アルゴリズム 52－9
（1996．7．24）

部分 k 木で辺素な道を見つけるアルゴリズム

周 暁　田村 朱麗　西関 隆夫
東北大学

本論文では部分 k 木 G の指定された端子対を結ぶ辺素な道を求める多項式時間アルゴリズムを3つ与える．G の点数を n として，端子対数を p としたとき，最初のアルゴリズムでは，$p = O(\log n)$ である限り，端子は G のどこにあってもよい．次のアルゴリズムでは端子対は何個あってもよいが，各端子対は分解木の同じ節点に入っていなければならない．これら2つを組み合せたのが3番目のアルゴリズムであり，ある条件を満足すれば端子対は必ずして分解木の同じ節点に入っていなくてもよいし，端子対が $O(\log n)$ より多くてもよい．

Finding Edge-Disjoint Paths in Partial k-Trees

Xiao Zhou,[1] *Syurei Tamura*[2] *and Takao Nishizeki*[2]

Abstract

For a given graph G and p pairs (s_i, t_i), $1 \leq i \leq p$, of vertices in G, the *edge-disjoint paths problem* is to find p pairwise edge-disjoint paths P_i, $1 \leq i \leq p$, connecting s_i and t_i. Many combinatorial problems can be efficiently solved for partial k-trees (graphs of treewidth bounded by a fixed integer k), but it has not been known whether the edge-disjoint paths problem can be solved in polynomial time for partial k-trees unless $p = O(1)$. This paper gives three algorithms for the edge-disjoint paths problem on partial k-trees. The first one solves the problem for any partial k-tree G and runs in polynomial time if $p = O(\log n)$ and in linear time if $p = O(1)$ where n is the number of vertices in G. The second solves the problem for any p in polynomial time if the graph G^+ obtained from G by adding p edges (s_i, t_i), $1 \leq i \leq p$, is also a partial k-tree. The third is a combination of the first and second, and solves the edge-disjoint paths problem under some resrtriction even if G^+ is not always a partial k-tree and p is not always $O(\log n)$.

1 Introduction

For a given graph G and p pairs (s_i, t_i), $1 \leq i \leq p$, of vertices in G, the *edge-disjoint paths problem* is to find p pairwise edge-disjoint paths P_i, $1 \leq i \leq p$, connecting terminals s_i and t_i in G. Figure 1 illustrates three edge-disjoint paths in a graph G. The *vertex-disjoint paths problem* is similarly defined. Both the edge-disjoint and vertex-disjoint paths problems are NP-complete even for planar graphs if p is not bounded [MP93]. If $p = O(1)$, then the vertex-disjoint paths problem can be solved in polynomial time for any graph by Robertson and Seymour's algorithm based on their series of papers on graph minor theory [RS95]. The edge-disjoint paths problem on a graph G can be reduced in polynomial time to the vertex-disjoint paths problem on a new graph G' constructed from G by replacing each vertex with a complete bipartite graph. Therefore, the edge-disjoint paths problem can also be solved in polynomial time for any graph by the algorithm if $p = O(1)$. However, the algorithm is not practically feasible due to the enormous constant factor [RS95]. On the other hand, practical algorithms for both problems with any p have been given for various classes of planar graphs or plane grids in which there are restrictions on the location of terminals or on the degrees of vertices, and these algorithms have been applied to VLSI-routings [Fra82, KM86, MNS85, MNS86, MP86, NSS85, SAN90, Seb93, SIN90, SNS89, WW92].

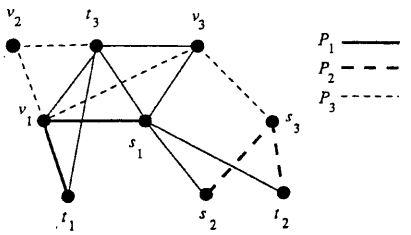


Figure 1: Three edge-disjoint paths P_1, P_2 and P_3 in a partial 3-tree G.

The class of partial k-trees includes trees ($k = 1$), series-parallel graphs ($k = 2$) [TNS82], Halin graphs ($k = 3$), and k-terminal recursive graphs. Any partial k-tree can be decomposed into a tree-like structure T of small "basis" graphs, each with at most $k+1$ vertices. Many problems can be solved efficiently for partial k-trees with bounded k by a dynamic programming algorithm based on the tree-decomposition [ACPS93, ALS91, BPT92, Cou90]. In particular,

[1]Education Center for Information Processing, Tohoku University, Sendai 980-77, JAPAN.
[2]Department of System Information Sciences, Graduate School of Information Sciences, Tohoku University, Sendai 980-77, JAPAN.

it is rather straightforward to design polynomial-time algorithms for vertex-type problems on partial k-trees. For example, the maximum independent vertex-set problem, minimum dominating vertex-set problem, vertex-coloring problem and vertex-disjoint paths problem with any p can be solved all in linear time for partial k-trees with bounded k [BPT92, Sch94, TP93]. (Indeed, Robertson and Seymour's algorithm for general graphs partly uses such an algorithm for partial k-trees.) However, this is not the case for edge-type problems such as the edge-coloring problem and the edge-disjoint paths problem with any p [BPT92]. It needs sophisticated treatment tailored for individual edge-type problems to design efficient algorithms. (See, for example, sophisticated linear-time algorithms for edge-coloring partial k-trees or series-parallel multigraphs [Bod90, ZNN, ZN95, ZSN96].) This is partly due to the following facts: the number of vertices in a basis graph (a node of a tree-decompostition T) is bounded by $k+1$ and hence the size of a DP table required to solve vertex-type problems can be easily bounded by a constant, say 2^{k+1} or $(k+1)^{k+1}$; however, the number of edges incident to vertices in a basis graph is not always bounded and hence it is difficult to bound the size of a DP table for edge-type problems by a constant.

In this paper we give three algorithms to solve the edge-disjoint paths problem for partial k-trees G with bounded k. Since the edge-disjoint paths problem with $p = O(1)$ can be expressed in the monadic second-order logic [Lag96], a linear-time algorithm can be automatically generated for partial k-trees by the general methods in [ALS91, BPT92, Cou90] if $p = O(1)$ although the constant factor is huge. On the other hand, the first one of our three algorithms solves the edge-disjoint paths problem with any p in time

$$O(n\{(p+k^2)p^{k(k+1)/2} + p(k+3)^{2(k+4)p+3}\}),$$

where n is the number of vertices in G. Since p is in a single exponent over a constant, our algorithm runs in polynomial-time if $p = O(\log n)$, and in linear time (with a relatively small constant factor) if $p = O(1)$. It should be noted that the edge-disjoint paths problem on a partial k-tree G can be reduced to the vertex-disjoint paths problem on a new graph G', but G' is not always a partial k-tree. Therefore, the vertex-disjoint paths algorithm for partial k-trees [Sch94] cannot be applied to the edge-disjoint paths problem. The second one of our three algorithms solves the edge-disjoint paths problem for partial k-tree G in time

$$O(n\{(p+k^2)p^{(k+1)(k+3)/2} + (k+1)^{6k+15}p^{(k+1)^{4k+8}}\})$$

if the graph G^+ obtained from G by adding p edges (s_i, t_i), $1 \leq i \leq p$, is also a partial k-tree. Since p is not in the exponents, the algorithm runs in polynomial time for any p. The third one is a combination of the first two, and solves the edge-disjoint paths problem for partial k-tree G in polynomial time under some conditions even if G^+ is not always a partial k-tree and p is not always $O(\log n)$. Our idea is to formulate the edge-disjoint paths problem as a new type of an edge-coloring problem, and to bound the size of a DP table by $(k+3)^{(k+3)p}$ or $(k+1)^{4(k+2)}p^{(k+1)^{4(k+2)}}$ applying and extending techniques developed for the ordinary edge-coloring problem [Bod90, ZN95, ZNN].

2 Terminology and Definitions

In this section we give some definitions. 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. We denote by n the number of vertices in G. The paper deals with *simple undirected* graphs without multiple edges or self-loops. An edge joining vertices u and v is denoted by (u, v). For $E' \subseteq E(G)$, $G[E']$ denotes the subgraph of G induced by the edges in E'; $G[E']$ contains every vertex of G to which at least one edge in E' is incident, and hence $G[E']$ contains no isolated vertex.

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

(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 $G' = (V \cup \{w\}, E \cup \{(v_i, w) | 1 \leq i \leq k\})$ 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).

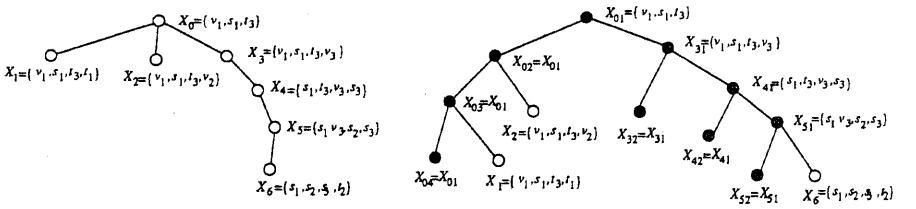
A graph is a *partial k-tree* if it is a subgraph of a k-tree. Thus a partial k-tree $G = (V, E)$ is a simple graph, and $|E| < kn$. The graph in Figure 1 is a partial 3-tree. In this paper we assume that k is a fixed constant.

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 [RS86]:

- $\bigcup_{X_i \in V_T} X_i = V$;
- 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_j lies on the path in T from node X_i to node X_l, then $X_i \cap X_l \subseteq X_j$.

Figure 2(a) illustrates a tree-decomposition of the partial 3-tree in Figure 1. The *width* of a tree-decomposition $T = (V_T, E_T)$ is $\max_{X_i \in V_T} |X_i| - 1$. The *treewidth* of graph G is the minimum width of a tree-decomposition of G, taken over all possible tree-decompositions of G. It is known that every graph with treewidth $\leq k$ is a partial k-tree, and conversely, that every partial k-tree has a tree-decomposition with width $\leq k$. Bodlaender has given a linear-time sequential algorithm to find a tree-decomposition of G with width $\leq k$ for fixed k [Bod93].

Consider a tree-decomposition of a partial k-tree G with treewidth $\leq k$. We transform it to a binary tree T as follows [Bod90]: regard T as a rooted tree with choosing an arbitrary node as the root X_0, 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_q} is the father of $X_{i_{q+1}}$ and the qth child X_{j_q} of X_i $(1 \leq q \leq r)$, and $X_{i_{r+1}}$ is a leaf of T. This transformation can be done in $O(n)$ time. T is a tree-decomposition of $G = (V, E)$ with the following characteristics:



(a) tree-decomposition (b) binary tree-decomposition

Figure 2: Tree-decompositions of the partial 3-tree in Figure 1.

- 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 [Bod90]. Figure 2(b) illustrates a binary transformation of the tree-decompostition in Figure 2(a). Clearly T has treewidth $\leq k$. 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 define an edge-set $E(X_i) \subseteq E$ for each node X_i of T as follows. If X_i is a leaf of T, then let $E(X_i) = \{e \in E |\ rep(e) \text{ is } X_i\}$. If X_i is an internal node of T having two children X_l and X_r, then let $E(X_i) = E(X_l) \cup E(X_r)$. Note that the two edge-sets $E(X_l)$ and $E(X_r)$ are disjoint. Thus node X_i of T corresponds to a subgraph $G[E(X_i)]$ of G induced by the edges in $E(X_i)$. The subgraph, denoted simply by $G[X_i]$, is an edge-disjoint union of two subgraphs $G[X_l]$ and $G[X_r]$, which share common vertices only in X_i because of the third property of a tree-decomposition.

3 The First Algorithm

In this section we give the first one of our three algorithms, proving the following theorem. Although all our algorithms only decide whether G has p edge-disjoint paths P_i, $1 \leq i \leq p$, connecting s_i and t_i, they can be easily modified so that they actually find such p edge-disjoint paths.

Theorem 3.1 *Let G be a partial k-tree of n vertices given by its tree-decomposition with treewidth $\leq k$. Let (s_i, t_i), $1 \leq i \leq p$, be p pairs of vertices in G. Then one can determine in time $O(n\{(p + k^2)p^{k(k+1)/2} + p(k + 3)^{2(k+4)p+3}\})$ whether G has p pairwise edge-disjoint paths P_i, $1 \leq i \leq p$, connecting s_i and t_i.*

If $k = O(1)$ and $p = O(\log n)$, then $p(k+3)^{2(k+4)p+3}$ is polynomial in n. Therefore we have the following corollary.

Corollary 3.2 *If $k = O(1)$ and $p = O(\log n)$, then the edge-disjoint paths problem can be solved in polynomial time for partial k-trees. If $k, p = O(1)$, then the edge-disjoint paths problem can be solved in linear time for partial k-trees.*

In the remaining of this section we will give a proof of Theorem 3.1. Our idea is to formulate the edge-disjoint paths problem as a new type of an edge-coloring problem, and then to solve the coloring problem using a dynamic programming with a table of size at most $(k+3)^{(k+4)p}$. We employ techniques developed for the ordinary edge-coloring problem [Bod90].

Let $G = (V, E)$ be a graph, and let (s_i, t_i), $1 \leq i \leq p$, be pairs of vertices in V called *terminals*. Let $C = \{1, 2, \cdots, p\}$ be the set of colors. Any mapping $f : E \rightarrow C$ is called a *coloring* of graph G. For a color $c \in C$, we denote by $G(f, c)$ the subgraph of G induced by the edges which are colored by c. We call f a *correct coloring* of G if, for each color $c \in C$, $G(f, c)$ has a connected component containing both terminals s_c and t_c. Figure 3 depicts a correct coloring of the graph in Figure 1. Clearly we easily have the following lemma holds.

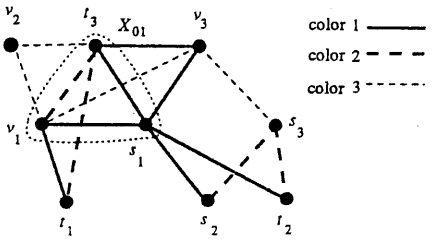


Figure 3: A correct coloring of the partial 3-tree G in Figure 1.

Lemma 3.3 *There are p edge-disjoint paths in a graph G if and only if there is a correct coloring of G.*

Thus the remaining problem is how to determine whether there is a correct coloring of G.

Let X_i be a node of a binary tree-decomposition T of a partial k-tree G. A coloring of graph $G[X_i]$ is *feasible* if it can be extended to a correct coloring of $G = G[X_{01}]$, where X_{01} is the root of T. A *color vector* $\mathcal{C}(X_i) = (\mathcal{Y}_1, \mathcal{Y}_2, \cdots, \mathcal{Y}_p)$ on X_i is defined to be a p-tuple of families $\mathcal{Y}_1, \mathcal{Y}_2, \cdots, \mathcal{Y}_p$ of vertex sets; $\mathcal{Y}_c$, $c \in C$, is a family of pairwise disjoint subsets of set $X_i \cup \{s_c, t_c\}$. A color vector $\mathcal{C}(X_i) = (\mathcal{Y}_1, \mathcal{Y}_2, \cdots, \mathcal{Y}_p)$ is defined *active* if $G[X_i]$ has a coloring f such that for each color $c \in C$

$$\mathcal{Y}_c = \{V(D) \cap (X_i \cup \{s_c, t_c\}) \mid D \text{ is a connected component of } G[X_i](f, c)\}.$$

Such $\mathcal{C}(X_i)$ is called the *color vector of the coloring* f of $G[X_i]$. We now have the following lemma.

Lemma 3.4 *Let X_i be any node of a tree-decomposition T of a partial k-tree G. Let f and g be colorings of $G[X_i]$ with the same color vector. Then f is feasible if and only if g is feasible.*

Thus a color vector on X_i can be seen as an equivalence class of colorings of $G[X_i]$. Since $|X_i| \leq k+1$, $|X_i \cup \{s_c, t_c\}| \leq k+3$. The total number of distinct partitions of set $X_i \cup \{s_c, t_c\}$ is the $(k+3)$-rd Bell number $B(k+3)$. Since $B(k+3) \leq (k+3)! \leq (k+3)^{k+3}$, the number of distinct families of pairwise disjoint subsets of $X_i \cup \{s_c, t_c\}$ is at most $(k+3)B(k+3) \leq (k+3)^{k+4}$. Therefore, the number of distinct families $\mathcal{Y}_c$ is at most $(k+3)^{k+4}$ for each color $c \in C$. Hence the total number of different color vectors on X_i is at most $(k+3)^{(k+4)p}$.

The main step of our algorithm is to compute a table of all active color vectors on each node of T from leaves to the root X_{01} of T by means of a dynamic programming. From the table on X_{01} one can easily check whether G has p edge-disjoint paths, as follows.

Lemma 3.5 *A partial k-tree G has a correct coloring and hence has p edge-disjoint paths if and only if the table on root X_{01} has at least one active color vector $\mathcal{C}(X_{01}) = (\mathcal{Y}_1, \mathcal{Y}_2, \cdots, \mathcal{Y}_p)$ such that each family $\mathcal{Y}_c$, $c \in C$, contains a set Q_{cj} such that $s_c, t_c \in Q_{cj}$.*

We first compute the table of all active color vectors on each leaf X_i of T as follows:

(1) enumerate all colorings f of $G[X_i]$; and

(2) compute all the active color vectors $(\mathcal{Y}_1, \mathcal{Y}_2, \cdots, \mathcal{Y}_p)$ on X_i from the colorings f of $G[X_i]$.

Since $|E(X_i)| \leq k(k+1)/2$, the number of colorings $f : E(X_i) \to C$ is $p^{k(k+1)/2}$. For each coloring f of $G[X_i]$, one can compute the color vector of f in time $O(p + k^2)$. Therefore, steps (1) and (2) can be executed for a leaf in time $O((p + k^2)p^{k(k+1)/2})$.

We next compute all active color vectors on each internal node X_i of T from leaves to the root. The following lemma shows how to compute all active color vectors on X_i from all active color vectors on the left and right children X_l and X_r of X_i. Let $C(X_l) = (\mathcal{Y}_{l1}, \mathcal{Y}_{l2}, \cdots, \mathcal{Y}_{lp})$ be an active color vector on X_l, and let $C(X_r) = (\mathcal{Y}_{r1}, \mathcal{Y}_{r2}, \cdots, \mathcal{Y}_{rp})$ be an active color vector on X_r. For each color $c \in C$, let $B_c = (\mathcal{Y}_{lc} \cup \mathcal{Y}_{rc}, E_c)$ be a bipartite graph with partite sets $\mathcal{Y}_{lc}$ and $\mathcal{Y}_{rc}$, where a vertex $L \in \mathcal{Y}_{lc}$ and a vertex $R \in \mathcal{Y}_{rc}$ are joined by an edge in E_c iff $L \cap R \neq \phi$. Let $B_{c1}, B_{c2}, \cdots, B_{cx}$ be the connected components of B_c, and for j, $1 \leq j \leq x$, let

$$W_{cj} = \bigcup_{S \in V(B_{cj})} S,$$

then W_{cj} corresponds to the vertex set of a connected component of $G[X_i](f, c)$ for a coloring f of $G[X_i]$. For a set $Y \subseteq V$ we define a family $U(\mathcal{Y}_{lc}, \mathcal{Y}_{rc}; Y)$ of vertex sets, as follows:

$$U(\mathcal{Y}_{lc}, \mathcal{Y}_{rc}; Y) = \{Y \cap W_{cj} \mid 1 \leq j \leq x\}.$$

We have the following lemma.

Lemma 3.6 *Let an internal node X_i of T have two children X_l and X_r. Then a color vector $\mathcal{C}(X_i) = (\mathcal{Y}_1, \mathcal{Y}_2, \cdots, \mathcal{Y}_p)$ on X_i is active if and only if there exists an active color vector $C(X_l) = (\mathcal{Y}_{l1}, \mathcal{Y}_{l2}, \cdots, \mathcal{Y}_{lp})$ on X_l and $C(X_r) = (\mathcal{Y}_{r1}, \mathcal{Y}_{r2}, \cdots, \mathcal{Y}_{rp})$ on X_r such that $\mathcal{Y}_c = U(\mathcal{Y}_{lc}, \mathcal{Y}_{rc}; X_i \cup \{s_c, t_c\})$ for each color $c \in C$.*

Since $|\mathcal{Y}_{lc}|, |\mathcal{Y}_{rc}| \leq k+3$, $|E_c| \leq (k+3)^2$. Clearly one can check in time $O(k+3)$ whether $L \cap R \neq \phi$. Therefore each bipartite graph B_c can be constructed in time $O((k+3)^3)$, and hence all p bipartite graphs can be constructed in time $O(p(k+3)^3)$. Thus one can compute an active color vector on X_i from a pair of active color vectors on X_l and on X_r in time $O(p(k+3)^3)$. There are at most $(k+3)^{2(k+4)p}$ pairs of active color vectors on X_l and on X_r. Therefore one can compute all active color vectors on X_i in time $O(p(k+3)^{2(k+4)p+3})$.

Since T has at most n internal nodes and at most n leaves, the total time required by the algorithm above is $O(n\{(p + k^2)p^{k(k+1)/2} + p(k+3)^{2(k+4)p+3}\})$.

This completes a proof of Theorem 3.1.

4 The Second and Third Algorithms

In this section, proving the following theorem, we first give the second one of our three algorithms, and then give the third one combining the first and second.

Theorem 4.1 *Let G be a partial k-tree of n vertices given by its binary tree-decomposition T with treewidth $\leq k$. Let (s_i, t_i), $1 \leq i \leq p$, be p pairs of vertices in G. Assume that for each i, $1 \leq i \leq p$, T has a node X such that $s_i, t_i \in X$. Then one can determine in time*

$$O(n\{(p+k^2)p^{(k+1)(k+3)/2} + (k+1)^{6k+15}p^{(k+1)^{4k+8}}\})$$

whether G has p pairwise edge-disjoint paths P_i, $1 \leq i \leq p$, connecting s_i and t_i.

Thus the algorithm runs in polynomial time for any p. For the partial k-tree G in Theorem 4.1, the graph G^+ is obtained from G by adding edge (s_i, t_i), $1 \leq i \leq p$, is a partial k-tree. In the remaining of this section we will give a proof of Theorem 4.1. One may assume without loss of generality that any vertex v in G is designated as at most one terminal and that every terminal has degree one; for each vertex v designated a terminal, add G a new vertex v', join v and v', and newly designate v' as the terminal. Note that the resulting graph is also a partial k-tree. Figure 4 illustrates the resulting graph obtained from the partial 3-tree in Figure 1.

Let $C = \{1, 2, \cdots, p\}$ be the set of colors. In this section we say that a coloring $f : E \to C$ of a graph $G = (V, E)$ is *correct* if every $G(f, c)$, $c \in C$, contains exactly one terminal pair, both in the same connected component of $G(f, c)$. Note that $G(f, c)$ does not always contains terminal pair s_c and t_c. Figure 4 depicts a correct coloring of the graph in Figure 4. Clearly the following lemma holds.

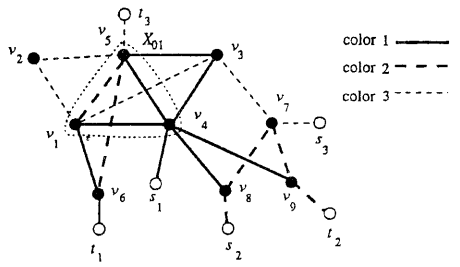


Figure 4: A correct coloring of the partial 3-tree G.

Lemma 4.2 *Graph G has edge-disjoint paths if and only if there is a correct coloring of G.*

Thus the remaining problem is how to determine whether there is a correct coloring of G.

Let X_i be a node of a tree-decomposition T of a partial k-tree G. Since every pair (s_c, t_c) of terminals are contained in the same node of tree T, either both s_c and t_c are contained in $G[X_i]$ or neither s_c nor t_c is contained in $G[X_i]$. Thus if a coloring $f : E(X_i) \to C$ of $G[X_i]$ can be extended to a colorect coloring of G, then every $G[X_i](f, c)$, $c \in C$, must satisfy the following i) and ii):

i) if $G[X_i](f, c)$ contains a terminal, then $G[X_i](f, c)$ contains exactly one pair of terminals; and

ii) if $G[X_i](f, c)$ contains exactly one pair of terminals which are in two different connected components of $G[X_i](f, c)$, then both of the components contain at least one vertex in X_i.

Such f is called a *valid coloirng of* $G[X_i]$. If $G[X_i](f, c)$ contains exactly one pair of terminas which are in two difference connected components G'_c and G''_c, but either $V(G'_c) \cap X_i = \phi$ or $V(G''_c) \cap X_i = \phi$, then the coloring f of $G[X_i]$ is not valid. Note that such f cannot be extended to a correct coloring of G. If $G[X_i](f, c)$ contains no terminals, then the pair of termianls corresponding to c is not contained in $G[X_i]$ but some edges in $G[X_i](f, c)$ may be used by a path connecting the pair later. Clearly the following lemma holds.

Lemma 4.3 *Let f be a coloring of $G[X_i]$, that is, $E(X_i) \to C$. Let $\varphi : C \to C$ be a permutation (bijection), and let $\varphi \circ f : E(X_i) \to C$ be the composite of f and φ. Then the coloring f is valid if and only if the coloring $\varphi \circ f$ is valid.*

Let $\mathcal{P}(X_i)$ be the set of all families of pairwise disjoint subsets of set X_i. In this section we define a *color vector* $C(X_i) = (\mathcal{Y}; \mathcal{Z})$ on X_i to be a tuple of $\mathcal{Y}$ and $\mathcal{Z}$, where $\mathcal{Y} = (\mathcal{Y}_1, \mathcal{Y}_2, \cdots, \mathcal{Y}_p)$ and $\mathcal{Z} = (\mathcal{Z}_1, \mathcal{Z}_2, \cdots, \mathcal{Z}_p)$ are p-tuples of families in $\mathcal{P}(X_i)$. Thus $C(X_i)$ is a $2p$-tuple of families in $\mathcal{P}(X_i)$. A color vector $C(X_i) = (\mathcal{Y}; \mathcal{Z})$ is *active* if $G[X_i]$ has a valid coloring f such that, for each color $c \in C$, $\mathcal{Y}_c$ and $\mathcal{Z}_c$ satisfies the following (a) and (b), respectively:

(a) $\mathcal{Y}_c = \{V(D) \cap X_i \mid D$ is a connected component of $G[X_i](f, c)\}$, and

(b) $\mathcal{Z}_c = \{V(D) \cap X_i \mid D$ is a connected component of $G[X_i](f, c)$ and D contains a terminal$\}$,

Thus $\mathcal{Z}_c \subseteq \mathcal{Y}_c$ and $|\mathcal{Z}_c| = 0, 1, 2$ for each $c \in C$. Such $C(X_i)$ is called the *color vector of the coloring* f of $G[X_i]$.

A valid coloring f with a color vector $C(X_i) = (\mathcal{Y}; \mathcal{Z})$ is *feasible* if it can be extended to a correct coloring f^* of $G = G[X_{01}]$, where X_{01} is the root of T. We now have the following lemma.

Lemma 4.4 *Let X_i be any node of a tree-decomposition T of a partial k-tree G. Let f and g be valid colorings of $G[X_i]$ with the same color vector $C(X_i)$. Then f is feasible if and only if g is feasible.*

Thus a color vector on X_i can be seen as an equivalence class of colorings of $G[X_i]$. A color vector $C(X_i)$ is *good* if $G[X_i]$ has a feasible coloring f with the color vector $C(X_i)$. Then we have the following lemma.

Lemma 4.5 *Let $(\mathcal{Y};\mathcal{Z})$ and $(\mathcal{Y}';\mathcal{Z}')$ be two color vectors on X_i, and let $\mathcal{Y}=(\mathcal{Y}_1,\mathcal{Y}_2,\cdots,\mathcal{Y}_p)$, $\mathcal{Z}=(\mathcal{Z}_1,\mathcal{Z}_2,\cdots,\mathcal{Z}_p)$, $\mathcal{Y}'=(\mathcal{Y}'_1,\mathcal{Y}'_2,\cdots,\mathcal{Y}'_p)$ and $\mathcal{Z}'=(\mathcal{Z}'_1,\mathcal{Z}'_2,\cdots,\mathcal{Z}'_p)$. Suppose that there exists a permutation $\varphi: C\to C$ such that $\mathcal{Y}_c=\mathcal{Y}'_{\varphi(c)}$ and $\mathcal{Z}_c=\mathcal{Z}'_{\varphi(c)}$ for each $c\in C$, that is, for any families $\mathcal{A},\mathcal{B}\in\mathcal{P}(X_i)$*

$$|\{c\in C\mid \mathcal{A}=\mathcal{Y}_c,\mathcal{B}=\mathcal{Z}_c\}|=|\{c\in C\mid \mathcal{A}=\mathcal{Y}'_c,\mathcal{B}=\mathcal{Z}'_c\}|.$$

Then $(\mathcal{Y};\mathcal{Z})$ is good if and only if $(\mathcal{Y}';\mathcal{Z}')$ is good.

We call a mapping $\gamma:\mathcal{P}(X_i)\times\mathcal{P}(X_i)\to\{0,1,2,\cdots,p\}$ a *count on node X_i*. A count γ on X_i is *active* if there exists an active color vector $C(X_i)=(\mathcal{Y};\mathcal{Z})$ such that $\gamma(\mathcal{A},\mathcal{B})=|\{c\in C\mid \mathcal{A}=\mathcal{Y}_c,\mathcal{B}=\mathcal{Z}_c\}|$ for all families $\mathcal{A},\mathcal{B}\in\mathcal{P}(X_i)$. Such γ is called a *count of the color vector $C(X_i)$*, and is called a *count of a coloring f* of $G[X_i]$ if $C(X_i)$ is the color vector of f.

By Lemma 4.5 γ is seen as an equivalence class of active counts on X_i. Clearly the total number of distinct γ's is at most $(p+1)^{|\mathcal{P}(X_i)|^2}$. The total number of all the partitions of set X_i is the $|X_i|$-th Bell number $B(|X_i|)$. Since $|X_i|\le k+1$, $B(|X_i|)\le(k+1)!\le(k+1)^{k+1}$. Since $\mathcal{P}(X_i)$ be the set of all families of pairwise disjoint subsets of set X_i, $|\mathcal{P}(X_i)|\le|X_i|B(|X_i|)$. Therefore, let $K=(k+1)^{k+2}$, then $|\mathcal{P}(X_i)|\le K$. Thus the total number of distinct γ's is at most $(p+1)^{K^2}$.

We easily have the following lemma.

Lemma 4.6 *Let f be a valid coloring of $G[X_i]$, and let $\varphi: C\to C$ be a permutation. Then the count of f is the same as the count of $\varphi\circ f$.*

The main step of our algorithm is to compute a table of all active counts on each node of T from leaves to the root X_{01} of T by means of a dynamic programming. From the table on X_{01} one can easily check whether G has p edge-disjoint paths, as follows.

Lemma 4.7 *A partial k-tree G has a correct coloring and hence has p edge-disjoint paths if and only if the table on root X_{01} has at least one active count γ such that $\gamma(\mathcal{A},\mathcal{B})=0$ for all families $\mathcal{A},\mathcal{B}\in\mathcal{P}(X_{01})$ with $|\mathcal{B}|\neq 1$.*

We first compute the table of all active counts on each leaf X_i of T as follows:

1) enumerate all colorings f of $G[X_i]$;

2) compute all the active color vectors $(\mathcal{Y};\mathcal{Z})$ on X_i from the colorings f of $G[X_i]$; and

3) compute all the active counts from the active color vectors.

Since $|E(X_i)|\le k(k+1)/2$, the number of distinct colorings $f: E(X_i)\to C$ is at most $p^{k(k+1)/2}$. For each coloring f of $G[X_i]$, one can compute the active color vector of f in time $O(p+k^2)$, and compute the active count of the active color vector in time $O(p+k^2)$. Therefore, steps 1)–3) can be executed for a leaf in time $O((p+k^2)p^{k(k+1)/2})$.

We next compute all active counts on each internal node X_i of T from leaves to the root. In order to compute all active counts on internal nodes X_i, we need to define an "active pair-count." Let X_i be an internal node with children X_l and X_r. We may assume that $X_i=X_l$. Note that $E(X_i)=E(X_l)\cup E(X_r)$ and $E(X_l)\cap E(X_r)=\phi$. We call a mapping $\pi:\mathcal{P}(X_l)\times\mathcal{P}(X_l)\times\mathcal{P}(X_r)\times\mathcal{P}(X_r)\to\{0,1,2,\cdots,p\}$ a *pair-count* on X_i. We define a pair-count π to be *active* if there is a valid coloring $f: E(X_i)\to C$ such that, for each 4-tuple $(\mathcal{A}_l,\mathcal{B}_l,\mathcal{A}_r,\mathcal{B}_r)$ with $\mathcal{A}_l,\mathcal{B}_l\in\mathcal{P}(X_l)$ and $\mathcal{A}_r,\mathcal{B}_r\in\mathcal{P}(X_r)$

$$\pi(\mathcal{A}_l,\mathcal{B}_l;\mathcal{A}_r,\mathcal{B}_r)=|\{c\in C\mid \mathcal{A}_l=\mathcal{Y}^l_c,\mathcal{B}_l=\mathcal{Z}^l_c,\mathcal{A}_r=\mathcal{Y}^r_c,\mathcal{B}_r=\mathcal{Z}^r_c\}|,$$

where $(\mathcal{Y}^l;\mathcal{Z}^l)=(\mathcal{Y}^l_1,\mathcal{Y}^l_2,\cdots,\mathcal{Y}^l_p;\mathcal{Z}^l_1,\mathcal{Z}^l_2,\cdots,\mathcal{Z}^l_p)$ is the color vector of colorings $f|G[X_l]$ on X_l, and $(\mathcal{Y}^r;\mathcal{Z}^r)=(\mathcal{Y}^r_1,\mathcal{Y}^r_2,\cdots,\mathcal{Y}^r_p;\mathcal{Z}^r_1,\mathcal{Z}^r_2,\cdots,\mathcal{Z}^r_p)$ is the color vector of $f|G[X_r]$ on X_r. Such π is called a *pair count of a valid coloring* f of $G[X_i]$. Then we have the following lemma.

Lemma 4.8 *Let an internal node X_i of T have two children X_l and X_r, and let $X_i=X_l$. Let π be a pair-count on X_i. Then π is active if and only if the following* (a) *and* (b) *hold:*

(a) $\pi(\mathcal{A}_l,\mathcal{B}_l;\mathcal{A}_r,\mathcal{B}_r)=0$ *if* $|\mathcal{B}_l|\ge 1$ *and* $|\mathcal{B}_r|\ge 1$;

(b) $\pi(\mathcal{A}_l,\mathcal{B}_l;\mathcal{A}_r,\mathcal{B}_r)=0$ *if* $|\mathcal{B}_r|=2$, $\mathcal{B}_r=\{S_1,S_2\}$, $X_i\cap S_1\neq\phi$ *and* $X_i\cap S_2\neq\phi$; *and*

(c) *there exist two active counts γ_l on X_l and γ_r on X_r such that*

1) $\gamma_l(\mathcal{A}_l,\mathcal{B}_l)=\sum_{\mathcal{A},\mathcal{B}\in\mathcal{P}(X_r)}\pi(\mathcal{A}_l,\mathcal{B}_l;\mathcal{A},\mathcal{B})$ *for each pair $\mathcal{A}_l,\mathcal{B}_l\in\mathcal{P}(X_l)$; and*

2) $\gamma_r(\mathcal{A}_r,\mathcal{B}_r)=\sum_{\mathcal{A},\mathcal{B}\in\mathcal{P}(X_l)}\pi(\mathcal{A},\mathcal{B},\mathcal{A}_r;\mathcal{B}_r)$ *for each pair $\mathcal{A}_r,\mathcal{B}_r\in\mathcal{P}(X_r)$.*

Clearly one can compute an active pair-count on X_i from a pair of active counts on X_l and X_r in time $O(|\mathcal{P}(X_l)|^2 |\mathcal{P}(X_r)|^2)$. There are at most $p^{|\mathcal{P}(X_l)|^2|\mathcal{P}(X_r)|^2}$ pairs of active color vectors on X_l and on X_r. Since $|X_l|, |X_r| \leq k+1$, $|\mathcal{P}(X_l)|, |\mathcal{P}(X_r)| \leq K = (k+1)^{k+2}$. Therefore one can compute all active pair-counts on X_i in time $O(K^4 p^{K^4})$.

We now show how to compute all active counts on an internal node X_i from all active pair-counts on X_i.

Lemma 4.9 *Let X_i be an internal node of T. A count γ on X_i is active if and only if there exists an active pair-count π on X_i such that $\gamma(\mathcal{A}, \mathcal{B}) = \sum \pi(\mathcal{A}_l, \mathcal{B}_l; \mathcal{A}_r, \mathcal{B}_r)$ for each pair $\mathcal{A}, \mathcal{B} \in \mathcal{P}(X_i)$ where the summation is taken over all 4-tuples $(\mathcal{A}_l, \mathcal{B}_l, \mathcal{A}_r, \mathcal{B}_r)$ such that*

(a) $\mathcal{A}_l, \mathcal{B}_l \in \mathcal{P}(X_l)$;

(b) $\mathcal{A}_r, \mathcal{B}_r \in \mathcal{P}(X_r)$;

(c) *either* $|\mathcal{B}_l| = 0$ *or* $|\mathcal{B}_r| = 0$;

(d) $\mathcal{A} = U(\mathcal{A}_l, \mathcal{A}_r; X_i)$, *where U has been defined in the previous section;*

(e) *if* $|\mathcal{B}| \geq 1$, *then either* $|\mathcal{B}_l| \geq 1$ *or* $|\mathcal{B}_r| \geq 1$;

(f) *if* $|\mathcal{B}| = 1$ *and* $\mathcal{B} = \{S\}$, *then family* $S \in \mathcal{A}$ *and* $\bigcup_{Y \in \mathcal{B}_l \cup \mathcal{B}_r} (Y \cap X_i) \subseteq S$;

(g) *if* $|\mathcal{B}| \geq 2$ *and* $|\mathcal{B}_l| \geq 1$, *then* $\mathcal{B}_l = \{B_{l1}, B_{l2}\}$ *and family* $\mathcal{A}$ *contains sets* S_1, S_2 $(\subseteq X_i)$ *such that* $B_{l1} \subseteq S_1$, $B_{l2} \subseteq S_2$ *and* $\mathcal{B} = \{S_1, S_2\}$; *and*

(h) *if* $|\mathcal{B}| \geq 2$ *and* $|\mathcal{B}_r| \geq 1$, *then* $\mathcal{B}_r = \{B_{r1}, B_{r2}\}$ *and family* $\mathcal{A}$ *contains sets* S_1, S_2 $(\subseteq X_i)$ *such that* $B_{r1} \cap X_i \subseteq S_1$, $B_{r2} \cap X_i \subseteq S_2$ *and* $\mathcal{B} = \{S_1, S_2\}$.

Since $|\mathcal{A}_l|, |\mathcal{A}_r| \leq k+1$, $|E_c| \leq (k+1)^2$ for the bipartite graph $B_c = (\mathcal{A}_l \cup \mathcal{A}_r, E_c)$ defined in the previous section. Clearly one can check in time $O((k+1)^3)$ whether $\mathcal{A} = U(\mathcal{A}_l, \mathcal{A}_r; X_i)$, constructing a bipartite graph for $\mathcal{A}_l$ and $\mathcal{A}_r$. Thus one can compute an active count on X_i from an active pair-count on X_i in time $O((k+1)^3 |\mathcal{P}(X_i)|^2 |\mathcal{P}(X_l)|^2 |\mathcal{P}(X_r)|^2) \leq (k+1)^3 K^6$ since $|\mathcal{P}(X_i)|, |\mathcal{P}(X_l)|, |\mathcal{P}(X_r)| \leq K$. There are at most $p^{|\mathcal{P}(X_l)|^2|\mathcal{P}(X_r)|^2} \leq p^{K^4}$ active pair-counts on X_i. Therefore one can compute all active counts on X_i in time $O((k+1)^3 K^6 p^{K^4})$. Since T has at most n internal nodes and at most n leaves, the total time required by the algorithm above is

$$O(n\{(p+k^2)p^{k(k+1)/2} + (k+1)^3 K^6 p^{K^4}\}) = O(n\{(p+k^2)p^{k(k+1)/2} + (k+1)^{6k+15} p^{(k+1)^{4k+8}}\}).$$

Note that $K = (k+1)^{k+2}$.

This completes a proof of Theorem 4.1.

Combining the algorithms in the previous section and this section, we have the following throrem.

Theorem 4.10 *Let G be a partial k-tree given by its tree-decomposition T with treewidth $\leq k$. Let (s_i, t_i), $1 \leq i \leq p$, be p pairs of terminals. Then one can determine in polynormial time whether G has p pairwise edge-disjoint paths P_i, $1 \leq i \leq p$, connecting s_i and t_i if there exist some nodes $X_1, X_2, \cdots, X_q$ of T such that*

- *each terminal pair s_i, t_i, $1 \leq i \leq p$, are contained in graph $G[X_j]$ for some j, $1 \leq j \leq q$, and*
- *each graph $G[X_j]$, $1 \leq j \leq q$, contains $O(\log n)$ terminals.*

The combined algorithm finds all active color vectors on each node X_j, $1 \leq j \leq q$, by the first algorothm, then computes all active counts on X_j from the color vectors, and finally finds all active counts on root by the second algorithm.

Middendorf and Pfeiffer announced in [MP93] that they were preparing a paper proving that the edge-disjoint paths problem for a partial k-tree G is solvable in polynomial time if the graph G^+ obtained from G by adding edges (s_i, t_i), $1 \leq i \leq p$, is a partial k-tree. However, they have not finished writing the paper [M96]. Theorem 4.1 implies the same claim as theirs, but provides a concrete algorithm of smaller complexity. Furthermore Theorem 4.10 cannot be derived from their claim.

5 Generalizations

Both of our algorithms can be extended as follows:

(a) An extended algorithm finds the maximum number of edge-disjoint paths connecting terminal pairs in the same amount of time.

(b) An extended algorithm solves the edge-disjoint paths problem for a multigraph whose underlying simple graph is a partial k-tree.

(c) Given a partial k-tree with nonnegative edge-lengths, an extended algorithm finds p edge-disjoint paths with the minimal total length in the same amount of time.

(d) Given a partial k-tree with directed edges, an extended algorithm finds p edge-disjoint directed paths in the same amount of time.

(e) Given a partial k-tree $G = (V, E)$ and subsets N_i, $1 \leq i \leq p$, of V (called *nets*), an algorithm finds p pairwise edge-disjoint trees B_i, $1 \leq i \leq p$, covering net N_i, that is, $N_i \subseteq V(B_i)$. The algorithm extended from the first one runs in polynomial time if $pq \log q = O(\log n)$ where $q = \max\{|N_1|, |N_2|, \cdots, |N_p|\}$. The algorithm extended from the second one runs in polynomial time if each N_i is contained in some node.

(f) Our algorithms can be implemented to NC parallel algorithms which find p edge-disjoint paths in $O(\log n)$ parallel time with a polynomial number of processors. The key idea is to apply the "tree contraction" to the bottom-up tree computation above in a sophisticated way like in [ZN95, ZNN].

Acknowledgment

We wish to thank Professors J. Lagergren and H. Suzuki for fruitful discussions.

References

[ACPS93] S. Arnborg, B. Courcelle, A. Proskurowski, and D. Seese. An algebraic theory of graph reduction. *Journal of the Association for Computing Machinery*, Vol. 40, No. 5, pp. 1134–1164, 1993.

[ALS91] S. Arnborg, J. Lagergren and D. Seese. Easy problems for tree-decomposable graphs. *Journal of Algorithms*, Vol. 12, No. 2, pp. 308–340, 1991.

[Bod90] H. L. Bodlaender. Polynomial algorithms for graph isomorphism and chromatic index on partial k-trees. *Journal of Algorithms*, Vol. 11, No. 4, pp. 631–643, 1990.

[Bod93] H. L. Bodlaender. 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.

[BPT92] R. B. Borie, R. G. Parker, and C. A. Tovey. Automatic generation of linear-time algorithms from predicate calculus descriptions of problems on recursively constructed graph families. *Algorithmica*, Vol. 7, pp. 555–581, 1992.

[Cou90] B. Courcelle. The monadic second-order logic of graphs I: Recognizable sets of finite graphs. *Information and Computation*, Vol. 85, pp. 12–75, 1990.

[Fra82] A. Frank. Disjoint paths in rectilinear grids. *Combinatorica*, Vol. 2, No. 4, pp. 361–371, 1982.

[KM86] M. Kaufmann and K. Mehlhorn. Routing through a generalized switchbox. *J. Algorithms*, Vol. 7, pp. 510–531, 1986.

[Lag96] J. Lagergren. Private communication, May 2, 1996.

[MNS85] K. Matsumoto, T. Nishizeki, and N. Saito. An efficient algorithm for finding multicommodity flows in planar networks. *SIAM J. Comput.*, Vol. 14, No. 2, pp. 289–302, 1985.

[MNS86] K. Matshmoto, T. Nishizeki, and N. Saito. Planar multicommodity flows, maximum matchings and negative cycles. *SIAM J. Comput.*, Vol. 15, No. 2, pp. 495–510, 1986.

[MP86] K. Mehlhorn and F.P. Preparata. Routing through a rectangle. *J. ACM*, Vol. 33, No. 1, pp. 60–85, 1986.

[M96] M. Middendorf. Private communication, May 1996.

[MP93] M. Middendorf and F. Pfeiffer. On the complexity of disjoint paths problem. *Combinatorica*, Vol. 13, No. 1, pp. 97–107, 1993.

[NSS85] T. Nishizeki, N. Saito, and K. Suzuki. A linear-time routing algorithm for convex grids. *IEEE Trans. Comput.-Aided Design*, Vol. 4, No. 1, pp. 68–76, 1985.

[RS86] N. Robertson and P.D. Seymour. Graph minors. II. Algorithmic aspects of tree-width. *Journal of Algorithms*, Vol. 7, pp. 309–322, 1986.

[RS95] N. Robertson and P.D. Seymour. Graph minors. XIII. The disjoint paths problem. *J. of Combin. Theory, Series B*, Vol. 63, No. 1, pp. 65–110, 1995.

[SAN90] H. Suzuki, T. Akama, and T. Nishizeki. Finding Steiner forests in planar graphs. In *Proc. of the First Ann. ACM-SIAM Sympo. on Discrete Algorithms*, pp. 444–453, 1990.

[Sch94] P. Scheffler. A practial linear time algorithm for disjoint paths in graphs with bounded tree-width. Technical Report, 396, Dept. of Mathematics, Technische Universität Berlin, 1994.

[Seb93] A. Sebő. Integer plane multiflows with a fixed number of demands. *J. of Combin. Theory, Series B*, Vol. 59, pp. 163–171, 1993.

[SIN90] H. Suzuki, A. Ishiguro, and T. Nishizeki. Edge-disjoint paths in a grid bounded by two nested rectangles. *Discrete Applied Mathematics*, Vol. 27, pp. 157–178, 1990.

[SNS89] H. Suzuki, T. Nishizeki, and N. Saito. Algorithms for multicommodity flows in planar graphs. *Algorithmica*, Vol. 4, pp. 471–501, 1989.

[TNS82] K. Takamizawa, T. Nishizeki, and N. Saito. Linear-time computability of combinatorial problems on series-parallel graphs. *Journal of ACM*, Vol. 29, No. 3, pp. 623–641, 1982.

[TP93] J.A. Telle, and A. Proskurowski. Practical algorithms on partial k-trees with an application to domination like problems. *Proc. of Workshop on Algorithms and Data Structures, WADS'93*, Montreal, Springer-Verlag Lect. Notes on Computer Science, 709, pp. 610–621, 1993.

[WW92] D. Wagner and K. Weihe. A linear-time algorithm for edge-disjoint paths in planar graphs. *Proc. of the First Europ. Symp. on Algorithms* (ESA'93), Springer-Verlag Lect. Notes in Computer Science, 726, pp. 384–395, 1992.

[ZN95] X. Zhou and T. Nishizeki. Optimal parallel algorithms for edge-coloring partial k-trees with bounded degrees. *IEICE Trans. on Fundamentals of Electronics, Communication and Computer Sciences*, Vol. E78-A, pp. 463–469, 1995.

[ZNN] X. Zhou, S. Nakano, and T. Nishizeki. Edge-coloring partial k-trees. *Journal of Algorithms, to appear.*

[ZSN96] X. Zhou, H. Suzuki, and T. Nishizeki. A linear algorithm for edge-coloring series-parallel multigraphs. *Journal of Algorithms*, Vol. 20, pp. 174–201, 1996.