

Multi-Coloring Trees

Magnús M. Halldórsson*

Guy Kortsarz†

Andrzej Proskurowski ‡

Ravit Salman§

Hadas Shachnai¶

Jan Arne Telle ||

Abstract

Scheduling jobs with pairwise conflicts is modeled by the graph *multi-coloring* problem. It occurs in two versions: preemptive and non-preemptive. We study these problems on trees under the sum-of-completion-times objective. In particular, we give a quadratic algorithm for the non-preemptive case, and a faster algorithm in the case that all job lengths are short, while we present a polynomial-time approximation scheme for the preemptive case.

1 Introduction

In many real-life situations, *non-sharable* resources need to be shared among users with *conflicting* requirements. This includes traffic intersection control [B92], frequency assignment to mobile phone users [DO85, Y73], and session management in local area networks [CCO93]. Each user can be identified with a *job*, the execution of which involves an exclusive use of some resource, in a given period of time. Indeed, scheduling such jobs, with pairwise conflicts, is a fundamental problem – in the above areas, as well as in distributed computing [L81, SP88].

The problem of scheduling dependent jobs is modeled as a graph *coloring* problem, when all jobs have the same (unit) execution times, and as graph *multi-coloring* for arbitrary execution times. The vertices of the graph represent the jobs and an edge in the graph between two vertices represents a dependency between the two corresponding jobs, which forbids scheduling these jobs at the same time. More formally, for a weighted undirected simple graph $G = (V, E)$ with n vertices, let the *length* of a vertex v be a positive integer denoted by $x(v)$ and called the *color requirement* of v . A multi-coloring of the vertices of G is a mapping into the power set of the positive integers, $\Psi : V \mapsto 2^N$, such that $|\Psi(v)| = x(v)$ and adjacent vertices receive non-intersecting sets of colors.

The traditional optimization goal is to minimize the total number of colors assigned to G . In the setting of a job system, this is equivalent to finding a schedule, in which the time when *all* the jobs have been completed is minimized. Such an optimization goal favors the system. However, from the point of view of the jobs themselves, an important goal is to minimize the average completion time of the jobs (or equivalently, the sum of the completion times). This optimization goal is the concern of this paper. Formally, in the *sum multi-coloring* (SMC) problem, we look for a multi-coloring Ψ that minimizes $\sum_{v \in V} f_\Psi(v)$, where $f_\Psi(v)$ is the largest color assigned to v by Ψ . This reduces to the *sum coloring* problem in the case of unit color requirements.

There are two variants of the sum multi-coloring problem. In the *preemptive* (p-SMC) problem, each vertex may get any set of colors, while in the *non-preemptive* (np-SMC) problem, the set

*Science Institute, University of Iceland, IS-107 Reykjavik, Iceland. E-mail: mmh@hi.is.

†Dept. of Computer Science, Open University, Ramat Aviv, Israel. E-mail: guyk@tavor.openu.ac.il.

‡Dept. of Computer Science, University of Oregon, Eugene, Oregon. E-mail: andrzej@cs.uoregon.edu.

§Dept. of Mathematics, Technion, Haifa 32000, Israel. E-mail: maravit@tx.technion.ac.il.

¶Dept. of Computer Science, Technion, Haifa 32000, Israel. E-mail: hadas@cs.technion.ac.il.

||Dept. of Informatics, University of Bergen, Bergen, Norway. E-mail: telle@ii.uib.no.

of colors assigned to each vertex has to be contiguous. The preemptive version corresponds to the scheduling approach commonly used in modern operating systems [SG98], where jobs may be interrupted during their execution and resumed at a later time. The non-preemptive version captures the execution model adopted in real-time systems, where scheduled jobs must run to completion.

In the current paper we study the sum multi-coloring problems on trees. Given the hardness of these problems on general graphs (see below), it is natural to seek out classes of graphs where effective solutions can be obtained efficiently. Trees constitute the boundary of what we know to be efficiently solvable, and represent perhaps the most frequently naturally occurring class of graphs.

A natural application, in which the resulting conflict graph is a tree, is packet routing on a tree network topology: each node can conflict over its neighboring links, either with its parent or children in the tree. Thus, the conflict graph is induced by the network topology. Conflicts among processes running on a single-user machine (e.g. PCs) are typically for shared data. In many operating systems, the creation of a new process is done by ‘splitting’ an existing process, via a ‘fork’ system call [SG98]. Thus, the set of processes form a tree where each process is a node. Conflicts over shared data typically occur between a process and its immediate descendents/ancestor in that tree, as these processes will share parts of their codes. Thus, the conflict graph is also a tree.

1.1 Related work

The sum multi-coloring problem was introduced by Bar-Noy et al. [BKH⁺98]. They presented a comprehensive study of the approximability of both the p-SMC and the np-SMC problems, on general and special classes of graphs. We are not aware of other published work on this problem.

The sum coloring problem was introduced by Kubicka [K89], who gave a polynomial algorithm for trees. Jansen [J97] extended the dynamic programming strategy to graphs of bounded treewidth. Also, the sum-coloring problem on general graphs was shown to be hard to approximate within $n^{1-\epsilon}$, for any $\epsilon > 0$ unless $NP = ZPP$ [FK96, BBH⁺98], as well as hard to approximate within some factor $c > 1$ on bipartite graphs [BK98]. These hardness results also carry over to the multi-coloring generalizations.

1.2 Our results

For the np-SMC problem, we give in Section 3 two exact algorithms, with incomparable complexity: the first one is quadratic, i.e., $O(n^2)$ where $|V| = n$, while the second is more effective if the maximum color requirement p is small, running in time $O(np)$. In both cases, non-trivial optimizations have been made to reduce the time complexity. The first algorithm is still more efficient for the special case of paths, running in time $O(n \cdot \log p / \log \log p)$.

Previous dynamic programming algorithms [K89] can be seen to generalize to multi-coloring, leading to algorithms that are polynomial in n and p , e.g. $O(p^2 n \log n)$. They are, however, not polynomial for large p .

For the case of p-SMC, the solvability for trees appears to be a hard question. We present in Section 4 a polynomial time approximation scheme, along with an exact algorithm for a limited special case. Finally, we discuss in Section 5 several generalizations of the problem, to which our algorithms continue to apply, and mention open problems for further study.

2 Definitions and Notation

An instance of a multi-coloring problem is a pair (G, x) where $G = (V, E)$ is a graph and $x : V \rightarrow \mathbf{N}$ is a vector of *color requirements* (or *lengths*) of the vertices. We denote by $p = \max_{v \in V} x(v)$ the

maximum color requirement.

A *multi-coloring* of G is an assignment $\Psi : V \rightarrow 2^N$, such that each vertex v is assigned $x(v)$ distinct colors and adjacent vertices receive non-intersecting sets of colors. The *start time* (*finish time*) of a vertex v under Ψ is the smallest (largest) color assigned to v , denoted by $s_\Psi(v) = \min\{i \in \Psi(v)\}$ ($f_\Psi(v) = \max\{i \in \Psi(v)\}$). A multi-coloring Ψ is *contiguous*, or non-preemptive, if for any v , $f_\Psi(v) = s_\Psi(v) + (x(v) - 1)$. The *sum* of a multi-coloring Ψ of an instance (G, x) is the sum of the finish times of the vertices $\sum_{v \in V} f_\Psi(v)$. The minimum sum of a preemptive (non-preemptive) multi-coloring of G is denoted by $\text{pSMC}(G)$ (npSMC).

We denote by n the number of vertices of the input instance. For a vertex v , $\text{deg}(v)$ is the degree and $N(v)$ the set of neighbors of v . In the context of rooted tree T , let T_v denote the subtree rooted at v , $\text{kids}(v)$ denote the set of children of v , and $\text{par}(v)$ its parent. Finally, let $[x, y]$ denote the interval of natural numbers $\{x, x + 1, \dots, y\}$.

We shall be needing the following bound on the number of colors in an optimal sum multi-coloring, whose proof is omitted.

Claim 1 *Optimal p -SMC and np -SMC colorings of a tree use at most $O(p \log n)$ colors.*

3 Non-preemptive multicoloring

We say that vertex v is *grounded* in a multi-coloring Ψ , if the smallest of v 's colors is 1, $s_\Psi(v) = 1$. We say that v is *flanked* in Ψ by a neighbor u , if the smallest color of v is one larger than the largest color of u , that is, $s_\Psi(v) = 1 + f_\Psi(u)$. We call a sequence of vertices v_0, v_1, \dots, v_m a *grounding sequence* of v_m , if v_0 is grounded and, for all $0 \leq i < m$, v_{i+1} is flanked by v_i .

The following observation is called for.

Observation 3.1 (Flanking property): *In an optimum npSMC coloring of a graph, each vertex v is either grounded or has a flanking neighbor.*

It is not difficult to see that this holds for any *minimal* coloring, where the coloring of any one vertex cannot be reduced without creating an improper coloring.

Proof. Let u be the last neighbor u of v finished before the execution of v starts. If no such u exists, then v can be safely grounded.

We claim that there is no other vertex w whose coloring starts within units $f_\Psi(u) + 1, \dots, s_\Psi(v) - 1$. This follows since otherwise the coloring of w is *completed* before the coloring of v starts (this remark follows by the fact that the coloring of the vertices is non-preemptive.) Now, if w ends before v starts, this disagrees with the definition of u being the last neighbor whose schedule is finished before v starts. Hence, we can safely start the coloring of v in color $s_\Psi(u) + 1$. \square

Several interesting comments follow from the Flanking property. A grounding sequence v_0, v_1, \dots, v_m of a vertex v_m completely determines the coloring of v_m . In fact, $s_\Psi(v_m)$ equals the sum of color requirements of v_0, \dots, v_{m-1} plus 1.

Also, we can make the following observation, that indicates that for any graph, the optimum non-preemptive sum multi-coloring can be computed in (exponential) time independent of p . This may be important within an applied context, for small values of n .

Claim 2 *An optimum npSMC coloring of a graph can be computed in time $n^{n+O(1)}$.*

Proof. One can guess for each v a flanking neighbor u . The number of legal choices is clearly bounded by $O(n^n)$. Now, any such choice completely defines the schedule, and we only need to pick the optimum legal one. \square

Note, that the number of legal choices for flanking vertices could approach n^n . In fact, consider some legal multi-coloring. Put a directed edge $u \rightarrow v$, if v is flanked by u (if there are more than one flanking vertices for v , choose one arbitrarily.) This induces a collection of directed trees, that

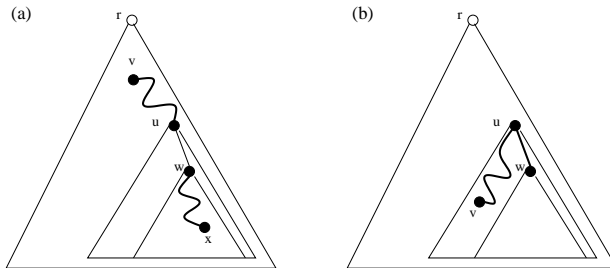


Figure 1: Computation of $A[u, v]$: (a) u grounded outside T_u , w grounded inside T_w , and (b) u grounded inside T_u , w flanked by u

contain all the vertices. Hence the number of choices for legal multi-colorings is bounded from below by the number of spanning trees of G , which can be as large as n^{n-2} .

In our search for an optimum npSMC coloring on trees, we examine possible grounding sequences. We note that there are at most n^2 paths (and grounding sequences) in a tree, which easily leads to a polynomial algorithm for trees. We shall introduce additional ideas to reduce the complexity to $O(n \min(n, p))$. Our general approach is based on dynamic programming, where we find a partial solution (projection of an optimal npSMC scheme on a subset of vertices) assuming a given grounding sequence and using previously computed partial solutions to related subproblems.

3.1 An $O(n^2)$ algorithm for np-SMC of trees

Assume that the tree T is arbitrarily rooted in vertex r . We give a dynamic programming algorithm that computes bottom-up a matrix A , where $A[u, v]$ contains the minimum cost of a coloring of a subtree T_u under the constraint that u is grounded in v . The desired solution is then given by $\min_u A[r, u]$.

Let $f_v(u)$ denote the finishing time of u when grounded in v . Namely, $f_v(u)$ is the total length of the grounding sequence of u on v , or the sum of the lengths of the vertices on the unique path from u to v .

A is computed as follows. $A[u, v] = f_v(u)$ for a trivial T_u (when u is a leaf of T), while in general, we compute $A[u, v]$ based on all values of $A[w, *]$ for every child w of u in T . This value could be $A[w, v]$, grounding w in the same vertex as its parent, or the minimum value of $A[w, z]$ over all those descendants z of w that result in a grounding of w that is compatible with the grounding of its parent u in v . This compatibility must satisfy two properties. First, if u is grounded in a vertex v that belongs to the subtree T_w of a child w of u , then w must also be grounded in v . Second, we must ensure that color ranges assigned to u and w do not overlap, $[f_v(u) - x(u) + 1, f_v(u)] \cap [f_z(w) - x(w) + 1, f_z(w)] = \emptyset$. Formally,

$$A[u, v] = f_v(u) + \sum_{w \in \text{kids}(u)} \min(A[w, v], \min_{x \in T_w} \{A[w, x] : f_v(u), f_x(w) \text{ compat.}\}) \quad (1)$$

Figure 3.1 illustrates two cases in computing $A[u, v]$.

The major inefficiency in computing the values of $A[u, v]$ according to (1) is the need for searching for compatible grounding in the subtrees T_w , resulting in a cubic algorithm. Instead, we ensure that optimal compatible solutions can be found quickly. We first compute, for each vertex in the tree, a sorted list of the n finishing times of the n different ways of grounding the vertex. The following lemma shows how to do this efficiently.

Lemma 3.2 *Given a tree T and a length function $x : V(T) \rightarrow R^+$, one can compute, for each vertex u of T , the sorted list of lengths of all paths in T originating in u , in $O(n^2)$ time.*

Proof. We root T in a vertex r . The algorithm has a bottom-up phase followed by a top-down phase. In the bottom-up phase we compute for each vertex u the sorted list of all lengths of paths from u to vertices in the subtree T_u rooted at u . For a leaf u this list contains only $x(u)$. The list for a non-leaf vertex u is obtained by merging the children's lists, then adding $x(u)$ to each element and prepending the element $x(u)$ to the resulting list. The bottom-up phase takes time proportional to $\sum_u \sum_{w \in N(u)} |T_w| \leq n \sum_u \deg(u) = O(n^2)$.

In the top-down phase, each non-root vertex u of T will process the complete sorted list of its parent $\text{par}(u)$. Out of that list, the entries involving descendants of u will appear in the same order as in the 'bottom-up' list of u and can thus be identified while scanning the two lists. The remaining entries on the list of $\text{par}(u)$, each augmented by $x(u)$, are merged with the 'bottom-up' list of u to produce the complete list of u in $O(n)$ time. \square

This computation of sorted lists of finishing times forms the first stage of our algorithm. We next process T in a bottom-up manner. Let the parent of a vertex u ($u \neq r$) have the sorted list of finishing times $f_{v_1}(\text{par}(u)) \leq \dots \leq f_{v_n}(\text{par}(u))$.

The processing of vertex u involves computing the list of optimum costs $c^{v_1}(u), \dots, c^{v_n}(u)$ where $c^{v_i}(u)$ is the optimum npSMC of the subtree T_u when the grounding of u is compatible with grounding its parent $\text{par}(u)$ in v_i . Assume the children of u have computed similar lists of optimum costs for their subtrees and let the sorted list of finishing times for u be $f_{z_1}(u) \leq \dots \leq f_{z_n}(u)$. We can now simplify Formula 1 computing the minimum cost npSMC of T_u when grounding u in z_i :

$$A[u, z_i] = f_{z_i}(u) + \sum_{w \in \text{kids}(u)} c^{z_i}(w) \quad (2)$$

Note that this is correct even if u is a leaf. For each value of i between 1 and n we also need to compute the "prefix minima" $P[u, i] = \min_{1 \leq j \leq i} \{A[u, z_j] : z_j \in T_u\}$ and "suffix minima" $S[u, i] = \min_{i \leq j \leq n} \{A[u, z_j] : z_j \in T_u\}$ over the entries in this array that reflect grounding in a descendant.

Consider the sorted list $f_{v_1}(\text{par}(u)) \leq \dots \leq f_{v_n}(\text{par}(u))$ of finishing times for the parent of u . For each $f_{v_i}(\text{par}(u))$, the compatible finishing times in the list $f_{z_1}(u) \leq \dots \leq f_{z_n}(u)$ of u can be found by a single scan through the two lists, producing for each i a left and right pointer l_i, r_i indicating that $f_{v_i}(\text{par}(u))$ is compatible with all finishing times from $f_{z_1}(u)$ up to $f_{z_{l_i}}(u)$ and all those from $f_{z_{r_i}}(u)$ up to $f_{z_n}(u)$. To find the minimum cost associated with these compatible finishing times, we use the l_i -th prefix and r_i -th suffix computed earlier:

$$c^{v_i}(u) = \begin{cases} A[u, v_i] & \text{if } v_i \in T_u, \\ \min(P[u, l_i], S[u, r_i], A[u, v_i]) & \text{otherwise.} \end{cases}$$

This gives us the desired list of optimum costs $c^{v_1}(u), \dots, c^{v_n}(u)$ with $c^{v_i}(u)$ the optimum npSMC of the subtree T_u where the grounding of u is compatible with grounding its parent in v_i . The value of the optimum cost npSMC for T overall is computed by the root vertex r and given by $P[r, n]$. The processing time for each vertex is linear.

Theorem 3.3 *The np-SMC problem can be solved for a tree in $O(n^2)$ time.* \square

In the case of paths, we can improve the complexity by observing that grounding sequences must be short.

Lemma 3.4 *The maximum number d of vertices in a grounding sequence v_1, \dots, v_d in a path is $O(\log p / \log \log p)$.*

Proof. Suppose the vertices v_0, v_1, \dots, v_d ($d > 2$) form a grounding sequence in an optimum npSMC coloring Ψ^* of a path. Then, we claim that, for each i , $2 \leq i < d$, $x(v_i) \geq (d-i) \sum_{0 \leq j < i} x(v_j)$. It then follows that

$$x(v_{d-1}) \geq (d-2)! \sum_{1 \leq j < 2} x(v_j) \geq (d-2)!$$

Since $p \geq x(v_{d-1}) = d^{\Omega(d)}$, we have the desired bound.

To prove the claim, consider the coloring obtained from Ψ^* by grounding the sequence v_i, \dots, v_{d-1} . This may necessitate flanking v_{i-1} by v_i . The former decreases the cost (with respect to $\text{SMC}(G, \Psi)$) by $(d-i) \sum_{0 \leq j < i} x(v_j)$, while the latter increases the cost by at most $x(v_i)$. The claim then follows from the assumed optimality of Ψ^* . \square

Corollary 3.5 *The np-SMC problem can be solved for a path in $O(n \log p / \log \log p)$ time.*

We note that Claim 1, combined with previous ideas, immediately gives a relatively simple optimum $O(n \cdot p \cdot \log n)$ algorithm for np-SMC on trees with the amount of information carried reduced to a minimum, saving on the space needed. In the next subsection, we improve this running time to $O(n \cdot p)$ via a slightly more complex algorithm.

3.2 An $O(n \cdot p)$ algorithm for np-SMC on trees

We now give an algorithm whose running time is linear in n , whenever $p \geq 1$ is a small constant. Let

$$B(v) = x(v) + \sum_{u \in N(v)} (x(u) + x(v) - 1).$$

Note, that the finish time of v under any “reasonable” coloring is at most $B(v)$, since each neighbor u of v can delay the completion of v by at most $x(u)$ steps, from its own length, plus $x(v) - 1$, from leaving a “gap” in the set of available colors for v .

We call a finishing time for vertex v an *optimal*, if there exists an optimal coloring of T_v where v has this finishing time. We say that a finishing time $f(v)$ *interferes* with a child u of v , if the respective coloring of v intersects with the coloring which corresponds to u 's optimal finish time in T_u .

The algorithm Tree-color proceeds by arbitrarily rooting the tree from some vertex r and then coloring it bottom-up. The coloring of v involves two tasks: (a) evaluating the cost of the possible finish times of v and selecting the optimal one, from which to derive the corresponding minimum multi-coloring sum of T_v , and (b) preparing a set of at most $x(v) + x(\text{par}(v)) - 1 < 2p$ alternative finish times for v , in the event that $\text{par}(v)$ chooses a finish time that interferes with v . The data required for these computations will be kept in the following integer arrays:

- $\text{inc}_v[B(v)]$, in which the i th entry gives the increase in the cost of the optimal coloring of T_v , when the finish time of v is i . The increase is computed relative to initial coloring of T_v , in which the algorithm assumes that $x(v) = 0$.
- $\text{alt}_v[2p]$, of alternative finish times for v : $\text{alt}_v[j]$ is the optimal finish time for v under the constraint that $\text{par}(v)$ has finish time j .

Each vertex v fills the arrays in four phases.

Phase 1: In the initial phase, v fills the array inc_v with values appropriate for the case that no collisions occur with the optimal colors of its children. That is, $\text{inc}_v[i] \leftarrow i$, for $x(v) \leq i \leq B(v)$.

Phase 2: In this phase, v updates the array inc_v . For each child $u \in \text{kids}(v)$, v adds in at most $x(u) + x(v) - 1$ entries the increase in the cost of the optimal coloring of T_u . Specifically, for any finish time i of v , which interferes with u , v updates the i th entry of inc_v , using the i th entry of the array alt_u , i.e. $\text{inc}_v[i] \leftarrow \text{inc}_v[i] + \text{inc}_u[\text{alt}_u[i]] - \text{inc}_u[f(u)]$. The optimal finish time, $f(v)$, is the value i that minimizes $\text{inc}_v[i]$.

Phase 3: We next compute two help vectors, containing prefix and suffix minimas of inc_v . That is, for $i = x(v), \dots, B(v)$, $p[i]$ is the index j that minimizes $\text{inc}_v[j]$ under the constraint $j \leq i$. It is computed inductively in linear time by: $p[i]$ is i , if $\text{inc}_v[i] \leq \text{inc}_v[p[i-1]]$, and $p[i-1]$ otherwise. Similarly, $s[i]$ is the index j that minimizes $\text{inc}_v[j]$ under the constraint $j \geq i$, computed by: $s[i]$ is

i if $inc_v[i] \leq inc_v[s[i+1]]$ and $s[i+1]$ otherwise. Here, $p[x(v)-1] = s[B(v)+1] = \infty$.

Phase 4: Finally, alternative finish times are computed. For each possible value j of a finish time of $par(j)$, $alt_v[j]$ should be the index minimizing inv_v . The constraint implies that either v is scheduled before $par(v)$, finishing no later than $j - x(par(v))$, or it is scheduled after $par(v)$, finishing no earlier than $j + x(v)$. The index minimizing inc_v in the former case is then given by $p[j - x(par(v))]$, while in the latter case it is given by $s[j + x(v)]$. Thus, we assign $alt_v[j]$ the better of the two possibilities.

Theorem 3.6 *Tree-color solves np-SMC on trees in $O(np)$ steps.*

Proof. The optimality of the algorithm follows from the fact that for any subtree T_v , we find a finishing time for v that minimizes the cost of coloring T_v . Formally, for any finish time $x(v) \leq i \leq B(v)$, the SMC of T_v is given by

$$SMC(T_v, i) = i + \sum_{u \in kids(v)} SMC(T_u, n_u(i))$$

where $n_u(i)$ is the finish time of child u in a minimum-cost coloring of T_u , when the finish time of v is i . Thus, the SMC of T_v is

$$SMC(T_v) = \min_{x(v) \leq i \leq B(v)} SMC(T_v, i) .$$

Since we color the vertices in the tree from the leaves upwards, the SMC of T is $SMC(T_r)$. The algorithm `Tree-color` follows the above dynamic programming scheme.

For the complexity of the algorithm, consider separately the phases performed by a vertex v . The first phase takes $O(B(v))$ steps. In the second phase, for each child u of v , $x(u) + x(v) - 1$ entries are updated in inc_v , for a combined complexity at most $O(B(v))$. The vectors p and s are computed in time $O(B(v))$, and the $O(p)$ entries of alt_v computed in constant time each. Observe, that $\sum_v B(v) = O(np)$, thus, summing up the complexity over all the vertices yields the theorem. \square

Corollary 3.7 *If each vertex requires exactly p colors, then `Tree-color` can be implemented in $O(n)$ steps.*

Proof. This case is equivalent to the sum coloring problem, when one color will represent a ‘block’ of colors of length p ; clearly, it is sufficient to consider in each pass on the vector inc only entries of finish times that are integral multiples of p (This can be shown inductively on the size of T). Thus, the complexity of the algorithm `Tree_color` reduces to $O(n)$. \square

4 Preemptive case

We turn our attention in this section to the preemptive version of the multi-coloring problem. Here, we do not have a polynomial algorithm for trees, nor a proof of NP-hardness. Instead, we give the next best possible: a polynomial-time approximation schema. We also mention an exact algorithm for the case of small color requirements.

4.1 PTAS for p-SMC of trees

The algorithm is a standard dynamic programming algorithm, but one that attempts to find a restricted type of a solution. These solutions have the property that there are at most $(1/\epsilon)^{O(\log p)}$ possible colorings of each vertex. Given such a property, a straightforward dynamic programming algorithm will examine the vertices bottom-up, trying each possible coloring of a vertex, and storing

the cost of the subtree for each such choice. The main part of the argument is to show the existence of a restricted solution whose sum is within $1 + \epsilon$ of optimal.

We first study the makespan problem (minimizing the maximum color used) on *bipartite* graphs, which has a rather trivial optimum solution. For simplicity, we allow multicolorings where at least $x(v)$ colors are assigned to each vertex v ; clearly, this does not make the problem any easier.

Lemma 4.1 *Let (G, x) be a bipartite instance, and let $\epsilon > 0$. Let $q = \max_{uv \in E} (x(u) + x(v))$ and let $s_i = \lfloor \epsilon i q \rfloor$, for $i = 0, \dots, \lceil 1/\epsilon \rceil$. Then, there is a contiguous coloring Ψ' of (G, x) using $\lfloor (1 + \epsilon)q \rfloor$ colors, such that for each vertex v there are integers j, j' such that Ψ' assigns to v the interval $[s_j, \dots, s_{j'} - 1]$.*

Proof. Let R, B be a bipartition of G , and let $r = \lfloor (1 + \epsilon)q \rfloor$. Consider the contiguous coloring Ψ_0 where

$$\Psi_0(v) = \begin{cases} [1, x(v)], & \text{when } v \in R \\ [r - x(v) + 1, r], & \text{when } v \in B. \end{cases}$$

Observe, that there are at least $r - q = \lfloor \epsilon q \rfloor$ values that separate the colors assigned to any pair of adjacent vertices. Hence, this coloring can be extended to a coloring Ψ' , given by

$$\Psi'(v) = \bigcup_j \{[s_j, s_{j+1} - 1] : [s_j, s_{j+1} - 1] \cap \Psi_0(v) \neq \emptyset\}. \quad \square$$

Let $\chi_\Psi = \max_v f_\Psi(v)$ be the makespan (maximum color used) of a multi-coloring Ψ . We now show how a given multi-coloring can be massaged into one satisfying several properties. The idea is to partition the range of possible colors into “layers” of geometrically increasing sizes. We apply Lemma 4.1 to schedule the colors of all vertices inside each layer, and to provide us with the desired restrictions on the possible colorings. The completion times of the vertices may increase for two reasons: the expansion factors of each level, and because of changes in the highest level that a vertex is colored in, but we can bound both factors by $1 + \epsilon$.

Theorem 4.2 *Let (G, x) be a bipartite instance, and $\epsilon > 0$. Then, for any multi-coloring Ψ of G , there is multi-coloring Ψ' , such that for each vertex v ,*

1. $f_{\Psi'}(v) \leq (1 + \epsilon)f_\Psi(v)$,
2. $\Psi'(v)$ is the union of at most $O(\log_{1+\epsilon} \chi_\Psi)$ contiguous segments, and
3. There are $O(1/\epsilon)$ choices for the beginning and the end of each segment.

Proof. Let $\epsilon_0 = \sqrt{1 + \epsilon} - 1$. For $1 \leq i \leq \lfloor \log_{1+\epsilon} \chi_\Psi \rfloor$, let $q_i = \lceil (1 + \epsilon_0)^i \rceil$ and $I_i = [q_{i-1}, q_i - 1]$. Define the instances (G, x_i) , where $x_i(v) = |\Psi(v) \cap I_i|$.

Apply Lemma 4.1 to obtain colorings Ψ'_i on (G, x_i) . Form Ψ' by concatenation:

$$\Psi'(v) = \bigcup_i \left\{ x + \sum_{j=0}^{i-1} \lfloor (1 + \epsilon_0)q_j \rfloor : x \in \Psi'_i(v) \right\}.$$

If the highest color of $\Psi(v)$ was in interval I_i , then $f(v) > q_i$, while

$$f_{\Psi'}(v) \leq \lfloor (1 + \epsilon_0)q_i \rfloor \leq (1 + \epsilon_0)^2 q_{i-1} \leq (1 + \epsilon)f_\Psi(v),$$

establishing part 1 of the theorem. Parts 2 and 3 also follow from properties of the Ψ'_i colorings of Lemma 4.1. Specifically, start and end points within each interval I_i are of the form $q_{i-1} + j \cdot \epsilon \cdot (q_i - q_{i-1})$ where $0 \leq j \leq \lfloor 1/\epsilon \rfloor$. \square

Theorem 4.3 *For each $\epsilon > 0$, the p -SMC problem on trees can be approximated within $1 + \epsilon$ factor in time $(p \cdot \log n)^{O(1/\epsilon \cdot \log(1/\epsilon))}$.*

Proof. Let Ψ be an optimal pSMC solution, and recall the properties of the solution Ψ' that Theorem 4.2 has shown to exist. We now argue that we can find a solution with such properties. For each vertex v and for each layer i , we try the $\lfloor 1/\epsilon \rfloor$ possibilities for the number of $\epsilon(q_i - q_{i-1})$ -sized “nibbles” that $\Psi'(v)$ contains in layer i . Thus, we need to maintain a table of size $(1/\epsilon)^{O(\log_{1+\epsilon} \chi_\Psi)}$. Since $\chi_\Psi = O(p \cdot \log n)$ by Claim 1, and $\ln(1 + \epsilon) \leq \epsilon$, the theorem follows. \square

When p is not polynomial in n , it is possible to show using a (rather non-trivial) scaling argument, that we can transform the instance to a new one with $p = O(n \cdot \log n)$, incurring only negligible loss in the approximation ratio (details omitted). This gives the result.

4.2 Exact algorithm for small lengths

Solving the p -SMC problem on trees by an exact algorithm seems to be a challenging task (even on paths). We only exhibit a very modest claim.

Claim 3 *The p -SMC problem on trees admits a polynomial solution when $p = O(\log n / \log \log n)$.*

Proof. Recall by Claim 1 that the number of colors used by an optimum solution for p -SMC is $O(p \cdot \log n)$. Thus, each vertex is to be assigned at most p colors in the range $1, \dots, O(p \cdot \log n)$. From the upper bound on p ,

$$\binom{O(p \cdot \log n)}{p} = O(\text{poly}(n)). \quad (3)$$

That is, the number of different possible preemptive assignments of colors to a vertex is polynomially bounded. Hence, the straightforward dynamic programming algorithm can compute an optimal solution in polynomial time, by exhaustively evaluating all possible assignments of colors to v . \square

One can also obtain an exact algorithm for trees in the case that color requirements are either all equal or all a small multiple of the same number. The proof is omitted.

Claim 4 *Let (G, x) be a p -SMC instance, where G is a tree and, for some natural number q , the color requirements are of the form $x(v) = y(v) \cdot q$. Then, taking an optimal sum multi-coloring of (G, y) and repeating each color q times yields an optimal sum multi-coloring of (G, x) .*

5 Discussion

The exact algorithms that we have given apply to several generalizations of the np-SMC problem on trees. We mention here two such generalizations.

The Optimum Chromatic Cost Problem (see [J97]) generalizes the Sum Coloring problem, in that the color classes come equipped with a cost function $c : Z^+ \rightarrow Z^+$, and the objective is to minimize the value of $\sum_{v \in V} c(f(v))$. We can generalize this to multi-colorings, in which case it is reasonable to assume that the color costs are non-decreasing. Our $O(n^2)$ and $O(np)$ algorithms hold then here as well.

The Channel Assignment problem comes with edge lengths $d : E \rightarrow Z^+$ and asks for an ordinary coloring, where the colors of adjacent vertices are further constrained to satisfy $|f(v) - f(w)| \geq d(vw)$. A non-preemptive multi-coloring instance corresponds roughly to the case where $d(vw) = (x(v) + x(w))/2$. Our algorithms handle this extension equally well, and can both handle the sum objective as well as minimizing the number of colors.

The argument for paths can be revised to hold for the generalized problems, in which case we can argue an $O(\log p)$ bound on the length of a grounding sequence.

Finally, our study leaves a few open problems. Is the p-SMC problem hard on trees? On paths? More generally, for which non-trivial, interesting classes of graphs, is the p-SMC problem solvable in polynomial time? Can we generalize the polynomial algorithms for np-SMC on trees to a larger class of graphs, e.g. outer-planar graphs? Our current arguments rely on a polynomial bound on the number of paths, which only holds for highly restricted extensions of trees.

References

- [B92] M. Bell. Future directions in traffic signal control. *Transportation Research Part A*, 26:303–313, 1992.
- [BBH⁺98] A. Bar-Noy, M. Bellare, M. M. Halldórsson, H. Shachnai, and T. Tamir. On chromatic sums and distributed resource allocation. *Information and Computation*, 140:183–202, 1998.
- [BKH⁺98] A. Bar-Noy, M. M. Halldórsson, G. Kortsarz, H. Shachnai, and R. Salman. Sum Multi-Coloring of Graphs. *Proceedings of the Fifth Annual European Symposium on Algorithms*, Prague, July 1999.
- [BK98] A. Bar-Noy and G. Kortsarz. The minimum color-sum of bipartite graphs. *Journal of Algorithms*, 28:339–365, 1998.
- [CCO93] J. Chen, I. Cidon and Y. Ofek. A local fairness algorithm for gigabit LANs/MANs with spatial reuse. *IEEE Journal on Selected Areas in Communications*, 11:1183–1192, 1993.
- [DO85] K. Daikoku and H. Ohdate. Optimal Design for Cellular Mobile Systems. *IEEE Trans. on Veh. Technology*, VT-34:3–12, 1985.
- [FK96] U. Feige and J. Kilian. Zero Knowledge and the Chromatic number. *Journal of Computer and System Sciences*, 57(2):187-199, October 1998.
- [J97] K. Jansen. The Optimum Cost Chromatic Partition Problem. *Proc. of the Third Italian Conference on Algorithms and Complexity (CIAC '97)*. LNCS 1203, 1997.
- [K89] E. Kubicka. The Chromatic Sum of a Graph. PhD thesis, Western Michigan University, 1989.
- [L81] N. Lynch. Upper Bounds for Static Resource Allocation in a Distributed System. *J. of Computer and System Sciences*, 23:254–278, 1981.
- [SP88] E. Steyer and G. Peterson. Improved Algorithms for Distributed Resource Allocation. *Proceedings of the Seventh Annual Symposium on Principles of Distributed Computing*, pp. 105–116, 1988.
- [SG98] A. Silberschatz and P. Galvin. *Operating System Concepts*. Addison-Wesley, 5th Edition, 1998.
- [Y73] W. R. Young. Advanced Mobile Phone Service, Introduction, Background and Objectives. *Bell Systems Technical Report*, 58:1–14, 1973.