A Polynomial-Time Algorithm for Finding Total Colorings of Partial k-Trees

Shuji Isobe, Xiao Zhou, and Takao Nishizeki

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

Abstract. A total coloring of a graph G is a coloring of all elements of G, i.e. vertices and edges, in such a way that no two adjacent or incident elements receive the same color. Many combinatorial problems can be efficiently solved for partial k-trees (graphs of treewidth bounded by a constant k). However, no polynomial-time algorithm has been known for the problem of finding a total coloring of a given partial k-tree with the minimum number of colors. This paper gives such a first polynomial-time algorithm.

1 Introduction

A *total coloring* of a graph G is a coloring of all elements of G, i.e. vertices and edges, so that no two adjacent or incident elements receive the same color. Figure 1 depicts a total coloring of a graph with four colors. This paper deals with the *total coloring problem* which asks to find a total coloring of a given graph G with the minimum number of colors. The minimum number of colors is called the *total chromatic number* $\chi_t(G)$ of G. The total coloring problem arises in many applications, including various scheduling and partitioning problems [Yap96]. The problem is NP-complete [Sán89], and hence it is very unlikely that there exists an algorithm to find a total coloring of a given graph G with $\chi_t(G)$ colors in polynomial time.

It is known that many combinatorial problems can be solved very efficiently for partial k-trees or series-parallel graphs [ACPS93, AL91, BPT92, Cou90, TNS82, ZNN96, ZSN96, ZTN96]. Partial k-trees are the same as graphs of treewidth at most k. In the paper we assume that $k = O(1)$. 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

J. Hromkovič, O. Sýkora (Eds.): WG'98, LNCS 1517, pp. 100–113, 1998.

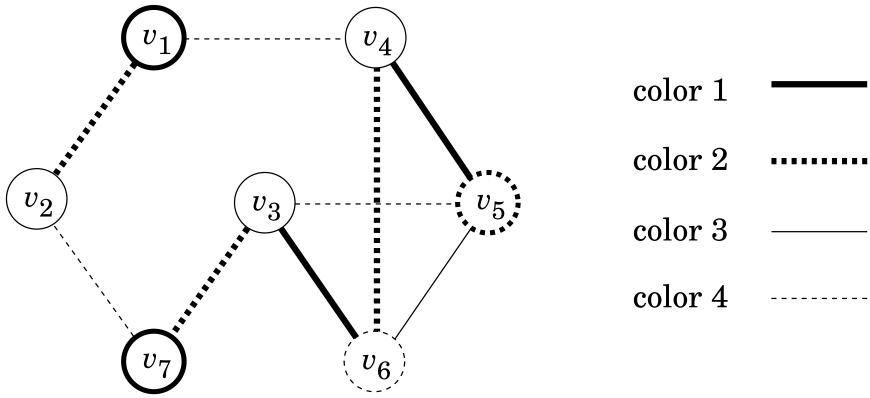

Fig. 1. A total coloring.

at most $k+1$ vertices. Many problems can be solved efficiently for partial k-trees by a dynamic programming (DP) algorithm based on the tree-decomposition [ACPS93,AL91,BPT92,Cou90]. In particular, it is rather straightforward to design polynomial-time algorithms for vertex-type problems on partial k-trees. For example, the vertex-coloring problem, the maximum independent vertex-set problem, the minimum dominating vertex-set problem, and the vertex-disjoint paths problem can be solved all in linear time for partial k-trees [BPT92,Sch94,TP97]. However, this is not the case for edge-type problems such as the edge-coloring problem and the edge-disjoint paths problem. It needs sophisticated treatment tailored for individual edge-type problems to design efficient algorithms. For example, the edge-coloring problem can be solved in linear time for partial k-trees and series-parallel multigraphs, but very sophisticated algorithms are needed [ZSN97,ZSN96]. On the other hand, the edge-disjoint paths problem is NP-complete even for partial k-trees [ZN98], although the problem can be solved in polynomial time for partial k-trees under a certain restriction on the number of terminal pairs or the location of terminal pairs [ZTN96]. The difficulty of edge-type problems stems from the following facts: the number of vertices in a basis graph (a node of a tree-decomposition 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 or a polynomial in the number of vertices in a partial k-tree.

Clearly the mixed type problem like the total coloring problem is more difficult in general than the vertex- and edge-type problems. Both the vertex-coloring problem and the edge-coloring problem can be solved in linear time for partial k-trees. Therefore a natural question is whether the total coloring problem can be efficiently solved for partial k-trees or not.

In this paper we give a polynomial-time algorithm to solve the total coloring problem for partial k-trees G. Our idea is to bound the size of a DP table by $O(n^{2^{2k+3}})$, applying and extending techniques developed for the edge-coloring problem [Bod90,ZN95,ZNN96]. The paper is organized as follows. In section 2 we present some preliminary definitions. In section 3 we give a polynomial-time algorithm for the total coloring problem on partial k-trees. Finally we conclude our result in section 4 with some comments on a parallel algorithm.

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). We denote by $\Delta(G)$ the *maximum degree* of G.

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

(K1) A complete graphs with k vertices is a k-tree.

(K2) If $G = (V, E)$ is a k-tree and k vertices $v_1, v_2, \ldots, v_k$ induce a complete subgraph of G, then $G' = (V \cup \{w\}, E \cup \{(v_i, w) : 1 \le i \le k\})$ is a k-tree where w is a new vertex not contained in G.

(K3) All k-trees can be formed with rules (K1) and (K2).

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 $|E| < kn$. Figure 2 illustrates a process of generating a 3-tree. The graph in Figure 1 is indeed a subgraph of a 3-tree in Figure 2, and hence is a partial 3-tree.

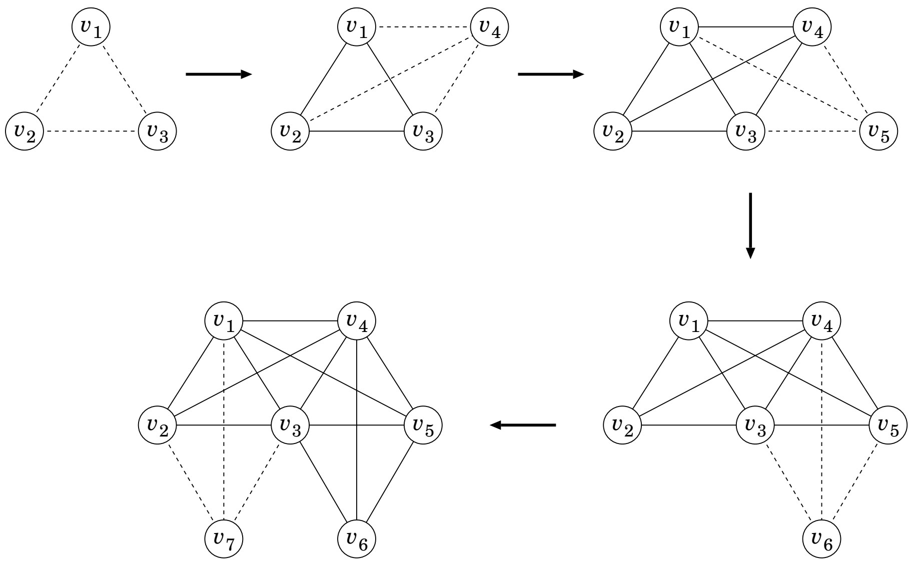

Fig. 2. 3-trees.

A *tree-decomposition* of G is a tree $T = (V_T, E_T)$ where V_T is a family of subset of V with the following three properties (a), (b) and (c):

(a) $\bigcup_{X \in V_T} X = V$;
(b) for each $e = (u, v) \in E$, there is a node $X \in V_T$ such that $u, v \in X$; and
(c) 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 3(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 $\max\{|X| - 1 : X \in V_T\}$. 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 $\leq k$.

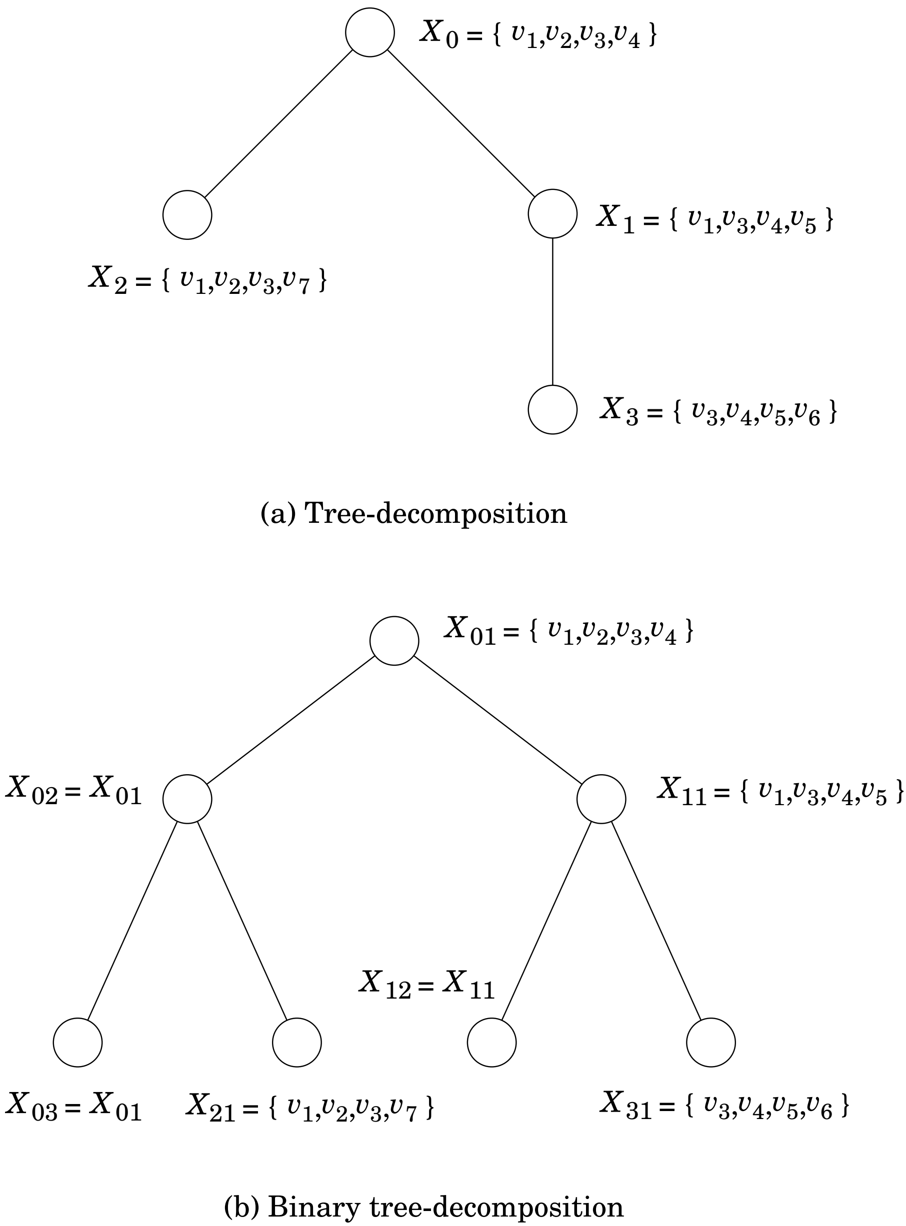

Fig. 3. Tree-decompositions of the partial 3-tree in Figure 1.

Bodlaender has given a linear time sequential algorithm to find a tree-decomposition of a given graph with width $\leq k$ for bounded k [Bod96]. We consider a tree-decomposition of a partial k-tree G with width $\leq k$. We transform it to a binary tree T as follows: regard the tree-decomposition as a rooted tree by choosing an arbitrary node as the root X_0 and replace every internal node X_i with r children $X_{j_1}, X_{j_2}, \ldots, X_{j_r}$ by $r+1$ new nodes $X_{i_1}, X_{i_2}, \ldots, X_{i_{r+1}}$ which are copies of X_i, where X_{i_1} has the same father as X_i, X_{i_q} is the father of $X_{i_{q+1}}$ and the q-th 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 linear time and doesn't change width [Bod90]. T is a tree-decomposition of G with the following properties:

(a) The number of nodes in T is $O(n)$.
(b) Each internal node X_i has exactly two children, say X_l and X_r, and either $X_i = X_l$ or $X_i = X_r$.
(c) For each edge $e = (u, v) \in E$, there is at least one leaf X_i with $u, v \in X_i$.

Such a tree T is called a *binary tree-decomposition*. Figure 3(b) illustrates a binary transformation of the tree-decomposition in Figure 3(a). Let T be a binary tree-decomposition with width $\leq k$ of a partial k-tree G. For each edge $e = (u, v) \in E(G)$, we choose an arbitrary leaf X_i with $u, v \in X_i$ and denote it by $\text{rep}(e)$. We define a vertex set $V_i \subseteq V(G)$ and an edge set $E_i \subseteq E(G)$ for each node X_i of T as follows: if X_i is a leaf, then let $V_i = X_i$ and $E_i = \{e \in E(G) : \text{rep}(e) = X_i\}$; if X_i is an internal node with children X_l and X_r, then let $V_i = V_l \cup V_r$ and $E_i = E_l \cup E_r$. Note that $V_l \cap V_r \subseteq X_i$ and $E_l \cap E_r = \emptyset$. We denote by G_i the graph with vertex set V_i and edge set E_i. Then graphs G_l and G_r share common vertices only in X_i because of the property (c) of a tree-decomposition.

3 A Polynomial-Time Algorithm

In this section we prove the following theorem.

Theorem 1. *Let $G = (V, E)$ be a partial k-tree of n vertices given by its tree-decomposition with width $\leq k$, let C be a set of colors,* *and let* $\alpha = |C|$. *Then it can be determined in time*

$$O(n(\alpha+1)^{2^{2k+3}})$$

whether G *has a total coloring:* $V \cup E \to C$.

One can easily know that the following lemma holds.

Lemma 1. *Every partial* k*-tree* G *satisfies*

$$\Delta(G) + 1 \leq \chi_t(G) \leq \Delta(G) + k + 2.$$

Proof: Clearly $\Delta(G) + 1 \leq \chi_t(G)$ for any graph.

Since a partial k-tree G is a simple graph, G has an edge-coloring with $\Delta(G)$ or $\Delta(G)+1$ colors by the classical Vizing theorem [FW77]. On the other hand, one can easily observe that a partial k-tree G has a vertex-coloring with at most $k+1$ colors. These two colorings immediately yield a total coloring of G with at most $\Delta(G) + k + 2$ colors. Thus $\chi_t(G) \leq \Delta(G) + k + 2$. □

Thus one can compute $\chi_t(G)$ by applying the algorithm in Theorem 1 to G for $k+2$ distinct values α, $\Delta(G) + 1 \leq |C| = \alpha \leq \Delta(G) + k + 2$. Furthermore, since $\alpha \leq n + k + 2$ and $k = O(1)$, the term $(\alpha+1)^{2^{2k+3}}$ is bounded by a polynomial in n. Thus we have the following corollary.

Corollary 1. *The total coloring problem can be solved in polynomial time for partial* k*-trees.*

In the remainder of this section we will give a proof of Theorem 1. Although we give an algorithm to decide whether $G = (V, E)$ has a total coloring $f : V \cup E \to C$ for a given set C of colors, it can be easily modified so that it actually finds a total coloring f with colors in C. Our idea is to reduce the size of a DP table to $O((\alpha+1)^{2^{2k+3}})$ by considering "pair-counts" and "quad-counts" defined below. A similar technique has been used for the ordinary edge-coloring and the f-coloring [Bod90,ZN95,ZNN96].

Let $C = \{1, 2, \ldots, \alpha\}$ be the set of colors. Let $G = (V, E)$ be a partial k-tree, and let X_i be a node of a binary tree-decomposition T of G. We say that a total coloring of graph G_i is *extensible* if it can be extended to a total coloring of $G = G_{01}$ without changing the coloring of G_i, where X_{01} is the root of T. Figure 4 illustrates total colorings of G_{02} and G_{11} for the partial 3-tree of G

in Figure 1 and its binary tree-decomposition T in Figure 3(b), where $X_{02} = \{v_1, v_2, v_3, v_4\}$ is the left child of the root X_{01} and $X_{11} = \{v_1, v_3, v_4, v_5\}$ is the right child. Both of the colorings are extensible because either can be extended to the total coloring of G in Figure 1.

For a total coloring f of G_i and a color $c \in C$, we define subsets $Y(X_i; f, c)$ and $Z(X_i; f, c)$ of X_i as follows:

$$Y(X_i; f, c) = \{v \in X_i : f(v) = c\}, \quad \text{and}$$
$$Z(X_i; f, c) = \{v \in X_i : G_i \text{ has an edge } (v, w) \text{ with } f((v, w)) = c\}.$$

Clearly,

$$Y(X_i; f, c) \cap Z(X_i; f, c) = \emptyset. \tag{1}$$

We call a mapping $\gamma : 2^{X_i} \times 2^{X_i} \rightarrow \{0, 1, 2, \ldots, \alpha\}$ a *pair-count* on a node X_i. A pair-count γ on X_i is defined to be *active* if G_i has a total coloring f such that

$$\gamma(A, B) = |\{c \in C : A = Y(X_i; f, c), B = Z(X_i; f, c)\}|$$

for each pair of $A, B \subseteq X_i$. Such a pair-count γ is called the *pair-count of the total coloring f*. Clearly, for any active pair-count γ,

$$\sum_{A,B \subseteq X_i} \gamma(A, B) = |C| = \alpha.$$

Furthermore, Eq. (1) implies that if $\gamma(A, B) \geq 1$ then $A \cap B = \emptyset$.

Let f be the total coloring f of $G = G_{01}$ for the root $X_{01} = \{v_1, v_2, v_3, v_4\}$ depicted in Figure 4(a), then

$$\begin{array}{ll} Y(X_{01}; f, 1) = \{v_1\}, & Z(X_{01}; f, 1) = \{v_3, v_4\}, \\ Y(X_{01}; f, 2) = \emptyset, & Z(X_{01}; f, 2) = \{v_1, v_2, v_3, v_4\}, \\ Y(X_{01}; f, 3) = \{v_2, v_3, v_4\}, & Z(X_{01}; f, 3) = \emptyset, \\ Y(X_{01}; f, 4) = \emptyset, & Z(X_{01}; f, 4) = \{v_1, v_2, v_3, v_4\}. \end{array}$$

Therefore f has the pair-count $\gamma_{X_{01}}$ such that

$$\begin{array}{l} \gamma_{X_{01}}(\{v_1\}, \{v_3, v_4\}) = 1, \\ \gamma_{X_{01}}(\emptyset, \{v_1, v_2, v_3, v_4\}) = 2, \\ \gamma_{X_{01}}(\{v_2, v_3, v_4\}, \emptyset) = 1, \end{array}$$

and $\gamma_{X_{01}}(A, B) = 0$ for any other pair of $A, B \subseteq X_{01}$. On the other hand, the total coloring of G_{02} for the left child $X_{02} = \{v_1, v_2, v_3, v_4\}$ of X_{01} depicted in Figure 4(b) has the pair-count $\gamma_{X_{02}}$ such that

$$\begin{aligned}
&\gamma_{X_{02}}(\{v_1\}, \emptyset) = 1, \\
&\gamma_{X_{02}}(\emptyset, \{v_1, v_2, v_3\}) = 1, \\
&\gamma_{X_{02}}(\{v_2, v_3, v_4\}, \emptyset) = 1, \\
&\gamma_{X_{02}}(\emptyset, \{v_1, v_2, v_4\}) = 1,
\end{aligned}$$

and $\gamma_{X_{02}}(A, B) = 0$ for any other pair of $A, B \subseteq X_{02}$. The total coloring of G_{11} for the right child $X_{11} = \{v_1, v_3, v_4, v_5\}$ of X_{01} depicted in Figure 4(c) has the pair-count $\gamma_{X_{11}}$ such that

$$\begin{aligned}
&\gamma_{X_{11}}(\{v_1\}, \{v_3, v_4, v_5\}) = 1, \\
&\gamma_{X_{11}}(\{v_5\}, \{v_4\}) = 1, \\
&\gamma_{X_{11}}(\{v_3, v_4\}, \{v_5\}) = 1, \\
&\gamma_{X_{11}}(\emptyset, \{v_3, v_5\}) = 1,
\end{aligned}$$

and $\gamma_{X_{11}}(A, B) = 0$ for any other pair of $A, B \subseteq X_{11}$.

We now have the following lemma.

Lemma 2. *Let two total colorings f and g of G_i for a node X_i of T have the same pair-count on X_i. Then f is extensible if and only if g is extensible.*

Thus an active pair-count on X_i characterizes an equivalence class of extensible total colorings of G_i. Since $|X_i| \leq k + 1$, there are at most $(\alpha + 1)^{2^{2(k+1)}}$ active pair-counts on X_i. The main step of our algorithm is to compute a table of all active pair-counts on each node of T from leaves to the root X_{01} of T by means of dynamic programming. From the table on the root X_{01} one can easily determine whether G has a total coloring using colors in C, as follows.

Lemma 3. *A partial k-tree G has a total coloring using colors in C if and only if the table on root X_{01} has at least one active pair-count.*

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

(1) enumerate all mappings:

$$V(G_i) \cup E(G_i) \to \{1, 2, \ldots, \min\{\alpha, (k + 1)(k + 2)/2\}\};$$

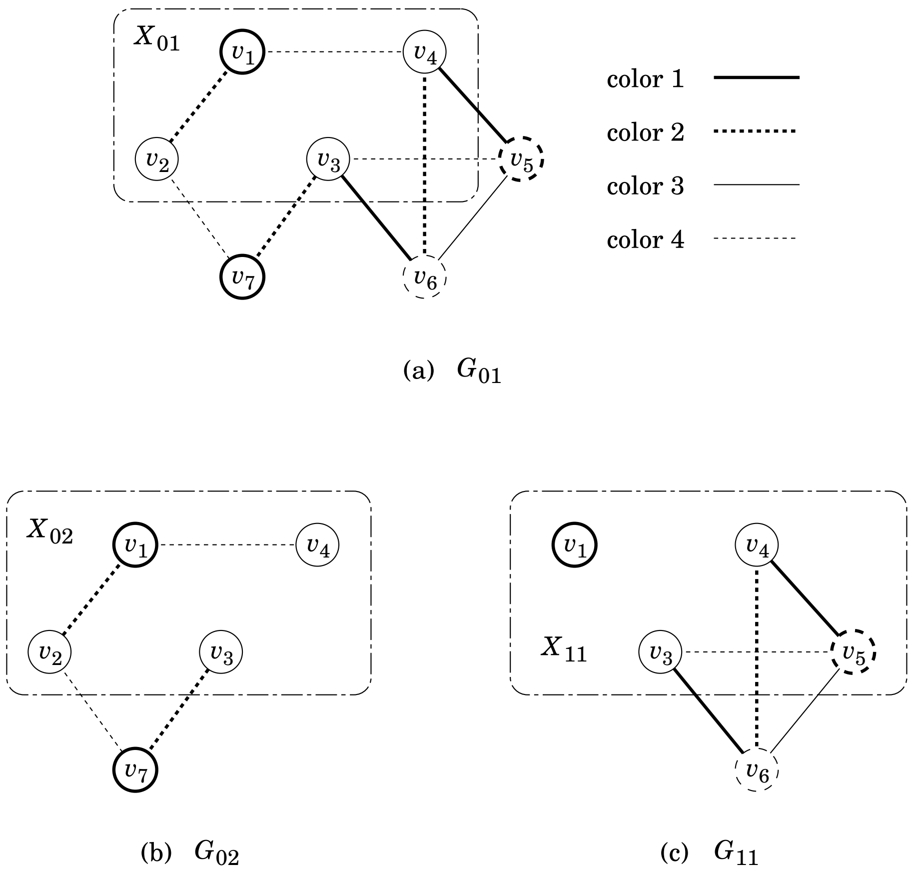

Fig. 4. Total colorings of (a) $G = G_{01}$, (b) G_{02}, and (c) G_{11}.

(2) find all total colorings of G_i from the mappings above; and
(3) compute all active pair-counts on X_i from the total colorings of G_i.

Since $|V_i| \leq k+1$ and $|E_i| \leq k(k+1)/2$ for leaf X_i, the number of distinct mappings $f : V(G_i) \cup E(G_i) \rightarrow \{1, 2, \ldots, \min\{\alpha, (k+1)(k+2)/2\}\}$ is at most $O(k^{k^2}) = O(1)$. For each mapping f of G_i, one can determine whether f is a total coloring of G_i in time $O(k^2) = O(1)$. For each total coloring f of G_i, one can compute the pair-count of f in time $O(k^2) = O(1)$. Therefore, steps (1), (2) and (3) can be done for a leaf in time $O(1)$. Since T has $O(n)$ leaves, the tables on all leaves can be computed in time $O(n)$.

We next compute all active pair-counts on each internal noode X_i of T from all active pair-counts of its children X_l and X_r. We may assume that $X_i = X_l$. Note that $V(G_i) = V(G_l) \cup V(G_r)$, $E(G_i) = E(G_l) \cup E(G_r)$ and $E(G_l) \cap E(G_r) = \emptyset$. We call a mapping $\rho : 2^{X_l} \times 2^{X_l} \times 2^{X_r} \times 2^{X_r} \rightarrow \{0, 1, 2, \ldots, \alpha\}$ a *quad-count* on X_i. We define a quad-count ρ to be *active* if G_i has a total coloring f such that, for each quadruplet (A_l, B_l, A_r, B_r) with $A_l, B_l \subseteq X_l$ and $A_r, B_r \subseteq X_r$

$$\rho(A_l, B_l; A_r, B_r) = |\{c \in C : A_l = Y(X_l; f_l, c), B_l = Z(X_l; f_l, c), \\ A_r = Y(X_r; f_r, c), B_r = Z(X_r; f_r, c)\}|$$

where $f_l = f|G_l$ is the restriction of f to $V(G_l) \cup E(G_l)$ and $f_r = f|G_r$ is the restriction of f to $V(G_r) \cup E(G_r)$. Such a quad-count is called the *quad-count of the total coloring f of G_i*. Then we have the following lemma.

Lemma 4. *Let an internal node X_i of T have two children X_l and X_r, and let $X_i = X_l$. Then a quad-count ρ on X_i is active if and only if ρ satisfies the following conditions* (a) *and* (b):

(a) *if $\rho(A_l, B_l; A_r, B_r) \geq 1$ then $A_l \cap X_r = A_r \cap X_l$ and $B_l \cap B_r = \emptyset$; and*
(b) *there are two active pair-counts γ_l on X_l and γ_r on X_r such that*
  (i) *for each pair $A_l, B_l \subseteq X_l$,*

$$\gamma_l(A_l, B_l) = \sum_{A,B \subseteq X_r} \rho(A_l, B_l; A, B);$$

(ii) *for each pair* $A_r, B_r \subseteq X_r$,

$$\gamma_r(A_r, B_r) = \sum_{A,B \subseteq X_l} \rho(A, B; A_r, B_r).$$

Using Lemma 4, we compute all active quad-counts ρ on X_i from all pairs of active pair-counts γ_l on X_l and γ_r on X_r. Since there are at most $(\alpha+1)^{2^{2k+3}}$ pairs of active pair-counts on X_l and X_r, there are at most $(\alpha+1)^{2^{2k+3}}$ distinct active quad-counts ρ. For each ρ of them, we determine whether ρ satisfies Conditions (a) and (b) in Lemma 4. For each ρ, one can determine in time $O(1)$ whether ρ satisfies Condition (a), because there are at most $2^{4(k+1)} = O(1)$ distinct quadruplets (A_l, B_l, A_r, B_r). Furthermore, checking Condition (b) for all possible ρ's can be done in time $O((\alpha+1)^{2^{2k+3}})$ since there are at most $(\alpha+1)^{2^{2k+3}}$ pairs of γ_l and γ_r. Thus we have shown that all active quad-counts ρ on X_i can be computed in time $O((\alpha+1)^{2^{2k+3}})$.

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

Lemma 5. *Let an internal node X_i of T have two children X_l and X_r with $X_i = X_l$. A pair-count γ on X_i is active if and only if there exists an active quad-count ρ on X_i such that for each pair $A, B \subseteq X_i$*

$$\gamma(A, B) = \sum \rho(A_l, B_l; A_r, B_r). \tag{2}$$

The summation above is taken over all quadruplets (A_l, B_l, A_r, B_r) such that $A = A_l$ and $B = (B_l \cup B_r) \cap X_l$.

Using Lemma 5 we compute all active pair-counts γ on X_i from all active quad-counts ρ on X_i. There are at most $(\alpha+1)^{2^{2k+3}}$ distinct active quad-counts ρ. From each ρ of them, we compute γ satisfying Eq. (2) in time $O(1)$ since $|A_l|, |B_l|, |A_r|, |B_r|, |X_i|, |X_l|, |X_r| \leq k+1$. Thus we have shown that all active pair-counts γ on X_i can be computed in time $O((\alpha+1)^{2^{2k+3}})$. Since T has $O(n)$ internal nodes, one can compute the tables for all internal nodes in time $O(n(\alpha+1)^{2^{2k+3}})$.

This completes a proof of Theorem 1.

4 Conclusion

In the paper we have given a polynomial-time algorithm to solve the total coloring problem for partial k-trees. One can immediately obtain a parallel algorithm to solve the total coloring problem for partial k-trees, slightly modifying the algorithm as follows. For a given tree-decomposition of a graph G with width at most k, one can obtain a binary tree-decomposition T of G with height $O(\log n)$ and width at most $3k+2$ in $O(\log n)$ parallel time using $O(n)$ operations on the EREW PRAM [BH95]. Since each leaf of T has at most $3k+3$ vertices, the tables of all active pair-counts on all leaves of T can be computed in $O(1)$ parallel time using $O(n)$ operations on the common CRCW PRAM. For each internal node X of T, the number of all active pair-counts on X is at most $(\alpha+1)^{2^{6(k+1)}}$ since $|X| \leq 3k+3$. Therefore the table on each internal node can be computed from all active pair-counts of the two children in $O(1)$ parallel time using $O((\alpha+1)^{2^{6k+7}})$ operations on the common CRCW PRAM. Since the height of the binary tree-decomposition T is $O(\log n)$, one can compute the table on the root in $O(\log n)$ parallel time using $O(n(\alpha+1)^{2^{6k+7}})$ operations on the CRCW PRAM. Thus the parallel algorithm runs in $O(\log n)$ parallel time using $O(n(\alpha+1)^{2^{6k+7}})$ operations on the common CRCW PRAM.

References

ACPS93. S. Arnborg, B. Courcelle, A. Proskurowski and D. Seese. An algebraic theory of graph reduction, *J. Assoc. Comput. Mach.*, 40(5), pp. 1134–1164, 1993.

AL91. S. Arnborg and J. Lagergren. Easy problems for tree-decomposable graphs, *Journal of Algorithms*, 12(2), pp. 308–340, 1991.

BH95. H. L. Bodlaender and T. Hagerup. Parallel algorithms with optimal speedup for bounded treewidth, *Proc. of the 22nd International Colloquium on Automata, Languages and Programming, Lecture Notes in Computer Science*, 944, pp. 268–279, Springer Verlag, 1995.

Bod90. H. L. Bodlaender. Polynomial algorithms for graph isomorphism and chromatic index on partial k-trees, *Journal of Algorithms*, 11(4), pp. 631–643, 1990.

Bod96. H. L. Bodlaender. A linear time algorithm for finding tree decompositions of small treewidth, *SIAM Journal on Computing*, 25, pp. 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, pp. 555–581, 1992.

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

FW77. S. Fiorini and R. J. Wilson. Edge-Colourings of Graphs, Pitman, London, 1977.

Sán89. A. Sánchez-Arroyo. Determining the total colouring number is NP-hard, *Discrete Math.*, 78, pp. 315–319, 1989.

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

TNS82. K. Takamizawa, T. Nishizeki and N. Saito. Linear-time computability of combinatorial problems on series-parallel graphs, *J. Assoc. Comput. Mach.*, 29(3), pp. 623–641, 1982.

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

Yap96. H. P. Yap. Total Colourings of Graphs, Lecture Notes in Mathematics, 1623, Springer-Verlag, Berlin, 1996.

ZN95. X. Zhou and T. Nishizeki. Algorithms for finding f-coloring of partial k-trees, In *Proc. of the Sixth International Symposium on Algorithms and Computation, Lecture Notes in Computer Science*, 1004, pp. 332–341, Springer-Verlag, 1995.

ZN98. X. Zhou and T. Nishizeki. The edge-disjoint paths problem is NP-complete for partial k-trees, submitted to a symposium.

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

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

ZSN97. X. Zhou, H. Suzuki and T. Nishizeki. An NC parallel algorithm for edge-coloring series-parallel multigraphs, *Journal of Algorithms*, 23, pp. 359–374, 1997.

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, Lecture Notes in Computer Science, Springer*, 1178, pp. 203–212, 1996.