PAPER Special Section on Discrete Mathematics and Its Applications

Optimal Parallel Algorithms for Edge-Coloring Partial k-Trees with Bounded Degrees

Xiao ZHOU† *and* Takao NISHIZEKI†, *Members*

SUMMARY Many combinatorial problems can be efficiently solved for partial k-trees (graphs of treewidth bounded by k). The edge-coloring problem is one of the well-known combinatorial problems for which no NC algorithms have been obtained for partial k-trees. This paper gives an optimal and first NC parallel algorithm to find an edge-coloring of any given partial k-tree with bounded degrees using a minimum number of colors. In the paper k is assumed to be bounded.

key words: *edge-coloring, parallel algorithm, DP algorithm*

1. Introduction

This paper deals with the edge-coloring problem which asks to color all edges of a given simple graph G, using a minimum number of colors, so that no two adjacent edges are colored with the same color. The minimum number is called the *chromatic index* $\chi'(G)$ of G. Vizing showed that $\chi'(G) = \Delta$ or $\Delta + 1$ for any simple graph G where Δ is the maximum degree of G [10]. The edge-coloring problem arises in many applications, including various scheduling and partitioning problems [10]. The problem is NP-complete [17], and hence it is very unlikely that there exists a sequential algorithm which edge-colors a given graph G with $\chi'(G)$ colors in polynomial time. On the other hand, there exist sequential algorithms which edge-color G with $\Delta+1$ colors in polynomial time [11], [22], [25]. However, no NC parallel algorithms for edge-coloring G with $\Delta+1$ colors have been obtained except for the case when Δ is bounded [19].

It is known that many combinatorial problems can be solved very efficiently for partial k-trees or series-parallel graphs [1], [2], [6], [9], [20], [24]. Such a class of problems has been characterized in terms of "forbidden graphs" or "extended monadic logic of second order" [1], [2], [6], [9], [24]. The edge-coloring problem does not belong to such a class of the "maximum (or minimum) subgraph problems," and is indeed one of the "edge-covering problems" which does not appear to be efficiently solved for partial k-trees [6]. However, Bodlaender gave a sequential dynamic programming algorithm which solves the edge-coloring problem on a partial k-tree G in time $O(n\Delta^{2^{2(k+1)}})$ [4]. Throughout the paper n denotes the number of vertices in G. His algorithm takes linear time if Δ is bounded. Furthermore the current authors have recently obtained a linear-time sequential algorithm for any partial k-tree whose Δ is not necessarily bounded [26]. On the other hand, NC parallel algorithms for the edge-coloring problem have not been obtained for partial k-trees [3], although NC parallel algorithms have been obtained for the following three restricted classes of graphs: planar graphs with maximum degree $\Delta \geqq 9$ [8]; outerplanar graphs [7], [13]; and series-parallel multigraphs [28]. Note that outerplanar graphs and series-parallel simple graphs are partial 2-trees.

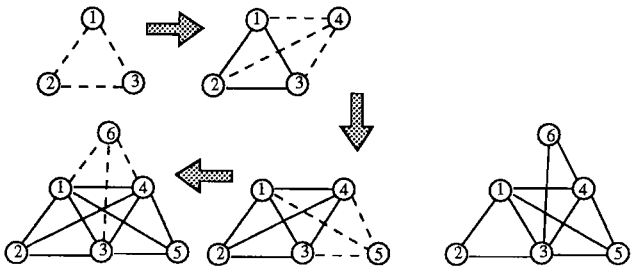
In this paper we give an optimal and first NC parallel algorithm which solves the edge-coloring problem for partial k-trees G with bounded Δ. Throughout the paper we assume that k is bounded. Given G with its tree-decomposition, our parallel algorithm finds an edge-coloring of G using $\chi'(G)$ colors in $O(\log n)$ time with $O(n/\log n)$ processors. It is known that a tree-decomposition of G can be found in $O(\log^2 n)$ time with $O(n/\log n)$ processors [5], [23]. Our algorithm uses three techniques, a tree contraction, a parallel dynamic programming, and a bottom-up tree computation: from a given tree-decomposition T of G, the algorithm constructs its "parsing tree" T^p of $O(\log n)$ height by means of a tree contraction, then solves the edge-coloring problem on G by means of a dynamic programming and a bottom-up tree computation on the parsing tree T^p. The parallel computation model we use is a concurrent-read exclusive-write parallel random access machine (CREW PRAM).

2. Terminology and Definitions

In this section we define some terms. Let $G = (V, E)$ denote a graph with vertex set V and edge set E. We often denote by $V(G)$ and $E(G)$ the vertex set and the edge set of G, respectively. The paper deals with *simple* graphs without multiple edges or self-loops. An edge joining vertices u and v is denoted by (u, v). The *degree* of vertex $v \in V$ is denoted by $d(v)$. The *maximum degree* of G is denoted by $\Delta(G)$ or simply by Δ. The graph obtained from G by deleting all vertices in $V' \subseteq V(G)$ is denoted by $G - V'$. The subgraph of G induced by the edges in a subset $E' \subseteq E(G)$ is denoted by $G[E']$. We

Manuscript received August 23, 1994.
Manuscript revised November 29, 1994.

†The authors are with the Department of System Information Sciences, Graduate School of Information Sciences, Tohoku University, Sendai-shi, 980-77 Japan.



(a) (b)

Fig. 1 (a) 3-trees and (b) a partial 3-tree.

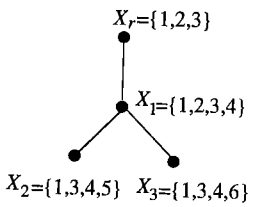


Fig. 2 A tree-decomposition of the partial 3-tree in Fig. 1 (b).

will use notions as: *leaf*, *node*, *internal node*, *child* and *father*, in their usual meaning.

The class of *k-trees* is defined recursively as follows [20]:

(a) A complete graph with k vertices is a k-tree.

(b) If $G = (V, E)$ is a k-tree and k vertices $v_1, v_2, \cdots, v_k$ induce a complete subgraph of G, then $G' = (V \cup \{w\}, E \cup \{(v_i, w) | 1 \leqq i \leqq k\})$ is a k-tree where w is a new vertex not contained in G.

(c) All k-trees can be formed with rules (a) and (b).

A graph is a *partial k-tree* if it is a subgraph of a k-tree. Thus partial k-trees $G = (V, E)$ are simple graphs, and $|E| < kn$. Figure 1 (a) illustrates a process of generating a 3-tree, and Fig. 1 (b) depicts a partial 3-tree. In this paper we assume that k is a constant.

A *tree-decomposition* of a graph $G = (V, E)$ is a tree $T = (V_T, E_T)$ with V_T a family of subsets of V satisfying the following properties [20]:

- $\bigcup_{X_i \in V_T} X_i = V$;
- for every edge $e = (v, w) \in E$, there is a node $X_i \in V_T$ with $v, w \in X_i$; and
- if node X_l lies on the path in T from node X_i to node X_j, then $X_i \cap X_j \subseteq X_l$.

Figure 2 depicts a tree-decomposition of the partial 3-tree in Fig. 1 (b). The *treewidth* of a tree-decomposition $T = (V_T, E_T)$ is $\max_{X_i \in V_T} |X_i| - 1$. The *treewidth* of G is the minimum treewidth of a tree-decomposition of G, taken over all possible tree-decompositions of G. It

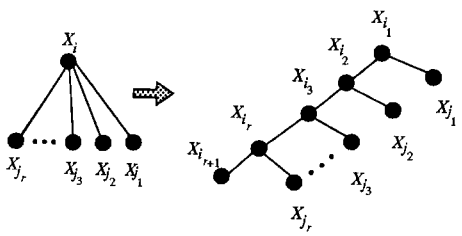


Fig. 3 Illustration of the binary transformation.

is known that every graph with treewidth $\leqq k$ is a partial k-tree, and conversely, that every partial k-tree has a tree-decomposition with treewidth $\leqq k$ [20].

Consider a tree-decomposition of a partial k-tree G with treewidth $\leqq k$. We transform it to a binary tree T as follows [4]: regard T as a rooted tree with choosing an arbitrary node as the root, and replace every internal node X_i of r children by $r+1$ new nodes $X_{i_1}, X_{i_2}, \cdots, X_{i_{r+1}}$ such that $X_i = X_{i_1} = X_{i_2} = \cdots = X_{i_{r+1}}$, where X_{i_1} has the same father as X_i, X_{i_p} is the father of $X_{i_{p+1}}$ and the pth child X_{j_p} of X_i $(1 \leqq p \leqq r)$, and $X_{i_{r+1}}$ is a leaf of T. (See Fig. 3.) This transformation can be done in $O(\log n)$ parallel time using a total of $O(n)$ operations. T is a tree-decomposition of $G = (V, E)$ with the following characteristics:

- the number of nodes in T is $O(n)$;
- each internal node X_i has exactly two children, say X_j and X_l, and either $X_i = X_j$ or $X_i = X_l$; and
- for each edge $(v, w) \in E$ there is at least one leaf X_l with $v, w \in X_l$.

Such T is called a *binary* tree-decomposition [4]. Clearly T has treewidth $\leqq k$. For each edge $e = (v, w) \in E$ we choose an arbitrary leaf X_j of T such that $v, w \in X_j$ and denote it by $rep(e)$.

We next introduce two new concepts: a two-terminal tree and its parsing tree. We define a *two-terminal tree* recursively as follows.

1. A tree T of a single edge is a two-terminal tree. The ends of the edge are called the *terminals* of T and denoted by $X_h(T)$ and $X_r(T)$, respectively.
2. Let T_i, $i = 1, 2, 3$, be two-terminal trees with terminals $X_h(T_i)$ and $X_r(T_i)$. Then a tree T obtained from T_1, T_2 and T_3 by identifying nodes $X_r(T_1)$, $X_h(T_2)$ and $X_h(T_3)$ is a two-terminal tree, whose terminals are $X_h(T) = X_h(T_1)$ and $X_r(T) = X_r(T_2)$ or $X_r(T_3)$. (See Fig. 4.)

Figure 4 illustrates the rule 2 above, where terminals are drawn by white circles.

Consider a **binary** tree-decomposition of a partial k-tree $G = (V, E)$. Add a new dummy node X_{root} to it as a father of the root and regard X_{root} and a leaf

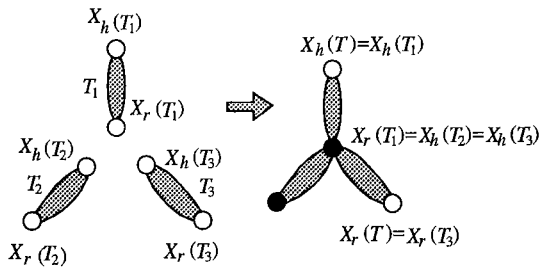


Fig. 4 Two-terminal trees.

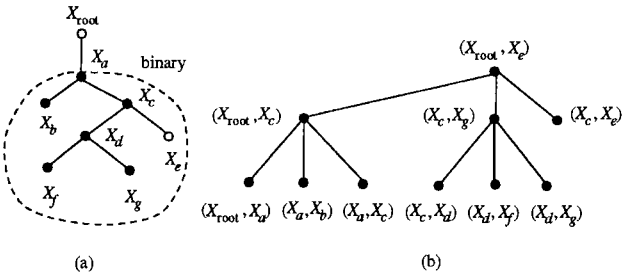


Fig. 5 (a) Two-terminal tree T and (b) its parsing tree T^p.

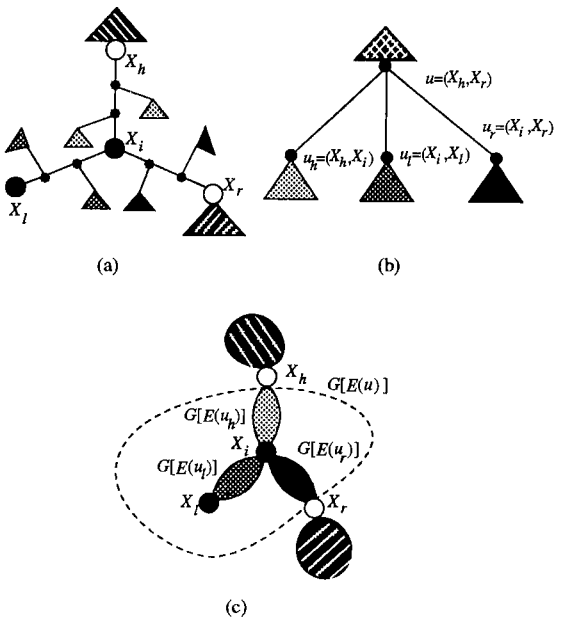


Fig. 6 (a) T, (b) T^p and (c) four subgraphs of G.

as terminals as depicted in Fig. 5 (a), then the resulting tree T is a two-terminal tree. Let $X_{root} = \phi$, then T is a tree-decomposition of G. Since a two-terminal tree T can be recursively constructed by the rules above, T can be represented by a "parsing tree" T^p. Every leaf of T^p represents a two-terminal tree induced by a single edge of T. On the other hand, each internal node u of T^p has exactly three children, say, u_h, u_l and u_r, and the two-terminal tree corresponding to u is obtained from the three two-terminal trees T_1, T_2 and T_3 corresponding to u_h, u_l and u_r by the rule 2 above. Thus the root of T^p corresponds to T. Figure 5 illustrates a two-terminal tree T having terminals X_{root} and X_e together with its parsing tree T^p, where each node u of tree T^p is denoted by the pair of terminals of a two-terminal subtree of T corresponding to u.

We next define an edge-set $E(u) \subseteq E$ for each node $u = (X_h, X_r)$ of T^p as follows. If u is a leaf of T^p, then $E(u) = \{e \in E |\ rep(e)$ is X_h or $X_r\}$. If u is an internal node of T^p having children $u_h = (X_h, X_i)$, $u_l = (X_i, X_l)$ and $u_r = (X_i, X_r)$, then $E(u) = E(u_h) \cup E(u_l) \cup E(u_r)$. Note that the three edge-sets $E(u_h)$, $E(u_l)$ and $E(u_r)$ are pairwise disjoint. Thus node $u = (X_h, X_r)$ of T^p corresponds to a subgraph $G[E(u)]$ of G induced by the edges in $E(u)$. The subgraph $G[E(u)]$ is an edge-disjoint union of three subgraphs $G[E(u_h)]$, $G[E(u_l)]$ and $G[E(u_r)]$, which share common vertices only in X_i because of the third property of a tree-decomposition. Similarly the subgraph $G[E(u)]$ shares common vertices with the other parts of G only in X_h and X_r. The root X_{root}, in particular, of T^p corresponds to G. Figure 6 illustrates T, T^p and four subgraphs $G[E(u)]$, $G[E(u_h)]$, $G[E(u_l)]$ and $G[E(u_r)]$.

3. Optimal Parallel DP Algorithm

In this section we prove the following theorem. Although the algorithm in the theorem only decides the chromatic index $\chi'(G)$ of G, it can be easily modified so that it actually edge-colors G with $\chi'(G)$ colors.

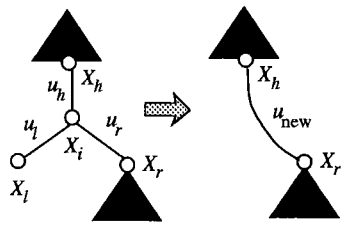
Theorem 1: Let G be a partial k-tree of n vertices given by its tree-decomposition with treewidth $\leqq k$. Then there is a parallel algorithm which determines the chromatic index $\chi'(G)$ in $O(\log n)$ parallel time using a total of $O(n\{\Delta 2^{6(k+1)(\Delta+1)} + (\Delta+1)^{(k^2+k+2)/2}\})$ operations.

If Δ is bounded, then $\{\Delta 2^{6(k+1)(\Delta+1)} + (\Delta + 1)^{(k^2+k+2)/2}\}$ is also bounded although it is very large, and hence the algorithm in Theorem 1 is an optimal parallel algorithm. The following general lemma is well-known [12], [18].

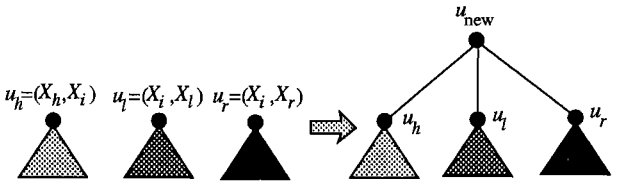
Lemma 1: Let A be a given algorithm with $O(\log n)$ parallel computation time. Suppose that A involves a total number of m operations. Then A can be implemented using p processors in $O(m/p + \log n)$ parallel time.

If there is an algorithm A which solves the edge-coloring problem in $O(\log n)$ parallel time using a total of $m = O(n)$ operations, then by adapting Lemma 1 with choosing $p = n/\log n$ one can know that A can be implemented using $O(n/\log n)$ processors in $O(\log n)$ parallel time. (We omit the "lower-level details" of implementing the algorithm with $O(n/\log n)$ processors [18].) Thus by Theorem 1 and Lemma 1 we have the following corollary.

Corollary 1: The chromatic index $\chi'(G)$ can be determined in $O(\log n)$ parallel time with $O(n/\log n)$ processors for a partial k-tree G given by its tree-



(a) Contraction of a two-terminal tree T



(b) Construction of a parsing tree T^p corresponding to (a)

Fig. 7 Illustration for the proof of Lemma 2.

decomposition with treewidth $\leqq k$ if k and the maximum degree Δ are bounded.

In the remaining of this section we will give a proof of Theorem 1. Our first idea is to construct a parsing tree T^p of $O(\log n)$ height. Since a tree-decomposition T of G has $O(n)$ nodes, a parsing tree T^p has $O(n)$ height in general. However, using the tree-contraction technique [15], [16], [18], we can necessarily construct a parsing tree T^p of $O(\log n)$ height, as follows.

Lemma 2: Any two-terminal tree T of $O(n)$ nodes has a parsing tree T^p of $O(\log n)$ height, and such a tree T^p can be constructed in $O(\log n)$ parallel time using $O(n)$ operations.

Proof: A required parsing tree $T^p = (V_p, E_p)$ can be constructed by the following algorithm. Note that we use adjacency lists to represent T and T^p.

begin
$V_p := \phi$; $E_p := \phi$;
choose the terminal $X_h(T)$ of T as the root of T;
for each edge $u = (X_h, X_r)$ of T **in parallel do**
$V_p := V_p \cup \{u\}$;
{ current T^p is a forest of isolated nodes }
for each leaf X_l of T **in parallel do**
index(X_l) $\leftarrow$ left-to-right leaf index X_l;
{ the leaves are numbered in left-to-right order }
while T has at least four nodes **do**
for each leaf X_l of T with odd index(X_l) such that X_l is the left child of X_l's father X_i **in parallel do**
begin { See Fig. 7}
let X_h be the father of X_i in T;
let X_r be the right child of X_i in T;
let $u_h = (X_h, X_i)$, $u_l = (X_i, X_l)$ and
$u_r = (X_i, X_r)$ be edges of the current tree T;
{ u_h, u_l and u_r are roots of trees in the current forest $T^p = (V_p, E_p)$ }
delete nodes X_l and X_i from T, and
add to T a new edge joining X_h and X_r;
$V_p := V_p \cup \{u_{new}\}$, where $u_{new} = (X_h, X_r)$;
{ add a new node u_{new} to T^p and join node u_{new} with nodes u_h, u_l and u_r in T^p }
$E_p := E_p \cup \{(u_{new}, u_h), (u_{new}, u_l), (u_{new}, u_r)\}$
end;
for each leaf X_r of T with odd index(X_r) such that X_r is the right child of X_r's father X_i **in parallel do**
begin
let X_h be the father of X_i in T;
let X_l be the left child of X_i in T;
let $u_h = (X_h, X_i)$, $u_l = (X_i, X_l)$ and
$u_r = (X_i, X_r)$ be edges of T;
delete nodes X_r and X_i from T, and
add to T a new edge joining X_h and X_l;
$V_p := V_p \cup \{u_{new}\}$, where $u_{new} = (X_h, X_l)$;
$E_p := E_p \cup \{(u_{new}, u_h), (u_{new}, u_l), (u_{new}, u_r)\}$
end;
for each leaf v of T **in parallel do**
index(v) $\leftarrow$ index(v)/2
end-while
end.

The number of leaves of T reduces by half whenever the **for**-loops in the **while** statement are executed. Therefore the **for**-loops are executed $O(\log n)$ times in total. Hence the algorithm above correctly finds a parsing tree T^p of $O(\log n)$ height in $O(\log n)$ parallel time using $O(n)$ operations. Note that each node of T^p corresponds to either an original edge in T or a new edge added to T during the tree-contraction process. □

Figure 5 illustrates a two-terminal tree and its parsing tree obtained by the algorithm above.

Our second idea is to solve the edge-coloring problem on G by means of a parallel dynamic programming and a bottom-up tree computation on the parsing tree T^p of $O(\log n)$ height: for each node u of T^p from leaves to the root, we construct all (equivalent classes of) edge-colorings of $G[E(u)]$ from those of three subgraphs $G[E(u_h)]$, $G[E(u_l)]$ and $G[E(u_r)]$ by means of a dynamic programming.

Let $C = \{1, 2, \cdots, |C|\}$ be the set of colors where $|C| = \Delta$ or $\Delta + 1$. A mapping (coloring) $f : E \rightarrow C$ is called a *total edge-coloring* of G if no two adjacent edges in E are colored with the same color. For a subset $E' \subseteqq E$ a mapping $f : E' \rightarrow C$ is called a *partial edge-coloring* of G if no two adjacent edges in E' are colored with the same color. For a partial edge-coloring f and each vertex $v \in V$, let $f(v) = \{f((v,x)) | (v,x) \in E'$ and $x \in V\}$, that is, $f(v)$ is the set of colors appearing at v in the edge-coloring f. A partial edge-coloring is *feasible* if it can be extended to a total edge-coloring.

For a node $u = (X_h, X_r)$ of T^p let $\mathcal{C}(h,r)$ be a $2|C|$-tuple of vertex sets such that

(a) $\mathcal{C}(h,r) = (\mathcal{S}, \mathcal{R})$;

(b) $\mathcal{S} = (S_1, \cdots, S_{|C|})$ and $\mathcal{R} = (R_1, \cdots, R_{|C|})$; and

(c) $S_c \subseteqq X_h$ and $R_c \subseteqq X_r$ for each color $c \in C$.

We call such $\mathcal{C}(h,r)$ a *color vector* on $u = (X_h, X_r)$. A color vector $\mathcal{C}(h,r)$ is *good* if there exists a partial edge-coloring $f : E(u) \rightarrow C$ such that $S_c = \{v \in X_h | f(v) \ni c\}$ and $R_c = \{v \in X_r | f(v) \ni c\}$ for each color $c \in C$.

Such $\mathcal{C}(h,r)$ is called the *color vector of the partial edge-coloring* f. Then we have the following lemma. (Bodlaender has proved a similar lemma [4].)

Lemma 3: Let f and g be partial edge-colorings on $u = (X_h, X_r)$ with the same color vector. Then f is feasible if and only if g is feasible.

Proof: Suppose that f is feasible. Then there exists a total edge-coloring f' such that $f'(e) = f(e)$ for $e \in E(u)$. One can extend g to a total edge-coloring g' as follows: $g'(e) = g(e)$ for $e \in E(u)$, and $g'(e) = f'(e)$ for $e \in E - E(u)$. Note that the subgraph $G[E(u)]$ of G shares common vertices with the other parts of G only in X_h and X_r. □

Thus a color vector on $u = (X_h, X_r)$ can be seen as an equivalence class of partial edge-colorings $f : E(u) \to C$. The total number of different color vectors on (X_h, X_r) is bounded by $2^{2(k+1)|C|}$ since $|X_h|, |X_r| \leqq k+1$.

The main step of our parallel algorithm is (i) to find a parsing tree T^p of T such that the height of T^p is $O(\log n)$, and (ii) to compute a table of all good color vectors on the root of T^p by means of a dynamic programming and a bottom-up tree computation on T^p. If the table has at least one good color vector, then the graph G corresponding to the root of T^p can be correctly colored with $|C|$ colors.

For each leaf $u = (X_h, X_r)$ of T^p, the table of all good color vectors on (X_h, X_r) can be computed in $O(1)$ parallel time using $O(|C|^{k(k+1)/2+1})$ operations as follows:

(1) enumerate all partial edge-colorings $f : E(u) \to C$; and

(2) compute all the color vectors $(\mathcal{S}, \mathcal{R})$ on (X_h, X_r) from the edge-colorings $f : E(u) \to C$.

Since $|E(u)| \leqq k(k+1)/2$, the number of edge-colorings $E(u) \to C$ is at most $|C|^{k(k+1)/2}$. Since k is bounded, steps (1) and (2) can be executed in $O(1)$ parallel time using $O(|C|^{k(k+1)/2+1})$ operations.

Next we show how to compute all good color vectors on the root of T^p. Let an internal node $u = (X_h, X_r)$ of T^p have three children $u_h = (X_h, X_i)$, $u_l = (X_i, X_l)$ and $u_r = (X_i, X_r)$. Let $\mathcal{C}_m(h,i,l,r)$ be a $6|C|$-tuple of vertex sets such that

(a) $\mathcal{C}_m(h,i,l,r) = (\mathcal{S}^h, \mathcal{R}^h; \mathcal{S}^l, \mathcal{R}^l; \mathcal{S}^r, \mathcal{R}^r)$,

(b) $\mathcal{S}^x = (S^x_1, \cdots, S^x_{|C|})$ and $\mathcal{R}^x = (R^x_1, \cdots, R^x_{|C|})$ for $x \in \{h, l, r\}$, and

(c) $S^h_c \subseteq X_h$, $R^h_c, S^l_c, S^r_c \subseteq X_i$, $R^l_c \subseteq X_l$, and $R^r_c \subseteq X_r$ for each color $c \in C$.

We call such $\mathcal{C}_m(h,i,l,r)$ a *multi-color vector* on (X_h, X_r). A multi-color vector $\mathcal{C}_m(h,i,l,r)$ is *good* if there exists a partial edge-coloring $f : E(u) \to C$ such that

$$S^h_c = \{v \in X_h | f_h(v) \ni c\}, R^h_c = \{v \in X_i | f_h(v) \ni c\},$$
$$S^l_c = \{v \in X_i | f_l(v) \ni c\}, R^l_c = \{v \in X_l | f_l(v) \ni c\},$$
$$S^r_c = \{v \in X_i | f_r(v) \ni c\}, R^r_c = \{v \in X_r | f_r(v) \ni c\},$$

for every color $c \in C$ where f_h, f_l and f_r are the partial edge-colorings of f restricted to $E(u_h)$, $E(u_l)$ and $E(u_r)$, respectively. Such $\mathcal{C}_m(h,i,l,r)$ is called the *multi-color vector of* f. Then we have the following lemma.

Lemma 4: Let an internal node $u = (X_h, X_r)$ of T^p have three children $u_h = (X_h, X_i)$, $u_l = (X_i, X_l)$ and $u_r = (X_i, X_r)$. Let $\mathcal{C}_m(h,i,l,r) = (\mathcal{S}^h, \mathcal{R}^h; \mathcal{S}^l, \mathcal{R}^l; \mathcal{S}^r, \mathcal{R}^r)$ be a multi-color vector on (X_h, X_r). Then $\mathcal{C}_m(h,i,l,r)$ is good if and only if the following (a) and (b) hold:

(a) $\mathcal{C}(h,i) = (\mathcal{S}^h, \mathcal{R}^h)$, $\mathcal{C}(i,l) = (\mathcal{S}^l, \mathcal{R}^l)$ and $\mathcal{C}(i,r) = (\mathcal{S}^r, \mathcal{R}^r)$ are good color vectors on (X_h, X_i), (X_i, X_l) and (X_i, X_r), respectively; and

(b) $(S^x_c \cup R^x_c) \cap (S^y_c \cup R^y_c) = \phi$ for any $x, y \in \{h, l, r\}$, $x \neq y$, and any $c \in C$.

Proof: $\Rightarrow$: Let $f : E(u) \to C$ be a partial edge-coloring with the multi-color vector $\mathcal{C}_m(h,i,l,r)$. Let f_h, f_l and f_r be partial edge-colorings of f restricted to $E(u_h)$, $E(u_l)$ and $E(u_r)$, respectively. Clearly $\mathcal{C}(h,i)$, $\mathcal{C}(i,l)$ and $\mathcal{C}(i,r)$ are color vectors of partial edge-colorings f_h, f_l and f_r, respectively, and hence $\mathcal{C}(h,i)$, $\mathcal{C}(i,l)$ and $\mathcal{C}(i,r)$ are good. Since f_h, f_l and f_r are partial edge-colorings of pairwise disjoint sets $E(u_h)$, $E(u_l)$ and $E(u_r)$ respectively and f is a partial edge-coloring of $E(u) = E(u_h) \cup E(u_l) \cup E(u_r)$, one can easily know that (b) holds true. (See Fig. 6 (c).)

$\Leftarrow$: Let g_h:$E(u_h) \to C$, g_l:$E(u_l) \to C$ and g_r:$E(u_r) \to C$ be partial edge-colorings with the color vectors $\mathcal{C}(h,i)$, $\mathcal{C}(i,l)$ and $\mathcal{C}(i,r)$, respectively. By the condition (b) clearly the following extension f of g_h, g_l and g_r

$$f(e) = \begin{cases} g_h(e) & \text{if } e \in E(u_h), \\ g_l(e) & \text{if } e \in E(u_l), \\ g_r(e) & \text{if } e \in E(u_r) \end{cases} \quad \text{and}$$

is a partial edge-coloring $E(u) \to C$, and $\mathcal{C}_m(h,i,l,r)$ is the multi-color vector of the partial edge-coloring f. Hence $\mathcal{C}_m(h,i,l,r)$ is good. □

The number of different multi-color vectors on any internal node $u = (X_h, X_r)$ of T^p is bounded by $2^{6(k+1)|C|}$. Therefore by Lemma 4 one can compute a table of all good multi-color vectors on u from the three given tables of all good color vectors on the children u_h, u_l and u_r in $O(1)$ parallel time using a total of $O(|C|2^{6(k+1)|C|})$ operations. Remember that $|C| = \Delta$ or $\Delta+1$. The following lemma shows how to compute the table of all good color vectors on u from the tables of all good multi-color vectors on u.

Lemma 5: Let an internal node $u = (X_h, X_r)$ of T^p have three children $u_h = (X_h, X_i)$, $u_l = (X_i, X_l)$ and $u_r = (X_i, X_r)$. Then a color vector $\mathcal{C}(h, r) = (\mathcal{S}, \mathcal{R})$ is good if and only if there exists a good multi-color vector $\mathcal{C}_m(h, i, l, r)$ such that $S_c = X_h \cap Y_c$ and $R_c = X_r \cap Y_c$ for every color $c \in C$, where $Y_c = S_c^h \cup R_c^h \cup S_c^l \cup R_c^l \cup S_c^r \cup R_c^r$.
Proof: Immediate from the definitions. □

From the lemmas above we have a parallel algorithm to determine whether a given partial k-tree G can be colored with $|C|$ colors:

(1) find a parsing tree T^p of a tree-decomposition T of G such that the height of T^p is $O(\log n)$;

(2) compute a table of all good color vectors on each leaf of T^p;

(3) for each internal node, all the three children of which are leaves, compute a table of all good color vectors on this node in parallel, and delete its children from T^p;

(4) repeat to do (3) until T^p has only one node; and

(5) finally check whether there exists a good color vector in the table corresponding to the root.

We apply the procedure above twice to G with choosing $|C| = \Delta$ and $|C| = \Delta + 1$ to determine the chromatic index of a partial k-tree G.

Since the height of T^p is $O(\log n)$, (3) is executed at most $O(\log n)$ times. Therefore one can easily execute the algorithm above in $O(\log n)$ parallel time. Clearly step (1) requires $O(n)$ operations in total. Step (2) requires $O((\Delta+1)^{(k^2+k+2)/2})$ operations for each leaf as mentioned before, and hence step (2) requires $O(n(\Delta+1)^{(k^2+k+2)/2})$ operations in total. As mentioned above, step (3) requires $O(\Delta 2^{6(k+1)(\Delta+1)})$ operations per node. Since (3) is executed for $O(n)$ nodes in total in step (4), step (4) requires $O(n\Delta 2^{6(k+1)(\Delta+1)})$ operations in total. Step (5) requires $O(2^{6(k+1)(\Delta+1)})$ operations in total. Thus the total number of required computational operations above is $O(n\{\Delta 2^{6(k+1)(\Delta+1)} + (\Delta+1)^{(k^2+k+2)/2}\})$.

This completes a proof of Theorem 1.

4. Conclusion

In this paper we gave an optimal and first NC parallel algorithm to solve the edge-coloring problem on partial k-trees G such that k and the maximum degree Δ of G are bounded. Given a partial k-tree G with its tree-decomposition, the algorithm takes in $O(\log n)$ parallel time with $O(n/\log n)$ processors where n is the number of vertices in G.

We have given an optimal parallel algorithm for decomposing a partial k-trees of large Δ into several subgraphs of small maximum degrees totaling exactly Δ [26]. Combining it with the algorithm in this paper, one can obtain an optimal parallel algorithm to solve the edge-coloring problem for any partial k-tree whose maximum degree is not necessarily bounded [27].

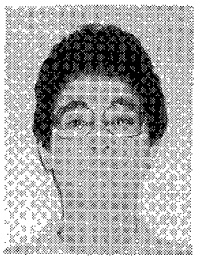
Our algorithm solves a single particular problem, that is, the edge-coloring problem. However the methods which we developed in this paper appear to be useful for many other problems, especially for the "edge-partition problem with respect to property π" which asks to partition the edge set of a given graph into a minimum number of subsets so that the subgraph induced by each subset satisfies the property π. For the edge-coloring problem, π is indeed a matching.

Consider for example a property π: the degree of each vertex v is not greater than $f(v)$, where $f(v)$ is a positive integer assigned to v. Clearly the edge-partition problem with respect to such a property π is the same as the so-called f-coloring problem [14], [21]. Our algorithm can be generalized to solve the f-coloring problem on partial k-trees in parallel.

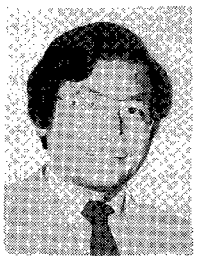
References

[1] Arnborg, S., Courcelle, B., Proskurowski, A. and Seese, D., "An algebraic theory of graph reduction," *Journal of the Association for Computing Machinery*, vol.40, no.5, pp.1134–1164, 1993.
[2] Arnborg, S. and Lagergren, J., "Easy problems for tree-decomposable graphs," *Journal of Algorithms*, vol.12, no.2, pp.308–340, 1991.
[3] Bodlaender, H.L., "NC-algorithm for graphs with small treewidth," In *Proc. of Int. Workshop WG'88 on Graph Theoretic Concepts in Computer Science*, Lecture Notes in Computer Science, 344, pp.1–10, Springer-Verlag, 1989.
[4] Bodlaender, H.L., "Polynomial algorithms for graph isomorphism and chromatic index on partial k-trees," *Journal of Algorithms*, vol.11, no.4, pp.631–643, 1990.
[5] Bodlaender, H.L., "A linear time algorithm for finding tree-decompositions of small treewidth," In *Proc. of the 25th Ann. ACM Symp. on Theory of Computing*, pp.226–234, San Diego, CA, 1993.
[6] Borie, R.B., Parker, R.G. and Tovey, C.A., "Automatic generation of linear-time algorithms from predicate calculus descriptions of problems on recursively constructed graph families," *Algorithmica*, vol.7, pp.555–581, 1992.
[7] Caspi, Y. and Dekel, E., "A near-optimal parallel algorithm for edge-coloring outerplanar graphs," manuscript, Computer Science Program, University of Texas at Dallas, Richardson, Tx 75083-0688A, 1992.
[8] Chrobak, M. and Nishizeki, T., "Improved edge-coloring algorithms for planar graphs," *Journal of Algorithms*, vol.11, pp.102–116, 1990.
[9] Courcelle, B., "The monadic second-order logic of graphs I: Recognizable sets of finite graphs," *Information and Computation*, vol.85, pp.12–75, 1990.
[10] Fiorini, S. and Wilson, R.J., *Edge-Colourings of Graphs*, Pitman, London, 1977.
[11] Gabow, H.N., Nishizeki, T., Kariv, O., Leven, D. and Terada., O., "Algorithms for edge-coloring graphs," Technical Report TRECIS-8501, Tohoku Univ., 1985.
[12] Gibbons, A. and Rytter, W., *Efficient Parallel Algorithms*, Cambridge Univ. Press, Cambridge, 1988.
[13] Gibbons, A.M. and Rytter, W., "Optimally edge-colouring outerplanar graphs is in NC," *Theoretical Computer Sci-*

ence, vol.71, pp.401–411, 1990.
[14] Hakimi, S.L. and Kariv, O., "On a generalization of edge-coloring in graphs," *Journal of Graph Theory*, vol.10, pp.139–154, 1986.
[15] He, X., "Efficient parallel algorithms for series-parallel graphs," *Journal of Algorithms*, vol.12, pp.409–430, 1991.
[16] He, X., and Yesha, Y., "Binary tree algebraic computations and parallel algorithm for simple graphs," *Journal of Algorithms*, vol.9, no.1, pp.92–113, 1988.
[17] Holyer, I., "The NP-completeness of edge-colouring," *SIAM J. Comput.*, vol.10, pp.718–720, 1981.
[18] JáJá, J., *An Introduction to Parallel Algorithms*, Addison-Wesley, New York, 1992.
[19] Karloff, J. and Shmoys, D.B., "Efficient parallel algorithms for edge-coloring problems," *Journal of Algorithms*, vol.8, pp.39–52, 1987.
[20] Kloks, T., *Treewidth – Computations and Approximations*, Lecture Notes in Computer Science, 842, Springer-Verlag, Berlin, 1994.
[21] Nakano, S., Nishizeki, T. and Saito, N., "On the f-coloring multigraphs," *IEEE Trans. on Circuits & Syst., CAS-35*, vol.3, pp.345–353, 1988.
[22] Nishizeki, T. and Chiba, N., *Planar Graphs: Theory and Algorithms*, North-Holland, Amsterdam, 1988.
[23] Reed, B.A., "Finding approximate separators and computing tree-width quickly," In *Proc. of the 24th Ann. ACM Symp. on Theory of Computing*, pp.221–228, 1992.
[24] Takamizawa, K., Nishizeki, T. and Saito, N., "Linear-time computability of combinatorial problems on series-parallel graphs," *Journal of ACM*, vol.29, no.3, pp.623–641, 1982.
[25] Terada, O. and Nishizeki, T., "Approximate algorithms for the edge-coloring of graphs," *Trans. Inst. of Electronics and Communication Eng. of Japan, J65-D*, vol.11, no.4, pp.1382–1389, 1982.
[26] Zhou, X., Nakano, S. and Nishizeki, T., "A linear algorithm for edge-coloring partial k-trees," In *Proc. of the First Europian Symposium on Algorithms*, Lecture Notes in Computer Science, 726, pp.409–418, Springer-Verlag, 1993.
[27] Zhou, X., Nakano, S. and Nishizeki, T., "A parallel algorithm for edge-coloring partial k-trees," In *Proc. of the Fourth Scandinavian Workshop on Algorithm Theory*, Lecture Notes in Computer Science, 824, pp.359–369, Springer-Verlag, 1994.
[28] Zhou, X., Suzuki, H. and Nishizeki, T., "Sequential and parallel algorithms for edge-coloring series-parallel multigraphs," In *Proc. of the Third Conference on Integer Programming and Combinatorial Optimization*, pp.129–145, 1993.



Xiao Zhou was born in Shanghai, China, on April 30, 1963. He received the B.E. degree from the East China Normal University, Shanghai, China, 1986, and the M.E. degree from Tohoku University, Sendai, Japan, in 1992. He is currently a Ph.D. student of Department of System Information Sciences, Graduate School of Information Sciences, Tohoku University. His areas of research interest are combinatorial algorithms, graph theory and computational complexity. He is a member of the Information Processing Society of Japan.



Takao Nishizeki was born in Sukagawa, Japan, on February 11, 1947. He received the B.E., M.E., and Dr.Eng. degrees from Tohoku University, Sendai, Japan, in 1969, 1971, and 1974, respectively, all in electrical communication engineering. He joined Tohoku University in 1974, and is currently Professor of Graduate School of Information Sciences. From 1977 to 1978 he was on leave at Mathematics Department of Carnegie-Mellon University, Pittsburgh, PA. His areas of research interest are combinatorial algorithms, VLSI routing, graph theory, network theory and computational complexity. He has published over 100 papers in these areas. He is coauthor of the book *Planar Graphs: Theory and Algorithms* (North-Holland, 1988), and is coediter of three books: *Graph Theory and Algorithms* (Springer-Verlag, 1980), *Discrete Algorithms and Complexity* (Academic Press, 1987), and *Algorithms* (Springer-Verlag, 1990). Dr. Nishizeki is a member of the Information Processing Society of Japan and the Association for Computing Machinery, and is a Fellow of IEEE.