Cost Total Colorings of Trees

Shuji ISOBE†, Nonmember, Xiao ZHOU†, and Takao NISHIZEKI†, Members

SUMMARY A total coloring of a graph $G$ is to color all vertices and edges of $G$ so that no two adjacent or incident elements receive the same color. Let $C$ be a set of colors, and let $\omega$ be a cost function which assigns to each color $c$ in $C$ a real number $\omega(c)$ as a cost of $c$. A total coloring $f$ of $G$ is called an optimal total coloring if the sum of costs $\omega(f(x))$ of colors $f(x)$ assigned to all vertices and edges $x$ is as small as possible. In this paper, we give an algorithm to find an optimal total coloring of any graph $G$. Let $\chi_T$ denote the minimum number of colors necessary for a total coloring of a tree $T$. The minimum number of colors necessary for a total coloring of $G$ is called the total chromatic number of $G$, and is denoted by $\chi(G)$. Let $\Delta$ be the maximum degree of $G$, then clearly $\chi(G) \geq \Delta + 1$. The total coloring problem asks to find a total coloring of a given graph $G$ with the minimum number $\chi(G)$ of colors. Figure 1 (a) depicts a total coloring of a tree $T$ with four colors $c_1, c_2, c_3$ and $c_4$, while Fig. 1 (b) depicts a total coloring of the same tree $T$ with five colors $c_1, c_2, \ldots, c_5$. Note that $\chi(T) = \Delta + 1 = 4$.

Let $\alpha \geq \chi(G)$, and let $C = \{c_1, c_2, \ldots, c_{\alpha}\}$ be a set of $\alpha$ colors. Let $\omega : C \rightarrow \mathbb{R}$ be a cost function which assigns to each color $c$ in $C$ a real number $\omega(c) \in \mathbb{R}$ as a cost of $c$. A cost $\omega(f)$ of a total coloring $f : V \cup E \rightarrow C$ of a graph $G = (V, E)$ is defined as follows:

$$\omega(f) = \sum_{x \in V \cup E} \omega(f(x)).$$

We say that a total coloring $f$ of $G$ is optimal (for $\omega$) if the cost $\omega(f)$ is as small as possible. The cost total coloring problem is to find an optimal total coloring of a given graph $G$. We call the cost $\omega(f)$ of an optimal total coloring $f$ of $G$ the minimum cost $\omega(G)$ of $G$. An optimal total coloring of $G$ does not always use exactly $\chi(G)$ colors. For example, suppose that $C = \{c_1, c_2, c_3, c_4, c_5\}$, $\omega(c_1) = \omega(c_2) = 1$ and $\omega(c_3) = \omega(c_4) = \omega(c_5) = 10$ for the tree $T$ in Fig. 1, then any total coloring of $T$ using $\chi(T) = 4$ colors in $C$ costs at least 47, because at least four elements must receive colors of cost 10 and at most seven elements can receive colors of cost 1, as illustrated in Fig. 1 (a). On the other hand, an optimal total coloring of $T$ costs 38 and uses $\chi(T) + 1 = 5$ colors in $C$ as illustrated in Fig. 1 (b). In Fig. 1 the vertices and edges receiving colors of cost 10 are drawn by thick lines and the others by thin lines.

The cost total coloring problem has a natural application in scheduling theory. Consider the scheduling problem of uniprocessor tasks and biprocessor tasks having unit execution time on dedicated machines. The problem can be modeled by a graph $G$ in which machines and uniprocessor tasks correspond to the vertices, and biprocessor tasks correspond to the edges [7], [8], [11]. A total coloring $f$ of $G$ with $\alpha$ colors corresponds to a schedule with the completion time $\alpha$; the vertices and edges colored with color $c_i \in C$ represent the collection of tasks that are executed in the $i$-th time slot. For each $i$, if a task is executed in the $i$-th time slot, then it takes a cost $\omega(c_i)$. The goal is to find a schedule that minimizes the total cost. This corresponds to the cost total coloring problem.

Since the total coloring problem is NP-hard [10], the cost total coloring problem is NP-hard in general. Therefore, it is unlikely that the cost total coloring problem can be efficiently solved. However, when restricted to trees, the cost total coloring problem would be efficiently solved as well as many other combinatorial problems. For example, the cost vertex-coloring problem, similarly defined as the cost total coloring problem, can be solved in linear time for trees [6], and the cost edge-coloring problem can be solved in time $O(n\Delta^2)$ for trees $T$ [12], where $n$ is the number of vertices in $T$ and $\Delta$ is the maximum degree of $T$.

In this paper, we give an algorithm to solve the cost total coloring problem for trees in time $O(n\Delta^3)$. The algorithm is based on a clever formation of dynamic programming, a reduction of constructing a DP table to finding a bi-
partite matching, and a batch processing of bipartite matchings. Thus the scheduling problem above can be solved in polynomial time for the case where the graph modeling the problem is a tree; for example, if the machines are connected in a tree-structure and each task is executed by either a single machine or a pair of adjacent machines, then the graph is a tree.

2. Algorithm

Our main result is the following theorem.

**Theorem 1**: An optimal total coloring of a tree \( T \) can be found in time \( O(n\Delta^3) \) where \( n \) is the number of vertices and \( \Delta \) is the maximum degree of \( T \).

In the remainder of this section we give a proof of Theorem 1. Although we give an algorithm to compute the minimum cost \( \omega(T) \) of a given tree \( T \), it can be easily modified so that it actually finds an optimal coloring of \( T \) having cost \( \omega(T) \).

Let \( T = (V,E) \) be a tree, let \( n \) be the number of vertices of \( T \), and let \( \Delta \) be the maximum degree of \( T \). One may assume that \( n \geq 2 \). We denote by \( d(v) \) the degree of a vertex \( v \) in \( T \). A tree \( T \) is a “free tree,” but we regard \( T \) as a rooted tree by choosing an arbitrary vertex \( r \) of degree 1 as a root, as illustrated in Fig. 2. We will use notations such as root, child, descendant and leaf in their usual meaning. We denote by \( e = (v,w) \) an edge of a rooted tree \( T \) joining a vertex \( v \) and a child \( w \) of \( v \). An edge \( e = (v,w) \) is called a leaf edge if \( w \) is a leaf of \( T \). For each edge \( e = (v,w) \) of \( T \), we denote by \( T_e = (V_e,E_e) \) the subtree of \( T \) induced by \( v \) and all descendants of \( w \) in \( T \). \( T_e \) is drawn by thick lines in Fig. 2. Let \( e \) be the edge incident to root \( r \), then \( T = T_e \).

One may assume without loss of generality that the colors \( c_1, c_2, \ldots, c_\alpha \) in \( C \) are ordered so that their costs are non-decreasing, that is, \( \omega(c_i) \leq \omega(c_j) \) for any indices \( i \) and \( j \), \( 1 \leq i < j \leq \alpha \). One can easily observe that \( \chi_1(T) = 3 \) if \( n = 2 \), and \( \chi_1(T) = \Delta + 1 \) if \( n \geq 3 \). We may assume without loss of generality that \( \alpha = |C| \geq 2\Delta + 1 \); otherwise, add to \( C \) new colors \( c_i, \alpha + 1 \leq i \leq 2\Delta + 1 \), such that \( \omega(c_i) = \infty \). We then have the following lemma.

**Lemma 2**: Every graph \( G = (V,E) \) has an optimal total coloring \( f \) such that

(a) \( f(e) \in \{c_1, c_2, \ldots, c_{2\Delta+1}\} \) for any edge \( e \in E \), and
(b) \( f(v) \in \{c_1, c_2, \ldots, c_{2d(v)+1}\} \) for any vertex \( v \in V \).

**Proof.** (a) Suppose that \( f \) is an optimal total coloring of \( G \) but \( f(e) \notin \{c_1, c_2, \ldots, c_{2\Delta+1}\} \) for some edge \( e = (v,w) \). Then there exists a color \( c \in \{c_1, c_2, \ldots, c_{2\Delta+1}\} \) which is assigned to none of the \( d(v) + d(w) \) elements adjacent or incident to \( e \); the ends \( v, w \), and the edges (other than \( e \)) incident to \( v \) or \( w \). Let \( f' : V \cup E \rightarrow C \) be a total coloring of \( G \) obtained from \( f \) by recoloring \( e \) with the color \( c \). Then \( \omega(f') = \omega(f) - \omega(f(e)) + \omega(c) \leq \omega(f) \) since \( \omega(c) \leq \omega(f(e)) \). Thus \( f' \) must be an optimal total coloring of \( G \), and \( c = f'(e) \in \{c_1, c_2, \ldots, c_{2\Delta+1}\} \).

(b) Similar to (a).

Although \( \alpha = |C| \geq 2\Delta + 1 \), by Lemma 2 we may assume without loss of generality that \( \alpha = 2\Delta + 1 \).

A dynamic programming method is a standard one to solve a combinatorial problem on trees. Our algorithm also uses it, and computes the minimum cost \( \omega(T) \) of \( T \) by the bottom-up tree-computation. The main idea of our algorithm is to introduce a parameter \( \omega(T_e,c_i,c_j) \) for each edge \( e = (v,w) \) of \( T \) and each pair of indices \( i \) and \( j \) such that \( i \leq 2d(v)+1 \) and \( j \leq \alpha = 2\Delta + 1 \). We say that a total coloring \( f \) of a subtree \( T_e \) is \((c_i,c_j)\)-optimal if \( f(v) = c_i, f(e) = c_j, \) and \( \omega(f) = \omega(f(e)) \) as small as possible among all such total colorings \( f \) of \( T_e \). (See Fig. 3.) We define \( \omega(T_e,c_i,c_j) \) to be the cost \( \omega(f) \) of a \((c_i,c_j)\)-optimal total coloring \( f \) of \( T_e \). Let \( \omega(T_e,c_i,c_j) = \infty \) if \( i = j \) and there is no such total coloring.

For each edge \( e = (v,w) \) of \( T \), our algorithm constructs a DP table of size \( (2d(v) + 1) \times \alpha = O(d(v)\Delta) \); the entries of the table are \( \omega(T_e,c_i,c_j) \) for all pairs of indices \( i \leq 2d(v)+1 \) and \( j \leq \alpha \). Figures 4(a) and (b) illustrate \((c_1,c_2)\)- and \((c_1,c_3)\)-optimal colorings of a subtree \( T_e \), respectively, for the costs of colors in Fig. 4(c), and Fig. 4(d) is a DP table for edge \( e \) where we assume \( \Delta = 3 \) and \( d(v) = 3 \) in \( T \).

One can compute the minimum cost \( \omega(T) \) of \( T \) immediately from the table for root \( r \), because \( 2d(r) + 1 = 3 \) and the following Lemma 3 is an immediate consequence of Lemma 2(b).

**Lemma 3**: Let \( e_r \) be the edge of \( T \) incident to the root \( r \), then

\[
\omega(T) = \min\{\omega(T_{e_r},c_i,c_j) \mid 1 \leq i \leq 3, 1 \leq j \leq \alpha \}.
\]
Thus our algorithm is outlined as follows.

Algorithm COST-TOTAL-COLOR(T)

Step 1. Construct a DP-table for each leaf edge of T.

Step 2. Construct a DP-table for each non-leaf edge from the bottom to the top (root) of T.

Step 3. Compute \( \omega(T) \) from the DP-table for edge \( e_r \) by Eq. (1).

We then explain Steps 1, 2 and 3 in detail.

**[Step 1]** Step 1 is to construct a DP table for each leaf edge \( e = (v, w) \) of T. Since \( T_e \) consists of a single edge \( e = (v, w) \), we have

\[
\omega(T_e, c_i, c_j) = \omega(c_i) + \omega(c_j) + \min\{\omega(c) | c \in C - \{c_i, c_j\}\}
\]

for any pair of distinct indices \( i \) and \( j \). Since \( C \) is sorted with respect to the costs of colors, the value \( \min\{\omega(c) | c \in C - \{c_i, c_j\}\} \) can be found in time \( O(1) \) for each pair of distinct indices \( i \leq 2d(v) + 1 \) and \( j \leq \alpha \). One can thus construct a DP table for a leaf edge \( e \) in time \( O(d(v)\Delta) \). Since \( d(v) \leq \Delta \), we have the following lemma.

**Lemma 4:** The DP table for a leaf edge of T can be constructed in time \( O(\Delta^2) \).

**[Step 2]** Step 2 is to compute a DP table for each non-leaf edge. Let \( e = (v, w) \) be a non-leaf edge, let \( e_u = (w, u) \) be an edge joining \( w \) and \( u \) for each child \( u \) of \( w \), and let \( E_w = \{e_u | u \text{ is a child of } w\} \), then \( |E_w| = d(w) - 1 \). (See Fig. 3.) One may assume that the DP tables for all edges in \( E_w \) have been already constructed, and that we are going to construct a DP table for \( e \).

We now have the following lemma.

**Lemma 5:** Let \( e = (v, w) \) be a non-leaf edge of T, and let \( i \leq 2d(v) + 1 \), \( j \leq \alpha \) and \( i \neq j \). Then \( T_e \) has a \((c_i, c_j)\)-optimal total coloring \( f \) such that

(a) \( f(w) = c_k \) for some index \( k \) such that \( k \neq i, j \) and \( k \leq 2d(w) + 1 \);
(b) all colors \( f(e_u), e_u \in E_w \), are distinct from each other and are contained in set \( C_k = C - \{c_i, c_j\} \);
(c) the restriction \( f_u \) of \( f \) to \( T_{e_u} \) is a \((c_k, f(e_u))\)-optimal total coloring of \( T_{e_u} \) for each edge \( e_u \in E_w \), where \( f_u(x) = f(x) \) for all vertices and edges \( x \) of \( T_{e_u} \).

**Proof.** Let \( f \) be a \((c_i, c_j)\)-optimal total coloring of \( T_e \).

(a) Since \( f \) is a total coloring, \( f(w) = c_k \) for some index \( k, k \neq i, j \). Similarly as in the proof of Lemma 2 one can easily show that \( T_e \) has a \((c_i, c_j)\)-optimal total coloring \( f \) such that \( k \leq 2d(w) + 1 \).

(b) Obvious.

(c) Suppose for a contradiction that the total coloring \( f_u \) of \( T_{e_u} \) is not \((c_k, f(e_u))\)-optimal for an edge \( e_u \in E_w \), and let \( g_u \) be a \((c_k, f(e_u))\)-optimal total coloring of \( T_{e_u} \). Then \( \omega(g_u) = \omega(T_{e_u}, c_k, f(e_u)) < \omega(f_u) \). Construct a new total coloring \( g \) of \( T_e \) from \( f \) and \( g_u \) by replacing \( f_u \) in \( f \) with \( g_u \). Then \( g(v) = c_i, g(e) = c_j \), but \( \omega(g) = \omega(f) - \omega(f_u) + \omega(g_u) < \omega(f) \), contrary to the assumption that \( f \) is a \((c_i, c_j)\)-optimal total coloring of \( T_e \).

Conversely, for a pair of indices \( i \) and \( j \) such that \( i \leq 2d(v) + 1 \), \( j \leq \alpha \) and \( i \neq j \), let us assume that

(a) \( k \) is an arbitrary index such that \( k \neq i, j \) and \( k \leq 2d(w) + 1 \); and
(b) \( \varphi_{jk} : E_w \to C_k \) is an arbitrary injection from set \( E_w \) into set \( C_k \), and hence all colors \( \varphi_{jk}(e_u), e_u \in E_w \), are distinct from each other and are contained in \( C_k \).

For each edge \( e_u \in E_w \), let \( g_u \) be a \((c_k, \varphi_{jk}(e_u))\)-optimal total coloring of \( T_{e_u} = (V_{e_u}, E_{e_u}) \). (See Fig. 3.) Extend all the colorings \( g_u \) to a coloring \( g_{\varphi_{jk}} \) of \( T_e \) as follows:

\[
g_{\varphi_{jk}}(x) = \begin{cases} c_i & \text{if } x = v; \\ c_j & \text{if } x = e; \\ g_u(x) & \text{if } e_u \in E_w \text{ and } x \in V_{e_u} \cup E_{e_u}. \end{cases}
\]

Then \( g_{\varphi_{jk}} \) is a total coloring of \( T_e \) such that \( g_{\varphi_{jk}}(v) = c_i, g_{\varphi_{jk}}(e) = c_j \) and \( g_{\varphi_{jk}}(w) = c_k \). By the definition of \( g_{\varphi_{jk}} \), we have

\[
\omega(g_{\varphi_{jk}}) = \omega(c_i) + \omega(c_j) + x_{\varphi_{jk}}
\]

where

\[
x_{\varphi_{jk}} = \sum_{e_u \in E_w} \omega(g_u) - (d(w) - 2)\omega(c_k)
\]

\[
= \sum_{e_u \in E_w} \omega(T_{e_u}, c_k, \varphi_{jk}(e_u)) - (d(w) - 2)\omega(c_k).
\]

From the argument above and Lemma 5 we immediately have the following lemma.

**Lemma 6:** Let \( e = (v, w) \) be a non-leaf edge of T, and let \( i \leq 2d(v) + 1 \), \( j \leq \alpha \), and \( i \neq j \). Let
Thus the values $y_{ij}$ for all pairs of distinct indices $i \leq 2d(w) + 1$ and $j \leq \alpha$. These values $y_{ij}$ can be computed from the values $z_{jk}$ defined as follows:

$$z_{jk} = \min_{\varphi_{jk}} \sum_{e_u \in E_w} \omega(T_{e_u}, c_k, \varphi_{jk}(e_u))$$

(5)

where the minimum is taken over all injections $\varphi_{jk} : E_w \rightarrow C_{jk}$. By Eqs. (2), (3) and (5) we have

$$y_{ij} = \min \{z_{jk} - (d(w) - 2)\omega(c_k)k \neq i, j, \\
1 \leq k \leq 2d(w) + 1\}.$$  

(6)

Thus the values $y_{ij}$ can be computed from the values $z_{jk}$ by Eq. (6). Hence it suffices to compute the values $z_{jk}$ for all indices $j \leq \alpha$ and $k \leq 2d(w) + 1$ to construct the DP table for $e$. This can be reduced to a bipartite matching problem as follows.

Let $B_{jk} = (E_w, C_{jk}; E_w \times C_{jk})$ be a weighted complete bipartite graph with bipartite vertex sets $E_w$ and $C_{jk}$, and let the weight of edge $(e_u, c_k)$ of $B_{jk}$ be $\omega(T_{e_u}, c_k, c_i)$. Then $z_{jk}$ is equal to the weight of a minimum matching $M_{jk}$ of $B_{jk}$ saturating all vertices in $E_w$. Figure 5 depicts $B_{jk}$, where $M_{jk}$ is indicated by thick lines. Since $|E_w| + |C_{jk}| = d(w) - 1 + \alpha - 2 \leq 3\Delta$, such a matching $M_{jk}$ can be found in time $O(\Delta^3)$ by a standard matching algorithm [9]. Thus a value $z_{jk}$ for a pair of indices $j$ and $k$ can be computed in time $O(\Delta^3)$. Thus the values $z_{jk}$ for all pairs of indices $j \leq \alpha$ and $k \leq 2d(w) + 1$ can be computed in time $O(d(w)\Delta^4)$.

The complexity of $O(d(w)\Delta^4)$ above can be improved to $O(d(w)\Delta^3)$ as follows. Our idea is to compute the values $z_{jk}$ for a fixed $k$ and all $j \leq \alpha$ together. Let $C_k = C - \{c_k\}$, let $B_k = (E_w, C_k; E_w \times C_k)$ be a weighted complete bipartite graph with bipartite vertex sets $E_w$ and $C_k$, and let the weight of edge $(e_u, c_k)$ of $B_k$ be $\omega(T_{e_u}, c_k, c_i)$. Then $B_{jk}$ can be obtained from $B_k$ by deleting vertex $c_j$, that is, $B_{jk} = B_k - c_j$. Using a batch processing method of Kao et al. [4, 5], one can find all the matchings $M_{jk}, 1 \leq j \leq \alpha$, in time $O(\Delta^3)$ by finding a matching $M_k$ of $B_k$ and updating $M_k$ to $M_{jk}$. Thus one can compute the values $z_{jk}$ for a fixed $k$ and all $j \leq \alpha$ in time $O(\Delta^3)$. Hence one can compute the values $z_{jk}$ for all pairs of indices $j \leq \alpha$ and $k \leq 2d(w) + 1$ in time $O(d(w)\Delta^3)$.

From the values $z_{jk}$ one can compute by Eq. (6) the values $y_{ij}$ for all pairs of distinct indices $i \leq 2d(v) + 1$ and $j \leq \alpha$ in time $O(\Delta^3)$, and from the values $y_{ij}$ one can compute by Eq. (4) the values $\omega(T_{e_i}, c_i, c_j)$ for all pairs of distinct indices $i \leq 2d(v) + 1$ and $j \leq \alpha$ in time $O(\Delta^2)$. Thus we have the following lemma.

**Lemma 7:** The DP table for a non-leaf edge $e = (v, w)$ of $T$ can be constructed in time $O(d(w)\Delta^3)$.

By Lemmas 4 and 7 one can know that the DP tables for all edges of $T$ can be constructed in time $O(n\Delta^3)$, because

$$\sum_{e \in (\omega(v)\omega(e))} d(w)\Delta^3 = \sum_{w \in (v)\omega(w)\omega(e)} d(w)\Delta^3 \leq (n - 1)\Delta^3.$$  

Remember that $r$ is the root of $T$, and note that the end $w$ of edge $e = (v, w) \in E$ is a child of $v$ in a rooted tree $T$ and that $T$ has $n - 1$ edges.

**[Step 3]** Step 3 is to compute the minimum cost $\omega(T)$ of $T$ from the DP table for the edge $e_r$, incident to root $r$ by Eq. (1). (See Fig. 2.) This can be done in time $O(\alpha = O(\Delta))$.

We have thus showed that $\omega(T)$ can be computed in time $O(n\Delta^3)$. This completes the proof of Theorem 1.

### 3. Conclusion

In this paper we have shown that the cost total coloring problem can be solved in time $O(n\alpha)$ for trees, where $\alpha$ is the maximum degree. Therefore, the problem can be solved in linear time if $\alpha = O(1)$.

If all the color costs are integers in the range $[-N, N]$, then one can solve the problem in time $O(n\alpha\cdot 2.5 \log(nN))$ by using the matching algorithm in [2] and the batch processing method of bipartite matchings in [4, 5].

Suppose that the cost for coloring vertices is different from that for coloring edges, and hence there are two cost functions $\omega_1 : C \rightarrow \mathbb{R}$ for vertices and $\omega_2 : C \rightarrow \mathbb{R}$ for edges, and we wish to find a total coloring $f : V \cup E \rightarrow C$ of a given tree $T = (V, E)$ whose cost

$$\omega(f) = \sum_{v \in V} \omega_1(f(v)) + \sum_{e \in E} \omega_2(f(e))$$

is as small as possible. For this problem, Lemma 2 is modified as follows: any tree $T$ has an optimal total coloring such that any vertex receives one of the cheapest $2d(v) + 1$ colors in $C$ with respect to $\omega_1$, and any edge receives one of the

![Fig. 5](image-url)
is as small as possible. For this problem, one may assume that \( \Delta + 1 \leq \alpha \) and \( \alpha \leq (2\Delta + 1)(|V| + |E|) = O(n\Delta) \), and the problem can be solved in time \( O(n(\alpha + \Delta)^3) = O(n^3) \) for trees similarly as above. The list total coloring problem asks whether, given a graph \( G = (V, E) \) together with lists (sets) \( L(x) \) of colors for all elements \( x \in V \cup E \), there is a total coloring \( f \) such that \( f(x) \in L(x) \) for all elements \( x \in V \cup E \). The list total coloring problem can be formulated as a cost total coloring in which \( \omega_x(c) = \infty \) for color \( c \in C \) if \( c \notin L(x) \). Thus the list total coloring problem can be solved in polynomial time for trees. Similarly the list edge-coloring problem can be solved in polynomial time for trees.

Consider another type of a total coloring, which we call a “pseudo total coloring.” A pseudo total coloring of a graph \( G \) is to color all vertices and edges in \( G \) so that any two adjacent edges receive different colors, and any vertex receives a color different from the colors of all edges incident to it. Thus two adjacent vertices may receive the same color. Any simple graph has a pseudo total coloring with \( \Delta + 1 \) colors. Using the cost edge-coloring algorithm in [12] with a slight modification, one can find the cheapest pseudo total coloring of any tree in time \( O(n\Delta^2) \).

One of the remaining problems is to improve the time complexity \( O(n\Delta^3) \) of our algorithm, although it looks difficult if one reduces the problem to a matching problem as we do. Another problem is to show either that the cost total coloring problem can be solved in polynomial time for partial \( k \)-trees [1], i.e. graphs of bounded tree-width, or that the problem is NP-hard even for partial \( k \)-trees. It is known that both the edge-coloring problem [13], [14] and the total coloring problem (without cost) [3] can be solved for partial \( k \)-trees in linear time, but it looks difficult to extend the algorithms in [3], [13], [14] to an algorithm for the cost total coloring problem. We finally remark that the cost total coloring problem can be solved in linear time for partial \( k \)-trees if \( \Delta = O(1) \).

References


Takao Nishizeki received the B.E., M.E. and Dr. Eng. degrees from Tohoku University, Japan, in 1969, 1971 and 1974, respectively, all in electrical communication engineering. He joined Tohoku University in 1974, and is currently Professor of Graduate School of Information Sciences. His areas of research interest are combinatorial algorithms, graph theory and Cryptology. Dr. Nishizeki is a member of the Information Processing Society of Japan and the Japan Society for Industrial and Applied Mathematics, and is a Fellow of ACM and IEEE.