Multicoloring 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 *multicoloring* problem. It occurs in two versions: in the *preemptive* case, each vertex may get any set of colors, while in the *non-preemptive* case, the set of colors assigned to each vertex has to be contiguous. We study these versions of the multicoloring problem 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 the 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 (see, e.g., [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 *multicoloring* 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 multicoloring 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 extended abstract of this paper appeared in COCOON '99 [HKP$^+$99].

[†]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.

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 multicoloring* (SMC) problem, we look for a multicoloring 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 multicoloring problem. In the *preemptive* (pSMC) problem, each vertex may get any set of colors, while in the *non-preemptive* (npSMC) problem, the set 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 multicoloring 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 (see e.g. [B86]). Thus, the set of processes forms 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 Our results

For the npSMC 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 $O(n \cdot \log p / \log \log p)$ (unless specified otherwise, all logarithms in this paper are base 2).

For the case of pSMC, the solvability for trees appears to be a hard question. We present in Section 4 a polynomial time approximation scheme (PTAS), along with an exact algorithm for a limited special case. A partitioning result of [HK99] allows us to improve the time-approximation tradeoffs of this method. Specifically, we give a PTAS for pSMC using at most $1/\epsilon^3 \cdot (\log 1/\epsilon)^2$ preemptions per vertex, running in time $exp((1/\epsilon \cdot \log 1/\epsilon)^3)n$. This implies that we can obtain $1+O((\log\log n/\log n)^{1/3})$-approximation in polynomial time, and for any fixed ϵ, we can achieve a $(1+\epsilon)$-approximation in linear time with a constant number of preemptions in the coloring of each vertex.

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.

1.2 Related work

The sum multicoloring problem was introduced by Bar-Noy et al. [BKH$^+$99]. They presented a comprehensive study of the approximability of both the pSMC and the npSMC problems, on general and special classes of graphs.

The sum coloring problem was introduced by Kubicka [K89], who gave a polynomial algorithm for trees. Jansen [J97] extended the dynamic programming strategy to partial k-trees. These dynamic programming algorithms can be seen to generalize to multicoloring, leading to algorithms that are polynomial in n and p, e.g. $O(p^2 n \log n)$. However, the additions in this paper are needed to obtain an algorithm polynomial in n only, or to reduce the complexity to $O(pn)$.

Known hardness results for the sum coloring problem carry over to the sum multicoloring problem. It is NP-hard on interval graphs [S99], planar graphs [HK99], and line graphs [BBH$^+$98], and NP-hard to approximate within some constant $c \geq 1$ on bipartite graphs [BK98]. On general graphs, it is hard to approximate within factor $n^{1-\epsilon}$, for any $\epsilon > 0$ unless $NP = ZPP$ [FK96, BBH$^+$98].

Resource-constrained scheduling has recently been investigated in the vast literature of scheduling algorithms (see e.g. [BK96, K96]). A special case involves the scheduling of multiprocessor tasks on dedicated processors. Kubale [K96] studies the complexity of scheduling biprocessor tasks, which corresponds to multicoloring line graphs. He also investigates special classes of graphs, and shows that npSMC of line graphs of trees is NP-hard in the weak sense, but leaves it open for pSMC.

Halldórsson and Kortsarz [HK99] have generalized some of the results of this paper to the class of partial k-trees (or, graphs of bounded treewidth). In particular, they gave an $O(n(p\log n)^{k+1})$ algorithm for npSMC, and a $(1+\epsilon)$-approximation in time $n^{O(1/\epsilon)^3}$ for pSMC. Notice that for both models, the algorithms of this paper have considerably better complexity bounds for the case of trees ($k=1$). The [HK99] paper also gave PTASes for planar graphs in both models.

2 Definitions and Notation

An instance of a multicoloring problem is a pair (G, x) where $G = (V, E)$ is a graph and $x : V \to \mathbf{N}$ 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 in G.

A *multicoloring* of G is an assignment $\Psi : V \to 2^N$, 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 multicoloring Ψ is *contiguous*, or non-preemptive, if for any v, $f_\Psi(v) = s_\Psi(v) + (x(v) - 1)$. The *sum* of a multicoloring 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) multicoloring of G is denoted by pSMC(G) (npSMC(G)).

We denote by n the number of vertices of the input instance. For a vertex v, $deg(v)$ is the degree and $N(v)$ is the set of neighbors of v. When T is a rooted tree, we denote by T_v the subtree rooted at v, $ch(v)$ denotes the set of children of v, and $p(v)$ its parent. Finally, we denote by $[x, y]$ the interval of natural numbers $\{x, x+1, \ldots, y\}$.

We shall be needing bounds on the number of colors used.

Lemma 2.1 *Consider an optimal sum multicoloring (preemptive or non-preemptive) of a bipartite graph, and let n' be the number of vertices that are not fully colored at some point. Then, at least $n'/2$ of these vertices are fully colored after additional $2p$ steps.*

Proof. We focus on the delay costs of the remaining n' vertices, i.e. the number of time steps before their completion during which they are not being colored. A coloring of these n' vertices that completes less than half of them in $2p$ steps incurs a delay of more than $pn'/2$.

Consider the following alternative coloring. If V_1, V_2 is a bipartition of the graph with $|V_1| > |V_2|$, color V_1 first to completion, followed by V_2. The delay incurred is at most $p|V_2| \leq pn'/2$. □

Lemma 2.1 implies the following claim, since at most one vertex remains after $2p \log n$ steps.

Claim 1 *Optimum sum multicolorings (preemptive or non-preemptive) of a bipartite graph use at most $O(p \cdot \log n)$ colors.*

A bound on the number of colors in an approximate solution was given in [HK99]. We state its preemptive version for bipartite graphs.

Claim 2 *Any bipartite graph G has a $(1+\epsilon)$-approximate preemptive sum multicoloring that uses at most $2p(\lg 1/\epsilon + 2)$ colors.*

Proof. Observe that a round-robin schedule of G, that colors the bipartitions alternately in odd and even time steps, completes each task within twice its length. Thus, pSMC$(G) \leq 2\mathcal{S}(G)$, where $\mathcal{S}(G) = \sum_{v \in V} x(v)$. It follows that in an optimal sum coloring, at most $\mathcal{S}(G)/p$ vertices remain to be completed by step $2p$. By repeated applications of Lemma 2.1, at most $\mathcal{S}(G) \cdot \epsilon/p$ remain after $2p \lg 1/\epsilon$ additional steps. If we now truncate the optimal coloring there, and 2-color the remaining vertices using $2p$ colors, the added cost of coloring these vertices is at most $p \cdot \mathcal{S}(G) \cdot \epsilon/p \leq$ pSMC$(G) \cdot \epsilon$. The total number of colors used will be $p(4 + 2\lg 1/\epsilon)$. □

3 Non-preemptive multicoloring

We say that vertex v is *grounded* in a multicoloring Ψ, if the smallest of v's colors is 1, i.e. $s_\Psi(v) = 1$, and v is *flanked* in Ψ by a neighbor u, if the smallest color of v is one larger than the largest color of u, i.e. $s_\Psi(v) = 1 + f_\Psi(u)$. We call a sequence of vertices $v_0, v_1, \ldots, v_m$ 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. It follows from the Flanking property that a grounding sequence $v_0, v_1, \ldots, v_m$ 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 $v_0, \ldots, v_{m-1}$ plus 1.

In our search for an optimum npSMC coloring on trees, we examine possible grounding sequences. We note that since each pair of vertices can be connected by a single path, the total number of paths is $\binom{n}{2}$; thus, the number of grounding sequences is n^2. This is the property of trees that is not shared by important larger classes of graphs. It 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. We arbitrarily root the tree, and give inductive definitions of some attributes of the vertices and their corresponding subtrees in terms of the attributes of their children. These attributes can be evaluated in any bottom-up order, e.g. within a DFS or postorder traversal of the tree. Essentially, we compute for each node v and for each plausible coloring of v, the cost of the optimal solution of the subtree rooted at v, assuming this particular coloring of v. The plausible colorings of v correspond, in the first algorithm, to the n possible groundings of v, and in the second algorithm, to all ways in which the neighbors of v can delay v.

We specify the coloring of vertices in terms of *finishing times*. The finishing times $f(u)$ and $f(v)$ of adjacent vertices u and v must satisfy

$$[f(u) - x(u) + 1, f(u)] \cap [f(v) - x(v) + 1, f(v)] = \emptyset \ ,$$

for the coloring to be valid, in which case we say they are *compatible.*

Remark: We observe that the optimum non-preemptive sum multicoloring can be computed in time independent of p. This may be important within an applied context, for small values of n. Namely, for each vertex v, there are at most $deg(v) + 1$ flanking choices for v: either flanking one of its neighbors, or being grounding. Thus, the number of minimal multicolorings is at most n^n. Each can be generated and checked in linear time, hence the complexity is $O(n^{n+1})$. This bound is essentially tight, since the number of minimal schedules in a clique on n vertices is $n! = \Omega(n/e)^n$.

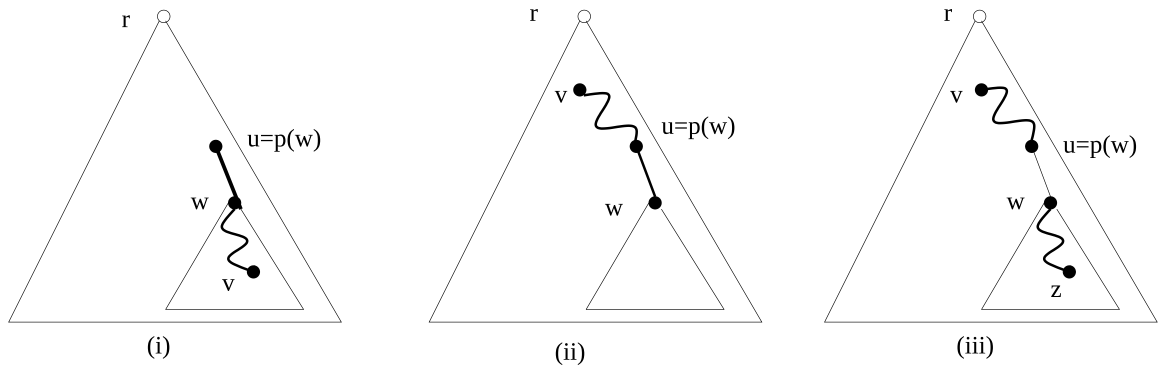

Figure 1: The three possible cases for grounding of w and $p(w)$.

3.1 An $O(n^2)$ algorithm for npSMC 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 the subtree T_u, under the constraint that u is grounded in v. The desired solution is then given by $\min_v A[r,v]$. Let $f_v(u)$ denote the *finishing time* of u when grounded in v. Namely, $f_v(u)$ is the sum of the lengths of the vertices on the unique path from v to u.

Adjacent vertices must satisfy the following constraints on their groundings. Let w be a non-root vertex with parent $u = p(w)$. There are only three possibilities for grounding of w and $p(w)$:

(i) w and $p(w)$ both grounded in $v \in T_w$. Then $p(w)$ is flanked by w.

(ii) w and $p(w)$ both grounded in $v \notin T_w$. Then w is flanked by $p(w)$.

(iii) w grounded in $z \in T_w$ and $p(w)$ grounded in $v \notin T_w$. Then $f_v(p(w))$ and $f_z(w)$ must be compatible finishing times.

Figure 3.1 illustrates the three cases.

The minimum cost of the subtree T_u when u is grounded in v, $A[u,v]$, is given by the finishing time of u when grounded in v, plus the minimum costs of grounding all subtrees of u in a compatible manner. We thus get the formula

$$A[u,v] = f_v(u) + \sum_{w \in ch(u)} \begin{cases} A[w,v] & v \in T_w \\ \min\limits_{z \in T_w \cup \{v\}} (A[w,z] : f_v(u), f_z(w) \text{ compatible}) & v \notin T_w. \end{cases} \tag{1}$$

Since the optimizations in the right-hand side of the formula for $A[u,v]$ involves only vertices in the subtree of u, this gives us a rule for computing the matrix A bottom-up, thus solving the problem. Note that when $u = p(w)$ is grounded in a vertex $v \in T_w$, w must also be grounded in v. This is the easy case (i). However, when $u = p(w)$ is grounded in a vertex $v \notin T_w$, we need to optimize over all groundings of w among those $z \in T_w$ that are compatible with grounding $p(w)$ in v. Because of this harder case (iii), we only have an $O(n)$ bound on the computation of each of the n^2 entries of A. giving an $O(n^3)$ algorithm overall. In the remainder of this section we show how to compute the entries in constant amortized time with some preprocessing, giving an $O(n^2)$ algorithm overall.

In computing case (iii), we need to find compatible finishing times. To do this quickly, we precompute, for each vertex in the tree, a sorted list of the finishing times corresponding to the n different ways of grounding this vertex. The following lemma shows how to do this efficiently.

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

Proof. 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 L_u of all lengths of paths from u to vertices in the subtree T_u. Each entry has the index of the originating vertex as a satellite data. For a leaf u, L_u contains only $x(u)$. For a non-leaf vertex u, L_u is obtained by merging the children's lists, then adding $x(u)$ to each entry and prepending the entry $x(u)$ to the resulting list.

In the top-down phase, each non-root vertex u of T processes the completed sorted list of its parent $p(u)$. The entries involving descendants of u will appear in the same order in that list as in L_u (with values that have been augmented by $x(p(u))$), and can thus be identified while scanning the two lists. We extract the entries of non-descendants of u, augment their values by $x(u)$, and merge the resulting list with L_u. This gives the complete list for u. The work done at each vertex in either phase is $O(n)$, for a total time complexity of $O(n^2)$. $\square$

For $u = p(w)$ grounded in vertex $v \notin T_w$ we now show how to deal with case (iii), *i.e.* how to compute efficiently $\min_{z \in T_w}(A[w,z] : f_v(u), f_z(w) \text{ compatible})$. Let z_i, $i = 1, \ldots, t$, be the vertices of T_w ordered such that $f_{z_1}(w) \leq \ldots \leq f_{z_t}(w)$.

First, we extract the list $A[w,z_1], \ldots, A[w,z_t]$. Next, we compute two vectors P and S, corresponding to *prefix and suffix minimas* of $A[w,z_1], \ldots, A[w,z_t]$. Namely,

$$P[w,i] = \min_{1 \leq j \leq i}\{A[w,z_j]\}, \qquad S[w,i] = \min_{i \leq j \leq t}\{A[w,z_j]\}.$$

Consider the sorted list $f_{v_1}(u) \leq \ldots \leq f_{v_n}(u)$ of all finishing times for the parent u of w. Observe that each $f_{v_i}(u)$ is incompatible only with $f_{z_j}(w)$, where j lies in some interval $j = l_i + 1, \ldots, r_i - 1$. Conversely, $f_{v_i}(u)$ is compatible with precisely $f_{z_1}(w), \ldots f_{z_{l_i}}(w)$ and with $f_{z_{r_i}}(w), \ldots, f_{z_t}(w)$. The minimum costs of these ranges are given by $P[w, l_i]$ and $S[w, r_i]$. Thus, given P and S, we can for u grounded in $v_i \notin T_w$ easily compute $\min_{z \in T_w}(A[w,z] : f_{v_i}(u), f_z(w) \text{ compatible}) = \min(P[w,l_i], S[w,r_i])$ in constant time per element. It remains to show how to compute the vectors l and r.

Observe that both the start and endpoints of these incompatibility intervals are monotone nondecreasing sequences. Thus, we can compute l_i and r_i, for all $i, 1 \leq i \leq n$, by a single scan through the two lists of finishing times for w and $p(w)$. Namely,

```
$f_{z_{t+1}}(w) \leftarrow \infty$.
$l_0 \leftarrow 0,\ r_0 \leftarrow 1$
for $i \leftarrow 1$ to $n$ do
  $l_i \leftarrow l_{i-1},\ r_i \leftarrow r_{i-1}$
  while ($f_{v_i}(p(w))$, $f_{z_{l_i+1}}(w)$ compatible and $r_i > l_i + 1$) $l_i{+}{+}$
```

```
        while ($(f_{v_i}(p(w)),\ f_{z_{r_i}}(w)$ are incompatible, or $f_{z_{r_i}}(u) < f_{v_i}(p(w)))$  $r_i++$
```

Observe that the processing time for computing the vectors P, S, l and r is $O(n)$ for each vertex. We can therefore compute $A[u, v]$ for all pairs u, v bottom-up over u in $O(n^2)$ time. The value of the overall optimum cost, npSMC of T, is given by $\min_{v \in T}(A[r, v])$. We have obtained the following theorem.

Theorem 3.3 *The* npSMC *problem can be solved for a tree in $O(n^2)$ time.* □

3.1.1 Special cases

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 $v_1, \ldots, v_d$ in a path is $O(\log p / \log\log p)$.*

Proof. Suppose the vertices $v_0, v_1, \ldots, v_d$ $(d > 2)$ form a grounding sequence in an optimum npSMC coloring Ψ^* of a path. Then, we claim that

$$x(v_i) \geq (d - i) \sum_{0 \leq j < i} x(v_j), \qquad \text{for } 2 \leq i < d. \tag{2}$$

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 show inequality (2), consider the coloring obtained from Ψ^* by grounding v_i the sequence $v_i, \ldots, v_{d-1}$. This may necessitate flanking v_{i-1} by v_i. The former decreases the cost (with respect to SMC(G, Ψ)) by $\sum_{0 \leq j < i} x(v_j)$, for each vertex $v_{i+1}, \ldots, v_{d-1}$, while the latter increases the cost by at most $x(v_i)$. Thus, the cost difference is $x(v_i) - (d - i) \sum_{0 \leq j < i} x(v_j)$, which by the assumed optimality of Ψ^* must be nonnegative. □

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

A reduction in the complexity of the tree algorithm can also be obtained when the tree has few distinct path lengths. We state the following claim without a proof. It implies, e.g. , that the npSMC of a tree of constant height with constant number of different lengths can be computed in linear time.

Claim 3 *Suppose a tree T has the property that from any vertex v, there are at most q different lengths of paths originating from v. Then,* npSMC *of T can be computed in time $O(qn)$.*

3.2 An $O(n \cdot p)$ algorithm for npSMC on trees

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

The algorithm Tree-color proceeds bottom-up on the rooted tree T. The coloring of each vertex v involves two tasks:

(a) Evaluate the cost of the possible finish times of v and select the optimal one, from which to derive the corresponding minimum multicolor sum of T_v.

(b) For $v \neq r$, prepare a set of at most $x(v) + x(p(v)) - 1 < 2p$ alternative finish times for v, in the event that $p(v)$ chooses a finish time that interferes with v.

Observe that the finish time of v in a minimal coloring is at most

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

Namely, 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.

The data required for these computations will be kept in the following integer arrays:
– $cost_v[B(v)]$, in which the ith entry gives the minimum cost of coloring T_v, when the finish time of v is set to be i.
– $alt_v[B(v)]$, of alternative finish times for v, in which the jth entry is the optimal finish time for v when $p(v)$ has finish time j.
Let $f(v)$ be the finish time of v that minimizes the cost of coloring T_v, and $minCost(v) = cost_v[f(v)]$ be that cost.

Each vertex v fills the arrays in four phases.

(i) In the initial phase, v fills the array $cost_v$ with values appropriate for the case that no collisions occur with the optimal colors of its children. Let

$$SubtreeCost(v) \leftarrow \sum_{u \in ch(v)} minCost(u).$$

Then, for $i = x(v), \ldots, B(v)$, set

$$cost_v[i] \leftarrow SubtreeCost(v) + i.$$

(ii) In the second phase, v adjusts the cost array to reflect collisions with the optimum colorings of the subtrees rooted at its children. Specifically, for any finish time i of v that is incompatible with $f(u)$, for $u \in ch(v)$, v updates the ith entry of $cost_v$, using the ith entry of the array alt_u.

Namely, for each $u \in ch(v)$ and $i = f(u) - x(u) + 1, \ldots, f(u) + x(v) - 1$,

$$cost_v[i] \leftarrow cost_v[i] + cost_u[alt_u[i]] - minCost(u).$$

The optimal finish time, $f(v)$, is the value i that minimizes $cost_v[i]$.

(iii) In this phase, two help vectors P and S are computed from $cost_v$. The *prefix index-minima* of i, $P[i]$, is the index in which $cost_v$ is minimal, in the range $[x(v), i]$. That is, for $i = x(v), \ldots, B(v)$,

$$P[i] = \arg \min_{x(v) \leq p \leq i} cost_v[p].$$

Thus, e.g., $cost_v[P[i]] \leq cost_v[p]$, for $x(v) \leq p \leq i$.

Similarly, the *suffix index-minima* of i, $S[i] = \arg\min_{i \leq s \leq B(v)} cost_v[s]$, is the index in the range $[i, B(v)]$ in which $cost_v$ is minimal.

(iv) Finally, alternative finish times are computed. For each possible finish time j for $p(v)$ that is incompatible with $f(v)$, $alt_v[j]$ should be the index minimizing $cost_v$. The constraint implies that either v is scheduled before $p(v)$, finishing no later than $j - x(p(v))$, or it is scheduled after $p(v)$, finishing no earlier than $j + x(v)$. The index minimizing $cost_v$ in the former case is then given by $P[j - x(p(v))]$, 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* npSMC *on trees in* $O(np)$ *time.*

Proof. We consider separately the phases performed by a vertex v. The first phase takes $O(B(v))$ steps. In phase (ii), for each child u of v, at most $x(u) + x(v) - 1$ entries are updated in $cost_v$, for a combined complexity $O(B(v))$. In phase (iii), the vectors P and S can be computed inductively, in $O(B(v))$ steps each. Initially, $P[x(v) - 1] \leftarrow S[B(v) + 1] \leftarrow \infty$, and for $x(v) \leq i \leq B(v)$,

$$P[i] \leftarrow \begin{cases} i & \text{if } cost_v[i] \leq cost_v[P[i-1]] \\ P[i-1] & \text{otherwise,} \end{cases} \qquad S[i] \leftarrow \begin{cases} i & \text{if } cost_v[i] \leq cost_v[S[i+1]] \\ S[i+1] & \text{otherwise.} \end{cases}$$

Finally, the $O(p)$ entries of alt_v are computed in constant time each. Observe, that

$$\sum_v B(v) = \sum_v (2d(v) + 1)x(v) \leq (4n - 3)p.$$

Thus, summing up the complexity over all the vertices yields the theorem. □

Corollary 3.7 *If the color requirement of each vertex is a multiple of q, for some integer q, then* npSMC *on trees can be solved in* $O(n(p/q))$ *time.*

Note, that if $x(v) = y(v) \cdot q$, $\forall\ v \in V$, then in any minimal coloring, every vertex will have finish time that is an integer multiple of q (this can be shown inductively on the size of the tree). Therefore, finding the minimum npSMC for the instance (G, x) is equivalent to finding the optimum npSMC for the reduced instance $(G, x/q)$. This can be done by the algorithm Tree-color in $O(n(p/q))$ steps.

4 Preemptive case

We turn our attention in this section to the preemptive version of the multicoloring 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 Algorithm overview

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 partition the color spectrum of an optimal coloring into *layers*, whose sizes are geometric powers of $1+\epsilon$. Consider the ith layer L_i and the coloring of a vertex v within that layer. Note that as long as $f(v) \notin L_i$, we may alter the colors assigned to v within L_i, as long as we do not "step out" of layer i. This follows since the objective function only takes into account the finish time $f(v)$. Now suppose that we know the *amount* of colors that each vertex has in layer L_i. Let $s(L_i)$ and $f(L_i)$ be the minimum and maximum colors in L_i. If we can "fit" all the required amounts of colors for each vertex v within the interval $[s(L_i), f(L_i)]$, this does not affect the $f(v)$ values, as long as $f(v) \notin L_i$. For each layer i, this results in a *makespan* (minimizing the number of colors used) instance: fit the required amount of colors per vertex in layer i so that the makespan is minimized. Using the minimum makespan coloring, we are guaranteed not to overstep L_i.

It is interesting to note that the makespan problem for bipartite graphs is trivially solvable using a natural greedy algorithm (see next subsection). From this discussion it follows that when given the quantities of colors per vertex in each layer, we can easily approximate the multicolor sum within $(1+\epsilon)$. Indeed, $f(v)$ may increase by $(1+\epsilon)$ due to the changes in the last layer of v (the layer i such that $f(v) \in L_i$). But since in all the other layers the colors do not overstep to the next layer this is the only increase.

If, on the other hand, we only "slightly overstep" the L_i layer, all the colors of v may be pushed upward, but only by a small amount. Indeed, we may expand each layer L_i by a factor of $1+\epsilon$, increasing $f(v)$ only by the same amount. We use this idea as follows. Let $c_i(v)$ be the exact number of colors assigned to v in L_i. "Guessing" the exact numbers $c_i(v)$ for each v turns out to be too expensive. Instead, we guess those quantities up to an additive factor of $\epsilon \cdot c_i(v)$. Namely, we guess the multiple of $\epsilon \cdot (f(L_i) - s(L_i))$ of colors that v has in each layer i. This decreases the number of possible choices down to $1/\epsilon$. We may be assigning up to $\epsilon \cdot (f(L_i) - s(L_i))$ extra colors per vertex, per level i. However, this only increases the finish time of each node by $1+\epsilon$, and the final multicoloring sum is within a factor of $(1+\epsilon)^2$ from optimal.

4.2 Polynomial time approximation scheme for pSMC of trees

We first study the *makespan* problem on bipartite graphs. For simplicity of exposition, 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 $i = 0, \ldots \lceil 1/\epsilon \rceil$. 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 $[s_j + 1, \ldots, s_{j'}]$ of colors.

Proof. Observe that q is a lower bound on the number of colors needed. 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

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

□

Let $\chi_\Psi = \max_v f_\Psi(v)$ be the makespan (maximum color used) of a multicoloring Ψ. We now show how a given multicoloring 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 multicoloring Ψ of G, there is multicoloring 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 $L_i = [q_{i-1}, q_i - 1]$. Define the instances (G, x_i), where $x_i(v) = |\Psi(v) \cap L_i|$.

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 the layer L_i, then $f_\Psi(v) > q_{i-1}$, 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 layer L_i are of the form $q_{i-1} + j \cdot \epsilon \cdot (q_i - q_{i-1})$ where $0 \leq j \leq \lfloor 1/\epsilon \rfloor$. □

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

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.

Traverse the tree in postorder, or any other bottom-up order. For each vertex we compute a table of size r^y, where $y = O(\log_{1+\epsilon} \chi_\Psi)$ is the number of segments in the coloring Ψ' and $r = 1/\epsilon$ is the number of possible starting or end points of each segment. There is an entry for each possible coloring of v under the constraints on Ψ' of Theorem 4.2, where we record the minimum cost of a coloring of the subtree rooted at v, given that coloring of v. For each such coloring, we search through the tables of the children of v for the cheapest colorings of their subtrees consistent with that assignment to v, and record the minimum.

The amount of computation for a given vertex v is then $r^{O(y)}d(v)$, for a combined time complexity of $r^{O(y)}n$. Since $\chi_\Psi = O(p \cdot \log n)$ by Claim 1, and $\ln(1+\epsilon) \le \epsilon$, the theorem follows. $\square$

As presented, the time complexity is only pseudo-polynomial. It is not hard to change the dependency on p to a dependency on n. However, following the early version of this paper [HKP+99], a structural result was given in [HK99] that leads to substantial improvements in the time complexity and/or approximation factors of the above approximation scheme.

Let $p_G = \max_{v \in G} x(v)$, and $l_G = \min_{v \in G} x(v)$. Let SMC$(G, \Psi)$ denote the sum of a multicoloring Ψ on G. The following is implicit in [HK99, Prop. 1].

Lemma 4.4 *Let G be a multicoloring instance and $q = q(n) \ge 1$ an integer. We can partition G in polynomial time into subgraphs $G_1, G_2, \ldots, G_t$ with the following two properties:*

1. *The ratio p_{G_i}/l_{G_i} of maximum to minimum color requirements is at most q.*
2. *Suppose we are given colorings Ψ_i of G_i, $i = 1, \ldots, t$, each using at most $k \cdot p_{G_i}$ colors, for some fixed number k. Then, we can concatenate the Ψ_i to obtain a coloring Ψ of G with*

$$\mathrm{SMC}(G, \Psi) \le \sum_{i=1}^{t} \mathrm{SMC}(G_i, \Psi_i) + \frac{k}{\sqrt{\ln q}} \cdot \mathrm{pSMC}(G).$$

Theorem 4.5 *There is a PTAS for* pSMC *using at most $O(1/\epsilon^3 \cdot (\log 1/\epsilon)^2)$ preemptions per node, running in time $exp((1/\epsilon \cdot \log 1/\epsilon)^3)n$.*

Proof. Let $\epsilon > 0$ be given, and set $\epsilon_2 = \epsilon/3$ and $\epsilon_1 = \epsilon/4$. Let $q = e^{(6/\epsilon \cdot (\lg 1/\epsilon + 4))^2}$.

Apply Lemma 4.4 with the above q, partitioning G into subgraphs G_i. Color each of the G_i independently as follows. By Claim 2, there is a $1+\epsilon_1$-approximate pSMC coloring Ψ_i using $2p_{G_i}(\lg 1/\epsilon_1 + 2)$ colors. Apply the dynamic programming strategy of Theorem 4.3 to find a coloring Ψ'_i that satisfies the properties of Theorem 4.2 for the ϵ_2 given. Finally, concatenate the colorings Ψ'_i to obtain a coloring Ψ of G.

Observe that the colorings Ψ'_i satisfy

$$\mathrm{SMC}(G_i, \Psi'_i) \le (1+\epsilon_2)\mathrm{SMC}(G_i, \Psi_i) \le (1+\epsilon_1)(1+\epsilon_2)\mathrm{pSMC}(G_i).$$

Note that $\lg 1/\epsilon_1 = \lg 1/\epsilon + 2$, and that $2(\lg 1/\epsilon + 4)/\sqrt{\ln q} = \epsilon/3$. By Lemma 4.4, the cost of Ψ is bounded by

$$\begin{aligned}
\mathrm{SMC}(G, \Psi) &\leq \sum_{i=1}^{t} \mathrm{SMC}(G_i, \Psi_i') + \frac{2(\lg 1/\epsilon_1 + 2)}{\sqrt{\ln q}} \cdot \mathrm{pSMC}(G) \\
&\leq ((1+\epsilon_1)(1+\epsilon_2) + \epsilon/3)\mathrm{pSMC}(G) \\
&\leq (1+\epsilon)\mathrm{pSMC}(G).
\end{aligned}$$

The complexity and preemption requirements are direct functions of the number of segments stipulated by Theorem 4.2 for each G_i. Part 2 of the statement of Theorem 4.2 can be strengthened to bound the number of segments by

$$O(\log_{1+\epsilon_2} \chi_{\Psi_i}/l_{G_i}) = O(\log_{1+\epsilon} q) = O(1/\epsilon^3 \cdot (\log 1/\epsilon)^2).$$

This is also the upper bound on the number of preemptions per vertex. By the argument of Theorem 4.3, the time complexity is bounded by

$$(1/\epsilon)^{O(\log_{1+\epsilon} q)} = 2^{O(1/\epsilon \cdot \lg 1/\epsilon)^3}$$

per node. □

In particular, for any fixed $\epsilon > 0$, a $1+\epsilon$-approximation using $O(1)$-preemptions can be computed in linear time, and a $1 + O(\log\log n/\log n)^{1/3}$-approximation using $O(\log n)$-preemptions can be computed in polynomial time.

4.3 Exact algorithm for small lengths

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

Claim 4 *The* pSMC *problem on trees admits a polynomial solution when* $p = O(\log n/\log\log n)$.

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

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

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. □

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 5 *Let* (G, x) *be a* pSMC *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 multicoloring of* (G, y) *and repeating each color* q *times yields an optimal sum multicoloring of* (G, x)*.*

5 Extensions

The exact algorithms that we have given apply to several generalizations of the npSMC problem on trees. We mention here a few 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 multicolorings, 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 multicoloring 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 this problem (and the OCCP problem), in which case we can argue an $O(\log p)$ bound on the length of a grounding sequence.

Various different measures and cost functions considered in scheduling theory can also be handled by our algorithms. The introduction of *release dates*, the points at which jobs become available, are accommodated by adjusting the feasibility of a proposed coloring of a node. Vertex will now be grounded if execution is initiated at its release time. Due dates and/or deadlines are treated by modifying the objective function, and the same holds for vertex weights. Common objective functions that can be handled include weighted sum of completion times, weighted number of late jobs, total tardiness, and the maximum (or sum) of monotonous non-decreasing functions of the completion times. Additionally, precedence constraints that follow the structure of the tree have the effect of directing the edges within the tree, and are easily accommodated by allowing only grounding consistent with those directions.

5.1 Open questions

Our study leaves a few open problems. Is the pSMC problem hard on trees? On paths? More generally, for which non-trivial, interesting classes of graphs, is the pSMC problem solvable in polynomial time? (It is possible to prove, that the problem can be easily solved on stars; we omit the details here). Can npSMC be optimally solved on other classes of graphs? Our current arguments rely on a polynomial bound on the number of paths, which only holds for highly restricted extensions of trees.

References

[B86] M. Bach. The Design of the UNIX Operating System Prentice Hall, 1986.

[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$^+$99] A. Bar-Noy, M. M. Halldórsson, G. Kortsarz, H. Shachnai, and R. Salman. Sum Multicoloring of Graphs. *Proceedings of the 7th 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.

[BK96] P. Brucker and A. Krämer. Polynomial algorithms for resource-constrainted and multiprocessor task scheduling problems. *European Journal of Operational Research,* 90:214–226, 1996.

[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.

[HK99] M. M. Halldórsson and G. Kortsarz. Multicoloring Planar Graphs and Partial k-trees. In *Proceedings of the Second International Workshop on Approximation algorithms* (APPROX '99). Lecture Notes in Computer Science Vol. 1671, Springer-Verlag, August 1999.

[HKP$^+$99] M. M. Halldórsson, G. Kortsarz, A. Proskurowski, R. Salman, H. Shachnai, and J. A. Telle. Multi-Coloring Trees. In *Proceedings of the Fifth International Computing and Combinatorics Conference (COCOON)*, Tokyo, Japan, Lecture Notes in Computer Science Vol. 1627, Springer-Verlag, July 1999.

[J97] K. Jansen. The Optimum Cost Chromatic Partition Problem. *Proc. of the Third Italian Conference on Algorithms and Complexity (CIAC '97).* LNCS 1203, 1997.

[K96] M. Kubale. Preemptive versus nonpreemptive scheduling of biprocessor tasks on dedicated processors. *European Journal of Operational Research* 94:242–251, 1996.

[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. *Journal of Computer and System Sciences*, 23:254–278, 1981.

[S99] T. Szkaliczki. Routing with Minimum Wire Length in the Dogleg-Free Manhattan Model is NP-complete. *SIAM Journal on Computing*, to appear.

[SG98] A. Silberschatz and P. Galvin. Operating System Concepts. Addison-Wesley, 5th Edition, 1998.

[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.

[Y73] W. R. Young. Advanced Mobile Phone Service, Introduction, Background and Objectives. *Bell Systems Technical Report,* 58:1–14, 1973.