アルゴリズム 62－4
（1998．5．20）

部分k木に対する辺素な道問題のNP-完全性

周 暁　　西関 隆夫
東北大学

数多くの組合せ問題は一般のグラフに対してNP-完全であるが，kが定数であるような部分k木に対しては多項式時間で，あるいはほとんどの場合線形時間で解ける．一方，いくつかの問題は部分k木に対してすらNP-完全である．しかしそのような問題は数少く，わずかに部分グラフ同形問題と帯域幅(bandwidth)問題等が部分k木に対してNP-完全であることが知られているにすぎない．しかもこれらの問題は$k=1$なる部分k木，すなわち通常の木あるいは林に対してすらNP-完全である．このように$k=1$なる部分k木に対しては多項式時間で解け，2以上で定数の部分k木に対してはNP-完全であるような問題は知られていなかった．本論文は辺素な道問題がそのような一例であることを示す．

The Edge-Disjoint Paths Problem is NP-Complete for Partial k-Trees

Xiao Zhou[1] and Takao Nishizeki[2]

Graduate School of Information Sciences
Tohoku University
Sendai 980-8579, JAPAN

Many combinatorial problems are NP-complete for general graphs, but are not NP-complete for partial k-trees (graphs of treewidth bounded by a constant k) and can be efficiently solved in polynomial time or mostly in linear time for partial k-trees. On the other hand, very few problems on unweighted graphs are known to be NP-complete for partial k-trees with bounded k. These include the subgraph isomorphism problem and the bandwidth problem. However, all these problems are NP-complete even for ordinary trees or forests, and there have been no known problems which are efficiently solvable for trees but NP-complete for partial k-trees. In this paper we present the first example of such problems, that is, we show that the edge-disjoint paths problem is NP-complete for partial k-trees with some bounded $k \geq 2$ although the problem is trivially solvable for trees.

[1] Email-address: zhou@ecei.tohoku.ac.jp
[2] Email-address: nishi@ecei.tohoku.ac.jp

1 Introduction

Many combinatorial problems are NP-complete for general graphs, and are unlikely to be solvable in polynomial time. However, many "natural" problems defined on unweighted graphs can be efficiently solved for partial k-trees (graphs of treewidth bounded by a constant k) in polynomial time or in linear time although not all problems are solvable for partial k-trees in polynomial time [ACPS93, ALS91, BPT92, Cou90, TP97, ZNN96]. On the other hand, a very few problems are known to be NP-complete for partial k-trees. These include the subgraph isomorphism problem and the bandwidth problem [DLP96, GN96, MT92, Sys83]. However, all these problems are NP-complete even for ordinary trees or forests [GJ79]. To the best of our knowledge there have been no known problems which are efficiently solvable for trees but NP-complete for partial k-trees with some bounded $k \geq 2$ and without any restriction on the connectivity or the maximum degree.

In this paper we present the first example of such problems, that is, we show the edge-disjoint paths problem is NP-complete for partial k-trees with some bounded $k \geq 2$ although the problem is trivially solvable for trees.

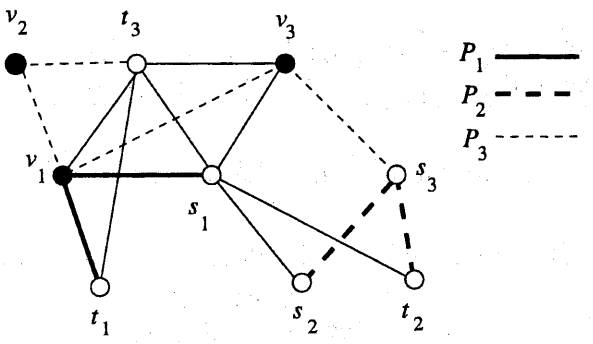

Figure 1: Three edge-disjoint paths P_1, P_2 and P_3 in a partial 3-tree.

The edge-disjoint paths problem asks whether there exist p pairwise edge-disjoint paths P_i, $1 \leq i \leq p$, connecting terminals s_i and t_i in a given graph G with p terminal pairs (s_i, t_i), $1 \leq i \leq p$, assigned to vertices of G. Figure 1 illustrates three edge-disjoint paths P_1, P_2 and P_3 in a partial 3-tree. The vertex-disjoint paths problem is similarly defined. These problems come up naturally when analyzing connectivity questions or generalizing (integral) network flow problems. Another reason for the growing interest is the variety of applications, e.g. in VLSI-design [SIN90, SNS89, WW95]. 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 similar to the line graph of G. Therefore, the edge-disjoint paths problem can also be solved in polynomial time for any graph by

the algorithm if $p = O(1)$. However, if p is not bounded, then both the edge-disjoint and vertex-disjoint paths problems are NP-complete even for planar graphs [MP93, Vyg95]. A natural question is whether the vertex-disjoint and edge-disjoint paths problems can be efficiently solved for another restricted class of graphs, say partial k-trees. Indeed Scheffler showed that the vertex-disjoint paths problem can be solved in linear time for partial k-trees even if p is not bounded [Sch94]. Zhou *et al.* showed that the edge-disjoint paths problem can be solved in polynomial time for partial k-trees if either $p = O(\log n)$ or the location of terminals satisfies some condition, where n denotes the number of vertices in a given partial k-tree [ZTN96]. The result implies that the edge-disjoint paths problem can be solved in polynomial time for a partial k-tree G if the graph obtained from G by adding p edges (s_i, t_i), $1 \leq i \leq p$, remains to be a partial k-tree. Furthermore, if a partial k-tree G has a bounded maximum degree, then clearly the edge-disjoint paths problem can be solved in linear time for G. However, it has not been known whether the edge-disjoint paths problem is NP-complete for partial k-trees if there is no restriction on the number of terminal pairs, the location of terminals, or the maximum degree. In this paper we show that the edge-disjoint paths problem is NP-complete in general for partial k-trees with some bounded k, say $k = 10$.

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. 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).

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 $|E| < kn$. 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 $X_i \cap X_l \subseteq X_j$.

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 [Bod96].

3 The Edge-Disjoint Paths Problem

Our main result is the following theorem.

Theorem 3.1 *The edge-disjoint paths problem is NP-complete for partial k-trees with some bounded k.*

In the remainder of this section we will give a proof of Theorem 3.1. Let $X = \{x_1, x_2, \cdots, x_n\}$ be a set of n Boolean variables. A *literal of* $x_i \in X$ is either a Boolean variable x_i or its negation $\overline{x}_i$. We denote by *3-CNF* the set of Boolean formulas in a conjunctive normal form over the n variables in X with at most 3 literals per clause. For a Boolean formula $f \in$ 3-CNF, the *3-SAT problem* which asks whether there is an assignment a of true-false values to the n variables such that $f(a)$ is true [Coo71]. If there is an assignment a such that $f(a)$ is true then we say that f is *satisfiable*. Clearly the edge-disjoint path problem is in NP. Therefore it suffices to show that the 3-SAT problem can be reduced in polynomial time to the edge-disjoint paths problem for partial k-trees with some bounded k.

For example, consider the following Boolean formula f with $n = 4$ variables and $m = 3$ clauses:

$$f = (\overline{x}_1 + x_2 + x_4)(x_1 + \overline{x}_3)(\overline{x}_2 + x_3 + \overline{x}_4).$$

As illustrated in Figure 2, we will construct a graph G_f which contains edge-disjoint paths connecting terminal pairs if and only if f is satisfiable. The formula f above is satisfiable for a true-false assignment such that $x_1 = 1$, $x_2 = 0$, $x_3 = 0$ and $x_4 = 1$, while G_f in Figure 2 has edge-disjoint paths connecting terminal pairs drawn in thick, dotted or weavy lines. Roughly speaking, G_f has n rows and $m + 2$ columns. The ith row corresponds to variable x_i for each i, $1 \leq i \leq n$. The leftmost column, i.e. 0th column, contains n quadrangles $s_i x_i s_{i0} \overline{x}_i$, $1 \leq i \leq n$, and the rightmost column, i.e. $(m+1)$th column, contains n quadrangles $t_i x'_i t_{im} \overline{x}'_i$, $1 \leq i \leq n$. Each of the other m columns corresponds to a clause and contains n "gadgets." Any two consecutive columns, jth and $(j+1)$th, are connected through exactly two vertices v_j and $\overline{v}_j$ for each j, $0 \leq j \leq m$. These vertices v_j and $\overline{v}_j$ are called *connection vertices*. The graph G_f constructed in this way is a partial k-tree for some bounded k as we will observe later.

We now describe how to construct G_f in detail. Consider a Boolean formula $f \in$ 3-CNF with m clauses $C_1, C_2, \cdots, C_m$ and n variables in X. We write $x_i \in C_j$ and $\overline{x}_i \in C_j$ if clause C_j contains literals x_i and $\overline{x}_i$, respectively. One may assume that, for any variable x_i, $1 \leq i \leq n$, and clause C_j, $1 \leq j \leq m$, exactly one of the following three cases occurs:

(1) $x_i, \overline{x}_i \notin C_j$,

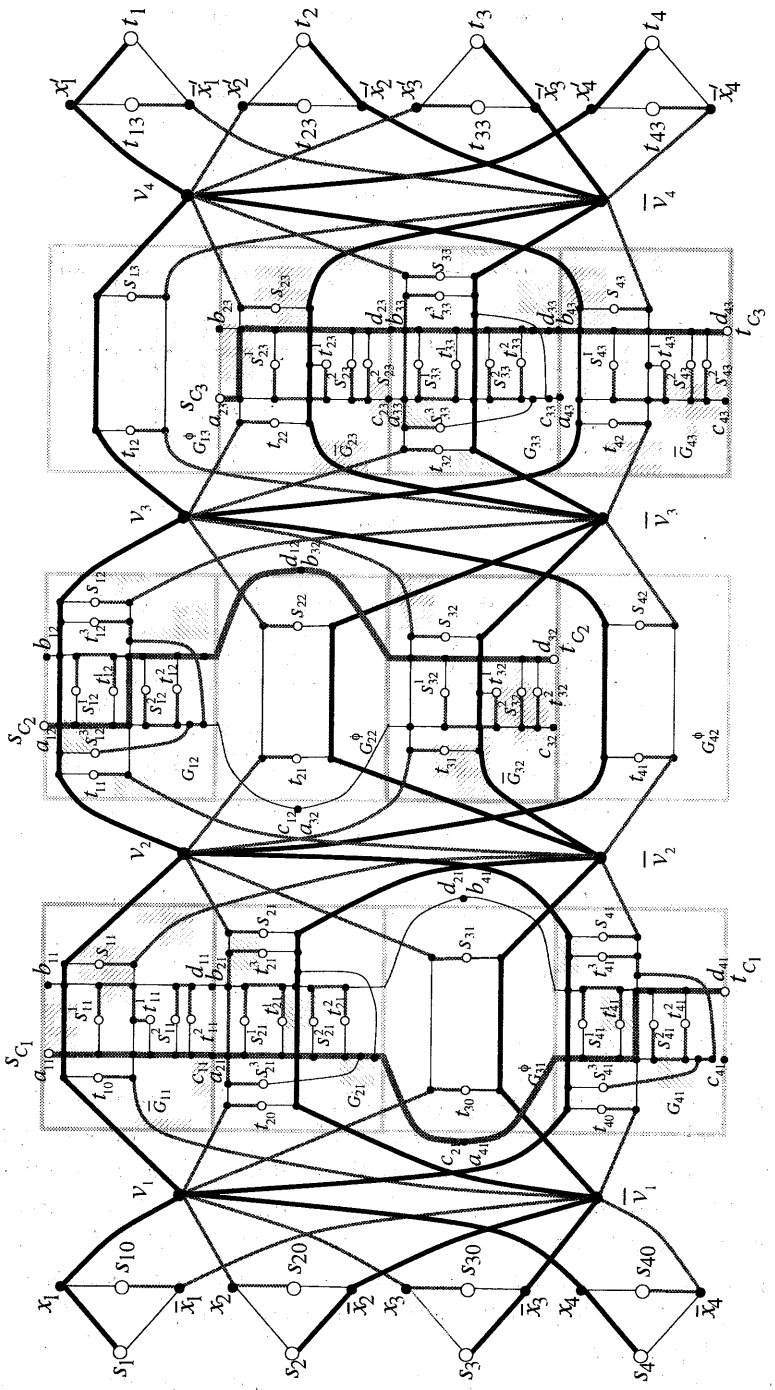

Figure 2: Graph G_f and edge-disjoint paths.

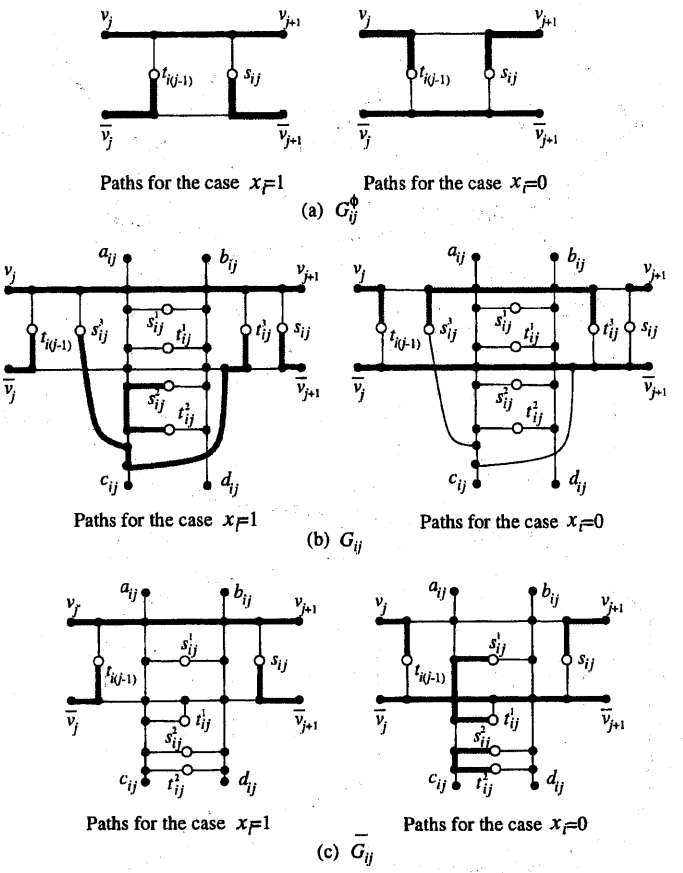

Figure 3: Three gadgets G_{ij}^{ϕ}, G_{ij}, $\overline{G}_{ij}$ and paths in them.

(2) $x_i \in C_j$ and $\overline{x}_i \notin C_j$, and

(3) $x_i \notin C_j$ and $\overline{x}_i \in C_j$.

For the three cases above we construct graphs G_{ij}^{ϕ}, G_{ij} and $\overline{G}_{ij}$, respectively, as three types of gadgets to build the whole graph G_f. That is, for each pair of variable x_i and clause C_j, we construct a graph G_{ij}^{ϕ} if $x_i, \overline{x}_i \notin C_j$, a graph G_{ij} if $x_i \in C_j$, and a graph $\overline{G}_{ij}$ if $\overline{x}_i \in C_j$.

The graph G_{ij}^{ϕ}, illustrated in Figure 3(a), contains exactly four outer vertices: two vertices on each of the two sides; vertices v_j and $\overline{v}_j$ on the left side, and the other two vertices v_{j+1} and $\overline{v}_{j+1}$ on the right. The graph G_{ij}^{ϕ} contains 6 inner vertices including two terminals $t_{i(j-1)}$ and s_{ij}. Thus G_{ij}^{ϕ} contains 10 vertices in total, and these vertices are connected as in Figure 3(a).

As illustrated in Figures 3(b) and (c), both G_{ij} and $\overline{G}_{ij}$ contain exactly eight outer vertices: two vertices on each of the four sides; vertices v_j and $\overline{v}_j$ on the left side, vertices

v_{j+1} and $\overline{v}_{j+1}$ on the right, vertices a_{ij} and b_{ij} on the top, and vertices c_{ij} and d_{ij} on the bottom. The graph G_{ij} contains 30 inner vertices including 8 terminals: two terminals $t_{i(j-1)}$ and s_{ij} and three pairs (s^1_{ij}, t^1_{ij}), (s^2_{ij}, t^2_{ij}) and (s^3_{ij}, t^3_{ij}). Thus G_{ij} contains 38 vertices in total, and these vertices are connected as in Figure 3(b). The graph $\overline{G}_{ij}$ contains 22 inner vertices including 6 terminals: two terminals $t_{i(j-1)}$ and s_{ij} and two pairs (s^1_{ij}, t^1_{ij}) and (s^2_{ij}, t^2_{ij}). Thus $\overline{G}_{ij}$ contains 30 vertices in total, and these vertices are connected as in Figure 3(c).

Graphs G^{ϕ}_{ij}, G_{ij} or $\overline{G}_{ij}$ in the ith row of the jth column contains terminals $t_{i(j-1)}$ and s_{ij}; $t_{i(j-1)}$ is paired with a terminal $s_{i(j-1)}$ in the ith row of the $(j-1)$th column, and s_{ij} is paired with a terminal t_{ij} in the ith row of the $(j+1)$th column in G_f. Only the outer vertices in G^{ϕ}_{ij}, G_{ij} and $\overline{G}_{ij}$ are connected to vertices in other gadgets or connection vertices.

We now construct a graph G_j for each clause C_j, $1 \leq j \leq m$. The graph G_j has n rows ordered from the top to the bottom by $1, 2, \cdots, n$; the ith row corresponds to the variable x_i. For each i, $1 \leq i \leq n$, we put to the ith row gadget G^{ϕ}_{ij} if $x_i, \overline{x}_i \notin C_j$, G_{ij} if $x_i \in C_j$, and $\overline{G}_{ij}$ if $\overline{x}_i \in C_j$. Identify all the n outer vertices v_j on the left side of gadgets with the connection vertex v_j; identify all the n outer vertices $\overline{v}_j$ on the left side of gadgets with the connection vertex $\overline{v}_j$; identify all the n outer vertices v_{j+1} on the right side of gadgets with the connection vertex v_{j+1}; and identify all the n outer vertices $\overline{v}_{j+1}$ on the right side of gadgets with the connection vertex $\overline{v}_{j+1}$. There are three cases on the number of literals in C_j; three literals in C_j, two in C_j and one in C_j. For the sake of convenience, we consider only the case when C_j has exactly three literals. The construction for the other two cases is similar. We thus assume that $C_j = (l_{i_1 j} + l_{i_2 j} + l_{i_3 j})$ and x_{i_1}, x_{i_2} and x_{i_3} are the Boolean variables corresponding to literals $l_{i_1 j}$, $l_{i_2 j}$ and $l_{i_3 j}$, respectively. Therefore $l_{i_1 j} = x_{i_1}$ or $\overline{x}_{i_1 j}$, $l_{i_2 j} = x_{i_2}$ or $\overline{x}_{i_2 j}$, and $l_{i_3 j} = x_{i_3}$ or $\overline{x}_{i_3 j}$. Furthermore we may assume without loss of generality that $i_1 < i_2 < i_3$. We identify vertex $c_{i_1 j}$ with $a_{i_2 j}$, $d_{i_1 j}$ with $b_{i_2 j}$, $c_{i_2 j}$ with $a_{i_3 j}$ and $d_{i_2 j}$ with $b_{i_3 j}$. Furthermore we add to G_j a new terminal pair (s_{C_j}, t_{C_j}) by setting $s_{C_j} = a_{i_1 j}$ and $t_{C_j} = d_{i_3 j}$.

We now construct the whole graph G_f by connecting $m+2$ graphs in cascade. The graph G_f has $m+2$ columns ordered from the left to the right by $0, 1, 2, \cdots, m, m+1$. The graph G_j is put to the jth column, $1 \leq j \leq m$. Any two consecutive columns G_j and G_{j+1}, $0 \leq j \leq m+1$, are connected through exactly two connection vertices v_{j+1} and $\overline{v}_{j+1}$. For each variable x_i, $1 \leq i \leq n$, we introduce a terminal pair (s_i, t_i) and two quadrangles $s_i x_i s_{i0} \overline{x}_i$ and $t_i x'_i t_{im} \overline{x}'_i$; these two quadrangles are put to the ith row of the 0th and $(m+1)$th columns, respectively, where $s_i, t_i, x_i, x'_i, \overline{x}_i, \overline{x}'_i, s_{i0}$ and t_{im} are vertices in graph G_f. Finally, add edges (x_i, v_1), $(\overline{x}_i, \overline{v}_1)$, (x'_i, v_{m+1}) and $(\overline{x}'_i, \overline{v}_{m+1})$, $1 \leq i \leq n$, to the graph G_f. Thus we have completed the construction of G_f. Let N be the number of vertices in G_f, and let p be the number of terminal pairs. Then $N = O(mn)$ and $p = O(N)$.

We have the following lemma.

Lemma 3.2 *G_f has edge-disjoint paths connecting terminal pairs if and only if f is satisfiable.*

By Lemma 3.2 it suffices to verify that G_f is a partial k-tree for some bounded k. For each j, $1 \leq j \leq m$, the deletion of four connection vertices v_j, $\overline{v}_j$, v_{j+1} and $\overline{v}_{j+1}$ from G_j leaves $n-2$ connected components of bounded size; one corresponds to the i_1th, i_2th and i_3th gadgets, and each of the other $n-3$ components corresponds to G^{ϕ}_{ij}. Thus one can observe that the resulting graph G_f is a partial k-tree for some bounded k, say $k = 10$. Actually graph G_f has a bounded pathwidth. Thus this completes the proof of Theorem 3.1.

References

[ACPS93] S. Arnborg, B. Courcelle, A. Proskurowski, and D. Seese. An algebraic theory of graph reduction. *Journal of the Association for Computing Machinery*, 40(5):1134–1164, 1993.

[ALS91] S. Arnborg, J. Lagergren, and D. Seese. Easy problems for tree-decomposable graphs. *Journal of Algorithms*, 12(2):308–340, 1991.

[Bod96] H. L. Bodlaender. A linear time algorithm for finding tree-decompositions of small treewidth. *SIAM Journal on Computing*, 25:1305–1317, 1996.

[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*, 7:555–581, 1992.

[Coo71] S.A. Cook. The complexity of theorem-proving procedures. In *Proc. of 3th Symp. Theory of Comp. Association for Computing Machinery*, pages 151–158, 1971.

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

[DLP96] A. Dessmark, A. Lingas, and A. Proskurowski. Faster algorithms for subgraph isomorphism of k-connected partial k-trees. In *Proc. of the Fourth Europian Symposium on Algorithms, Lect. Notes in Computer Science, Springer-Verlag*, volume 1136, pages 501–513, 1996.

[GJ79] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman & Co., New York, 1979.

[GN96] A. Gupta and N. Nishimura. The complexity of subgraph isomorphism for classes of partial k-trees. *Theoretical Computer Science*, 164:287–297, 1996.

[MP93] M. Middendorf and F. Pfeiffer. On the complexity of disjoint paths problem. *Combinatorica*, 13(1):97–107, 1993.

[MT92] J. Matoušek and R. Thomas. On the complexity of finding iso- and other morphisms for partial k-trees. *Discrete Mathematics*, 108:343–364, 1992.

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

[RS95] N. Robertson and P.D. Seymour. Graph minors XIII. The disjoint paths problem. *J. of Combin. Theory, Series B*, 63(1):65–110, 1995.

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

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

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

[Sys83] M. Sysło. NP-complete problems on some tree-structured graphs: a review. In *Proc. WG'83 International Workshop on Graph Theoretic Concepts in Computer Science*, pages 478–489, Univ. Verlag Rudolf Trauner, Linz, West Germany, 1983.

[TP97] J. A. Telle and A. Proskurowski. Algorithms for vertex partitioning problems on partial k-trees. *SIAM J. Discrete Math.*, 10:529–550, 1997.

[Vyg95] J. Vygen. NP-completeness of some edge-disjoint paths problems. *Discrete Appl. Math.*, 61:83–90, 1995.

[WW95] D. Wagner and K. Weihe. A linear-time algorithm for edge-disjoint paths in planar graphs. *Combinatorica.*, 15:135–150, 1995.

[ZNN96] X. Zhou, S. Nakano, and T. Nishizeki. Edge-coloring partial k-trees. *Journal of Algorithms*, 21:598–617, 1996.

[ZTN96] X. Zhou, S. Tamura, and T. Nishizeki. Finding edge-disjoint paths in partial k-trees. In *Proc. of the Seventh International Symposium on Algorithms and Computation, Lect. Notes in Computer Science, Springer*, volume 1178, pages 203–212, 1996.