The Complexity of Tree Multicolorings

Dániel Marx*

Dept. of Computer Science and Information Theory,
Budapest University of Technology and Economics
dmarx@cs.bme.hu

Abstract. The multicoloring problem is that given a graph G and integer demands $x(v)$ for every vertex v, assign a set of $x(v)$ colors to vertex v, such that neighboring vertices have disjoint sets of colors. In the *preemptive sum multicoloring problem* the *finish time* of a vertex is defined to be the highest color assigned to it. The goal is to minimize the sum of the finish times. The study of this problem is motivated by applications in scheduling. Answering a question of Halldórsson et al. [4], we show that the problem is strongly **NP**-hard in binary trees. As a first step toward this result we prove that list multicoloring of binary trees is **NP**-complete.

1 Introduction

Graph multicoloring problems are often used to model scheduling of dependent jobs. Given a set of jobs, one has to assign a set of time slots to every job. The constraints are the following: every job has a length, which is the number of time slots it requires, and there are interfering pairs of jobs which cannot be active in the same time slot. In the *preemptive* scheduling model it is assumed that the jobs can be interrupted arbitrarily, the time slots assigned to a job do not have to be consecutive. This scheduling problem can be translated into a multicoloring problem on graphs as follows. The vertices of a graph correspond to the jobs and two jobs are connected if they cannot be executed at the same time. The colors correspond to the time slots and every vertex has a color requirement $x(v)$, which is the length of the job. In a multicoloring $x(v)$ colors have to be assigned to every vertex v such that neighboring vertices have disjoint sets of colors. Clearly, there is one to one correspondence between the feasible preemptive schedulings of the jobs and the feasible multicolorings of the graph.

One traditional optimization goal is to minimize the total completion time (makespan) of the scheduling, that is, the highest color assigned to the vertices (or, equivalently, the total number of different colors assigned). This problem is called *multicoloring* or *weighted coloring*. Another well-studied optimization goal is to minimize the average completion time of the jobs, which is the same as to minimize the sum of the completion times. This problem, *preemptive minimum sum multicoloring*, will be studied in this paper. It can be stated formally as follows:

* Research supported by grant OTKA 30122 of the Hungarian National Science Fund.

Preemptive Sum Multicoloring (pSMC)
Input: A graph $G(V, E)$ and a *demand function* x: $V \to \mathbb{N}$
Output: A *multicoloring* Ψ: $V \to 2^{\mathbb{N}}$ such that $|\Psi(v)| = x(v)$ for every $v \in V$, and $\Psi(u) \cap \Psi(v) = \emptyset$ if u and v are neighbors in G.
Goal: Let the *finish time* of vertex v in coloring Ψ be the highest color assigned to it, $f_\Psi(v) = \max\{i \in \Psi(v)\}$. The goal is to minimize $\sum_{v \in V} f_\Psi(v)$, the *sum* of the coloring Ψ.

If every demand is 1, i.e., $x(v) \equiv 1$, then we obtain the *chromatic sum* problem as a special case. The study of chromatic sums were started in [9,11,10]. The complexity and approximability of the chromatic sum in certain restricted classes of graphs were investigated in several papers [2,6,12,13].

Approximation results for arbitrary demand function $x(v)$ on general and k-colorable graphs were given by Bar-Noy et al. [1]. A polynomial time approximation scheme for preemptive minimum sum multicoloring is known for trees [4], for partial k-trees and planar graphs [3]. In [4] it is shown that the problem can be solved optimally in polynomial time in trees if every demand is bounded by a fixed constant. However, in general, the complexity of the problem in trees (and in paths) remained an open question. The main result of the paper is to show that the problem is **NP**-hard on binary trees, even if every demand is polynomially bounded. As a first step, we also prove the **NP**-completeness of another variant of multicoloring, the so-called list multicoloring.

In Section 2, we introduce some notations and present the result on list multicoloring. Section 3 defines penalty gadgets, which are the most important tools of the reduction in Section 4.

2 Preliminaries

We slightly extend the problem by allowing $x(v) = 0$. Clearly this does not make the problem more difficult, but it will be needed for technical reasons. If $x(v) = 0$, then define $f_\Psi(v) = 0$ in every coloring Ψ. Notice that by using this definition the trivial inequality $f_\Psi(v) \geq x(v)$ holds even if $x(v) = 0$.

Let us introduce some notations. If $V' \subseteq V$ and Ψ is a coloring then let $f_\Psi(V') = \sum_{v \in V'} f_\Psi(v)$. Similarly, $x(V') = \sum_{v \in V'} x(v)$. The sum of the optimum coloring of (G, x) is denoted by $\mathrm{OPT}(G, x)$, or by $\mathrm{OPT}(G)$ if the function $x(v)$ is clear from the context. The notation $[a, b]$ stands for the set $\{a, a + 1, \ldots, b\}$ if $a \leq b$, otherwise it is the empty set.

The size of the input to the multicoloring problem is the size of the graph, and it does not include the size of the demand function.

Instead of the preemptive sum multicoloring problem, we start with the **NP**-completeness of another multicoloring problem. The following is the obvious common generalization of list coloring and multicoloring (for a thorough overview on list coloring and related problems, see [14]):

List Multicoloring
Input: A graph $G(V, E)$, a *demand function* x: $V \to \mathbb{N}$, a set of colors C and a *color list* L: $V \to 2^C$ for each vertex
Question: Is there a *multicoloring* Ψ: $V \to 2^C$ such that $|\Psi(v)| = x(v)$, $\Psi(v) \subseteq L(v)$ for every $v \in V$, and $\Psi(u) \cap \Psi(v) = \emptyset$ if u and v are neighbors in G?

Clearly, this problem is **NP**-complete in every class of graphs where either multicoloring or list coloring is **NP**-complete. List coloring is **NP**-complete in bipartite graphs [5,8], but both problems can be solved in polynomial time in trees (see [7] for a linear time list coloring algorithm in trees). On the other hand, list multicoloring of trees is **NP**-complete:

Theorem 2.1. *The list multicoloring problem remains* **NP***-complete restricted to trees.*

Proof. The reduction is from the maximum independent set problem. For every graph $G(V, E)$ and integer k, we will construct a tree T (in fact, a star), a demand function, and a color list for each node, such that the tree can be colored with the lists if and only if G has an independent set of size k. The colors correspond to the vertices of G, the leaves of the star correspond to the edges of G. The construction will ensure that the colors given to the central node correspond to an independent set in G.

Let $e_1, e_2, \ldots, e_m$ be the edges of G and denote by $u_{i,1}$ and $u_{i,2}$ the two end vertices of edge e_i. The tree T is a star with a central node v and m leaves $v_1, \ldots, v_m$. The demand of v is k and the demand of every leaf is 1. The set of colors C corresponds to the vertex set V. The color list of the central node v is the set C, the list of node v_i is the set $\{u_{i,1}, u_{i,2}\}$.

Assume that there is a proper list coloring of T. It assigns k colors to v. The corresponding set of k vertices will be independent in G: at least one end vertex of each edge e_i is not contained in this set since node v_i must be colored with either $u_{i,1}$ or $u_{i,2}$. On the other hand, if there is an independent set of size k in G, then we can assign this k colors to v and extend the coloring to the nodes v_i: either $u_{i,1}$ or $u_{i,2}$ is not contained in the independent set, thus it can be assigned to v_i. □

There are two main difficulties in adapting these ideas for the minimum sum coloring problem.

- We want to prove **NP**-completeness in binary trees. The central node of the star has high degree.
- There are no lists in the minimum sum coloring problem. How can we forbid a node from using certain colors?

The first problem can be solved quite easily with a 'color copying' trick. To demonstrate this, we present a stronger form of Theorem 2.1:

Theorem 2.2. *The list multicoloring problem remains* **NP***-complete restricted to binary trees.*

Proof. The proof is essentially the same as in Theorem 2.1, but the degree m central node of the star is replaced by a path $v'_1, v'_2, \ldots, v'_{2m-1}$ of $2m-1$ nodes. The m neighbors of v are connected to the m nodes $v'_1, v'_3, \ldots, v'_{2m-1}$ one by one. The list of every node v'_i is C, the demands are $x(v'_{2i+1}) = k$ and $x(v'_{2i}) = |C| - k$. It is easy to see that in every proper multicoloring of the tree, the nodes $v'_1, v'_3, \ldots, v'_{2m-1}$ receive the same set of k colors. Furthermore, as in the previous proof, this set corresponds to an independent set in G.

□

To solve the second problem, certain 'penalty gadgets' will be constructed, Section 3 is devoted to this task.

3 The penalty gadgets

The goal of the penalty gadgets is that by connecting such a gadget to a node v, we can force v not to use certain colors: if node v uses a forbidden color, then the gadget can be colored only with a 'very large' penalty.

For offset t, demand size d and penalty C we define a tree $T_{t,d,C}$. The root r of this tree will be connected to some node v. When the root r of this tree uses the set $[t+1, t+d]$, then the tree can be colored optimally. On the other hand, if v uses even one color from $[t+1, t+d]$, then r cannot have the set $[t+1, t+d]$ and so $f_\Psi(T_{t,d,C}) \geq OPT(T_{t,d,C}, x) + C$. When C is sufficiently large, then this will force v to use colors not in $[t+1, t+d]$.

Proposition 3.1. *For integers $d, C > t \geq 0$ there is a binary tree $T_{t,d,C}$ and a demand function $x(v)$ such that*

1. *The root r has demand $x(r) = d$.*
2. *$\Psi(r) = [t+1, t+d]$ in every optimum coloring Ψ.*
3. *If $\Psi(r) \neq [t+1, t+d]$ for a coloring Ψ, then $f_\Psi(T_{t,d,C}) \geq OPT(T_{t,d,C}, x) + C$.*
4. *The demand x of every vertex is polynomially bounded by d and C.*

Furthermore, there is an algorithm which, given t, d and C, outputs the tree $T_{t,d,C}$, the demand function x and the value $OPT(T_{t,d,C}, x)$ in time polynomial in d and C.

Proof. Let $k = \lceil \log_2(C+t) \rceil$ and $\widehat{C} = 2^k$. Obviously, $C + t \leq \widehat{C} < 2(C+t)$. The tree $T_{t,d,C}$ consists of a complete binary tree and some attached paths. The complete binary tree T_0 has $k+1$ levels, the root r is on level 1 and the leaves, $\ell_1, \ell_2, \ldots, \ell_{\widehat{C}}$, are on level $k+1$. Attach a path of $k+3$ nodes to every leaf: node ℓ_i ($1 \leq i \leq \widehat{C}$) is connected to path P_i: $a_{i,k+2}, a_{i,k+1}, \ldots, a_{i,2}, a_{i,1}, a_{i,0}$ (nodes ℓ_i and $a_{i,k+2}$ are neighbors). Figure 1 shows the construction for $t = 2$, $d = 4$, $C = 6$. Clearly, $T_{t,d,C}$ has $2\widehat{C} - 1 + (k+3)\widehat{C}$ nodes, which is polynomially bounded in C.

We say that a node is of type j if it is either on the jth level of T_0 or it is an $a_{i,j}$ for some $1 \leq i \leq \widehat{C}$.

The demand $x(v)$ will depend only on the type of node v. Let

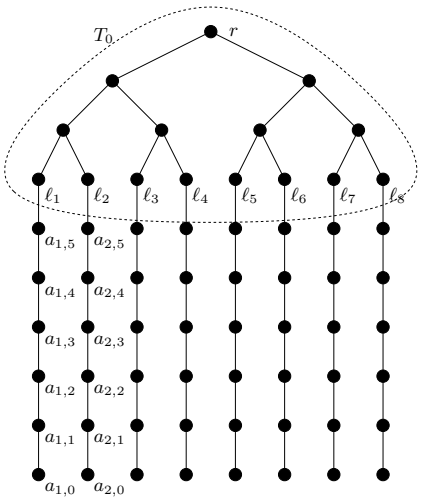
$$g(0) = t,$$
$$g(1) = d,$$
$$g(n) = (3d + t + C) \cdot 4^{n-2} \text{ for } n \geq 2.$$

Obviously, $g(n)$ is monotone and it is easy to see that

$$g(i+1) \geq 3g(i) + C + t \geq g(i-1) + C + t$$

for all $i \geq 1$ (these inequalities will be used later).

For a node v of type i let $x(v) = g(i)$. This implies that $x(r) = g(1) = d$ for the root r. The maximum value of $x(v)$ is $g(k+2) = (3d + t + C) \cdot 4^k$, which is bounded by a polynomial of d and C.



Type	Demand	Set of colors assigned by Ψ
1	4	$[3,6]$
2	20	$[1,2]$ $[7,24]$
3	80	$[3,6]$ $[25,100]$
4	320	$[1,2]$ $[7,24]$ $[101,400]$
5	1280	$[3,6]$ $[25,100]$ $[401,1600]$
4	320	$[1,2]$ $[7,24]$ $[101,400]$
3	80	$[3,6]$ $[25,100]$
2	20	$[1,2]$ $[7,24]$
1	4	$[3,6]$
0	2	$[1,2]$

Fig. 1. The tree $T_{t,d,C}$ for $t = 2$, $d = 4$ and $C = 6$. The nodes on the same level have the same type. On the right are the demands and also the colors assigned by the optimum coloring.

We describe a proper multicoloring Ψ, which will turn out to be the unique optimum solution. The same color set is assigned to the nodes of the same type. Start with $\Psi(v) = [1, t]$ for every node v of type 0. Then color the different types in increasing order: assign to the nodes of type i the first $g(i)$ colors not used by the type $i-1$ nodes. This gives a proper coloring since the already colored neighbors of type i nodes are type $i-1$ nodes. Notice that the root r receives the set $[t+1, t+d]$, as required. It is easy to prove that the finish time of a node v of type i is $f_\Psi(v) = g(i) + g(i-1) = x(i) + x(i-1)$ since there will be

exactly $g(i-1)$ 'skipped' colors and the finish time of nodes of type i is greater then the finish time of the nodes of type $i-1$ because $g(i) > g(i-2)$. The following simple observation will be used later: if u is a type i node and v is its type $i+1$ neighbor, then in every coloring Φ, the equalities $\Phi(u) = \Psi(u)$ and $f_\Phi(v) = f_\Psi(v) = g(i+1) + g(i)$ imply $\Phi(v) = \Psi(v)$. This follows directly from the definition of Ψ: there is just one way of choosing the first $x(v) = g(i+1)$ colors not used by u.

The following three lemmas show that Ψ is an optimum coloring, and if a coloring Φ assigns to r a set different from $\Psi(r) = [t+1, t+d]$, then $f_\Phi(T_{t,d,C}) \geq f_\Psi(T_{t,d,C}) + C$.

Lemma 3.2. *(a) $f_\Phi(T_0) \geq f_\Psi(T_0) - t$ holds for every coloring Φ of $(T_{t,d,C}, x)$.*
(b) If $\Phi(r) = \Psi(r)$, then $f_\Phi(T_0) \geq f_\Psi(T_0)$.
(c) If there is a $v \in T_0 \setminus \{r\}$ such that $f_\Phi(v) < f_\Psi(v)$, then $f_\Phi(T_0) \geq f_\Psi(T_0) + C$.

Proof. Let $L = \{v \in T_0 : f_\Phi(v) < f_\Psi(v)\}$ and let $H = T_0 \setminus L$ be its complement in T_0. We note that L is an independent set. To see this, let v and u be neighbors of type i and $i+1$, respectively. The sum of their demand is $g(i)+g(i+1)$, thus at least one of them must have finish time not smaller than $g(i) + g(i+1)$. Clearly this makes it impossible to have $f_\Phi(v) < f_\Psi(v) = g(i) + g(i-1)$ and $f_\Phi(u) < f_\Psi(u) = g(i) + g(i+1)$ simultaneously.

Partition the vertices of T_0 as follows. Define a subset S_v for every node $v \in H$. Let $v \in S_v$ for every $v \in H$, and $u \in L$ is in S_v iff v is the parent of u. When the root r is in L then r forms a set itself, $S^* = \{r\}$. It is clear that this defines a partition, every vertex is in exactly one subset. Apart from S^*, every subset contains a node from H and zero, one or two nodes from L.

Assume that the set S_v contains no node from L. Then $f_\Phi(S_v) \geq f_\Psi(S_v)$ follows from the definition of H and L. Now consider a set S_v which has at least one node from L. It contains a type i node v from H and one or two type $i+1$ nodes (u_1, u_2) from L. Since v and u_z $(z = 1, 2)$ are neighbors and the sum of their demand is $g(i)+g(i+1)$, at least one of them must have finish time at least $g(i)+g(i+1)$. Since u_z is in L, we have $f_\Phi(u_z) < f_\Psi(u_z) = g(i)+g(i+1)$, thus $f_\Phi(v) \geq g(i)+g(i+1)$. Therefore, $f_\Phi(v) - f_\Psi(v) \geq (g(i)+g(i+1)) - (g(i-1)+g(i)) = g(i+1) - g(i-1)$. Since $f_\Psi(u_z) = g(i+1) + g(i)$ and $x(u_z) = g(i+1)$, clearly $f_\Phi(u_z) - f_\Psi(u_z) \geq -g(i)$. Now

$$f_\Phi(S_v) - f_\Psi(S_v) \geq (g(i+1) - g(i-1)) - 2g(i) \geq g(i+1) - 3g(i) \geq C + t,$$

where the last inequality follows from $g(i+1) \geq 3g(i) + C + t$.

If r is in S^*, then $f_\Phi(S^*) = f_\Psi(S^*) + (f_\Phi(r) - f_\Psi(r))$ holds. Therefore $f_\Phi(T_0) \geq f_\Psi(T_0) + (f_\Phi(r) - f_\Psi(r)) \geq f_\Psi(T_0) - t$, since $f_\Phi(r) \geq d$. This proves statement (a), and (b) also follows because $\Phi(r) = \Psi(r)$ implies $f_\Phi(r) - f_\Psi(r) = 0$. Furthermore, if $f_\Phi(u) < f_\Psi(u)$ for some $u \in T_0 \setminus \{r\}$, then $f_\Phi(S_v) \geq f_\Psi(S_v) + C + t$ for the set S_v of the partition that contains u. This proves statement (c).

□

Lemma 3.3. *$f_\Phi(P_i) > f_\Psi(P_i)$ holds for every coloring $\Phi \neq \Psi$ of $T_{t,d,C}$ and for every $1 \leq i \leq \widehat{C}$.*

Proof. Assume that $f_\Phi(P_i) \leq f_\Psi(P_i)$, define $L = \{v \in P_i : f_\Phi(v) < f_\Psi(v)\}$ and $H = P_i \setminus L$. If $f_\Phi(P_i) \leq f_\Psi(P_i)$ and Φ is different from Ψ, then there is a $v \in P_i$ such that $f_\Phi(v) < f_\Psi(v)$, thus L is not empty. As in Lemma 3.2, it is easy to see that L is an independent set. The nodes of P_i are partitioned into $|H|$ classes: if $v \in H$ then v is in S_v, if $u \in L$ then u is in S_v, where v is the child of u. Notice that $a_{i,0} \in H$ since $f_\Psi(a_{i,0}) = x(a_{i,0}) = g(0) \leq f_\Phi(a_{i,0})$.

We prove that $f_\Phi(S_v) \geq f_\Psi(S_v)$ for every S_v. If $S_v = \{v\}$, then it is clear that $f_\Phi(S_v) \geq f_\Psi(S_v)$ holds. Assume that $S_v = \{u, v\}$, node $u \in L$ is type $j+1$, and $v \in H$ (its child) is type $j \geq 0$. The finish time of node v is at least $x(u) + x(v) = g(j+1) + g(j)$, therefore

$$f_\Phi(S_v) \geq x(u) + (x(u) + x(v)) = g(j+1) + (g(j+1) + g(j))$$

holds. On the other hand, if $j \geq 1$, then $f_\Psi(S_v) = (g(j+1) + g(j)) + (g(j) + g(j-1))$, thus $f_\Phi(S_v) > f_\Psi(S_v)$ follows from $g(j+1) > g(j) + g(j-1)$. In the case $j = 0$, we have $f_\Psi(S_v) = t + (t+d) = g(j) + (g(j) + g(j+1)) < f_\Phi(S_v)$, since $f_\Phi(S_v) \geq g(j+1) + (g(j) + g(j+1)) = d + (t+d)$ (recall that $t < d$). Since H is not empty, there is at least one subset S_v in the partition with $f_\Phi(S_v) > f_\Psi(S_v)$, contradicting $f_\Phi(P_i) \leq f_\Psi(P_i)$. □

Lemma 3.4. *If $\Phi(r) \neq \Psi(r) = [t+1, t+d]$, then $f_\Phi(T_{t,d,C}) \geq f_\Psi(T_{t,d,C}) + C$.*

Proof. Denote by $P^* = \bigcup_{i=1}^{\widehat{C}} P_i = T_{t,d,C} \setminus T_0$ the union of the paths. If there is a node $v \in T_0 \setminus \{r\}$ with $f_\Phi(v) < f_\Psi(v)$, then by part (c) of Lemma 3.2 $f_\Phi(T_0) \geq f_\Psi(T_0) + C$, and by Lemma 3.3 $f_\Phi(P^*) \geq f_\Psi(P^*)$ follows, which implies $f_\Phi(T_{t,d,C}) \geq f_\Psi(T_{t,d,C}) + C$, and we are ready. Therefore it can be assumed that $f_\Phi(v) \geq f_\Psi(v)$ for every node $v \in T_0 \setminus \{r\}$. Furthermore, if there is a $v \in T_0$ with $f_\Phi(v) \geq f_\Psi(v) + C + t$, then $f_\Phi(T_0) \geq f_\Psi(T_0) + C$, thus $f_\Phi(P^*) \geq f_\Psi(P^*)$ implies $f_\Phi(T_{t,d,C}) \geq f_\Psi(T_{t,d,C}) + C$. In the following, it will be assumed that $f_\Psi(v) \leq f_\Phi(v) \leq f_\Psi(v) + C + t$ holds for every $v \in T_0 \setminus \{r\}$.

Call a vertex v 'changed' in Φ if $\Phi(v) \neq \Psi(v)$. The goal is to show that if the root r is changed, then all the nodes $a_{1,k+2}, a_{2,k+2}, \ldots, a_{\widehat{C},k+2}$ are changed. Let v be a node of type i in T_0 and let u be one of its children, a node of type $i+1$. If v is changed, then there is a color $j \in \Phi(v)$ and $j \notin \Psi(v)$. We consider two cases. If $j \leq f_\Psi(u)$, then by the fact that $j \notin \Psi(v)$ and the way Ψ was defined $j \in \Psi(u)$ follows. Therefore u is also changed since $j \in \Phi(v)$ implies $j \notin \Phi(u)$. In the second case, where $j > f_\Psi(u) = g(i+1) + g(i)$ we have

$$\begin{aligned} f_\Phi(v) \geq j > g(i+1) + g(i) &= (g(i+1) - g(i-1)) + (g(i) + g(i-1)) \\ &= g(i+1) - g(i-1) + f_\Psi(v) \geq f_\Psi(v) + C + t, \end{aligned}$$

contradicting the assumption $f_\Phi(v) \leq f_\Psi(v) + C + t$.

Assume that $f_\Phi(T_{t,d,C}) < f_\Psi(T_{t,d,C}) + C$. By applying the previous result inductively, one finds that all the leaves ℓ_i and their children $a_{i,k+2}$ $(1 \leq i \leq \widehat{C})$ are changed. Lemma 3.3 ensures that Φ is not an optimum coloring of P_i, thus $f_\Phi(P_i) \geq f_\Psi(P_i) + 1$ and $f_\Phi(P^*) \geq f_\Psi(P^*) + \widehat{C} \geq f_\Psi(P^*) + C + t$. By Lemma 3.2, $f_\Phi(T_0) \geq f_\Psi(T_0) - t$, hence $f_\Phi(T_{t,d,C}) \geq f_\Psi(T_{t,d,C}) + C$. □

To prove Prop. 3.1, we have to show that requirements 2 and 3 hold. If $\Phi(r) = \Psi(r)$, then by part (b) of Lemma 3.2 and by Lemma 3.3, $f_\Phi(T_{t,d,C}) \geq f_\Psi(T_{t,d,C})$. If $\Phi(r) \neq \Psi(r)$, then by Lemma 3.4, $f_\Phi(T_{t,d,C}) \geq f_\Psi(T_{t,d,C}) + C$. Therefore the coloring Ψ is an optimum coloring and the tree satisfies the requirements of the proposition.

Clearly, the described tree $T_{t,d,C}$ and the demand function x can be constructed in polynomial time. The sum of the optimum solution can be also calculated, by adding the appropriate finish time of every node. □

4 The reduction

We will reduce the maximum independent set problem to the minimum sum coloring problem in binary trees. In the decision version of the minimum sum coloring problem, the input is a graph G, a demand function $x(v)$, and an integer K, the question is whether there exists a multicoloring Ψ with sum less than K. The reduction is based on the proof of Theorem 2.2. The penalty gadgets $T_{t,d,C}$ of Section 3 are used to imitate the effect of the color lists.

More precisely, the penalty gadget is used in two different ways: as a lower penalty gadget and as an upper penalty gadget. The *lower penalty gadget* $T^L_{d,C}$ is a tree $T_{0,d,C}$. By connecting the root of such a tree to a node v, the node v is forced to use only colors greater than d: otherwise the gadget can be colored only with a penalty C. A tree will be called a tree of type T^L if it is the tree $T^L_{d,C}$ for some d and C.

The *upper penalty gadget* $T^U_{d,C}$ is a tree $T_{d,C,C}$. If this gadget is connected to a node v, then this forces v to use only colors not greater than d. If v uses only colors not greater than d, then its finish time is at most d, and the gadget can be colored optimally. If v uses a color greater than d but not greater than $d+C$, then the gadget can be colored only with a penalty of C. If v uses colors greater than $d+C$, then it has finish time at least $d+C$, which is a penalty of at least C compared to the case when v uses only colors at most d.

Theorem 4.1. *The minimum sum preemptive multicoloring problem is* **NP***-complete on binary trees when the value of the demand function is polynomially bounded.*

Proof. Let a graph $G(V, E)$ and an integer k be given. Denote $n = |V|$, $m = |E|$ and let $C = 8mn$. Let integers $u_{i,1} < u_{i,2}$ denote the two end vertices of the ith edge in G.

We define a binary tree T, which consists of a core $\widehat{T}$ and some attached subtrees of type T^L and T^U. We start with a path of $2m-1$ nodes, $a_1, b_1, a_2, b_2, \ldots,$ a_{m-1}, b_{m-1}, a_m. Define $x(a_i) = k$ $(1 \leq i \leq m)$ and $x(b_i) = C + n - k$ $(1 \leq i \leq m-1)$. For every $1 \leq i \leq m$ attach a path of 6 nodes to a_i. Let these nodes be $c_{i,1}, d_{i,1}, c_{i,2}, d_{i,2}, c_{i,3}, d_{i,3}$. Let $x(c_{i,j}) = 1$, $x(d_{i,j}) = C+n-1$ $(j = 1, 2)$ and $x(c_{i,3}) = 1$, $x(d_{i,3}) = u_{i,2} - u_{i,1} - 1$. Clearly, $x(v) \geq 0$ for every node v. This completes the definition of $\widehat{T}$. Now attach trees of type T^L and T^U to $\widehat{T}$ as follows (see Figure 2):

– a $T^U_{C+n,2C}$ to every node b_i $(1 \le i \le m-1)$,
– a $T^U_{n,C}$ to the node a_1,
– a $T^U_{C+n,2C}$ to every node $d_{i,j}$ $(1 \le i \le m,\ j = 1, 2)$,
– a $T^U_{u_{i,2}+1,C}$ to every node $c_{i,1}$ $(1 \le i \le m)$,
– a $T^L_{u_{i,1}-1,C}$ to every node $c_{i,2}$ $(1 \le i \le m)$,
– a $T^L_{u_{i,1},C}$ and a $T^U_{u_{i,2}-1,C}$ to every node $d_{i,3}$ $(1 \le i \le m)$.

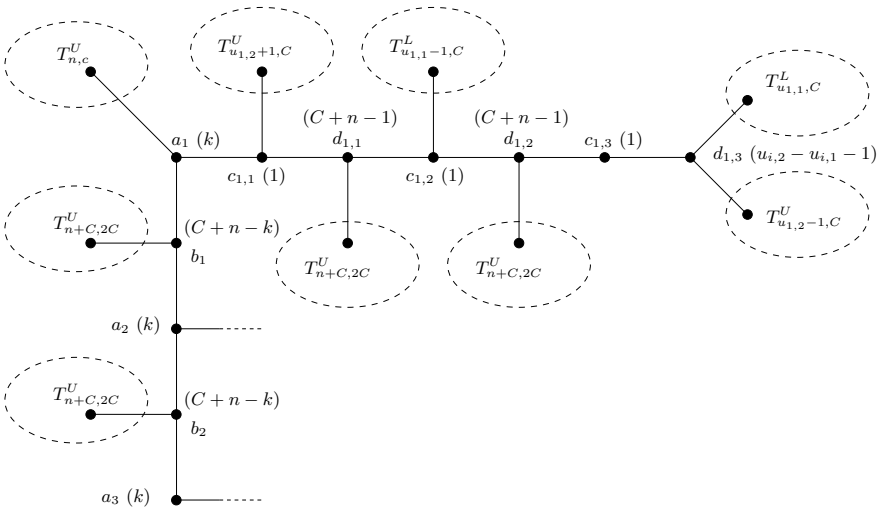


Fig. 2. The tree T for $m = 3$. For the sake of clarity, the nodes $c_{i,j}, d_{i,j}$ for $i \ge 2$ and the subtrees connected to these nodes are omitted. The numbers in parentheses are the demand of the vertices.

It is clear that the size of the resulting tree T is polynomial in n, the number of vertices of G, because $\widehat{T}$ has $8m-1$ nodes and we attach $7m$ trees to it, each of size bounded by a polynomial in $C+n$.

As required by Prop. 3.1, the algorithm that constructs the trees of type T^U and T^L also outputs the minimum sum of these $7m$ trees, that is, the value of $\mathrm{OPT}(T \setminus \widehat{T})$. Let $K = \mathrm{OPT}(T \setminus \widehat{T}) + x(\widehat{T}) + C$.

The intuition behind the construction is that in a 'well-behaved' solution, when the coloring of the T^L and T^U trees are optimal, for every i, the three nodes $c_{i,1}, c_{i,2}, c_{i,3}$ have the same color. The trees attached to these nodes ensure that this color must be either $u_{i,1}$ or $u_{i,2}$, one of the end nodes of the ith edge in G. This color cannot appear in a_i, this is the reason why the k colors assigned to the nodes a_i form an independent set, at least one end node of each edge is not in the set.

First we prove that if there is an independent set S of size k, then T can be colored with sum smaller than K. Let $\widehat{u}_i \in \{u_{i,1}, u_{i,2}\}$, $\widehat{u}_i \notin S$ be an end node of

the ith edge. Assume that Ψ colors all the trees of type T^U and T^L optimally, i.e., $f_\Psi(T \setminus \widehat{T}) = \mathrm{OPT}(T \setminus \widehat{T})$ and let

- $\Psi(a_i) = S$ $(1 \le i \le m)$,
- $\Psi(b_i) = [1, C+n] \setminus S$ $(1 \le i \le m-1)$,
- $\Psi(c_{i,j}) = \{\widehat{u}_i\}$ $(1 \le i \le m, j = 1,2,3)$,
- $\Psi(d_{i,j}) = [1, C+n] \setminus \{\widehat{u}_i\}$ $(1 \le i \le m,\ j = 1,2)$,
- $\Psi(d_{i,3}) = [u_{i,1}+1, u_{i,2}-1]$ $(1 \le i \le m)$.

It is straightforward to verify that Ψ is a proper coloring of T. Notice that $f_\Psi(v) \le x(v) + n$ holds for every node v of $\widehat{T}$, thus $f_\Psi(\widehat{T})$ can be bounded by $x(\widehat{T}) + |\widehat{T}|n$. Therefore $f_\Psi(T) = f_\Psi(T \setminus \widehat{T}) + f_\Psi(\widehat{T}) \le \mathrm{OPT}(T \setminus \widehat{T}) + x(\widehat{T}) + |\widehat{T}|n = \mathrm{OPT}(T \setminus \widehat{T}) + x(\widehat{T}) + (8m-1)n < \mathrm{OPT}(T \setminus \widehat{T}) + x(\widehat{T}) + C = K$, what we had to show.

To prove the other direction, we will show that when there is a coloring Ψ with sum $f_\Psi(T) < K$, then there is a set of k independent vertices in G. Obviously $f_\Psi(T) = f_\Psi(\widehat{T}) + f_\Psi(T - \widehat{T}) \ge x(\widehat{T}) + \mathrm{OPT}(T \setminus \widehat{T})$. If there is even one node $v \in \widehat{T}$ such that $f_\Psi(v) \ge x(v) + C$, then $f_\Psi(\widehat{T}) \ge x(\widehat{T}) + C$ and $f_\Psi(T) \ge \mathrm{OPT}(T \setminus \widehat{T}) + x(\widehat{T}) + C = K$. Thus it can be assumed that $f_\Psi(v) < x(v) + C$ for every $v \in \widehat{T}$. Now consider a tree T_v of type T^L or T^U attached to some node $v \in \widehat{T}$. If $f_\Psi(T_v) \ge \mathrm{OPT}(T_v) + C$, then $f_\Psi(T) \ge x(\widehat{T}) + \mathrm{OPT}(T \setminus \widehat{T}) + C = K$. Thus it can be assumed that $f_\Psi(T_v) < \mathrm{OPT}(T_v) + C$. Therefore, by the definition of T_v, if it is a $T^L_{d,C}$ (resp. $T^U_{d,C}$) tree, then Ψ assigns to its root the set $[1,d]$ (resp. $[d+1, d+C]$). Obviously, it follows that the node v cannot use the colors in this set.

By the argument in the previous paragraph, $f_\Psi(a_1) < x(a_1) + C \le n + C$ and $\Psi(a_1) \cap [n+1, n+C] = \emptyset$, which implies that $\Psi(a_1)$ contains only colors not greater than n. Similarly, $f_\Psi(b_1) < x(b_1) + C \le n + 2C$ and $\Psi(b_1) \cap [n+C+1, n+3C] = \emptyset$, which implies that the $n - k + C$ colors in $\Psi(b_1)$ are not greater than $n + C$. This set of colors must be disjoint from the k colors in $\Psi(a_1)$, therefore we have $\Psi(b_1) = [1, n+C] \setminus \Psi(a_1)$. Furthermore, $f_\Psi(a_2) < x(a_2) + C \le n + C$, hence it must use the k colors not used by b_1, therefore $\Psi(a_2) = \Psi(a_1)$. Continuing on this way, we get $\Psi(a_i) = \Psi(a_1) = S$ for all $2 \le i \le m$ and S contains k colors not greater than n.

Assume that the set S is not independent, that is, both end vertices of some edge of G is in this set, $u_{i,1}, u_{i,2} \in S$. From the assumption $f_\Psi(T) < K$ follows that $c_{i,1}$ cannot use either of these colors.

We have seen that $f_\Psi(c_{i,1}) < 1 + C$ and $\Psi(c_{i,1}) \cap [u_{i,2}+1, u_{i,2}+C] = \emptyset$ follow from the assumption $f_\Psi(T) < K$, which implies that the color of $c_{i,1}$ is at most $u_{i,2} \le n$. Moreover, since $f_\Psi(d_{i,1}) < 2C + n - 1$ and $\Psi(d_{i,1}) \cap [n+C+1, n+3C] = \emptyset$, thus node $d_{i,1}$ must use the first $C + n - 1$ colors missing from $c_{i,1}$, therefore we have $\Psi(d_{i,1}) = [1, C+n] \setminus \Psi(c_{i,1})$. Similarly as in the case of the nodes a_j and b_j, it follows that $\Psi(c_{i,1}) = \Psi(c_{i,2}) = \Psi(c_{i,3}) = \{u\}$. Furthermore, notice that $u \ge u_{i,1}$, since $c_{i,2}$ cannot use the colors below $u_{i,1}$: these colors are assigned to the root of the attached tree $T^L_{u_{i,1}-1,C}$. Similarly, u cannot be in $[u_{i,2}+1, u_{i,2}+C]$ since $c_{i,1}$ cannot use these colors. Finally, observe that $d_{i,3}$ must have the colors

$[u_{i,1}+1, u_{i,2}-1]$ which forbids $c_{i,3}$ from using a color between $u_{i,1}$ and $u_{i,2}$. Since u is a color not greater than C, thus it must be either $u_{i,1}$ or $u_{i,2}$.

If the demands are polynomially bounded, then the problem is obviously in **NP**: a proper coloring with the given sum is a polynomial size certificate, which finishes the proof of **NP**-completeness.

□

5 Acknowledgments

I'm grateful to Katalin Friedl for useful discussions and for helpful comments, which considerably improved the presentation of the paper. The comments of Judit Csima were also very valuable.

References

1. Amotz Bar-Noy, Magnús M. Halldórsson, Guy Kortsarz, Ravit Salman, and Hadas Shachnai. Sum multicoloring of graphs. *J. Algorithms*, 37(2):422–450, 2000.
2. Amotz Bar-Noy and Guy Kortsarz. Minimum color sum of bipartite graphs. *J. Algorithms*, 28(2):339–365, 1998.
3. Magnús M. Halldórsson and Guy Kortsarz. Multicoloring planar graphs and partial k-trees. In *Randomization, approximation, and combinatorial optimization (Berkeley, CA, 1999)*, pages 73–84. Springer, Berlin, 1999.
4. Magnús M. Halldórsson, Guy Kortsarz, Andrzej Proskurowski, Ravit Salman, Hadas Shachnai, and Jan Arne Telle. Multi-coloring trees. In *Computing and combinatorics (Tokyo, 1999)*, pages 271–280. Springer, Berlin, 1999.
5. M. Hujter and Zs. Tuza. Precoloring extension. II. Graph classes related to bipartite graphs. *Acta Mathematica Universitatis Comenianae*, 62(1):1–11, 1993.
6. Klaus Jansen. The optimum cost chromatic partition problem. In *Algorithms and complexity (Rome, 1997)*, pages 25–36. Springer, Berlin, 1997.
7. Klaus Jansen and Petra Scheffler. Generalized coloring for tree-like graphs. *Discrete Appl. Math.*, 75(2):135–155, 1997.
8. J. Kratochvíl. Precoloring extension with fixed color bound. *Acta Mathematica Universitatis Comenianae*, 62(2):139–153, 1993.
9. Ewa Kubicka. *The Chromatic Sum of a Graph.* PhD thesis, Western Michigan University, 1989.
10. Ewa Kubicka, Grzegorz Kubicki, and Dionisios Kountanis. Approximation algorithms for the chromatic sum. In *Computing in the 90's (Kalamazoo, MI, 1989)*, pages 15–21. Springer, Berlin, 1991.
11. Ewa Kubicka and Allan J. Schwenk. An introduction to chromatic sums. In *Proceedings of the ACM Computer Science Conf.*, pages 15–21. Springer, Berlin, 1989.
12. S. Nicoloso, M. Sarrafzadeh, and X. Song. On the sum coloring problem on interval graphs. *Algorithmica*, 23(2):109–126, 1999.
13. Tibor Szkaliczki. Routing with minimum wire length in the dogleg-free Manhattan model is NP-complete. *SIAM J. Comput.*, 29(1):274–287, 1999.
14. Zsolt Tuza. Graph colorings with local constraints—a survey. *Discuss. Math. Graph Theory*, 17(2):161–228, 1997.