アルゴリズム 54－10
（1996．10．17）

部分 k-木を $[g, f]$-辺彩色する多項式時間アルゴリズム

周 暁　布施 一樹　西関 隆夫
東北大学

グラフの辺彩色とは各点に接続する辺の色が必ず異なるようにグラフの辺を彩色することである．これに対しグラフの $[g, f]$-辺彩色とは各点 v に接続している辺のうち同じ色で彩色されるのが $g(v)$ 本以上かつ $f(v)$ 本以下になるように辺に彩色することである．本論文は部分 k 木を最少色数で $[g, f]$-辺彩色する多項式時間アルゴリズムを与える．

An Algorithm for Finding $[g, f]$-Colorings of Partial k-Trees

Xiao Zhou[1], *Kazuki Fuse*[2] *and Takao Nishizeki*[3]
Graduate School of Information Sciences
Tohoku University, Sendai 980-77, JAPAN.

Abstract

In an ordinary edge-coloring of a graph $G = (V, E)$ each color appears at each vertex $v \in V$ at most once. A $[g, f]$-coloring is a generalized edge-coloring in which each color appears at each vertex $v \in V$ at least $g(v)$ and at most $f(v)$ times, where $g(v)$ and $f(v)$ are nonnegative and positive integers, respectively, assigned to v. This paper gives a polynomial-time algorithm to find a $[g, f]$-coloring of a given partial k-tree with the minimum number of colors if such a coloring exists.

1 Introduction

This paper deals with generalized edge-coloring, called a $[g, f]$-coloring, of partial k-trees. A partial k-tree, formally defined later, is a simple graph, that is, it has no multiple edges or self-loops. A partial 3-tree is depicted in Figure 1. Throughout the paper we consider only partial k-trees with bounded k, that is, graphs with bounded treewidths. The class of partial k-trees includes many interesting classes of graphs such as outer planar graphs, series-parallel graphs and Halin graphs.

An *edge-coloring* of a graph G is to color all the edges of G so that no two adjacent edges are colored with the same color. There are linear-time sequential and optimal parallel algorithms to edge-color partial k-trees with the minimum number of colors [8, 9, 10, 12].

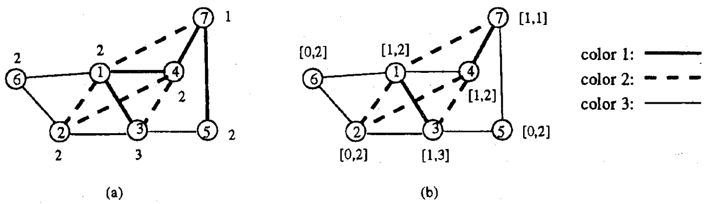

Figure 1: (a) An f-coloring and (b) a $[g, f]$-coloring.

Let f be a function which assigns a positive integer $f(v)$ to each vertex $v \in V$. Then an *f-coloring* of G is to color all the edges of G so that, for each vertex $v \in V$, at most $f(v)$ edges incident to v are colored with the same color. Figure 1(a) illustrates an f-coloring of a partial 3-tree with three colors, where numbers next to vertices v are $f(v)$'s and three colors are indicated by thin, thick and dotted lines. The ordinary edge-coloring is a special case of an f-coloring for which $f(v) = 1$ for every vertex $v \in V$.

[1]Email: zhou@ecip.tohoku.ac.jp
[2]Email: fuse@nishizeki.ecei.tohoku.ac.jp
[3]Email: nishi@ecei.tohoku.ac.jp

We call the minimum number of colors needed for an f-coloring the *f-chromatic index* of G, and denote it by $\chi'_f(G)$. We call $\Delta_f(G) = \max_{v \in V} \lceil d(v)/f(v) \rceil$ the *maximum f-degree* of graph G, where $d(v)$ is the degree of vertex v. It is known that $\chi'_f(G) = \Delta_f(G)$ or $\Delta_f(G) + 1$ for any simple graph G [4]. The *f-coloring problem* is to find an *optimal f-coloring* of G, that is, an f-coloring using $\chi'_f(G)$ colors. The problem arises in many applications, including the scheduling of file transfers in computer networks [6]. There is a polynomial-time algorithm to solve the f-coloring problem for partial k-trees [11].

Let G be a graph without isolated vertices. Let g be a function which assigns a nonnegative integer $g(v)$ to each vertex $v \in V$. Then a *$[g, f]$-coloring* of G is to color all the edges of G so that, for each vertex $v \in V$, at least $g(v)$ and at most $f(v)$ edges incident to v are colored with the same color. Figure 1(b) illustrates a $[g, f]$-coloring of a partial 3-tree with three colors, where numbers next to vertices v are $g(v)$'s and $f(v)$'s and three colors are indicated by thin, thick and dotted lines. The f-coloring is a special case of a $[g, f]$-coloring for which $g(v) = 0$ for every vertex $v \in V$. The *minimum g-degree* $\delta_g(G)$ of graph G is defined as follows:

$$\delta_g(G) = \min_{v \in V} \lfloor d(v)/g(v) \rfloor,$$

where we set $\lfloor d(v)/g(v) \rfloor = \infty$ if $g(v) = 0$. For a $[g, f]$-coloring φ of G, we denote by $\#\varphi$ the number of colors used by φ. It is clear that for any $[g, f]$-coloring φ of G

$$\Delta_f(G) \leq \#\varphi \leq \delta_g(G).$$

It should be noted that there does not always exist a $[g, f]$-coloring of a given graph. The *$[g, f]$-coloring problem* is to find a $[g, f]$-coloring of G using the minimum number of colors if it exists. The problem arises in many applications, including the scheduling of file transfers in computer networks [6].

Since the ordinary edge-coloring problem is NP-complete [5], the $[g, f]$-coloring problem is also NP-complete in general. Therefore it is very unlikely that there exists a sequential algorithm which solves the $[g, f]$-coloring problem in polynomial time. Furthermore, the $[g, f]$-coloring problem is one of the typical "edge-partitioning problems" which, as mentioned in [3], does not appear to be efficiently solvable even for partial k-trees although many combinatorial problems of other types can be solved very efficiently for partial k-trees.

In this paper we give a polynomial-time algorithm to solve the $[g, f]$-coloring problem for partial k-trees for the case all $f(v)$'s are bounded. Our idea is to bound the size of a DP table by $n^{O(1)}$, applying and extending techniques developed for the ordinary edge-coloring problem and the f-coloring problem [1, 8, 11, 12], where n is the number of vertices in G. This paper is organized as follows. In Section 2 we present some preliminary definitions. In Section 3 we give a polynomial-time algorithm for the $[g, f]$-coloring problem on partial k-trees. In Section 4 we conclude with some generalizations of our 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 isolated vertices, multiple edges or self-loops. An edge joining vertices u and v is denoted by (u, v). The *degree* of vertex $v \in V(G)$ is denoted by $d(v, G)$ or simply by $d(v)$. The *maximum degree* of G is denoted by $\Delta(G)$ or simply by Δ. 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 $|E| < kn$. Figure 2 illustrates a process of generating a 3-tree. The graph in Figure 1 is indeed a subgraph of the 3-tree, and hence 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 [7]:

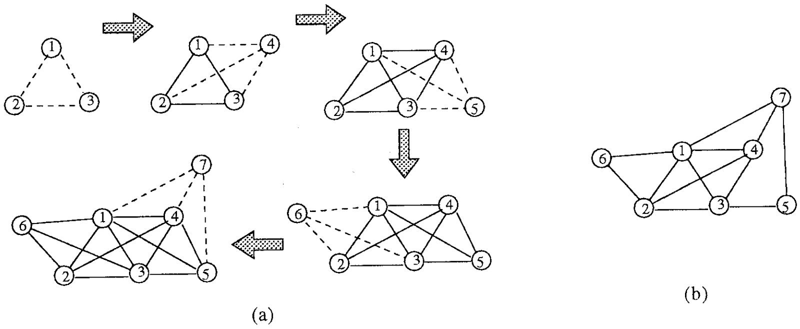

Figure 2: A process of generating a 3-tree.

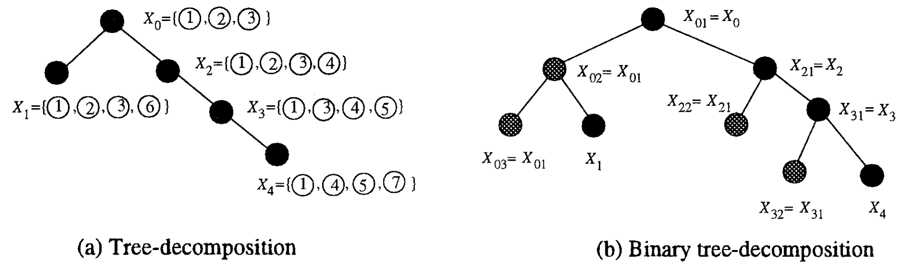

Figure 3: Tree-decompositions of the partial 3-tree in Figure 1.

- $\bigcup_{X_i \in V_T} X_i = V$;
- 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 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 $\max_{X_i \in V_T} |X_i| - 1$. 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 bounded k [2].

Consider a tree-decomposition of a partial k-tree G with width $\leq k$. We transform it to a binary tree T as follows [1]: regard T as a rooted tree by choosing an arbitrary node as the root X_0, and replace every internal node X_i with 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. (See Figure 4.) This transformation can be done in $O(n)$ time. 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_l and X_r, and either $X_i = X_l$ or $X_i = X_r$; and
- for each edge $(v, w) \in E$ there is at least one leaf X_i with $v, w \in X_i$.

Such a tree T is called a *binary* tree-decomposition [1]. Figure 3(b) illustrates a binary transformation of the tree-decomposition in Figure 3(a). Clearly T has width $\leq k$. For each edge $e = (v, w) \in E$, we choose an arbitrary leaf X_i of T such that $v, w \in X_i$ 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 $E(X_i) = \{e \in E \mid rep(e) = X_i\}$. 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

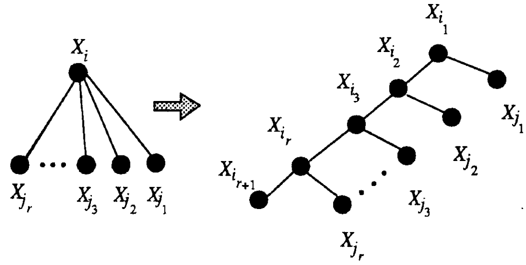

Figure 4: Illustration of the binary transformation.

T corresponds to a subgraph $G[E(X_i)]$ of G induced by the edges in $E(X_i)$. The subgraph $G[E(X_i)]$ is denoted simply by $G[X_i]$ or G_i. Then G_i is an edge-disjoint union of two subgraphs G_l and G_r, which share common vertices only in X_i because of the third property of a tree-decomposition. ($G_{02} = G[X_{02}]$ and $G_{21} = G[X_{21}]$ for the partial 3-tree G in Figure 1 will be depicted later in Figure 5.)

Let f be a function which assigns a natural number $f(v)$ to each vertex $v \in V$. Let $m_f = \max\{f(v) \mid v \in V(G)\}$. One may assume without loss of generality that $f(v) \leq d(v)$ for each vertex $v \in V(G)$. Furthermore we assume in the paper that all $f(v)$'s are bounded, that is, m_f is bounded. We call $d_f(v,G) = \lceil d(v,G)/f(v) \rceil$ the *f-degree* of vertex v, and often denote it by $d_f(v,G)$ or simply by $d_f(v)$. We call $\Delta_f(G) = \max\{d_f(v) | v \in V(G)\}$ the *maximum f-degree* of G.

Let g be a function which assigns a nonnegative integer $g(v)$ to each vertex $v \in V$. The *g-degree* of vertex $v \in V$ is defined as follows:

$$d_g(v,G) = \begin{cases} \lfloor d(v)/g(v) \rfloor & \text{if } g(v) \geq 1, \\ \infty & \text{if } g(v) = 0. \end{cases}$$

We often denote $d_g(v,G)$ simply by $d_g(v)$. We call $\delta_g(G) = \min\{d_g(v) | v \in V(G)\}$ the *minimum g-degree* of G.

Let C be a set of colors. Any mapping (coloring) $\varphi : E \to C$ is called a *$[g,f]$-coloring* of G if $g(v) \leq d_\varphi(v,c) \leq f(v)$ for each vertex $v \in V$ and each color $c \in C$, where $d_\varphi(v,c)$ is the number of edges of G which are incident to v and colored with c by φ. Then we have the following lemma.

Lemma 2.1 *If a graph G has a $[g,f]$-coloring φ, then*

$$\Delta_f(G) \leq \#\varphi \leq \delta_g(G)$$

and G has a $[g,f]$-coloring φ' such that $\#\varphi' \leq \min\{\Delta + 1, \delta_g(G)\}$.

3 A Polynomial-Time Algorithm

In this section we prove the following theorem. Although the algorithm in the theorem only decides whether $G = (V,E)$ has a $[g,f]$-coloring: $E \to C$ for a given set C of colors, it can be easily modified so that it actually finds a $[g,f]$-coloring: $E \to C$ of G.

Theorem 3.1 *Let G be a partial k-tree given by its tree-decomposition with width $\leq k$, and let C be a set of colors. Then there is an algorithm to determine whether G has a $[g,f]$-coloring: $E \to C$ in polynomial time for bounded k and m_f.*

By Lemma 2.1 and Theorem 3.1 we have the following theorem.

Theorem 3.2 *Let G be a partial k-tree given by its tree-decomposition with width $\leq k$. Then there is an algorithm to find a $[g,f]$-coloring of G with the minimum number of colors in polynomial time if G has a $[g,f]$-coloring and both k and m_f are bounded.*

In the remainder of this section we will give a proof of Theorem 3.1. Our idea is to solve the $[g,f]$-coloring problem using dynamic programming with a table of size at most $O(n^{(m_f+1)^{2(k+1)}})$. We employ techniques developed for the ordinary edge-coloring problem [1].

Let $C = \{1, 2, \cdots, \alpha\}$ be the set of colors. Let $G = (V,E)$ be a partial k-tree. Let X_i be a node of a tree-decomposition T of G. If a mapping (coloring) $\varphi : E(X_i) \to C$ of G_i can be extended to a $[g,f]$-coloring of G, then φ must satisfy the following conditions i) and ii):

i) if vertex $v \in V(G_i) - X_i$, then $g(v) \le d_\varphi(v,c) \le f(v)$ for each color $c \in C$, and

ii) if vertex $v \in X_i$, then $d_\varphi(v,c) \le f(v)$ for each color $c \in C$.

Such a coloring φ is called a *valid* $[g,f]$*-coloring of* G_i. There may exist a vertex $v \in X_i$ such that $d_\varphi(v,c) < g(v)$ for some color $c \in C$, but some edges $(v,x) \in E - E(X_i)$ incident to v may be colored by c later. A valid $[g,f]$-coloring φ of G_i is *extensible* if it can be extended to a $[g,f]$-coloring φ^* of $G = G[X_{01}]$, where X_{01} is the root of a binary tree-decomposition T. Figure 5(a) depicts a $[g,f]$-coloring of the partial 3-tree G in Figure 1, Figures 5(b) a valid $[g,f]$-coloring of $G_{02} = G[X_{02}]$, and Figures 5(c) a valid $[g,f]$-coloring of $G_{21} = G[X_{21}]$, where $X_{02} = \{①, ②, ③\}$ and $X_{21} = \{①, ②, ③, ④\}$ are respectively the left and right children of the root X_{01} of the binary tree-decomposition T in Figure 3(b). Both of the valid $[g,f]$-colorings in Figures 5(b) and (c) are extensible, because either can be extended to the $[g,f]$-coloring of $G = G[X_{01}]$ in Figure 5(a).

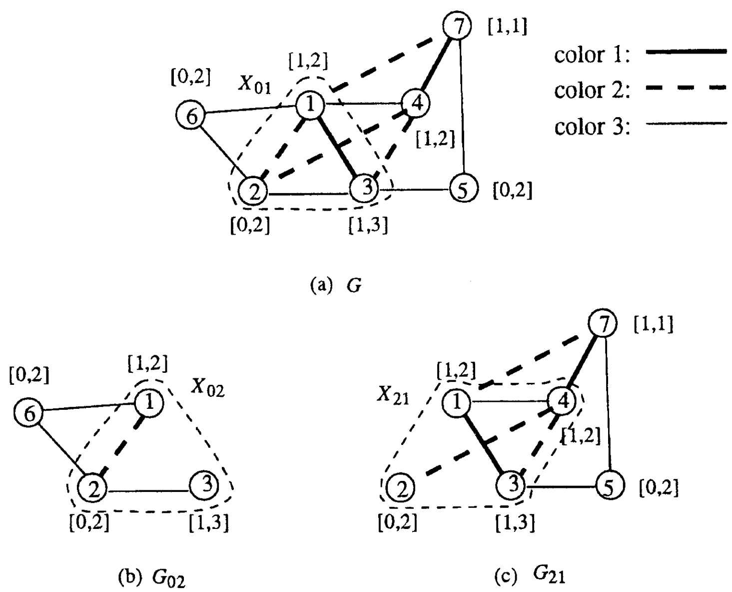

Figure 5: A $[g,f]$-coloring of (a) G and valid $[g,f]$-colorings of (b) G_{02} and (c) G_{21}.

Clearly the following lemma holds.

Lemma 3.3 *Let* $\varphi : E(X_i) \to C$ *be a valid* $[g,f]$*-coloring of* $G_i = G[X_i]$. *Let* $\xi : C \to C$ *be a permutation (bijection) of* C. *Then the composite* $\xi \circ \varphi : E(X_i) \to C$ *of* φ *and* ξ *is a valid* $[g,f]$*-coloring of* G_i.

Define a set $\mathcal{P}(X_i, m_f)$ of (m_f+1)-tuples of sets as follows:

$$\mathcal{P}(X_i, m_f) = \{\mathbf{S} = (S_0, S_1, \cdots, S_{m_f}) \mid S_0, S_1, \cdots, S_{m_f} \text{ are pairwise disjoint subsets of } X_i \text{ and } \bigcup_{j=1}^{m_f} S_j = X_i\}.$$

Since $|X_i| \le k+1$, then

$$|\mathcal{P}(X_i, m_f)| \le (m_f+1)^{k+1}. \tag{1}$$

A *color vector* $\mathbf{C}(X_i) = (\mathbf{S}_1, \mathbf{S}_2, \cdots, \mathbf{S}_\alpha)$ on a node X_i of tree-decomposition T is defined to be a α-tuple of vectors $\mathbf{S}_c \in \mathcal{P}(X_i, m_f)$, $c \in C$. A color vector $\mathbf{C}(X_i) = (\mathbf{S}_1, \mathbf{S}_2, \cdots, \mathbf{S}_\alpha)$ is *active* if G_i has a valid $[g,f]$-coloring such that for each color $c \in C$ and each integer j, $0 \le j \le m_f$,

$$S_{cj} = \{v \in X_i | d_\varphi(v,c) = j\}.$$

The valid $[g,f]$-coloring of $G = G[X_{01}]$ depicted in Figure 5(a) has a color vector $\mathbf{C}(X_{01}) = (\mathbf{S}_{X_{01}1}, \mathbf{S}_{X_{01}2}, \mathbf{S}_{X_{01}3})$ such that

$$\begin{aligned} \mathbf{S}_{X_{01}1} &= (\{②\},\{①,③\},\phi,\phi) \\ \mathbf{S}_{X_{01}2} &= (\phi,\{③\},\{①,②\},\phi) \text{ and} \\ \mathbf{S}_{X_{01}3} &= (\phi,\phi,\{①,②,③\},\phi). \end{aligned}$$

Note that $m_f = 3$.

We now have the following lemma.

Lemma 3.4 *Let two valid $[g,f]$-colorings φ and ψ of $G_i = G[X_i]$ have the same color vector. Then φ is extensible if and only if ψ is extensible.*

Thus a color vector on X_i characterizes an equivalence class of valid $[g,f]$-colorings of G_i. By Eq. (1) the total number of different color vectors $\mathbf{C}(X_i)$ on X_i is

$$(m_f + 1)^{(k+1)\alpha}.$$

Since α is not always bounded, the total number of different color vectors is not polynomially bounded. However, the number of distinct "counts" classfying all valid $[g,f]$-colorings is polynomially bounded, as follows. We call a mapping $\gamma : \mathcal{P}(X_i, m_f) \to \{0,1,2,\cdots,\alpha\}$ a *count on a node* X_i. A count γ on X_i is defined to be ***active*** if there exists an active color vector $\mathbf{C}(X_i) = (\mathbf{S}_1, \mathbf{S}_2, \cdots, \mathbf{S}_\alpha)$ such that $\gamma(\mathbf{S}) = |\{c \in C \mid \mathbf{S} = \mathbf{S}_c\}|$ for each $\mathbf{S} \in \mathcal{P}(X_i, m_f)$. Clearly

$$\sum_{\mathbf{S} \in \mathcal{P}(X_i, m_f)} \gamma(\mathbf{S}) = \alpha.$$

Such γ is called a *count of the color vector* $\mathbf{C}(X_i)$, and is called a *count of a valid $[g,f]$-coloring* φ of G_i if $\mathbf{C}(X_i)$ is the color vector of φ.

Thus, the $[g,f]$-coloring of $G = G[X_{01}]$ for the root $X_{01} = \{①,②,③\}$ depicted in Figure 5(a) has the count $\gamma_{X_{01}}$ such that

$$\begin{aligned} \gamma_{X_{01}1}((\{②\},\{①,③\},\phi,\phi)) &= 1, \\ \gamma_{X_{01}2}((\phi,\{③\},\{①,②\},\phi)) &= 1, \\ \gamma_{X_{01}3}((\phi,\phi,\{①,②,③\},\phi)) &= 1, \end{aligned}$$

and $\gamma_{X_{01}}(\mathbf{S}) = 0$ for any other $\mathbf{S} \in \mathcal{P}(X_i, m_f)$.

We now have the following lemma.

Lemma 3.5 *Let two valid $[g,f]$-colorings φ and ψ of G_i have the same count. Then φ is extensible if and only if ψ is extensible.*

By Lemma 3.5 an active count γ characterizes an equivalence class of valid $[g,f]$-colorings of G_i. Since $|\mathcal{P}(X_i, m_f)| \le (m_f+1)^{k+1}$, the number n_γ of distinct active counts $\gamma : \mathcal{P}(X_i, m_f) \to \{0,1,\cdots,\alpha\}$ is at most

$$n_\gamma \le (\alpha+1)^{(m_f+1)^{k+1}} = O\left(n^{(m_f+1)^{k+1}}\right) \tag{2}$$

since $\alpha \le |E| \le kn$. Thus n_γ is bounded by a polynomial in n.

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 dynamic programming. From the table on X_{01} one can easily check whether G has a $[g,f]$-coloring, as follows.

Lemma 3.6 *A partial k-tree $G = (V,E)$ has a $[g,f]$-coloring : $E \to C$ if and only if the table on root X_{01} has at least one active count γ such that, for any $\mathbf{S} = (S_0, S_1, \cdots, S_{m_f}) \in \mathcal{P}(X_i, m_f)$, if $\gamma(\mathbf{S}) \ge 1$ then $g(v) \le j$ for every vertex $v \in S_j$, $0 \le j \le m_f$.*

We next compute all active counts on each internal node X_i of T from all active counts of its children. 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

$$\rho : \mathcal{P}(X_l, m_f) \times \mathcal{P}(X_r, m_f) \to \{0,1,2,\cdots,\alpha\}$$

a *pair-count* on X_i. We define a pair-count ρ to be *active* if there is a valid $[g, f]$-coloring $\varphi : E(X_i) \to C$ such that, for each pair $(\mathbf{S}^l, \mathbf{S}^r)$ with $\mathbf{S}^l \in \mathcal{P}(X_l, m_f)$ and $\mathbf{S}^r \in \mathcal{P}(X_r, m_f)$,

$$\rho(\mathbf{S}^l, \mathbf{S}^r) = |\{c \in C \mid \mathbf{S}^l = \mathbf{S}^l_c, \mathbf{S}^r = \mathbf{S}^r_c\}|,$$

where

$$\mathbf{C}(X_l) = (\mathbf{S}^l_1, \mathbf{S}^l_2, \cdots, \mathbf{S}^l_\alpha)$$

is the color vector of the restriction of φ to $E(X_l)$, and

$$\mathbf{C}(X_r) = (\mathbf{S}^r_1, \mathbf{S}^r_2, \cdots, \mathbf{S}^r_\alpha)$$

is the color vector of the restriction of φ to $E(X_r)$. Such a mapping ρ is called a *pair count of a valid coloring* φ of G_i. Then we have the following lemma.

Lemma 3.7 *Let an internal node X_i of T have two children X_l and X_r, and let $X_i = X_l$. Then a pair-count ρ on X_i is active if and only if the following conditions* (a) *and* (b) *hold:*

(a) *for each pair of $\mathbf{S}^l = (S^l_0, S^l_1, \cdots, S^l_{m_f}) \in \mathcal{P}(X_l, m_f)$ and $\mathbf{S}^r = (S^r_0, S^r_1, \cdots, S^r_{m_f}) \in \mathcal{P}(X_r, m_f)$, $\rho(\mathbf{S}^l, \mathbf{S}^r) \geq 1$ implies the following* (a1) *and* (a2)*:*

   (a1) *if $v \in S^l_{j_l} \cap S^r_{j_r}$, $0 \leq j_l, j_r \leq m_f$, then $j_l + j_r \leq f(v)$, and*

   (a2) *if $v \in S^r_{j_r} - X_l$, $0 \leq j_r \leq m_f$, then $g(v) \leq j_r$; and*

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

   (b1) *for each $\mathbf{S}^l \in \mathcal{P}(X_l, m_f)$*
   $$\gamma_l(\mathbf{S}^l) = \sum_{\mathbf{S} \in \mathcal{P}(X_r, m_f)} \rho(\mathbf{S}^l, \mathbf{S}); \quad \textit{and}$$

   (b2) *for each $\mathbf{S}^r \in \mathcal{P}(X_r, m_f)$*
   $$\gamma_r(\mathbf{S}^r) = \sum_{\mathbf{S} \in \mathcal{P}(X_l, m_f)} \rho(\mathbf{S}, \mathbf{S}^r).$$

Using Lemma 3.7, we compute all active pair-counts on X_i from all pairs of active counts on X_l and X_r. By Eq. (1) we have

$$|\mathcal{P}(X_l, m_f)|, |\mathcal{P}(X_r, m_f)| \leq (m_f + 1)^{k+1}.$$

Therefore there are at most $(\alpha + 1)^{(m_f+1)^{2(k+1)}}$ distinct active pair-counts. For each ρ of them, we check whether ρ satisfies the conditions (a) and (b) in Lemma 3.7. For each pair $(\mathbf{S}^l, \mathbf{S}^r)$, one can check in time $O(1)$ whether $\rho(\mathbf{S}^l, \mathbf{S}^r) \geq 1$ implies the conditions (a1) and (a2) since $|S^l_{j_l}|$, $|S^r_{j_r}| \leq k + 1$ for all j_l and j_r, $0 \leq j_l, j_r \leq m_f$. There are at most $(m_f + 1)^{2(k+1)} = O(1)$ pairs $(\mathbf{S}^l, \mathbf{S}^r)$. Therefore, for each ρ, one can check in time $O(1)$ whether ρ satisfies the condition (a). Hence, checking the condition (a) for all possible ρ's can be done in time $O((\alpha + 1)^{(m_f+1)^{2(k+1)}})$. Furthermore, checking the condition (b) for each ρ can be done in time $O(1)$ since there are at most $O(1)$ pairs $(\mathbf{S}^l, \mathbf{S}^r)$. Therefore, checking the condition (b) for all possible ρ's can be done in time $O((\alpha + 1)^{(m_f+1)^{2(k+1)}})$. Thus we have shown that all active pair-counts ρ on X_i can be computed in time

$$O\left((\alpha + 1)^{(m_f+1)^{2(k+1)}}\right) = O\left(n^{(m_f+1)^{2(k+1)}}\right),$$

since $\alpha \leq |E| \leq kn$. We now show how to compute all active counts on an internal node X_i from all active pair-counts on X_i.

Lemma 3.8 *Let an internal node X_i of T have two children X_l and X_r with $X_i = X_l$. A count γ on X_i is active if and only if there exists an active pair-count ρ on X_i such that for each $\mathbf{S} = (S_0, S_1, \cdots, S_{m_f}) \in \mathcal{P}(X_i, m_f)$*

$$\gamma(\mathbf{S}) = \sum \rho(\mathbf{S}^l, \mathbf{S}^r). \tag{3}$$

The summation above is taken over all pairs $(\mathbf{S}^l, \mathbf{S}^r)$ such that

(a) $\mathbf{S}^l = (S_0^l, S_1^l, \cdots, S_{m_f}^l) \in \mathcal{P}(X_l, m_f)$ *and* $\mathbf{S}^r = (S_0^r, S_1^r, \cdots, S_{m_f}^r) \in \mathcal{P}(X_r, m_f)$*;*

(b) *if* $v \in S_{j_l}^l \cap S_{j_r}^r$, $0 \le j_l, j_r \le m_f$, *then* $v \in S_{j_l+j_r}$*; and*

(c) *if* $v \in S_{j_l}^l - X_r$, $0 \le j_l \le m_f$, *then* $v \in S_{j_l}$.

Using Lemma 3.8, we compute all active counts on X_i from all active pair-counts on X_i. Since $|\mathcal{P}(X_i, m_f)| \le (m_f+1)^{k+1}$, there are at most $(\alpha+1)^{(m_f+1)^{2(k+1)}}$ distinct active pair-counts ρ. From each ρ of them, we compute γ satisfying Eq. (3). There are at most $(m_f+1)^{2(k+1)} = O(1)$ pairs $(\mathbf{S}^l, \mathbf{S}^r)$. For each pair $(\mathbf{S}^l, \mathbf{S}^r)$ we can check the conditions (b) and (c) in $O(1)$ time since $|S_{j_l}^l|, |S_{j_r}^r| \le k+1$ for all j_l and j_r, $0 \le j_l, j_r \le m_f$. Therefore one can check the conditions (a), (b) and (c) total in $O(1)$ time. Thus we have shown that all active counts γ on X_i can be computed in time

$$O\left((\alpha+1)^{(m_f+1)^{2(k+1)}}\right) = O\left(n^{(m_f+1)^{2(k+1)}}\right),$$

since $\alpha \le |E(G)| \le kn$.

The number of internal nodes in tree-decomposition T is $O(n)$. Therefore we can compute the table on root of T from leaves to root by means of dynamic programming total in time $O\left(n^{1+(m_f+1)^{2(k+1)}}\right)$. By Lemma 3.6 one can check whether there is a $[g, f]$-coloring: $E \to C$ in time $O\left(n^{(m_f+1)^{2(k+1)}}\right)$. Thus the algorithm can be done total in time $O\left(n^{1+(m_f+1)^{2(k+1)}}\right)$.

This completes a proof of Theorem 3.1.

4 Conclusion

In this paper we gave a polynomial-time algorithm for the $[g, f]$-coloring problem on partial k-trees. Our algorithms solve a single particular problem, that is, the $[g, f]$-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 the 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 between $g(v)$ and $f(v)$, where $g(v)$ and $f(v)$ are nonnegative and positive integers, respectively, assigned to v. Clearly the edge-partition problem with respect to such a property π is the same as the $[g, f]$-coloring problem.

References

[1] 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.

[2] 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.

[3] 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.

[4] S. L. Hakimi and O. Kariv. On a generalization of edge-coloring in graphs. *Journal of Graph Theory*, Vol. 10, pp. 139–154, 1986.

[5] I. Holyer. The NP-completeness of edge-colouring. *SIAM J. Comput.*, Vol. 10, pp. 718–720, 1981.

[6] S. Nakano and T. Nishizeki. Scheduling file transfers under port and channel constraints. *Int. J. Found. of Comput. Sci.*, Vol. 4, No. 2, pp. 101–115, 1993.

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

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

[9] X. Zhou, S. Nakano, and T. Nishizeki. A linear algorithm for edge-coloring partial k-trees. In *Proc. of the First Europian Symposium on Algorithms, Lect. Notes in Computer Science, Springer-Verlag*, Vol. 726, pp. 409–418, 1993.

[10] X. Zhou, S. Nakano, and T. Nishizeki. A parallel algorithm for edge-coloring partial k-trees. In *Proc. of the Fourth Scandinavian Workshop on Algorithm Theory, Lect. Notes in Computer Science, Springer-Verlag*, Vol. 824, pp. 359–369, 1994.

[11] X. Zhou and T. Nishizeki. Algorithms for finding f-colorings of partial k-trees. In *Proc. of the Sixth International Symposium on Algorithms and Computation, Lect. Notes in Computer Science, Springer*, Vol. 1004, pp. 332–341, 1995.

[12] 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.