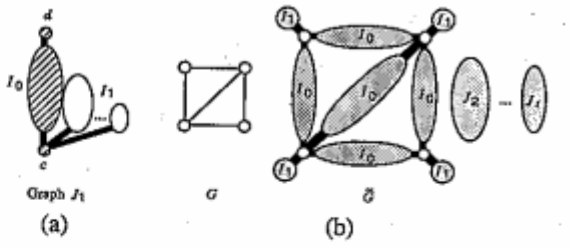

Fig. 4.1

Notation For subsets V_1, V_2 of a linearly ordered set V, we denote $V_1 < V_2$ if $v_1 < v_2$ for any $v_1 \in V_1$ and any $v_2 \in V_2$.

Proof of Theorem 4.1 As shown in Lewis and Yannakakis (1980), if a property π is nontrivial and hereditary, it follows from Ramsey's Theorem that either π is satisfied by all independent sets of vertices or π is satisfied by all cliques.

Since π is nontrivial, there exists an undirected graph J such that $\beta_J = \min\{\beta_G \mid G \text{ is an undirected graph violating } \pi\}$. Let $J_1, ..., J_t$ be the connected components of J that are sorted as $\alpha_{J_1} \geq \cdots \geq \alpha_{J_t}$. Hence the β-sequence of J is $(\alpha_{J_1}, ..., \alpha_{J_t})$. Let c be c_J and I_0 be the largest connected component of J_1 relative to c.

Case 1. π is satisfied by all independent sets: Since π is hereditary and satisfied by all independent sets, I_0 must contain an edge. Therefore there is a vertex d of I_0 with $d \neq c$. Let I_1 be the graph obtained by deleting I_0 except c from J_1 (Fig 4.1 (a)).

We reduce LFMIS to the problem. For an undirected graph $G = (V, E)$, we construct a graph $\tilde{G} = (\tilde{V}, \tilde{E})$ as follows (Fig. 4.1 (b)): For each vertex u of G, a copy of I_1 is attached by identifying u with c. Then each edge $\{u, v\}$ in E is replaced by a copy of I_0 by identifying u (resp. v) with c (resp. d). Finally, independent graphs $J_2, ..., J_t$ are added. It follows from the choice of J that for any independent set U of G the induced subgraph of $(\tilde{V} - V) \cup U$ has a β-sequence smaller than β_J and for each edge $\{u, v\}$ in E the induced subgraph of $(\tilde{V} - V) \cup \{u, v\}$ violates π. Therefore we define an order

on $\tilde{V}$ so that $\tilde{V} - V < V$, the order on V is the same as G and the order on $\tilde{V} - V$ is arbitrary. With this order, first all vertices in $\tilde{V} - V$, which were newly added, are chosen. Then the vertices in the lfm independent set of G are chosen according to the order on V. Thus the lfm independent set U of G and the lfm subset $\tilde{U}$ of $\tilde{V}$ whose induced subgraph satisfies π are related as $\tilde{U} = (\tilde{V} - V) \cup U$.

Case 2. π is satisfied by all cliques: This case can be solved in the same way as Case 1 by considering complement graphs.□

Proof of Theorem 4.2 We say that a directed graph $D = (V, A)$ with $V = \{1, ..., n\}$ is *complete antisymmetric transitive* (c.a.t.) (resp. an *independent set*, *complete symmetric*) if $(i, j) \in A$ but $(j, i) \notin A$ for all $1 \leq i < j \leq n$ (resp. $A = \emptyset$, $A = V \times V$). By Ramsey's Theorem, π is satisfied by either (i) all independent sets, or (ii) all complete symmetric directed graphs, or (iii) all complete antisymmetric transitive directed graphs. The cases for (i) and (ii) can be proved in the same way as in the case of undirected graphs (Theorem 4.1). We deal with the case (iii). Assume that π is satisfied by all c.a.t. directed graphs but not by all independent sets. Let s be the largest number such that every graph consisting of any c.a.t. directed graph and s independent vertices $u_1, ..., u_s$ satisfies π but there is a c.a.t. directed graph C such that the graph consisting of C and $u_1, ..., u_{s+1}$ violates π. Since some independent set violates π, such number s exists.

We give a reduction from LFMIS. For an instance $G = (V, E)$ of LFMIS, we define a directed graph $D = (V_D, A_D)$ as follows. Let k be the number of vertices in C and let $V = \{1, ..., n\}$. We use k copies of the complement graph $\bar{G}$ of G. We denote the ith copy of $\bar{G}$ by $\bar{G}_i = (V_i, \bar{E}_i)$, where $V_i = \{v_{i1}, ..., v_{in}\}$. Then we define $V_D = \bigcup_{i=1}^{k} V_i \cup \{u_1, ..., u_s\}$. The edges of D are added so that the induced subgraph of $\{v_{1j}, v_{2j}, ..., v_{kj}\}$ is the c.a.t. directed graph of k vertices. Namely,

$$\begin{aligned} A_D = & \{(v_{pj}, v_{qj}) \mid 1 \leq p < q \leq k,\ 1 \leq j \leq n\} \\ & \cup \bigcup_{p=1}^{k} \{(v_{pi}, v_{pj}) \mid \{i, j\} \notin E,\ 1 \leq i < j \leq n\} \\ & \cup \{(v_{pi}, v_{qj}) \mid 1 \leq p < q \leq k,\ 1 \leq i, j \leq n, \\ & \qquad \{i, j\} \notin E\} \end{aligned}$$

The order on V_G is

$$u_1 < \cdots < u_s < v_{11} < v_{21} < \cdots < v_{k1} < \cdots < v_{1n} < v_{2n} < \cdots < v_{kn}$$

Let U be the lfm independent set of G and U_D be the lfm subset of V_D whose induced subgraph satisfies π. Then by induction we shall show the following relation.

(1) $U_D = \{u_1, ..., u_s\} \cup \{v_{pi} \mid i \in U,\ 1 \leq p \leq k\}$.

Since the induced subgraph of $\{v_{11}, v_{21}, ..., v_{k1}\}$ is the c.a.t. directed graph of size k, it follows from the choice of s and C that the vertices $u_1, ..., u_s$ and $v_{11}, v_{21}, ..., v_{k1}$ can be chosen into U_D. Let $U_D^{(j)} = U_D \cap (\{u_1, ..., u_s\} \cup \{v_{pi} \mid 1 \leq i \leq j,\ 1 \leq p \leq k\})$ and $U^{(j)} = U \cap \{1, ..., j\}$. Let j be the next vertex in V to be tested for choice into $U^{(j)}$. It suffices to note the following two facts.

(2) If j is not adjacent to any vertex in $U^{(j-1)}$, then all vertices $v_{1j}, ..., v_{kj}$ can be chosen since the resulting induced subgraph of $U_D^{(j)}$ consists of a c.a.t. directed graph and s independent vertices.

(3) If j is adjacent to some vertex i in $U^{(j-1)}$, then there is no edge between v_{pi} and v_{pj} for all $p = 1, ..., k$. Therefore none of $v_{1j}, ..., v_{kj}$ can be chosen since adding an independent vertex v_{pj} produces a graph consisting of C and $u_1, ..., u_s, v_{pj}$ that violates π by the choice of s and C.□

Proof of Theorem 4.3 The idea is very similar to the proof of Theorem 4.1 except that we consider planar bipartite graphs. Since π is hereditary and satisfied by all independent edges[2], I_0 must also contain an edge.

Case 1. I_0 is not a single edge: We give a reduction from LFMIS restricted planar graphs (Lemma 4.4 (1)). We can choose the vertex d so that c and d are on the same face in its planar layout and the distance between c and d is even. The construction is the same as that in Case 1 of the proof of Theorem 4.1.

Case 2. I_0 is a single edge: We reduce the lfm subgraph of maximum degree 1 problem restricted to planar bipartite graphs (Lemma 4.5 (3)) to the problem. We call the complete bipartite graph $K_{1,n}$ the *n-star*. The (m, n)-*double star* is a graph obtained by connecting the centers of an m-star and an n-star by an edge. Since I_0 is a single edge, J_1 is a star and therefore so are $J_2, ..., J_t$. Assume that J_1 has r edges. Since π is satisfied by all independent edges, we see $r \geq 2$. Note that graphs consisting of any number of independent $(r-1)$-stars and $J_2, ..., J_t$ have β-sequences smaller than β_J and satisfy π. Therefore there exist integers q, k, p such that the following facts holds.

[2] A collection of disjoint edges is called *independent edges*.

(1) Graphs consisting of any number of $(r-2, q-1)$-double stars, $k-1$ number of $(r-2, q)$-double stars and $J_2, ..., J_t$ satisfy π.

(2) The graph consisting of p number of $(r-2, q-1)$-double stars, k number of $(r-2, q)$-double stars and $J_2, ..., J_t$ violates π.

For a planar bipartite graph $G = (V, E)$, where V is partitioned as $V = N \cup M$, we construct a graph $\tilde{G} = (\tilde{V}, \tilde{E})$ as follows: For each vertex u in N (resp. M), an $(r-2)$-star (resp. a q-star) is attached by identifying u with the center of the star. Then $J_2, ..., J_t$, $k-1$ number of $(r-2, q)$-double stars and p number of $(r-2, q-1)$-double stars are added. Obviously the resulting graph is bipartite and planar. An order on $\tilde{V}$ is given so that $\tilde{V} - V < V$, the order on V is the same as G and the order on newly added vertices is arbitrary.

Then it can be shown that the lfm subset $\tilde{U}$ of $\tilde{G}$ whose induced subgraph satisfies π is $(\tilde{V}-V) \cup U$, where U is the lfm subset of V whose induced subgraph is of degree at most one. We omit the detail.□

Remark Theorem 4.1 also holds when the instances are restricted to either planar graphs or bipartite graphs. This can be shown by using Lemma 4.4 (1) and (2).

5 FINDING PARALLELISM

This section presents an idea that finds parallelism in problems. Although the method is not universal, many problems can be shown to be in NC. The basic idea is due to Rytter (1985).

5.1 Inference Systems

Definition An *inference system* is a pair $Q = (X, R)$, where X is a finite set of *sentences* and R is the set of *rules*. A rule is of the form $\beta_1, ..., \beta_n \Rightarrow \alpha$, where $\alpha, \beta_1, ..., \beta_n \in X$ $(n \geq 0)$. In particular, if a rule is of the form $\Rightarrow \alpha$, we call the sentence α an *axiom*. $\mathbf{TH}(Q)$ is the least set of sentences containing all axioms and closed under the rules. An element in $\mathbf{TH}(Q)$ is called a *theorem*.

Definition A *proof tree* $T(\gamma)$ for $\gamma \in \mathbf{TH}(Q)$ is a finite labelled tree such that the root is labelled with γ, the leaves are all labelled with axioms, and the internal vertices satisfy the following condition:

If a vertex with label $\alpha \in X$ has sons labelled with $\beta_1, ..., \beta_n$, then $\beta_1, ..., \beta_n \Rightarrow \alpha$ is a rule in R.

We denote by $size(T(\gamma))$ and $height(T(\gamma))$ the number of leaves in $T(\gamma)$ and the height of $T(\gamma)$, respectively. Then we define $size(\gamma) = \min\{size(T(\gamma)) \mid T(\gamma)$ is a proof tree for $\gamma\}$ and $size(Q) = \max\{size(\gamma) \mid \gamma \in \mathbf{TH}(Q)\}$.

For each rule $\beta_1, ..., \beta_n \Rightarrow \alpha$ with $n \geq 3$, we replace it by $(\beta_1, \beta_2 \Rightarrow \gamma_1)$, $(\gamma_1, \beta_3 \Rightarrow \gamma_2)$,..., $(\gamma_{n-2}, \beta_n \Rightarrow \alpha)$ by introducing new sentences $\gamma_1, ..., \gamma_{n-2}$. By this replacement, we assume hereafter that the left side of each rule of $Q = (X, R)$ has at most two sentences. It should be noticed that this change increases the number of rules only linearly.

5.2 Balancing Technique

For an inference system $Q = (X, R)$, we define the following inference system $\hat{Q} = (\hat{X}, \hat{R})$ called the *balanced inference system*, where $\hat{X} = X \cup (X \times X) \cup (X \times (X \times X))$. $\hat{R}$ consists of the following rules of (1)-(4).

(1) For each rule in R, the following rules are in $\hat{R}$.

Rule in R	Rule in $\hat{R}$
$\Rightarrow x$	$\Rightarrow x$
$y \Rightarrow x$	$\Rightarrow (y, x)$
$z, y \Rightarrow x$	$\Rightarrow (y, (z, x))$
$z, y \Rightarrow x$	$\Rightarrow (z, (y, x))$

(2) $y, (y, x) \Rightarrow x$.
(3) $z, (z, (y, x)) \Rightarrow (y, x)$.
(4) $(z, y), (y, x) \Rightarrow (z, x)$.

Proposition 5.1 $\mathbf{TH}(Q) = \mathbf{TH}(\hat{Q}) \cap X$.

Proposition 5.2 *For each theorem $\gamma \in \mathbf{TH}(Q)$, let $T(\gamma)$ be a proof tree for γ in Q. Then there is a proof tree $T'(\gamma)$ for γ in $\hat{Q}$ satisfying*

$$height(T'(\gamma)) \leq 7 \cdot \log(size(T(\gamma))).$$

Proposition 5.2 is proved by the following separator theorem for trees.

Lemma 5.3 *Let T be a k-ary tree with $k \geq 2$. Then there is a vertex v of T such that*

$$size(T') \leq \frac{k}{k+1} size(T), \quad size(T'') \leq \frac{k}{k+1} size(T),$$

where T' is the subtree of T rooted at v and T'' is the tree obtained by pruning T' from T.

```
begin
  for k = 1 to 7·log(size(Q))
    pardo for each β1,...,βn ⇒ α ∈ R̂
      if {β1,...,βn} ⊆TH(Q̂)
          then TH(Q̂) ←TH(Q̂) ∪ {α}
    end pardo
  end for
end
```

Algorithm 5.1

Propositions 5.1, 5.2 guarantee that Algorithm 5.1 on a CRCW PRAM computes $\mathbf{TH}(Q)$ with polynomial number of processors although the polynomial is of rather large degree. Possibly, $size(Q)$ is exponential with respect to $|X|$. However, if $size(Q)$ is polynomially bounded, it runs in $O(\log n)$ time. This observation is also found in Ullman and van Gelder (1986).

Definition We say that a family $\{Q_i = (X_i, R_i)\}$ of inference systems has *polynomial size proof trees* if there is a polynomial $p(n)$ such that every theorem γ of Q_i has a proof tree $T_i(\gamma)$ for γ in Q_i with $size(T(\gamma)) \leq p(|X_i|)$.

For $Q = (X, R)$, we define a directed graph $D_G = (X, A)$ by setting $A = \{(x, y) \mid (x, y) \in \mathbf{TH}(\hat{Q})\}$. We say that Q has the *unique path property* if D_G is uniconnected, that is, there is at most one directed path between any pair of distinct vertices.

Theorem 5.4 (Rytter 1985) *Let $\{Q_i = (X_i, R_i)\}$ be a family of inference systems with polynomial size proof trees. Let $n = |X_i|$.*

(1) $\mathbf{TH}(Q_i)$ can be computed on a CRCW PRAM with polynomial number of processors in $O(\log n)$ time.

(2) If the unique path property is satisfied, then $\mathbf{TH}(Q_i)$ can be computed on a CREW PRAM with polynomial number of processors in $O(\log n)$ time.

5.3 Applications

An interesting feature of Theorems 5.4 is that it gives a method of transforming sequential algorithms to parallel algorithms. We show some applications of the theorem.

Example 5.1 The lfm independent set problem is known P-complete but the problem can be parallelized by Theorem 5.4 (2) if the instances are restricted to forests. For a forest $G = (V, E)$ with $V = \{1, ..., n\}$, we construct an inference system as follows: For $i \in V$, let $N(i) = \{j \mid \{i, j\} \in E \text{ and } j < i\}$. The rules are defined by

(1) $j \Rightarrow \neg i$ for each $j \in N(i)$.

(2) $\neg j_1, ..., \neg j_t \Rightarrow i$, where $N(i) = \{j_1, ..., j_t\}$.

It is easy to see that vertex i is in the lfm independent set if and only if i is a theorem of the inference system. Further, the resulting inference system has the unique path property. Hence the problem is solvable on CREW PRAM in time $O(\log n)$.

Example 5.2 The lexicographically first maximal triangle free edge-induced subgraph problem is, given a graph $G = (V, E)$ with a linear order on E, to find the lfm edge set $F \subseteq E$ such that the graph formed by F contains no cycle of length three. This problem is P-complete for graphs with degree 6 (Miyano 1987). However, it can be shown by constructing an inference system from a graph that the problem restricted to outerplanar graphs is parallelized by Theorem 5.4 (1) (Miyano 1988a).

Example 5.3 2-satisfiability problem (2SAT) is also solved on a CRCW PRAM in $O(\log n)$ time. Let $S = \{\{\alpha_1, \beta_1\}, \{\alpha_2, \beta_2\}, ..., \{\alpha_m, \beta_m\}\}$ be a set of clauses of size two, where α_i, β_i are in $\{x_1, \neg x_1, ..., x_n, \neg x_n\}$. The problem is to decide whether S is satisfiable. For S, we construct an inference system $Q_S = (X_S, R_S)$ as follows:

$$X_S = \{\{\alpha, \beta\} \mid \alpha, \beta \in \{x_1, \neg x_1, ..., x_n, \neg x_n\}\} \cup \{\square\}$$

The rules of Q_S are

$$\Rightarrow \{\alpha_i, \beta_i\} \quad \text{for } i = 1, ..., m$$
$$\{\alpha, \beta\}, \{\neg\beta, \gamma\} \Rightarrow \{\alpha, \gamma\}$$
$$\{\beta\}, \{\neg\beta, \gamma\} \Rightarrow \{\gamma\}$$
$$\{\beta\}, \{\neg\beta\} \Rightarrow \square$$

where α, β, γ are literals in $\{x_1, \neg x_1, ..., x_n, \neg x_n\}$.

It is known (Jones et al. 1976) that S is not satisfiable if and only if there is a sequence of literals $\gamma_1, ..., \gamma_k$ such that (1) holds, and either (2) or (3) holds.

(1) $\{\neg\gamma_i, \gamma_{i+1}\}$ is in S for all $i = 1, ..., k-1$.

(2) $\{\gamma_1\} \in S$ and $\gamma_k = \neg\gamma_1$.

(3) $\gamma_1 = \neg\gamma_j = \gamma_k$ for some $1 < j < k$.

Then it can be shown that $\square$ is in $\mathbf{theorem}(Q_S)$ if and only if S is not satisfiable. Moreover, it is not hard to show that $size(Q_S)$ is linear with respect to m.

Example 5.4 The class AuxDPDA($n^{O(1)}, \log n$) of functions computable by deterministic auxiliary pushdown machines (Hopcroft and Ullman 1979) which run in polynomial time using $O(\log n)$ worktape space contains many important problems, for example, sorting, pattern matching, parsing, etc.

We show by constructing inference systems that problems in this class are computable on a CREW PRAM with polynomial number of processors in $O(\log n)$ time. This class is known to be contained in NC^2 (Ruzzo 1980, 1981).

Assume that an auxiliary pushdown machine M runs in polynomial time, say $p(n)$, using $O(\log n)$ worktape space. A *surface configuration* $C = (I, A)$ of M on an input x of length n consists of the top symbol A of the pushdown store and a configuration I except the pushdown store describing the current state, the input head position, and the contents of the worktape together with the worktape head location.

For each x, we construct an inference system as follows: Let C, C', D, D', E be surface configurations of M on x. For an indexed pair $(C, D)_h^k$ of surface configurations, it means that M on x can move from C to D keeping the stack height at least h. The rules are defined as follows:

$$\begin{array}{lrcl} 1. & & \Rightarrow & (C, C)_h^0 \\ 2. & (C, D)_h^1, (D, E)_h^k & \Rightarrow & (C, E)_h^{k+1} \\ 3. & (C, D)_h^k & \Rightarrow & (C', D')_{h-1}^1 \end{array}$$

where it is assumed in 3 that $C' \vdash C$ by $push(A)$ and $D \vdash D'$ by $pop(A)$ for some A, and the lower index h is between 1 and $p(n)$ and the upper index k is between 0 and $p(n)$.

If the pair $(C_0, D_0)_1^k$ of the initial and final surface configurations is a theorem of the inference system for some k, then M accepts x, and vice versa. We can show that the above family of inference systems indexed by inputs has polynomial size proof trees and the unique path property.

Moreover, if M is nondeterministic, the unique path property may be lost but it still has polynomial size proof trees. Therefore Theorem 5.4 (1) is applicable.

Example 5.5 Unambiguous context-free languages are recognizable in time $O(\log n)$ on CREW PRAM (Rytter 1985). This result can be also proved by directly constructing inference systems. We show the construction by an example. Let G be an unambiguous context-free grammar with productions

$$S \to SS,\ S \to (S),\ S \to (),$$

where "(" and ")" are terminal symbols. For a string $w = x_1 \cdots x_n$ of terminal symbols, we define an inference system $Q_w = (X_w, R_w)$ by setting $X_w = \{S[i, j] \mid 0 \leq i < j \leq n,\ j - i \text{ is even}\}$. R_w consists of the following rules:

(1) $\Rightarrow S[i, i+2]$ if $x_{i+1}x_{i+2} = ()$.

(2) $S[i, j] \Rightarrow S[i-1, j+1]$ if $x_i = (\ \text{and}\ x_{j+1} =)$.

(3) $S[i, j], S[j, k] \Rightarrow S[i, k]$.

Then $S[0, n] \in \mathbf{TH}(Q_w) \iff S \Rightarrow^* w$ and the family $\{Q_w\}$ satisfies the conditions of Theorem 5.4 (2).

6 CONCLUSION

Given a problem in **P**, there may be two ways to go. The first reaches P-completeness. The other falls into **NC**. In these respects, this paper contributed in the following points:

1. Very general P-completeness theorems are found that derives a new series of P-complete problems that are solvable by simple greedy algorithms.
2. By using inference systems, a method of finding parallelism in sequential algorithms is introduced.

While the search for a really efficient and exact algorithm is always expected, the P-completeness would light up the way to go. Also, a variety of parallelizing techniques should be searched in detail.

REFERENCES

Alt, H., Hagerup, T., Mehlhorn, K. and Preparata, F.P. (1987), Deterministic simulation of idealized parallel computers on more realistic ones, *SIAM J. Comput.* **16**, 808-835.

Anderson, R. and Mayr, E.W. (1984), Parallelism and greedy algorithms, Report No. STAN-CS-84-1003, Department of Computer Science, Stanford University.

Anderson, R. and Mayr, E.W. (1987), Parallelism and the maximal path problem, *Inf. Process. Lett.* **24**, 121-126.

Asano, T. and Hirata, T. (1982), Edge-deletion and edge-

contraction problems, *Proc. 14th ACM STOC*, 245-254. 279-295.

Cole, R.J. and Ó'Dúnlaing, C. (1986), Note on the AKS sorting network, Ultracomputer Note No. 109, Courant Institute of Mathematical Sciences, New York University.

Cook, S.A. (1974), An observation on time-storage trade off, *J. Comput. System Sci.* **9**, 308-316.

Cook, S.A. (1981), Towards a complexity theory of synchronous parallel computation, *Enseign. Math.* **27**, 99-124.

Cook, S.A. (1983), The classification of problems which have fast parallel algorithms, *Proc. 1983 International FCT Conference* (Lecture Notes in Computer Science **158**), 78-93.

Cook, S.A. (1985), A taxonomy of problems with fast parallel algorithms, *Inform. and Control* **64**, 2-22.

Dobkin, D., Lipton, R.J. and Reiss, S. (1979), Linear programming is log-space hard for P, Inf. Process. Lett. **9**, 96-97.

Dwork, D., Kanellakis, P.C. and Michell, J.C. (1984), On the sequential nature of unification, *J. Logic Programming* **1**, 35-50.

Furst, M., Saxe, J.B. and Sipser, M. (1981), Parity, circuits, and polynomial-time hierarchy, *Proc. 22nd IEEE FOCS*, 260-270.

Goldschlager, L.M. (1977), The monotone and planar circuit value problems are log space complete for P, *SIGACT News* **9**, 25-29.

Goldschlager, L.M., Shaw, R.A. and Staples, J. (1982), The maximum flow problem is log-space complete for P, *Theoret. Comput. Sci.* **21**, 105-111.

Hopcroft, J.E. and Ullman, J.D. (1979), *Introduction to Automata Theory, Languages, and Computation*, Addison-Wesley Publishing Company.

Jones, N.D., Lien, Y.E. and Laaser, W.T. (1976), New problems complete for nondeterministic log space, *Math. Systems Theory* **10**, 1-17.

Jones, N.D. and Laaser, W.T. (1977), Complete problems for deterministic polynomial time, *Theoret. Comput. Sci.* **3**, 105-117.

Kuck, D.J. (1977), A survey of parallel machine organization and programming, *Comput. Survey* **9**, 29-59.

Ladner, R.E. (1977), The circuit value problem is log space complete for P, *SIGACT News* **7**, 583-590.

Lewis, J.M. and Yannakakis, M. (1980), The node deletion problems for hereditary properties is NP-complete, *J. Comput. System Sci.* **20**, 219-230.

Luby, M. (1985), A simple parallel algorithm for the maximal independent set problem, *Proc. 17th ACM STOC*, 1-10.

Miyano, S. (1987), The lexicographically first maximal subgraph problems: P-completeness and NC algorithms, *Proc. 13th ICALP* (Lecture Notes in Computer Science **267**), 425-434.

Miyano, S. (1988a), A parallelizable lexicographically first maximal edge-induced subgraph problem, *Inf. Process. Lett.* **27**, 75-78.

Miyano, S. (1988b), Δ_2^p-complete lexicographically first maximal subgraph problems, *Proc. 13th MFCS* (Lecture Notes in Computer Science **317**),454-462.

Miyano, S., Shiraishi, S. and Shoudai, T. (1988), A list of P-complete problems, in preparation.

Pippenger, N. (1979), On simultaneous resource bounds, *Proc. 20th IEEE FOCS*, 307-311.

Ranade, A.G. (1987), How to emulate shared memory, *Proc. 28th IEEE FOCS*, 185-194.

Reif, J.H. (1985), Depth-first search is inherently sequential, *Inf. Process. Lett.* **20**, 229-234.

Ruzzo, W.L. (1980), Tree-size bounded alternation, *J. Comput. System Sci.* **21**, 218-235.

Ruzzo, W.L. (1981), On uniform circuit complexity, *J. Comput. System Sci.* **22**, 365-383.

Rytter, W. (1985), Parallel time $O(\log n)$ recognition of unambiguous cfl's, *Proc. Fundamentals of Computation Theory* (Lecture Notes in Computer Science **199**), 380-389.

Shiloach, Y. and Vishkin, U. (1982), An $O(n^2 \log n)$ parallel MAX-FLOW algorithm, *J. Algorithms* **3**, 128-146.

Stockmeyer, L. and Vishkin, U. (1984), Simulation of parallel random access machines by circuits, *SIAM J. Comput.* **13**, 409-422.

Ullman, J.D. and van Gelder, A. (1986), Parallel complexity of logical query programs, *Proc. 27th IEEE FOCS*, 438-454.

Valiant, L.G. (1982), Parallel computation, *Proc. 7th IBM Symp. on Mathematical Foundations of Computer Science*, 172-189.

Watanabe, T., Ae, T. and Nakamura, A. (1981), On the removal of forbidden graphs by edge-deletion or by edge-contraction, *Discrete Appl. Math.* **3**, 151-153.

Yannakakis, M. (1981), Node-deletion problems on bipartite graphs, *SIAM J. Comput.* **10**, 310-327.

Yasuura, H. (1984), On parallel computational complexity of unification, *Proc. FGCS'84*, 235-243.