Complexity of the Packing Coloring Problem of Trees[*]

Jiří Fiala[†] Petr A. Golovach[‡]

Abstract

Packing coloring is a partitioning of the vertex set of a graph with the property that vertices in the i-th class have pairwise distance greater than i. We solve an open problem of Goddard et al. and show that the decision whether a tree allows a packing coloring with at most k classes is NP-complete.

We accompany this negative result by a polynomial time algorithm for trees for closely related variant of the packing coloring problem where the lower bounds on the distances between vertices inside color classes are determined by an infinite nondecreasing sequence of bounded integers.

Keywords: Packing coloring, computational complexity, graph algorithm, chordal graph.

1 Introduction

The concept of packing coloring comes from the area of frequency planning in wireless networks. This model emphasizes the fact that some frequencies might be used more sparely than the others.

[*]The research was initiated at WFAP'07 — Second Workshop on Frequency Assignment Problems in Wireless Networks organized in September 23–27, 2007 at Sádek u Třebíče by DIMATIA and ITI, whose generous support of both authors is gratefully acknowledged.

[†]Department of Applied Mathematics and Institute for Theoretical Computer Science (ITI), Charles University, Prague, Czech Republic, email: `fiala@kam.mff.cuni.cz`. ITI is supported by the Ministry of Education of the Czech Republic as project 1M0021620808.

[‡]Department of Informatics, University of Bergen, 5020 Bergen, Norway, email: `petrg@ii.uib.no`. Supported by Norwegian Research Council.

In graph terms, we ask for a partitioning of the vertex set of a graph G into disjoint classes $X_1, \dots, X_k$ (representing frequency usage) according to the following constraints. Each color class X_i should be an *i-packing* i.e. a set of vertices with the property that any distinct pair $u, v \in X_i$ satisfies that $\mathrm{dist}(u, v) > i$. Here $\mathrm{dist}(u, v)$ is the *distance* between u and v, i.e. the length of some shortest path from u to v and it is declared to be infinite when u and v belong to distinct components of connectivity.

Such partitioning into k classes is called a *packing k-coloring*, even though it is allowed that some sets X_i can be empty. The smallest integer k for which exists a packing k-coloring of G is called the *packing chromatic number* of G, and it is denoted by $\chi_p(G)$. The notion of the packing chromatic number was established by Goddard et al. [7] under the name *broadcast chromatic number*. The term packing chromatic number was introduced by Brešar et al. [2].

Determining the packing chromatic number is difficult, even for special graph classes. For example, Sloper [11] showed that for trees of maximum degree three the the upper bound is seven, while χ_p is unbounded already on trees of maximum degree four. Goddard et al. [7] provided polynomial time algorithms for cographs and split graphs.

The packing chromatic number of the hexagonal grid is also seven as was shown by Brešar et al. [2] (the lower bound) and by Fiala and Lidický [personal communication] (the upper bound). Goddard et al. [7] also showed that the χ_p of the infinite two-dimensional square grid lies between 9 and 22. On the other hand, Finbow and Rall [10] proved that the packing chromatic numbers of the triangular infinite lattice as well as of the infinite three-dimensional square grid are unbounded.

The following decision problem arises naturally:

PACKING COLORING
Instance: A graph G and a positive integer k.
Question: Does G allow a packing k-coloring?

Goddard et al. [7] showed that the PACKING COLORING problem is NP-complete for general graphs and $k = 4$. They also asked about the computational complexity of this problem for trees. It was suggested by Brešar et al. [2] that the problem for trees can be difficult. Our main result is an affirmative proof of this conjecture:

Theorem 1. *The* PACKING COLORING *problem is* NP*-complete for trees.*

In contrary, the existence of a packing k-coloring can be expressed by a formula in Monadic Second Order Logic (MSOL), when k becomes fixed. It follows from work of Courcelle [3] that the PACKING COLORING problem is solvable in polynomial time for bounded treewidth graphs when k is fixed. In addition, we get the following corollaries for closely related graph classes:

Corollary 1. *The* PACKING COLORING *problem is fixed parameter tractable for chordal graphs with respect to the parameter k.*

Proof. If the given chordal graph G has a clique of size greater than k, then no packing k-coloring exists, since vertices of the clique have to be colored by distinct colors.

Otherwise, G has bounded clique size. Consequently it has also bounded treewidth and the result follows. □

Consequently, even in the case when k is not fixed, the PACKING COLORING problem becomes is easy for special tree-like graphs:

Corollary 2. *The* PACKING COLORING *problem is solvable in polynomial time for graphs of bounded treewidth and of bounded diameter.*

Proof. Let us consider the following maximization problem: for a given graph G we ask for some induced subgraph G' of G of maximum size that allows a packing d-coloring. By the results of Arnborg et al. [1] this problem can be solved by a linear algorithm on graphs of restricted treewidth for any fixed d, if the tree decomposition is given. (Also follows from a result of Courcelle et al. [4] on an adaptation of MSOL for optimization problems.)

Suppose that d is the upper bound on diameters of the considered graph class. If $k \leq d$ then the PACKING COLORING problem can be solved in polynomial time. Otherwise any color $c > d$ can be used on at most one vertex of G. Therefore, $\chi_p(G) = d + |V_G \setminus V_{G'}|$, where G' is an optimal solution of the auxiliary maximization problem. □

Finally, we focus our attention to more general concept of S-packing coloring introduced by Goddard et al. [7]. Let S be an infinite nondecreasing sequence of positive integers. In this new setting, vertices in the i-th class X_i are required to have distance greater than s_i. For example, the concept of the ordinary packing coloring is the S-packing coloring for $S = (1, 2, \dots)$. We address the following decision problem:

S-Packing Coloring *Parameter:* A nondecreasing sequence S. *Instance:* A graph G and a positive integer k. *Question:* Does G allow an S-packing coloring with at most k color classes?

We show that minimum number of color classes can be determined in polynomial time when the sequence S is bounded from above.

Theorem 2. *For nondecreasing sequences S with values bounded by a constant t the S-Packing Coloring problem can be solved for trees by an algorithm with running time $O(n^{2t+3})$.*

Our algorithm involves dynamic programming to evaluate all partial colorings for the initial classes with $s_i < t$ while minimizing the maximal number of uncolored vertices that are pairwise at distance at most t (i.e. the number of remaining color classes).

To our knowledge the machinery of MSOL developed by Courcelle et al. [4] cannot be used directly for this minimax optimization problem. Hence, we provide an explicit algorithm for the S-Packing Coloring problem to prove Theorem 2.

2 Proof of Theorem 1

For integers $a \leq b$ we define discrete intervals as $[a, b] := \{a, a+1, \ldots, b\}$.

2.1 Auxiliary constructions

We first construct a gadget where some vertices are forced predetermined colors in an arbitrary packing k-coloring.

Construction 1. Let $t \leq k$ be a positive integer. Construct a tree S_t with three levels as follows: The only vertex v_0 of the first level, called the *central* vertex, is of degree $t-1$, and all its neighbors $v_1, v_2, \ldots, v_{t-1}$ are of degree k. The vertices $v_0, v_1, \ldots, v_{t-1}$ are called the *inner* vertices of S_t.

Lemma 1. *For every packing k-coloring of S_t the inner vertices are colored by distinct colors. Also for every subset I of $[1, k]$ of size at least t, a packing k-coloring of S_t exists such that the inner vertices are colored by distinct colors from I.*

Proof. If a packing k-coloring of S_t exists, then none of vertices v_i, $i \in [1, t-1]$ is colored by color 1, since it would be impossible to find k distinct colors in $[2, k]$ to color the neighbors of v_i. Hence, the colors of all inner vertices are greater or equal to 2, and each may present at most once as the maximal distance on the inner vertices is two.

For the second claim we construct the packing k-coloring from I as follows: Use elements of I bijectively on the inner vertices with the rule that the central vertex is colored by 1, if it presents in I. All leaves in the third level are colored by the color 1. □

Given some tree S_t, choose one of its leaves arbitrarily and call it the *root of* S_t. To simplify some expressions we involve an auxiliary parameter $d := 28$.

Construction 2. For an odd $k > d$ and any $i \in [d+1, k]$ we construct the tree T_i as follows:

1. Take a copy of the tree S_i with the root u_3.
2. If $i < k$, then add a copy of S_k and join u_3 with the root of S_k by a path $u_3, u_4, \ldots, u_{k-3}$ of length $k-6$.
3. If $i < k-2$ then for each odd j such that $i < j < k$ we add two copies of the tree S_j. The root of one of the two copies of S_j, called the top copy, is joined by a path of length $\lceil \frac{j}{2} \rceil - 3$ to the vertex $u_{\lceil j/2 \rceil}$. The root of the other one, called the bottom copy, is joined to the same $u_{\lceil j/2 \rceil}$ by a path of length $\lceil \frac{j}{2} \rceil - 4$.
4. Finally, if $i < k-1$ and i is odd, we add an extra copy of S_{i+1} and join its root to $u_{\lceil \frac{i}{2} \rceil}$ by a path of length $\lceil \frac{i}{2} \rceil - 3$.

Denote by U the set of inner vertices of the copy of S_i in T_i. We choose the *root of* T_i as some leaf in the copy of S_i that is at distance four from u_3.

The construction of the tree T_{k-8} is depicted in Figure 1.

Lemma 2. *If i and k satisfy assumptions of Construction 2 then*

1. *the vertices of U are colored by different colors from the set $[1, i]$ in any packing k-coloring of T_i;*

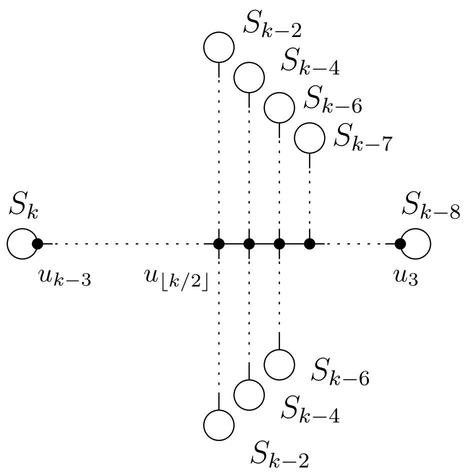

Figure 1: The tree T_{k-8}

2. *the tree T_i admits a packing k-coloring such that the vertices colored by a color $c \in [i+1, k]$ are at distance more than c from the root of T_i. Moreover, the root of T_i is colored by the color 1, and vertices colored by the colors $i, i-1, i-2$ are at distance at least three from the root.*

Proof. Assume first that a packing k-coloring of T_i is given. By Lemma 1 the inner vertices vertices of S_k are colored only by colors $[1, k]$, which proves the lemma in the case $i = k$.

We now determine the maximal distances between the inner vertices of used copies of S_j. They are summarized in the following table:

From	To	Max. distance
top S_j	S_k	$3+\lceil\frac{j}{2}\rceil-3+(k-3-\lceil\frac{j}{2}\rceil)+3=k$
	top $S'_j, j'>j$	$\lceil\frac{j}{2}\rceil+\frac{j'-j}{2}+\lceil\frac{j'}{2}\rceil=j'+1$
	bottom $S'_j, j'>j$	$\lceil\frac{j}{2}\rceil+\frac{j'-j}{2}+\lceil\frac{j'}{2}\rceil-1=j'$
bottom S_j	S_k	$\lceil\frac{j}{2}\rceil-1+(k-\lceil\frac{j}{2}\rceil)=k-1$
	top $S'_j, j'>j$	$\lceil\frac{j}{2}\rceil+\frac{j'-j}{2}+\lceil\frac{j'}{2}\rceil=j'+1$
	bottom $S'_j, j'>j$	$\lceil\frac{j}{2}\rceil+\frac{j'-j}{2}+\lceil\frac{j'}{2}\rceil-1=j'$
	top S_j	$\lceil\frac{j}{2}\rceil+\lceil\frac{j}{2}\rceil-1=j$
S_{i+1}	S_k	$\lceil\frac{i}{2}\rceil+(k-\lceil\frac{i}{2}\rceil)=k$
	top S_j	$\lceil\frac{i}{2}\rceil+\frac{j-i}{2}+\lceil\frac{j}{2}\rceil=j+1$
	bottom S_j	$\lceil\frac{i}{2}\rceil+\frac{j-i}{2}+\lceil\frac{j}{2}\rceil-1=j$
S_i	S_k	$3+k-6+3=k$
	top S_j	$\lceil\frac{j}{2}\rceil+\lceil\frac{j}{2}\rceil=j+1$
	bottom S_j	$\lceil\frac{j}{2}\rceil+\lceil\frac{j}{2}\rceil-1=j$
	S_{i+1}	$\lceil\frac{i}{2}\rceil+\lceil\frac{i}{2}\rceil=i+1$

On the bottom copy of S_j cannot be used any color $j'>j$ since they are used either on the top copies of $S_{j'-1}$ — for even j' — or on the bottom copies of $S_{j'}$ — for odd j'. The case of S_{k-2} has to be treated separately, but in this case applies distance $k-1$ to S_k. Hence, by Lemma 1 the interval $[1,j]$ is used on any bottom copy of S_j.

For the top copies of S_j holds an analogous argument with close distance to copies of $S_{j'}$ with $j'>j$. In addition, the color j is forbidden there, since it is used on the bottom copy of S_j and the distance is at most j. Again, by Lemma 1 the set $[1,j+1]\setminus\{j\}$ is used on the inner vertices of S_j.

The cases of S_{i+1} and S_i are treated in the same way.

Now we describe a packing k-coloring of T_i which satisfies the second claim. On each copy of S_j we use the coloring with colors from $[1,j]$, such that the center and all leaves are colored by the color 1, and the neighbor of the root of S_i is colored by 2. In addition, the neighbor of the root of T_i is colored by 3. We continue with the part of the tree around vertices $u_{\lfloor k/2\rfloor},\dots,u_{\lceil i/2\rceil}$. The periodic coloring pattern is depicted in Fig. 2 and uses only colors from the interval $[1,9]$. What remains yet uncolored are paths, each of length at least eight. Along these paths we use pattern $1,2,1,3,1,2,\dots$ with possible appearance of the color 4 so the path coloring fits well with the coloring determined so far. By careful observation of distances between inner sets of trees S_j one can verify that we get a valid packing k-coloring. Moreover, vertices colored by the color $j>i$ are at

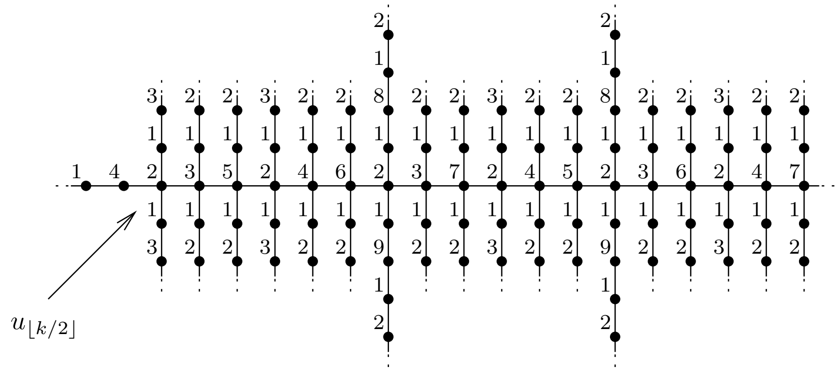

Figure 2: The periodic coloring pattern used for the central part of T_i

distance more than j from the root of T_i as it was required. □

Construction 3. Given $L \subset [d+1, k]$ of at most three elements we construct a tree T_L as follows. Take a copy of the tree S_{d+1}, and choose an inner vertex u arbitrarily. For every $j \in [d+1, k] \setminus L$ take an extra copy of the tree T_j and connect its root with a unique leaf neighbor of u by a path of length $j - 6$ (i.e., use different neighbors of u for different trees T_j).

The *root of* T_L is any leaf of S_{d+1} that is at distance three from u.

Lemma 3. *If L and k satisfy assumptions of Construction 3, then*

1. *for every packing k-coloring of T_L the inner vertices of S_{d+1} are colored by different colors from the set $[1, d] \cup L$;*

2. *for every packing k-coloring of T_L at least one inner vertex of S_{d+1} is colored by the color from the set L;*

3. *for every set $I \subset [1, d] \cup L$, $|I| = d + 1$, the tree T_L admits a packing k-coloring such that*

   - *the inner vertices of S_{d+1} are colored by the colors from I,*
   - *vertices colored by the color j for $j \in [d+1, k] \setminus I$ are at distance more than j from the root of T_L, and*
   - *vertices colored by the colors from L are 3-distant from the root.*

Proof. The first two claims follow immediately from the Lemmas 2 and 1.

For the proof of the third claim we construct the required packing k-coloring of T_L as follows: All vertices adjacent to the inner vertices of S_{d+1} are colored by the color 1. If $1 \in I$ then the central vertex of S_{d+1} is also colored by 1 as well. Then $t = |L|$ inner vertices of S_{d+1} which are different from the central vertex and from u, and that are not adjacent to the root, are chosen and colored by the colors from L. The remaining inner vertices of S_{d+1} are colored by the remaining colors from I. The vertices of trees T_j are colored according to the second claim of Lemma 2. Finally, every path between the root of some T_j and u is colored by colors 1, 2, 3, with possible one appearance of the color 4. □

2.2 Polynomial reduction

We proceed with reduction of the well known NP-complete 3-Satisfiability problem [6, problem L02, page 259] to our Packing Coloring problem for trees.

Let Φ be a boolean formula in conjunctive normal form with variables $x_1, x_2, \ldots, x_n$ and clauses $c_1, c_2, \ldots, c_m$. Each clause consists of three literals. We choose $k := 4n + 2d - 1$ and $r := 2(d + n - 1)$. For every variable x_i we define the set $X_i := \{2i + r, 2i + r + 1\}$.

For every clause c_j a three element set $C_j \subset [1, k]$ is constructed as follows: If the clause c_j contains the literal x_i then the integer $2i + r$ is included to the set C_j. On the other hand, if $\overline{x}_i \in c_j$ then $2i + r + 1 \in C_j$.

Construction 4. We construct the final tree T_Φ from the disjoint union of trees T_{X_i} over all variables x_i together with trees T_{C_j} over all clauses c_j. In addition we insert an extra new vertex u and join it to the roots of trees $T_{X_1}, T_{X_2}, \ldots, T_{X_n}$ by paths of length $d - 3$. We also join u with the roots of $T_{C_1}, T_{C_2}, \ldots, T_{C_m}$ by paths of length $\lceil \frac{k}{2} \rceil - 3$.

Lemma 4. *The tree T_Φ has a packing k-coloring if and only if the formula Φ can be satisfied.*

Proof. Suppose that a packing k-coloring of T_Φ exists. According to the second claim of Lemma 3 at least one element of the set X_i is used for among colors of the inner vertices of S_{d+1} in any T_{X_i} (in the sequel we denote this set of inner vertices by U_i). If the color $2i + t$ is used then we set $x_i := \mathsf{false}$, and $x_i := \mathsf{true}$ otherwise.

For every $j \in [1, m]$ at least one color $c \in C_j$ is used on an inner vertex of S_{d+1} in T_{C_i} (this set we denote by W_j). Suppose that $c = 2i + t$ for some

$i \in [1,n]$. Then the clause C_j contains the literal x_i. Since vertices of W_j and U_i are at distance at most $d + \lceil \frac{k}{2} \rceil = 2n + 2d \leq 2i + 2(d+n-1) = 2i + r$, the color $2i + r$ is not on the set U_i, and the variable x_i has to be assigned true.

Analogously, if $c = 2i + r + 1$ for some $i \in [1,n]$, then the clause c_j contains literal $\overline{x}_i$. By the same arguments as before, the color $2i + r + 1$ is not used on U_i, and $x_i = \mathsf{false}$.

Assume that a satisfying assignment of variables $x_1, x_2, \ldots, x_n$ for the formula Φ exists. For every $i \in [1,n]$ vertices of On any tree T_{X_i} we use the coloring described in the third statement of Lemma 3 arranged such that the vertices of U_i are colored by the set $[1,d] \cup \{2i+r+1\}$ if $x_i = \mathsf{true}$, and by the set $[1,d] \cup \{2i + r\}$ in the case when $x_i = \mathsf{false}$.

Note that the distance between different sets U_i is least $2d - 4$. Also, if some color $c \in [d+1,k]$ is used among the sets U_i then it is used only for a single vertex in a single set. Suppose that given clause C_j is satisfied by positively evaluated literal $x_i = \mathsf{true}$. Then the vertices of W_j are colored by the colors of the set $[1,d] \cup \{2i+r\}$ as described in Lemma 3. If C_j is satisfied by a literal $\overline{x}_i = true$, then vertices of W_j are colored by $[1,d] \cup \{2i+r+1\}$.

The distance between different sets W_i is least $2\lceil \frac{k}{2} \rceil - 4$, and by Lemma 3 vertices of different sets W_j which are colored by the colors from $[d+1,k]$ are at distance $2\lceil \frac{k}{2} \rceil > k = 4n + 2d - 1 = 2n + 1 + r \geq 2i + 1 + r$ for any $i \in [1,n]$. Also if a color $c \in [d+1,,k]$ is used for coloring of vertices of W_j then it can not be used on any set U_i.

Finally, we complete the packing k-coloring of T_Φ on the vertex u and the vertices from the paths between u and trees T_{X_i} and T_{C_j}. We proceed similarly as in the previous constructions — color u by 4, and use pattern $1, 2, 1, 3, 1, 2, \ldots$ on the paths, with possible one more appearance of the color 4, if necessary. □

Since trees S_i have $O(k^2)$ vertices, and trees T_i have $O(k^3)$ vertices, the final tree T_Φ has $O(n^4(n+m))$ vertices. Hence our reduction is polynomial and the proof Theorem 1 is finished.

3 Proof of Theorem 2

Without loss of generality assume that s_r is the last element of S smaller than t. For every $k \leq r$ the S-packing coloring problem can be solved polynomially for trees (and for graphs of restricted treewidth), e.g., by the machinery of MSOL.

We construct a dynamic programming algorithm under assumptions that $k > r$ and $s_2 > 1$. (If $s_2 = 1$ then two color classes always suffices for any tree; one class can be used only if the tree has only one vertex.)

Assume that T is a rooted tree on n vertices. If W is a subset of children of some node v then we denote by $T_{v,W}$ subtree of T rooted in v and containing all vertices from W together with all their descendants.

For a tree $T_{v,W}$ we explore all its partial $(s_1, \dots, s_r)$-packing colorings with respect to the following parameters:

- the distances d_i between the root v and the closest vertex from the i-th class for every $i \in [1, r]$; it's only essential to know the distance only when $d_i \le s_i$,
- the numbers p_j of uncolored vertices that are at distance at most $j \le t$ from v for every $j \in [0, t]$.

Formally, we encode these two sets of parameters by sequences $D = (d_1, d_2, \dots, d_{r-1})$ such that $d_i \in [0, s_i] \cup \{\infty\}$, and $P = (p_0, p_1, \dots, p_t)$ such that $0 \le p_1 \le p_2 \le \dots \le n$.

Among those partial colorings that provide the same parameters we identify the maximal number of uncolored vertices that are pairwise at distance smaller than t and choose the coloring that minimizes this value. In particular, our algorithm computes for each triple $T_{v,W}, D, P$ the minimal integer $c(T_{v,W}, D, P)$ such that there is a partition of $V(T_{v,W})$ into sets $X_1, \dots, X_r, Y$ for which the following conditions are fulfilled:

- for every $i \in [1, r]$ the set X_i is an S_i packing in $T_{v,W}$,
- for every $i \in [1, r] : d_i = \min\{\mathrm{dist}(v, z) \colon z \in X_i, \mathrm{dist}(v, z) \le s_i\}$;
  it is assumed that $d_i = \infty$ if no such z exists,
- for every $j \in [0, t] : p_j = |\{z \in Y : \mathrm{dist}(v, z) \le j\}|$,
- for every $Z \subset Y$, satisfying $u, v \in Z : \mathrm{dist}(u, v) \le t$, holds that $|Z| \le c(T_{v,W}, D, P)$.

If no such partition exists then we define $c(T_{v,W}, D, P) = \infty$.

The sequences D are used to properly extend partial packing colorings, while sequences P allow us to determine the maximum size of the set Z. In other words Z induces a clique in the t-th power of T. (In the t-th power vertices are adjacent if and only if they are at distance at most t in the original graph). Here we strongly rely on the well known fact that powers of trees are chordal [8, 9], and their chromatic numbers are equal to the size of their maximum clique.

The algorithm consists from three subroutines. The first subroutine `Leaf` is called if T_v has only one vertex v (i.e. v is a leaf of T).

The subroutine `NewRoot` is called for a vertex v with a child w, and it computes $c(T_{v,\{w\}}, D, P)$ from the values of $c(T_{w,N(w)}, D', P')$. Here $N(w)$ stands for the set of children of w.

The last subroutine `Join` is called for vertices of T which are not leaves. It computes from the tables of values $c(T_{v,W_i}, D_i, P_i)$ for two subtrees T_{v,W_1} and T_{v,W_2} with a unique common vertex v, which is the root of the trees, the value of $c(T_{v,W}, D, P)$ for the union of these trees $T_{v,W}$, where $W = W_1 \cup W_2$.

Our algorithm starts from leaves of the tree T and constructs for them tables of values $c(T_{v,\emptyset}, D, P)$ by the subroutine `Leaf`. If v is not a leaf then we use the subroutine `NewRoot` if it has only one child. If v has more children $w_1, w_2, \ldots, w_l$ then the subroutine `NewRoot` is used for the construction of auxiliary tables for values $c(T_{v,\{w_i\}}, D, P)$ for all $i \in [1, l]$. Then we use the subroutine `Join` and construct consecutively tables for trees $T_{v,\{w_1,w_2,\ldots,w_i\}}$ for $i = 2, 3, \ldots, l$. Finally, table is constructed for the root u. Then if there are D and P for which $c(T_{u,N(u)}, D, P) + r > k$ then the tree T allows an S-packing k-coloring. Otherwise no such coloring exists.

Due the space restrictions the explicit description of these routines is given in the appendix.

Now we estimate the time complexity. Since the sequence S is fixed, there is a constant number of sequences D. There are $O(n^{t+1})$ sequences P and all such sequences can be listed in time $O(n^{t+2})$. Note that we have to list these sequences only once. The construction of the tables with all values $c(T_{v,\emptyset}, D, P)$ for leaves of T by the subroutine `Leaf` demands $O(n^{t+2})$ operations, since T has no more than n leaves. Each call of the subroutine `NewRoot` takes $O(n)$ operations. Since we use this subroutine for every edge of T, the total number of operations is $O(n^{t+2})$. At every call of the subroutine `Join` all possible sequences P_1, P_2, D_1 and D_2 are considered. For any sequence P there are $O(n^{t+1})$ such sequences, and all such sequences for all P can be listed in time n^{2t+3}. We can use this table of sequences for the all calls of the subroutine. Then every call of the subroutine demands $O(n^{t+1})$ operations, and the table of all values of $c(T_{v,W}, D, P)$ can be constructed in time $O(n^{2t+2})$. The total number of such tables is no more than the number of edges of T. Correspondingly, the total number of operations is $O(n^{2t+3})$.

4 Conclusion and open problems

We have shown that for bounded sequences the S-Packing Coloring problem is solvable in polynomial time for trees. On the other hand, Theorem 1 shows that it is NP-complete for the sequence $(1, 2, 3, \dots)$. It would be interesting to classify computational complexity of the S-Packing Coloring problem for different sequences. It can be easily seen that the proof of Theorem 1 can be extended for sequences $S = (s_1, s_2, s_3, \dots)$ of different positive integers such that $s_i = \Theta(i^c)$ for some constant c.

Another interesting question is the complexity of the S-Packing Coloring problem for bounded sequences S and graphs of restricted treewidth. The fact that powers of trees are chordal graphs does not apply in this case. But it is known [12] that if $S = (s, s, s, \dots)$ for some constant s then the problem can be solved polynomially for the graphs of restricted treewidth. On the other hand, the S-Packing Coloring problem might belong to the family of coloring problems that allow an polynomial time algorithm for trees, but that are NP-complete for graphs of restricted treewidth — see our paper [5] for an example.

As a consequence of a result of Sloper [11] and Corollary 1 the packing chromatic number can be computed polynomially for trees of maximum degree three. This raises the question whether χ_p can be determined efficiently for bounded degree trees.

References

[1] S. Arnborg, J. Lagergren, and D. Seese, *Easy problems for tree-decomposable graphs*, J. Algorithms, 12 (1991), pp. 308–340.

[2] B. Brešar, S. Klavžar, and D. F. Rall, *On the packing chromatic number of cartesian products, hexagonal lattice, and trees*, Discrete Applied Mathematics, 155 (2007), pp. 2303–2311.

[3] B. Courcelle, *The monadic second-order logic of graphs iii: tree-decompositions, minor and complexity issues*, ITA, 26 (1992), pp. 257–286.

[4] B. Courcelle, J. Makowsky, and U. Rotics, *Linear time solvable optimization problems on graphs of bounded clique width*, in WG, J. Hromkovic and O. Sýkora, eds., vol. 1517 of Lecture Notes in Computer Science, Springer, 1998, pp. 1–16.

[5] J. Fiala, P. A. Golovach, and J. Kratochvíl, *Distance constrained labelings of graphs of bounded treewidth*, in ICALP, L. Caires, G. F. Italiano, L. Monteiro, C. Palamidessi, and M. Yung, eds., vol. 3580 of Lecture Notes in Computer Science, Springer, 2005, pp. 360–372.

[6] M. Garey and D. Johnson, *Computers and Intractability. A Guide to the Theory of NP-Completeness*, W.H. Freeman and Company, 1979.

[7] W. Goddard, S. M. Hedetniemi, S. T. Hedetniemi, J. M. Harris, and D. F. Rall, *Broadcast chromatic numbers of graphs*, Ars Combinatoria, 86 (2008).

[8] P. E. Kearney and D. G. Corneil, *Tree powers*, J. Algorithms, 29 (1998), pp. 111–131.

[9] Y. L. Lin and S. S. Skiena, *Algorithms for square roots of graphs*, SIAM J. Discrete Math., 8 (1995), pp. 99–118.

[10] D. F. Rall and A. Finbow, *On the packing chromatic number of some infinite graphs.* manuscript, 2007.

[11] C. Sloper, *An eccentric coloring of trees*, Australas. J. Combin., 29 (2004), pp. 309–321.

[12] T. Zhou, Y. Kanari, and T. Nishizeki, *Generalized vertex-coloring of partial k-trees*, IEICE Trans. on Fundamentals of Electronics, Communication and Computer Sciences, E83-A (2000), pp. 671–678.

A Subroutines

```
Subroutine Leaf(T_v, D, P);
if d_1 = d_2 = ··· = d_r = ∞ and p_0 = p_1 = ··· = p_t = 1 then
    c(T_{v,∅}, D, P) := 1;
else
    if ∃i ∈ [1, r] such that d_i = 0 and d_j = ∞ for all j ∈ [1, r] \ {i},
        and p_0 = p_1 = ··· = p_t = 0 then
        c(T_{v,∅}, D, P) := 0;
    else
        c(T_{v,∅}, D, P) := ∞;
Return c(T_{v,∅}, D, P)
```

```
Subroutine NewRoot(T_{v,{w}}, D, P);
c(T_{v,{w}}, D, P) := ∞;
if ( ∃i ∈ [1, r] such that d_i = 0 and d_j > 0 for all j ∈ [1, r] \ {i}, and
p_0 = 0 )
    or ( d_j > 0 for all j ∈ [1, r] and p_0 = 1 ) then
    for j := 1 to t do
        p'_{j-1} := p_j;
    J := {j ∈ [1, r]: d_j = 0 or d_j = ∞};
    forall j ∈ [1, r] \ J do
        d'_j := d_j − 1;
    for p'_t := p'_{t-1} to n do
        P' := (p'_0, p'_1, ..., p'_t);
        for every choice d'_j ∈ {s_j, ∞} for all j ∈ J do
            D' := (d'_1, d'_2, ..., d'_r);
            if c(T_{v,{w}}, D, P) > c(T_{w,N(w)}, D', P') then
                c(T_{v,{w}}, D, P) := c(T_{w,N(w)}, D', P');
Return c(T_{v,{w}}, D, P)
```

```
Subroutine Join(T_{v,W_1}, T_{v,W_2}, D, P);
c(T_{v,W}, D, P) := ∞;
if ∃i ∈ [1,r] such that d_i = 0 and d_j > 0 for all j ∈ [1,r] \ {i}, and
p_0 = 0 then
    forall P_1 := (p_0^(1), p_1^(1), ..., p_t^(1)) and P_2 := (p_0^(2), p_1^(2), ..., p_t^(2))
        such that p_j^(1) + p_j^(2) = p_j for all j ∈ [0,t] do
        forall D_1 := (d_1^(1), d_2^(1), ..., d_r^(1)) and
        D_2 := (d_1^(2), d_2^(2), ..., d_r^(2))
            such that d_i = min{d_j^(1), d_j^(2)} for all j ∈ [1,r],
            and d_j^(1) + d_j^(2) > s_j for all j ∈ [1,r] \ {i} do
            m := max{c(T_{v,W_1}, D_1, P_1), c(T_{v,W_2}, D_2, P_2)};
            for j := 0 to t do
                if m < p_j^(1) + p_{t-j}^(2) then m := p_j^(1) + p_{t-j}^(2);
            if c(T_{v,W}, D, P) > m then c(T_{v,W}, D, P) := m;
if d_i > 0 for all i ∈ [1,r] and p_0 = 1 then
    forall P_1 := (p_0^(1), p_1^(1), ..., p_t^(1)) and P_2 := (p_0^(2), p_1^(2), ..., p_t^(2))
        such that p_j^(1) + p_j^(2) = p_j for all j ∈ [1,t] and p_0^(1) = p_0^(2) = 1
    do
        forall D_1 := (d_1^(1), d_2^(1), ..., d_r^(1)) and
        D_2 := (d_1^(2), d_2^(2), ..., d_r^(2))
            such that d_i = min{d_j^(1), d_j^(2)} and d_j^(1) + d_j^(2) > s_j for all
        j ∈ [1,r] do
            m := max{c(T_{v,W_1}, D_1, P_1), c(T_{v,W_2}, D_2, P_2)};
            for j := 0 to t do
                if m < p_j^(1) + p_{t-j}^(2) - 1 then m := p_j^(1) + p_{t-j}^(2) - 1;
            if c(T_{v,W}, D, P) > m then c(T_{v,W}, D, P) := m
Return c(T_{v,W}, D, P)
```