Tools for Multicoloring with Applications to Planar Graphs and Partial k-Trees*

Magnús M. Halldórsson† Guy Kortsarz‡

November 30, 2001

Abstract

We study graph multicoloring problems, motivated by the scheduling of dependent jobs on multiple machines. In multicoloring problems, vertices have lengths which determine the number of colors they must receive, and the desired coloring can be either contiguous (non-preemptive schedule) or arbitrary (preemptive schedule). We consider both the sum-of-completion times measure, or the sum of the last color assigned to each vertex, as well as the more common makespan measure, or the number of colors used.

In this paper, we study two fundamental classes of graphs: planar graphs and partial k-trees. For both classes, we give a polynomial time approximation scheme (PTAS) for the multicoloring sum, for both the preemptive and non-preemptive cases. On the other hand, we show the problem to be strongly NP-hard on planar graphs, even in the unweighted case, known as the Sum Coloring problem. For non-preemptive multicoloring sum of partial k-trees, we obtain a fully polynomial time approximation scheme. This is based on a pseudo-polynomial time algorithm that holds for a general class of cost functions. Finally, we give a PTAS for the makespan of a preemptive multicoloring of partial k-trees that uses only $O(\log n)$ preemptions.

These results are based on several properties of multicolorings and tools for manipulating them, which may be of more general applicability.

*Preliminary version appears in APPROX '99 [HK99].

†Iceland Genomics Corp., Snorrabraut 60, IS-105 Reykjavik, Iceland. E-mail: mmh@hi.is.

‡Dept. of Computer Science, Rutgers – The State University of New Jersey, Camden, NJ E-mail: guyk@camden.rutgers.edu.

1 Introduction

In multiprocessor systems certain resources may not be shared concurrently by some sets of jobs. Scheduling dependent jobs on multiple machines 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 that forbids scheduling these jobs at the same time.

An *instance* to multicoloring problems is a pair (G, x), where $G = (V, E)$ is a graph, and x is a vector of *color requirements* (or *lengths*) of the vertices. For a given instance, we denote by n the number of vertices, by $p = \max_{v \in V} x(v)$ the maximum color requirement. A *multicoloring* of G is an assignment $\psi : V \to 2^N$, such that each vertex $v \in V$ is assigned a set of $x(v)$ distinct colors and adjacent vertices receive non-intersecting sets of colors.

A multicoloring ψ is called *non-preemptive* if the colors assigned to v are contiguous, i.e. if for any $v \in V$, $(\max_{i \in \psi(v)} i) - (\min_{i \in \psi(v)} i) + 1 = x(v)$. If arbitrary sets of colors are allowed, the coloring is *preemptive*. 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.

One of the traditional optimization goals 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 minimizing the time within which *all* the jobs have been completed. Such an optimization goal favors the system. However, from the point of view of the jobs themselves, another important goal is to minimize the average completion time of the jobs.

We study multicoloring graphs in both the preemptive and non-preemptive models, under both the makespan and sum-of-completion times measures defined as follows. Denote by $f_\psi(v) = \max_{i \in \psi(v)} i$ the largest color assigned to v by multicoloring ψ. The *multisum* of ψ on G is

$$\texttt{SMC}(G, \psi) = \sum_{v \in V} f_\psi(v) \ .$$

Minimizing the makespan is simply minimizing $\max_v \{f_\psi(v)\}$. The problem of finding a preemptive (non-preemptive) multicoloring with minimum sum (makespan) is denoted `p-sum` (`p-makespan`), while the non-preemptive version is `np-sum` (`np-makespan`, respectively). When all the color requirements are equal to 1, the makespan problem is simply the usual coloring problem, while the sum versions reduce to the well-known *sum coloring* (`SC`) problem.

The Sum Multicoloring (`SMC`) problem has numerous applications, including traffic intersection control [B92, BH94], session scheduling in local-area networks [CCO93], compiler design and VLSI routing [NSS99].

1.1 Related work

The `p-makespan` problem is also known as *weighted coloring* [GLS88] or *minimum integer weighted coloring* [X98]. Grötschel, Lovász, and Schrijver [GLS88] gave a polynomial time algorithm on perfect graphs. For many classes of perfect graphs, preemptive multicoloring with the makespan objective can be translated to the ordinary coloring problem. A vertex v with color-requirement $x(v)$ is replaced by a clique $C(v)$ of size $x(v)$ (connecting a copy of v to a copy of u if u and v are connected in G). This reduction is polynomial if p is polynomial in n, but can often be done implicitly for large values of p. In the context of makespan this reduction preserves optimum solution. Such reduction is possible for families of graphs closed under cliqueing; e.g. chordal (and thus interval graphs). For a faster multicoloring algorithm on chordal graphs see [H94].

The p-makespan problem is NP-hard even when restricted to hexagon graphs (which are planar graphs) [MR97], while a 4/3-approximation is known [NS97]. The hexagon graphs are important for their use in *cellular networks*. The problem is polynomial solvable on outerplanar graphs [NS97], and trivial on bipartite graphs (cf. [NS97]).

Non-preemptive multicoloring has been studied in several context. On interval graphs it corresponds to the *Dynamic Storage Allocation* problem, for which the best approximation known is 5 [Ger96]. On line graphs, it forms the basis of the *Minimum File Transfer Scheduling* problem, which is approximable within 2.5 [Cof85].

The sum coloring problem was first directly studied in [KS89], followed by [KKK89]. Recent research has concentrated on finding approximation algorithms and proving hardness results [BBH$^+$98, BK98, HKS01]. The paper [BBH$^+$98] addressed general graphs, bounded degree graphs, and line graphs, and the paper [BK98] studied bipartite graphs. For partial k-trees, Jansen [J97] gave a polynomial algorithm for the Optimal Chromatic Cost Problem (OCCP) that generalizes the sum coloring problem. See [HKS01] for a recent summary of known results.

The paper most relevant to our study is [BHK$^+$98], where the np-sum and p-sum problems are thoroughly studied and the following results presented, among others. A constant factor approximation is given in the preemptive case for graphs where the *Maximum Independent Set* (MIS) problem is polynomially solvable (e.g. perfect graphs), while an $O(\rho)$-approximation holds for graphs where MIS is ρ-approximable. In the non-preemptive case, an $O(\log n)$-approximation is given for graphs where MIS is polynomial solvable, which translates to an $O(\rho \cdot \log n)$-approximation when MIS is ρ-approximable. Further results are given for bipartite, bounded-degree, and line graphs. In [HK$^+$99], efficient exact algorithms for np-sum are given for trees and paths, while a polynomial time approximation scheme (PTAS) is given for the preemptive case. In [HKS01], constant factor approximation is given for np-sum on line graphs and k-claw free graphs.

In [BBH$^+$98] it was proven that if there exists an $f(n)$-approximation algorithm for the sum coloring problem for a given hereditary class of graphs, then there exists a $g(n)$-approximation algorithm for Graph Coloring on the same class of graphs, where $g(n) = O(f(n) \log n)$. If further $f(n) = \Omega(n^c)$ for some $c > 0$, then $g(n) = O(f(n))$. Based on the hardness result for the minimum coloring problem, from [FK98], this indicates that the sum coloring problem cannot be approximated within $n^{1-\epsilon}$, for any $\epsilon > 0$, unless $NP = ZPP$ [BBH$^+$98, FK98]. For bipartite graphs, the sum coloring problem is NP-hard to approximate within some factor $c > 1$, unless $P = NP$ [BK98]. Clearly, these limitations carry over to the sum multicoloring problems.

1.2 Our results

We continue the line of work initiated in [BHK$^+$98]. A problem has a polynomial-time approximation scheme (PTAS) if it can be approximated in polynomial time within $1 + \epsilon$, for any constant $\epsilon > 0$. If in addition, the complexity of the scheme is polynomial in both n and $1/\epsilon$, it has a *fully* polynomial approximation scheme (FPTAS).

Partial k-trees: In Section 3 we deal with multicoloring problems on partial k-trees, graphs whose treewidth is bounded by k. We design a general algorithm CompSum for that goal, that outputs an optimum solution for np-makespan and np-sum on partial k-trees in time $O(n \cdot (p \cdot \log n)^{k+1})$. Note that the algorithm is only pseudo-polynomial, namely does not run in polynomial time for large p. The same algorithm solves p-sum and p-makespan on partial k-trees in polynomial time when p is small ($O(\log n / \log \log n)$).

In Section 4, we give PTASs for both p-makespan and p-sum, that hold for any p. The respective running times are $n^{O(k^2/\epsilon^3)}$ for the p-makespan problem, and $n^{O(k^2/\epsilon^5)}$ for the p-sum problem. These schemes satisfy the additional property that only $O(\log n)$ preemptions

are used. These results are applied in the approximation of planar graphs. We also give an FPTAS for the non-preemptive case, with a running time of $O(n^{2k+3}\epsilon^{-(k+1)})$.

Planar graphs: In Section 5, we give PTASs for `p-sum` and `np-sum` on planar graphs. The running times are $n^{O(1/\epsilon^5)}$ in the preemptive case, and $n \cdot exp(\ln\ln n + exp(O(\epsilon^{-1}\log\epsilon^{-1})^2))$ in the non-preemptive case. This also implies the first PTAS for the sum coloring problem. These algorithms are complemented with a matching NP-hardness result for sum coloring.

For both `p-sum` and `np-sum`, the previously best known bounds on planar graphs and partial k-trees were fixed constant factors [BHK+98].

In order to establish our results, we have derived in Section 2 a number of tools for analyzing and manipulating general multicoloring instances, which could be useful for further research:

- Bounds on the number of colors used by any optimal (multisum) solution, and transformations that reduce this number at a relatively small cost to the objective function.
- Scaling and rounding transformations, that allow reductions of the problems to instances with small color requirements.
- Partitions of the instances into subinstances of relatively uniform color requirement. This involves a lemma that generalizes the Markov inequality and may be of independent interest.

The preemptive and non-preemptive cases turn out to require different treatments. Non-preemptiveness restricts the form of valid solutions, which helps dramatically in designing efficient exact algorithms. On the other hand, approximation also becomes more difficult due to these restrictions. The added dimension of lengths of jobs, whose distribution can be arbitrary, introduces new difficulties into the already hard problems of coloring graphs. Perhaps our main contribution lies in the techniques of partitioning the multicoloring instance, both "horizontally" into segments of the color requirements (Section 4) and "vertically" into subgraphs of similar length vertices (Section 5).

2 Properties and tools for multicolorings

Multicolorings have certain features and complications that distinguish them from (uni)colorings. In unit-length instances, all vertices are created equal and need only a single value as a color. In multicoloring instances, vertices not only need multiple value, but their requirements can be arbitrarily varied. Thus, greedy approaches, e.g., can fail quite dramatically, because short jobs cannot afford to wait for intermediate jobs which in turn cannot wait for long jobs. What we need is a toolbox for managing and manipulating these weighted instances.

After giving key definitions in Section 2.1, we study in Subsection 2.2 the number of colors needed in optimal and approximate sum multicolorings. This is crucial for limiting our search for appropriate colorings.

We then study how and to what extent we can reduce the instances to ones with small color requirements. This is done via classical approaches from scheduling theory involving rounding and scaling the color requirements. As elsewhere in this paper, we have on top of the usual scheduling instance a graph structure that must be taken into account.

An orthogonal reduction technique that we develop involves partitioning the vertex set according to the color requirements. Each set in the partition contains nodes with color requirements in a given interval, and these intervals are carefully chosen so that, intuitively speaking, the average color requirement in each set is much lower than the smallest color requirement in the following set. Using scaling, this allows us to reduce the problem to that on instances with small maximum

color requirements, with confidence that the solutions for the small jobs will not unduly delay the long jobs. The formation of this partition involves a generalization of Markov inequality, which may be interesting in its own right.

2.1 Notation

A multicoloring instance $I = (G, x)$ consists of a graph G and color requirements vector x, but we may also use G to refer to the instance. Let $\mathcal{S}(G) = \sum_v x(v)$ be the sum of the color requirements of the vertices. More generally, for a set of numbers X, let $\mathcal{S}(X)$ denote $\sum_{x_i \in X} x_i$. Let p_{min} denote $\min_v x(v)$ and recall that p denotes the maximum color requirement. Let $\tau(G) = p/p_{min}$ denote the ratio between the maximum and minimum color requirements of vertices in G. When all the color requirements $x(v)$ are divisible by q, let I/q denote the instance resulting from dividing each $x(v)$ by q.

The minimum *multisum* (or multicoloring sum) of a graph G, denoted by $\texttt{pSMC}(G)$, is the minimum $\texttt{SMC}(G, \psi)$ over all multicolorings ψ. We denote the minimum contiguous (non-preemptive) multisum of G by $\texttt{npSMC}(G)$. The minimum number of colors in an ordinary uni-valued coloring of G (i.e. ignoring the color requirements) is denoted by $\chi(G)$. We denote by $\mu(I)$ the minimum makespan, i.e. the number of colors required to preemptively multicolor $I = (G, x)$. We let $OPT(G)$ denote the cost of the optimal solution, for the respective problem at hand.

Utilizing the relationship between coloring problems and scheduling problems, we view our algorithms as procedures that color in (synchronous) rounds. Each round involves one time unit, and colors some independent set (whose vertices have not been fully colored) by some color.

Definition 2.1 *When a vertex v is not colored in a given round, we say that the vertex is* delayed *by the independent set chosen in this round.*

In particular, the finish time $f(v)$ equals $x(v)$ plus the total delay of v.

2.2 Bounding the number of colors needed

We can bound the number of colors used by any optimal sum multicoloring. In this subsection, multicolorings can be either preemptive or non-preemptive. We note that coloring the graph with minimum number of colors does not always help in solving even the SC problem. For instance, while bipartite graphs can be colored with two colors, there exist bipartite graphs (in fact, trees) for which any optimal solution for the sum coloring problem uses $\Omega(\log n)$ colors [KS89]. We show here that this ratio bound is also tight in the multicoloring case.

Lemma 2.1 (Color count) *At most $n/2^i$ vertices remain to be completed after $2i\mu(G)$ rounds of an optimal sum multicoloring, for any $i = 1, \ldots, \log n$.*

Proof. Suppose the claim fails, in which case let i_0 be the smallest value for i for which it fails. Then, at most $n/2^{i_0-1}$ vertices remain after step $X = 2(i_0 - 1)\mu(G)$. The delay during the next $2\mu(G)$ rounds is at least $2\mu(G)$ per vertex, or strictly more than $2\mu(G) \cdot n/2^{i_0} = \mu(G) \cdot n/2^{i_0-1}$ in total.

Consider the alternative coloring, that colors all vertices not completed at step X by $\mu(G)$ colors. For that, we use an arbitrary minimum makespan coloring with colors $X + 1, \ldots, X + \mu(G)$. The delay caused is at most $\mu(G)$ per vertex, or $\mu(G)n/2^{i_0-1}$ in total. This contradicts the assumption that the coloring above was optimal (namely, it is better to use in the last $2\mu(G)$ rounds this trivial algorithm). □

It follows from Lemma 2.1 that an optimal sum multicoloring uses at most $p + 2\mu(G)\log n$ colors, since after $2\mu(G)\log n$ rounds, at most one vertex can remain.

Corollary 2.2 *An optimal sum multicoloring uses at most $p + 2\mu(G)\log n$ colors.*

Note that for $O(1)$-colorable graphs, $\mu(G) = O(p)$, even in the non-preemptive case. Indeed, we can fully color each color-class one after the other, getting at most $O(p \cdot \chi(G)) = O(p)$ delay per vertex.

Remark: In the appendix, we will see that the color-count lemma holds for a general family of objective functions.

Another corollary concerns the number of colors in approximate solutions. Note that this is an existence result.

Lemma 2.3 *There is a $1+\epsilon$-approximate solution for the sum multicoloring problems that uses at most $O(\mu(G)\log(k/\epsilon))$ colors, on a k-colorable graph.*

Proof. Let $i = \log(k/\epsilon)$. Use the first $2i\mu(G)$ colors of an optimal sum multicoloring, followed by a $\mu(G)$-coloring of the remaining vertices. At most $n/2^i$ vertices remained after the first $2i\mu(G)$ colors. Thus, the additional cost caused by the second part is at most $\mu(G)n/2^i = \epsilon\mu(G)n/k \leq \epsilon\,\mathcal{S}(G)$. The last inequality follows as $\mu(G) \leq kp$. □

Rounding and scaling instances

Here we present general methods for reducing p while paying a small price in the approximation factor. One of the main tool we use is the following scaling lemma.

Lemma 2.4 (Preemptive scaling.) *Let $\epsilon > 0$ and $c = c_\epsilon$ be large enough. Let $I = (G, x)$ be a multicoloring instance where for each v, $x(v)$ is divisible by q and $x(v)/q \geq c \cdot \ln n$. Then, the makespan of I and I/q are related by*

$$\mu(I) \leq q \cdot \mu(I/q) \leq (1+\epsilon) \cdot \mu(I). \tag{1}$$

Proof. The first inequality of (1) is verified as follows. Use an optimum coloring of I/q for coloring I, by repeating q times each color class of I/q. Observe that the number of colors used for each vertex v is $q \cdot x(v)/q = x(v)$ as required. In addition, the total number of colors used is bounded by $q \cdot \mu(I/q)$, hence the inequality. We now prove the second inequality using a probabilistic argument.

Consider an optimum makespan solution $OPT(I)$. We shall form a solution ψ for I/q. Include each color class of $OPT(I)$ into ψ with probability $(1+\epsilon_2)/q$ (with ϵ_2 to be determined later). The expected makespan (which is the expected number of independent sets selected) is $((1+\epsilon_2)/q) \cdot \mu(I)$. So, by Markov inequality (cf. [MR95]), the makespan is at most $((1+\epsilon_2)(1+\epsilon_3)/q) \cdot \mu(I)$, with probability at least $1 - 1/(1+\epsilon_3)$.

For this solution to be legal for I/q, we need to show that each vertex v gets at least $x(v)/q$ colors. We show that this holds with non-zero probability. The number of colors each v gets is a binomial variable with mean $(1+\epsilon_2) \cdot x(v)/q \geq (1+\epsilon_2) \cdot c \cdot \log n$. For a binomial variable X with mean ζ, and for any δ, $0 \leq \delta \leq 1$, Chernoff bound (cf. [MR95]) gives that

$$\Pr(X < (1-\delta)\zeta) \leq exp(-\delta^2 \cdot \zeta/2).$$

By choosing $c = 4(1+\epsilon_2)/\epsilon_2^2$, we bound the probability that v receives fewer than $x(v)/q$ colors by $1/n^2$. Hence, with probability at least $1 - 1/(1+\epsilon_3) - 1/n$, all vertices get their required number of colors, and simultaneously the makespan is at most $(1+\epsilon_3) \cdot (1+\epsilon_2) \cdot \mu(I)/q$. Therefore, there exists a coloring of I/q achieving these properties.

Select ϵ_2 and ϵ_3 so that $(1+\epsilon_2)(1+\epsilon_3) = (1+\epsilon)$. We then form a coloring for I by repeating q times each color class of the coloring for I/q. The resulting makespan is at most $(1+\epsilon)\mu(I)$, yielding Inequality (1). □

A modification of this lemma yields a similar bound for the preemptive multisum. This involves bounding separately for every vertex the probability that the finish time in the coloring of vertex I/q is greater than $(1+\epsilon)/q$ times its finish time in the optimal sum coloring of I.

Corollary 2.5 *Let $\epsilon > 0$ and $c = c_\epsilon$ be large enough. Let $I = (G, x)$ be a multicoloring instance where for each v, $x(v)$ is divisible by q and $x(v)/q \geq c \cdot \ln n$. Then, the multisums of I and I/q are related by*

$$\texttt{pSMC}(I) \leq q \cdot \texttt{pSMC}(I/q) \leq (1+\epsilon) \cdot \texttt{pSMC}(I). \tag{2}$$

An important consequence of the above lemma when minimizing the preemptive makespan is that at a small price, we may assume p is only logarithmic in n.

Lemma 2.6 (Logarithmic length) *Consider the* `p-makespan` *problem in k-colorable graphs. For any ϵ it is possible to reduce the instance I to an instance $\bar{I}$ so that:*

1. *The maximum color requirement in $\bar{I}$ is $\bar{p} = O(1/\epsilon^3 \cdot \log n)$.*
2. *A ρ-ratio solution ψ to $\bar{I}$ can be transformed into a $\rho(1+\epsilon)$-ratio solution of I.*
3. *The number of preemptions in the final transformed solution for I is logarithmic in n.*

Proof. Let c' be a constant larger than both $2k$ and $c_{\epsilon/2}$ of Lemma 2.4. Let $\omega = \lfloor \lg p \rfloor + 1$ denote the number of bits needed to represent p, and let $\alpha = \lceil \lg(c'/\epsilon) + \log\log n + 1 \rceil$. Let $q = 2^{\omega-\alpha}$. We shall partition the color requirements x into two parts x' and x'', yielding instances $I' = (G, x')$ and $I'' = (G, x'')$ such that $x(v) = x'(v) + x''(v)$. More specifically, x' is derived by zeroing in x the $\omega - \alpha$ less significant bits, and $x'' = x$ modulo $2^{\omega-\alpha}$ represents the less significant bits in x. Intuitively, it does not cause a large delay to schedule x'' first so that only color requirements x' remain. This follows as x'' is small relative to x'. Now, note that x' are all divisible by $2^{\omega-\alpha}$. We then use Lemma 2.4 to scale down the x' into α bits numbers and solve it and derive a solution for x.

Observe that $2^\omega \leq 2p$, and thus $q = 2^{\omega-\alpha} \leq \epsilon 2^\omega/(2c' \log n) \leq \epsilon p/(c' \log n)$. Consider the instance $\bar{I} = I'/q$ with color requirements x' divided by q. Given a solution ψ for $\bar{I}$, a solution to I is composed as follows. I'' is scheduled non-preemptively by any graph k-coloring of G. Later, we take the schedule of ψ and repeat each independent set q times. As it is easily seen that the resulting schedule is feasible for I the reduction is completed.

We now show the required properties. By the Scaling Inequality (1), the makespan of the resulting schedule of I' is bounded by $q \cdot \mu(I'/q) \leq (1+\epsilon/2)\mu(I')$. Also, if $c' \geq 2k$, the makespan of the k-coloring of I'' is at most $kp\epsilon/c' < (\epsilon/2)p < (\epsilon/2)\mu(I)$. Thus, the makespan of the combined schedule of I is at most a $1+\epsilon$ factor from optimal. The length $\bar{p}$ of the longest task in $\bar{I}$ is at most $2^\alpha = O(c'/\epsilon \cdot \log n)$. Since the graph is k-colorable, $\mu(\bar{I}) \leq \bar{p}k = O(\log n)$. Finally, the number of preemptions used in the reduction overall is at most $\bar{p}k + k = O(c'/\epsilon \cdot \log n)$. □

Thus, when seeking an approximate preemptive makespan multi-coloring of a graph, one can as a first step make p logarithmic in n at a very small cost. We believe that this reduction may be of independent interest. It will be used in Section 4 on partial k-trees.

For the `p-sum` problem, such a logarithmic reduction is not in sight. We prove a weaker result that transforms p to a value polynomial in n at a low cost.

Lemma 2.7 (Linear lengths) *In the* `p-sum` *problem on $O(1)$-colorable graphs, one may assume that $p = O(n/\epsilon)$ at the cost of increasing the cost of the solution by at most a $1+\epsilon$ factor.*

Proof. Let ϵ_1 be a number to be determined later. Choose j such that $\lceil n/\epsilon_1 \rceil \leq \lfloor p/2^j \rfloor \leq 2 \cdot \lceil n/\epsilon_1 \rceil$, and put $x' = \lfloor x/2^j \rfloor$. The number x' gives (roughly) the $\log(n/\epsilon_1)$ most significant bits in x. The "small part" of the color requirements $x_s = x - x' \cdot 2^j$ sum to at most $O(\epsilon_1 \cdot p)$. This follows since

$$\frac{p}{x_s} \geq \frac{n}{2\epsilon_1}, \tag{3}$$

for any x.

Now the schedule is described. Let I_s be the instance induced by the color requirements $x - x' \cdot 2^j$. Choose a minimum coloring of G, and schedule first I_s non-preemptively one color-class after the other. We separate the cost incurred into two parts: The sum of the color contribution of the small numbers, and the delay inflicted on the large numbers. Regarding the color-sum of the small numbers, it follows from Inequality (3) that since the graph is $O(1)$-colorable, this cost (which is bounded by $O(1)$ times the sum of the small numbers) is at most $O(\epsilon_1 \cdot p) = O(\epsilon_1 \cdot \mathcal{S})$. In addition, this coloring of the small numbers gives a delay for the large part of the numbers. The number of colors used in coloring the small numbers is bounded by (an order of) the maximum color-requirement in the small part of the numbers. By Inequality (3) this is bounded by $O(\epsilon_1 \cdot p/n) = O(\epsilon_1 \cdot \mathcal{S}/n)$ incurring an $O(\epsilon_1 \cdot \mathcal{S})$ delay for the remaining vertices. Now, removing the zeros from the remaining ("large") numbers, we get an instance with color requirements x' with maximum of $O(1/\epsilon_1 \cdot n)$. We now solve this instance with the assumed algorithm, and take q copies of each resulting set. Now, with an appropriate choice of ϵ_1, we get by Corollary (2.5) that only a $(1+\epsilon)$ increase in the cost is incurred. □

It is interesting to note that this means that our approximation will hold even when color requirements are super-polynomial and optimal solutions may not be polynomial representable. Also, for general graphs, a similar argument follows with $p = O(n \cdot \chi(G))$.

In the non-preemptive case, scaling can be done without error.

Lemma 2.8 (Exact non-preemptive scaling) *Let $I = (G, x)$ be a non-preemptive multicoloring instance where for each v, $x(v)$ is divisible by q. Then,*

$$q \cdot \texttt{npSMC}(I/q) = \texttt{npSMC}(I). \tag{4}$$

Proof. Let $\psi^*(I)$ be an optimal `np-sum` coloring of I, and let C_i be the set of vertices colored with color i. By induction, the finish time of any vertex in $\psi^*(I)$ is a multiple of q. Now, consider the coloring ψ formed by every q-th color of ψ^*, $\psi = C_q, C_{2q}, \ldots$. Then, ψ is a proper coloring of I/q. Hence, $\texttt{npSMC}(I/q) \leq \texttt{npSMC}(I)/q$. On the other hand, given a coloring of I/q, we can form a coloring of I by repeating each color q times. Thus, $\texttt{npSMC}(I) \leq q \cdot \texttt{npSMC}(I/q)$. □

Rounding non-preemptive instances The non-preemptive case is easier to round-and-scale, as we can reduce the minimum length down to a constant while paying only a slight overhead factor. Let $I/q = (G, x')$ be the instance obtained by $x'(v) = \lfloor x(v)/q \rfloor$. Recall that p_{min} denotes $\min_v x(v)$.

Lemma 2.9 (Non-preemptive rounding) *Let $I = (G, x)$ be a multicoloring instance, and $\epsilon > 0$ given. Suppose $p_{min} \geq 3/\epsilon$ and suppose we can approximate* `np-sum` *on I/q within ratio ρ, whenever $q = O(\epsilon p_{min})$. Then, we can approximate* `np-sum` *on I within ratio $(1+\epsilon)\rho$.*

Proof. Let ϵ_1 to be determined, and let $q = \lceil \epsilon_1 p_{min} \rceil$. Let $x'(v) = \lfloor x(v)/q \rfloor$ and let $I/q = (G, x')$ be the corresponding instance. Given a multicoloring ψ' of I/q, form a schedule ψ'' by repeating each color of I/q q times. Observe that each $x'(v) \geq 1/\epsilon_1$, for each vertex v. Note that $q \cdot x'(v) \geq x(v) - (q-1) \geq (1-\epsilon_1)x(v)$.

Let $t = \lfloor (1-\epsilon_1)/\epsilon_1 \rfloor$. We finally form a schedule ψ, by repeating once every $2t$-th color class of ψ'', i.e. if ψ'' consists of the sequence of independent sets $C_1, C_2, \ldots,$, then ψ consists of all of these sets, along with double occurrences of the sets $C_{i \cdot 2t}$, $i = 1, 2, \ldots$. Since each job is of length at least t, the number of colors they receive is multiplied by $1 + 1/t$. Hence, $\psi(v)$ contains at least

$$qx'(v)(1 + 1/t) \geq (1 - \epsilon_1)x(v)(1 + 1/t) \geq x(v)$$

colors.

The finish time of v in ψ is at most $(1 + 2/t)\psi''(v) = (1 + 2/t)q\psi'(v)$. Then, the multisum of ψ'' is at most

$$(1 + 2/t)qOPT(I/q) \leq (1 + 2/t)OPT(I).$$

Set $\epsilon = 2/t = 2\lfloor (1-\epsilon_1)/\epsilon_1 \rfloor$ satisfies the lemma. □

We shall be using this lemma for partial k-trees when the ratio of maximum to minimum color requirement is relatively small.

2.3 Partitioning into subgraphs of relatively uniform color requirement

The main technique introduced here involves splitting the instance into subgraphs in which all vertices have similar color requirements.

Call a class $\mathcal{G}$ of graphs *hereditary* if any induced subgraph of a member of the class is also a member. Let `SMC` refer to either `p-sum` or `np-sum`. Recall that $\tau(G)$ denotes the ratio p/p_{min}.

Theorem 2.10 *Let n, $q = q(n)$ and σ be given. Suppose that for any G in a hereditary class $\mathcal{G}$ that has at most n vertices and has $\tau(G) \leq q$, we can approximate* `SMC` *within a factor $1 + \epsilon(n)$ using $\sigma \cdot p(G)$ colors in time $t(n)$. Then, we can approximate* `SMC` *on any graph in $\mathcal{G}$ within a factor $1 + \epsilon(n) + \sigma/\sqrt{\ln q}$ using at most $2\sigma \cdot p(G)$ colors in $O(t(n))$ time.*

We shall later see how to approximate `np-sum` efficiently on planar graphs with τ relatively small. That, combined with the above theorem, then yields a PTAS for `np-sum` on planar graphs. We first prove two lemmas.

Markov inequality shows that at most $1/\ell$ fraction of the elements of a set $X = \{x_1, x_2, \ldots, x_n\}$ of non-negative numbers are greater than ℓ times the average value $\overline{x}$ (cf. [MR95]). Define $g(x)$ to be the number of x_i greater than or equal to x, i.e. $g(x) = |\{x_i : x_i \geq x\}|$. Then, Markov inequality corresponds to

$$g(\ell \cdot \overline{x}) \leq \frac{1}{\ell}\overline{x}n.$$

Rewriting $t = \ell n$, and $\overline{x} = \mathcal{S}(X)/n$, it corresponds to

$$g(t) \leq \frac{\mathcal{S}(X)}{t},$$

which holds for every $t \geq 0$. It is easy to show it to be tight for any fixed value of t but not for multiple values of t simultaneously. We show that if we are free to choose t from a range of values, the resulting bound on the tail is improved by a logarithmic factor. We state this first for an arbitrary function f.

Lemma 2.11 *Let r and s be real numbers, $s < r$, and let f be a function defined on $[s, r]$. Then, for some $t \in [s, r]$,*

$$tf(t) \leq \frac{1}{\ln(r/s)} \int_s^r f(x)dx.$$

Proof. Let t be the value x in the interval $[s, r]$ that minimizes $xf(x)$. Then,

$$\int_s^r f(x)dx = \int_s^r xf(x) \cdot \frac{1}{x}dx \geq tf(t) \int_s^r \frac{1}{x}dx = tf(t) \ln(r/s). \tag{5}$$

□

A weaker version of the lemma gives perhaps the most indicative improvement on Markov inequality. It uses the fact that g is positive and integer-valued. Its bound can be shown to be tight.

Corollary 2.12 *There is a t, $s \leq t \leq r$, such that*

$$g(t) \leq \frac{1}{\ln(r/s)} \cdot \frac{\mathcal{S}(X)}{t}.$$

Proof. Define the indicator functions $I_i(x)$ as 1 where $x \leq x_i$ and 0 elsewhere. Thus, $g(x) = \sum_i I_i(x)$. Then,

$$\int_0^\infty g(x)dx = \sum_i \int_0^\infty I_i(x)dx = \sum_i x_i = \mathcal{S}(X). \tag{6}$$

From Lemma 2.11 we have that $tg(t) \leq \frac{1}{\ln(r/s)} \int_0^\infty g(x)dx = \frac{1}{\ln(r/s)}\mathcal{S}(X)$. □

We use Lemma 2.11 to partition the instance into compact segments with good average weight properties.

Proposition 2.13 *Let $X = \{x_1, \ldots, x_n\}$ be a set of non-negative reals, and let q be a natural number. Then, there is a polynomial time algorithm that generates a sequence of integral breakpoints b_i, $i = 1, 2, \ldots$, with $\sqrt{q} \leq b_{i+1}/b_i \leq q$, such that*

$$\sum_{i=1}^{m} g(b_i) \cdot b_i \leq \frac{1}{\ln \sqrt{q}} \mathcal{S}(X).$$

Proof. Let b_0 be the smallest x_i value, and inductively let b_i be the breakpoint obtained by the Lemma 2.11 on the set $X_i = \{x_j : x_j \geq b_{i-1}\}$ with $s = b_{i-1} \cdot \sqrt{q}$ and $r = b_{i-1} \cdot q$. Terminate the sequence once b_i exceeds the maximum length p.

Since $b_i \geq b_{i-1}\sqrt{q}$, we have that $b_i \geq q^{i/2}$, and the loop terminates within $2 \log_q p$ iterations. In each iteration, the ratio r/s is at least $\sqrt{q}$. By Lemma 2.11,

$$b_i \cdot g(b_i) \leq \frac{1}{\ln \sqrt{q}} \int_{b_{i-1}\sqrt{q}}^{b_{i-1}q} g(x)dx.$$

Note that $b_i \geq b_{i-1}\sqrt{q}$ and thus the intervals $[b_{i-1}\sqrt{q}, b_{i-1}q)$ are disjoint. Hence,

$$\sum_i b_i g(b_i) \leq \frac{1}{\ln \sqrt{q}} \cdot \sum_i \int_{b_{i-1}\sqrt{q}}^{b_{i-1}q} g(x)dx \leq \frac{1}{\ln \sqrt{q}} \int_0^\infty g(x)dx = \frac{\mathcal{S}(X)}{\ln \sqrt{q}}.$$

The algorithm that finds the b_i partition can be easily implemented in linear time. □

Proof of Theorem 2.10.

Our approximation algorithm for an arbitrary graph G in $\mathcal{G}$ is as follows:

1. Find breakpoints $b_1, b_2, \ldots$ of the color requirements $x(v_1), \ldots, x(v_n)$ by Proposition 2.13 for the given value of q.

2. Partition G into subgraphs G_i, induced by $V_i = \{v : b_{i-1} \leq x(v) < b_i\}$, for which $\tau(G_i) \leq q$.

3. Solve instances (G_i, x) independently (by the algorithm assumed in Theorem 2.10) and schedule them in that order.

The reason why we can schedule the subgraphs G_i in order is that we have ensured with our choice of breakpoints that the smaller jobs don't delay the longer jobs much.

The cost of the multicoloring is derived from two parts: the sum of the costs of the subproblems, and the delay costs incurred by the colorings of the subproblems. We consider separately the delay caused by each G_i. Namely, the total delay is broken into the delays caused by each individual subproblem. For each subproblem, the delay occurred is reflected by the number of colors used in this subproblem, times the number of yet uncolored vertices (namely, the number of colors used times the total number of vertices included in later problems which are vertices of higher lengths). The number of colors used in G_i is assumed to be $O(\sigma \cdot b_i)$, while $g(b_i)$ represents the number of vertices delayed. By Proposition 2.13, this combined cost is thus $O(\frac{\sigma}{\sqrt{\ln q}} \cdot \mathcal{S}(G))$.

The cost of subproblem i is, by assumption, at most $(1+\epsilon(n))OPT(G_i)$. Thus, the sum of the costs of the subproblems, excluding the delay is $\sum_i (1+\epsilon(|G_i|))OPT(G_i) \leq (1+\epsilon(n))OPT(G)$. The total cost of the coloring is thus at most $(1+\epsilon(n)+O(\frac{\sigma}{\sqrt{\ln q}}))OPT(G)$. The total number of colors used is at most $\sum_i \sigma b_i \leq \sigma \sum_{i=0}^{\infty} p/q^i = (1+1/(q-1))\sigma \cdot p$. □

3 Exact multicolorings of partial k-trees

In this section, we study exact algorithms for multicoloring partial k-trees, particularly for `np-sum`. We note that the results here hold for a fairly general type of a cost function or measure that includes makespan and multisum functions. The definition of this family in its most general form is deferred to the appendix. The algorithm is described for the sum of completion time measure, while slight changes give a solution for the makespan measure.

The scenario is as follows. We are given a family $\mathcal{F}$ of colorings, and we look for the best coloring in this family. A trivial example would be for the family $\mathcal{F}$ to contain all possible colorings. This may, however, not be tractable. Instead, $\mathcal{F}$ may contain a family of coloring of some restrictive form where the best one is only an *approximation* of the optimum coloring. The family $\mathcal{F}$ must be uniformly well behaved in the sense that $\mathcal{F}$ must contain a good approximation for *any instance*. Hence, $\mathcal{F}$ cannot be too small.

The algorithm CompSum we present below follows a path similar to [J97] and is described for the sake of completeness. As we deal with graphs of fixed treewidth, which are $k+1$-colorable, we may assume by Corollary 2.2 that the number of colors used by an optimal coloring is at most $c \cdot p \cdot \log n$, where $c \leq 2k+3$. We assume without loss of generality that the graph contains no isolated vertices.

Partial k-trees are graphs that can be represented by the following tree structure. In a tree decomposition, we are given a collection $\mathcal{X}$ of at most $n = |V|$ subsets $X_i \subseteq V$ of vertices. Each subset X_i contains at most $k+1$ vertices. In addition, the subsets X_i are the vertices in a *supertree*, $\mathcal{T}(\mathcal{X}, E)$ with the following properties.

(I) **Edge-covering property:** The subsets X_i *cover* the edges of G, namely, for each $(v,u) \in E$, there exists a subset $X_i \in \mathcal{X}$ such that $v, u \in X_i$.

(II) **Connectivity:** For every vertex v, the subtree induced by the subsets X_j containing v is connected.

The vertices of the partial k-tree correspond to the nodes of the supertree, and two vertices are adjacent if they are both contained in some set X_i. Trees are partial 1-trees, where the edges form

the sets X_i. Partial k-trees draw their usefulness from their susceptibility to dynamic programming solutions. This uses the fact that it is possible to decompose the problem cleanly: if we delete all $k+1$ elements of a set X_i from the instance, we break the graph into disjoint components.

It is well known (see, e.g., [J97]) that a small modification in this tree structure, allows to keep the above properties and extend it with the following properties. It is possible to root $\mathcal{T}$ such that

1. $\mathcal{T}$ is a binary tree.
2. If supervertex X_i has a single child X_j, then $||X_i| - |X_j|| = 1$ and $X_i \subseteq X_j$ or $X_j \subseteq X_i$.
3. If supervertex X_i has two children X_k and X_j, then $X_i = X_k = X_j$.

The algorithm CompSum follows the prototypical dynamic programming paradigm on trees. Each node in the supertree involves up to $k+1$ vertices in the graph. For every meaningful coloring of these vertices, we compute the cost of the cheapest consistent coloring of the subtree rooted at the node. The computation proceeds bottom-up, with each node depending only on values computed at its children.

For every X_i, let $\mathcal{T}_i$ be the tree rooted by X_i, and let $U_i = \bigcup_{X_j \in \mathcal{T}_i} X_j$. Let $g_i : X_i \to 2^{Z^+}$, $g_i \in \mathcal{F}$ be a function that assigns $x(v)$ colors to the vertices $v \in X_i$. Namely, g_i is some multicoloring of the set X_i. Denote by $R_{g_i}(v)$ this set of colors assigned to v by g_i.

Define $M_i(g_i)$ to be the minimum SMC value of all multicolorings $\bar{g}_i$ (in $\mathcal{F}$) that extend g_i to the vertices in U_i. Formally, let $\mathcal{Q}_i$ be the set of functions $\bar{g}_i : U_i \to \{1, \ldots, c \cdot p \cdot \log n\}$, $\bar{g}_i \in \mathcal{F}$ with $\bar{g}_i(v) = g_i(v)$, for all $v \in X_i$. Denote

$$M_i(g_i) = \min_{\bar{g}_i \in \mathcal{Q}_i} \{\mathtt{SMC}(U_i, \bar{g}_i)\}.$$

These values are computed in a bottom-up manner. In a leaf X_i we have for any function g_i

$$M_i(g_i) = \begin{cases} \mathtt{SMC}(X_i, g_i), & \text{if } R_{g_i}(v) \cap R_{g_i}(u) = \emptyset \text{ for all } u, v \in X_i \text{ s.t. } (u,v) \in E, \\ \infty, & \text{otherwise.} \end{cases}$$

In what follows, we consider only functions g_i that correspond to legal colorings of X_i, namely, $R_{g_i}(v) \cap R_{g_i}(u) = \emptyset$ for all $u, v \in X_i$ & $(u,v) \in E$.

Now, consider an internal node X_i, with a single child X_j in the tree. Assume first that $X_i \subseteq X_j$. Let $\{v\} = X_j \setminus X_i$. Given a function g_i, let $\mathcal{L}_{g_i}$ be the set of legal colorings for X_j that are consistent with g_i on X_i. (We show later that we can bound the cardinality of $\mathcal{L}_{g_i}$ by a natural parameter of the family $\mathcal{F}$.) In this case,

$$M_i(g_i) = \min_{\bar{g}_i \in \mathcal{L}_{g_i}} \{M_j(\bar{g}_i)\}.$$

Otherwise, we have $X_j \subseteq X_i$. Let $\{v\} = X_i \setminus X_j$. In this case, for a function g_i giving values to X_i, let $\bar{g}_i$ be the restriction of g_i to X_j. By previous computation, we know $M_j(\bar{g}_i)$; the optimum extension of X_j to U_j. Then

$$M_i(g_i) = M_j(\bar{g}_i) + f_{g_i}(v).$$

Namely, we add the finish time given to v by g_i to the minimum color-sum over U_j

Finally, we have the case of X_i with two children X_j and X_k. Note that by the Connectivity property, $U_j \cap U_k = X_i$. Let g_i be a function on $X_i (= X_k = X_j)$. By previous computation, we know that values $M_j(g_i)$ and $M_k(g_i)$, the costs of the respective functions extending g_i optimally to respectively U_k and U_j. By adding these together, we have accounted for the cost of coloring all of U_i, but doubly counted the finish times of vertices in X_i. Thus,

$$M_i(g_i) = M_j(g_i) + M_k(g_i) - \sum_{v \in X_i} f_{g_i}(v).$$

We note that by the edge-covering property, all the finite values $M_t(g_i)$ for the root X_t represent legal colorings g_i of all the vertices of G. Hence the minimum multicolor sum is given by $\min_{g_i}\{M_t(g_i)\}$. It is easy to compute the actual minimum coloring from the above.

The CompSum algorithm requires only slight changes to find the optimum makespan of the partial k-trees.

Analysis The following parameter is used in measuring the running time of the procedure. We denote by $\mathcal{D} = \mathcal{D}(\mathcal{F}, v)$ the number of different colorings v has among the family $\mathcal{F}$. Let $\mathcal{D}(\mathcal{F})$ denote $\max_v\{\mathcal{D}(\mathcal{F}, v)\}$. "Good" families $\mathcal{F}$ are ones with $\mathcal{D}(\mathcal{F})$ small.

Let us now estimate the time complexity. We need to compute $M_i(g_i)$ for all the appropriate functions g_i. For a given function g_i, the required time is $O(k)$ per edge in the supertree. The number of different functions g_i on X_i is bounded by $\mathcal{D}^{k+1}$, since each of the at most $k+1$ vertices in X_i has at most $\mathcal{D}$ different colorings. Since the number of vertices of $\mathcal{T}$ is bounded by n, the resulting complexity is bounded by $O(n \cdot \mathcal{D}^{k+1})$.

Definition 3.1 *A family F of colorings is* searchable *if, for each vertex, the number of different colorings for v is polynomial and they can be generated in polynomial time.*

Theorem 3.1 *Given a searchable family $\mathcal{F}$,* CompSum *can find either the optimum multisum or the optimum makespan in time $O(n\mathcal{D}^{k+1})$.*

Remark: The main problem with the above time complexity is that the number $\mathcal{D}$ may be very large. In particular, while we know that the number of non-redundant non-preemptive colorings of a vertex are fewer than $(2\chi(G)+1)p$, where $\chi(G)$ denotes the chromatic number of G, the number of preemptive colorings may be as large as $\binom{p\log n}{p}$, or exponential in p.

3.1 Two corollaries

Our first result is for `np-sum` on partial k-trees. In this case, by Corollary 2.2, the number of possible colorings of a vertex v, $\mathcal{D}(v)$, is bounded by $\mathcal{D} = O(k \cdot p \cdot \log n)$, since we only need to specify the first color with the rest of the colors being consecutive.

Corollary 3.2 *The* `np-sum` *and* `np-makespan` *problems admit an exact algorithm for partial k-trees that runs in $O(n \cdot (kp \log n)^{k+1})$ time.* □

For the special case of trees, i.e. 1-trees, `np-sum` can be solved in time $O(np)$ and $O(n^2)$ [HK+99]. It is not clear if an algorithm exists for partial k-trees that does not depend on p, but it is likely that the polynomiality and log factors could be improved.

In general, the situation in the preemptive case seems harder, as great many colorings are possible for a single vertex. However, consider the case of preemptive coloring when color requirements are small, which may be a reasonable restriction. We know by Corollary 2.2 that the number of colors used by an optimum solution for `p-sum` on partial k-trees is at most $O(p \cdot \log n)$. For each vertex we need to choose up to p colors in the range $1, \ldots, O(p \cdot \log n)$. The number of different possible preemptive assignments of colors to a vertex v is $\binom{O(p\cdot\log n)}{p}$. This is polynomial in n due to the bound on p when $p = O(\log n / \log\log n)$. Hence, the following corollary.

Corollary 3.3 *The* `p-sum` *and* `p-makespan` *problems on partial k-trees admit polynomial solutions in the case $p = O(\log n / \log\log n)$, k fixed.* □

4 Approximate multicolorings of partial k-trees

We show in this section that one can obtain near-optimal solutions to `p-makespan` on partial k-trees with the additional property of using few preemptions. This leads to a good approximation of `p-sum`,

that also uses few preemptions. First, we give a still better approximation for the non-preemptive case.

4.1 FPTAS for `np-sum` on partial k-trees

The algorithm CompSum is polynomial only when p is polynomially bounded. We now show `np-sum` can in general be approximated within a very small ratio. The same can be shown to hold for `np-makespan`.

Theorem 4.1 `np-sum` *admits a fully-polynomial time approximation scheme (FPTAS) on partial k-trees that runs in $(n/\epsilon)^{O(k)}$ time.*

Proof. Let ϵ be given and let $q = \lfloor \epsilon p/n^2 \rfloor$. We may assume that $q \geq 1$, as otherwise algorithm CompSum yields an exact solution in polynomial time by Corollary 3.2.

The algorithm is as follows. Form a new instance $I' = (G, x')$ by rounding the color requirements of the input $I = (G, x)$ upwards to the nearest multiple of q. Apply the algorithm CompSum on the quotient instance I'/q, and obtain an optimal solution to I'/q. By Lemma 2.8, this also gives an optimal solution to I' formed by repeating each color q times.

To analyze the same solution for I, let us first relate solutions of I' to those of I. Given a solution of I, we can form a solution of I' by repeating q times, for each node v, some color class containg v. The combined delay caused by repeating a single color class is at most qn. In total, we must repeat (at most) n color classes, one per each vertex. Thus, the total additional cost is bounded above by qn^2. Hence,

$$OPT(I) \leq OPT(I') \leq OPT(I) + qn^2 \leq OPT(I) + \epsilon p \leq (1+\epsilon)OPT(I).$$

Thus, the solution produced by our algorithm is within a factor of $1+\epsilon$ from optimal.

The time complexity of the method is polynomial in $p/q \approx n^2/\epsilon$, amounting to $O((n^2\epsilon^{-1}\log n)^{k+1} = O(n^{2k+3}\epsilon^{-(k+1)})$. □

4.2 PTAS for `p-makespan` using $O(\log n)$ preemptions

Preemptions are a resource that may be desirable to conserve. From the point of view of a scheduler, a preemption is likely to cost some overhead in changing to and from an active state. Limited use of preemptions is also preferable for our multisum approximations, in hindsight of our algorithms; in a sense, such colorings have low entropy.

In our case we can bound the number of rounds when some vertex turns active, and refer to this as the maximum number of preemptions per vertex.

Theorem 4.2 *The* `p-makespan` *problem on partial k-trees, k fixed, admits a PTAS that uses $O(\log n)$ preemptions per vertex and runs in time $n^{O(k^2/\epsilon^3)}$.*

The theorem follows immediately from Lemma 4.3 below along with Theorem 3.1. A family of colorings will here be said to be *universal* if it depends only on n, p, and a given k-partition of the n vertices. Universal families do not depend on the actual structure of the graph, nor on the color requirements.

Lemma 4.3 *There is a searchable universal family of multicolorings $\mathcal{F}$ with $\mathcal{D}(\mathcal{F})$ polynomial in n, such that for any k-colorable graph G, $\mathcal{F}$ includes a coloring that approximates the makespan of G by a $1+\epsilon$ factor. Additionally, the number of preemptions per vertex used by any coloring in $\mathcal{F}$ is $O(\log n)$.*

Proof. We recall the family in the proof of Lemma 2.6. The instance I is split into an instance I'' and an instance $\bar{I}$, where $\bar{I}$ has $\bar{p} = O(\log n/\epsilon)$. A coloring of I corresponds to a non-preemptive coloring of I'' and after that a coloring of $\bar{I}$ repeated $q = 2^{\omega-\alpha}$ times (see the lemma). Now, count

the number of possible colorings of a vertex v. Recall that the maximum color-requirement in I'' is bounded by $kp\epsilon/c < (\epsilon/2)p$. Further, I'' is colored non-preemptively one color class after the other. Hence, we can assign in advance $p\epsilon/c$ fixed (disjoint) colors to each color-class (independent of the instance). Hence, for the first (roughly) $\epsilon/2\mu(I)$ colors the coloring is independent of the instance (depends only on k and p) and is of one of (only) k types. What determines $\mathcal{D}$ is thus the coloring of $\bar{I}$. Now,

$$\mathcal{D}(\mathcal{F}, v) \leq \binom{\bar{p} \cdot k}{\bar{p}} \leq 2^{\bar{p}k} = n^{O(kc/\epsilon_1)} = n^{O(k/\epsilon^3)}.$$

Taking q consecutive copies of each color does not affect this bound. □

It may seem somewhat surprising that such a universal coloring can approximate simultaneously all the multicoloring instances. It should be clear from the proof that the different colorings of a given vertex can be computed efficiently.

4.3 PTAS for p-sum using $O(\log n)$ preemptions

We now give a PTAS for the sum measure, building on the makespan result.

Theorem 4.4 *The* p-sum *problem on partial k-trees admits a PTAS using $O(\log n)$ preemptions per vertex with a running time of $n^{O(k^2/\epsilon^5)}$.*

This theorem follows immediately from the following lemma. The existence of a family with $\mathcal{D}$ polynomial means that we can use CompSum to find the desired coloring.

Lemma 4.5 *There is a searchable universal family of multicolorings $\mathcal{F}$ with $\mathcal{D}$ polynomial in n, such that for any k-colorable graph G, $\mathcal{F}$ includes a schedule that approximates* p-sum(G) *within $1+\epsilon$. Additionally, each coloring in $\mathcal{F}$ has $O(\log n)$ preemptions per vertex.*

We show this by transforming an optimum p-sum solution to an approximate solution with the desired restricted structure that colorings in $\mathcal{F}$ have. The exact schedule is divided into segments, and each segment considered as a makespan instance, for which we use the restricted approximate solutions of Lemma 4.3. We also want to ensure that the coloring of each vertex is limited to a compact interval. Thus, we delete all very small and very large colors assigned to the vertex, and schedule them in small blocks in between the intermediate segments. This will ensure that only constant number of segments are active for any given vertex, and therefore there is a $1+\epsilon$-approximate searchable family of colorings for the graph.

Proof. Let ϵ_4 and ϵ_5 be small values to be chosen later, and let $\epsilon' = \epsilon_5/k$. Partition the colors $\{1, 2, \ldots\}$ into geometrically increasing segments, where the length d_i of the i-th segment L_i is $d_i = (1+\epsilon_4)^i$ (ignoring round-off). The set $M(v) = \{(\epsilon'/2)x(v), \ldots, (2/\epsilon')x(v)\}$ is the area of the color space to which we want to confine the coloring of v. Let ψ^* be an optimal p-sum schedule of G. Let $x'(v) = |\psi^*(v) \setminus M(v)|$, and $x_i(v) = |\psi^*(v) \cap L_i \cap M(v)|$. Thus, we obtain instances (G, x') and (G, x_i), $i = 1, 2, \ldots$, that partition the instance (G, x). Namely, $x(v) = x'(v) + \sum_i x_i(v)$.

Our coloring ψ will consist of segments, each containing a *main block* for treating x_i and k *round-robin blocks* for the remaining color requirements x'. The size of the main block is $(1+\epsilon_5)d_i$, and the size of each of the k round-robin blocks is $\epsilon' d_i$, for a total segment size of $(1+2\epsilon_5)d_i$. The starting point of segment i is therefore $z_i = (1+2\epsilon_5)\sum_{j=1}^{i-1} d_j = (1+2\epsilon_5)\frac{(1+\epsilon_4)^i}{2+\epsilon_4}$.

Since (G, x_i) could be colored properly in $d_i = (1+\epsilon_4)^i$ colors, we can obtain by Lemma 4.3 a schedule ψ_i of (G, x_i) with makespan at most $(1+\epsilon_5)d_i$, which we use in the main block.

At most one of the k sets of round-robin blocks is used for any given vertex v. For that purpose, we make use of a k-coloring of G, $\Upsilon : V \to \{0, 1, \ldots, k-1\}$. For each vertex v, and for each segment i, exactly one (more precisely, the one indexed by $\Upsilon(v)$) of the round-robin blocks is used to satisfy the $x'(v)$ requirements.

Let $a = \min_i\{L_i \neq \emptyset\} = \log_{1+\epsilon_4}(\epsilon'/2)x(v)$ be the index of the smallest segment with a non-empty main block. Similarly, let $b = \max_i\{L_i \neq \emptyset\}$ be the index of the largest such segment.

The set of colors assigned to v can be specified precisely as follows. For a set S of integers and integer t, define the set $[S + t] = \{x + t : x \in S\}$. Then,

$$\psi(v) = \bigcup_{i=a}^{b} [\psi_i(v) + z_i] \cup [\{1, 2, \ldots, \epsilon' d_i\} + (z_i + (1 + \epsilon_5)d_i + \Upsilon(v)\epsilon' d_i)].$$

The round-robin block is offset by z_i, the beginning of segment i, $(1 + \epsilon_5)d_i$, the size of the main block, and $\Upsilon(v)\epsilon' d_i$, the combined size of the previous round-robin blocks.

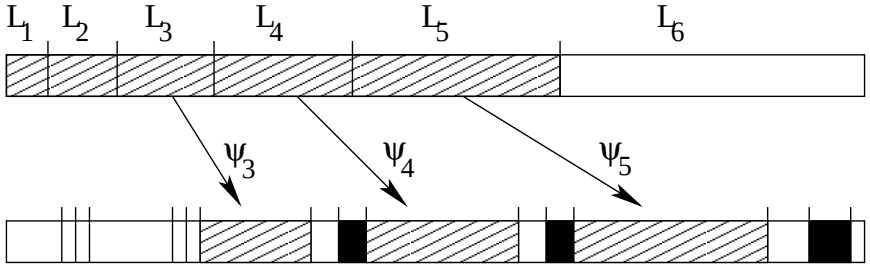

Figure 1: Transformation of $\psi^*(v)$ to $\psi(v)$

We illustrate the construction in Figure 1 for a 2-colorable graph. The upper block shows $\psi^*(v)$ cut into segments L_i. Here, $a = 3$ and $b = 5$, so segments L_3 through L_5 are transformed into main blocks $\psi_3(v)$ through $\psi_5(v)$ in $\psi(v)$. The second of two round-robin blocks are used in segments 3 through 5 for satisfying $x'(v)$, which here amounts to the the colors used in L_1 and L_2.

We first verify that this forms a valid coloring of (G, x). The coloring is proper for each ψ_i, and thus for the corresponding parts of $\psi(v)$. Also, two vertices u and v use the same round-robin segment, then it must be that $\Upsilon(u) = \Upsilon(v)$, so they are non-adjacent and their colorings do not conflict.

We now argue that the round-robin blocks used cover the remaining color requirements $x'(v)$. We may allocate more colors than necessary, but that does not hurt. Observe that the round-robin blocks available to v are a ϵ' fraction of the size of the corresponding main blocks. The value $x'(v)$ is composed of two parts: values of $\psi^*(v)$ less than $(\epsilon'/2)x(v)$ and those greater than $(2/\epsilon')x(v)$. Consider first the small values. Let r be the smallest number such that $\sum_{i=1}^{q} d_i \geq x(v)$, i.e. $r = \lceil \log_{1+\epsilon_4} x(v) \rceil$. The round-robin segments allocated to v in segments $1, 2, \ldots, r$ then contain space for $\epsilon' x(v)$ elements. Thus, the segments $a, a + 1, \ldots, r$ contain space for at least $(\epsilon'/2)x(v)$ elements, since $\sum_{i=1}^{r} d_i \geq 2\sum_{i=1}^{a-1} d_i$. Therefore, since $b \geq r$, the small values do not delay the coloring of v in ψ, independent of how v was colored by ψ^*,

Consider now the case when $x'(v)$ contained some large values of ψ'. Thus, the index of the largest segment containing a non-empty main block is $b = \log_{1+\epsilon_4}(2/\epsilon')x(v)$. Then, the round-robin blocks available to v in segments $1, 2, \ldots, b$ contain space for $2x(v)$ elements. Hence, those in segments $a, a + 1, \ldots, b$ contain space for at least $x(v)$ elements.

The finish time of a vertex v in $\psi(v)$ depends on two factors: its final segment in $\psi^*(v)$, and its location within that segment of $\psi(v)$. The segments expand from $\psi^*(v)$ to $\psi(v)$ by a factor of $1 + 2\epsilon_5$. The colors that a vertex receives within a segment is beyond our control, as the segments are handled as makespan instances where only the maximum number of colors used is relevant. This may delay the vertex by what amounts the full size of the last segment in which it was colored, or $\epsilon_4 x(v)$. Hence, $f_\psi(v)$ is bounded above by $(1 + 2\epsilon_5 + \epsilon_4)f_{\psi^*}(v)$.We choose $\epsilon_4 = \epsilon_5 = \epsilon/3$ to ensure $f_\psi(v) \leq (1 + \epsilon)f_{\psi^*}(v)$.

The number of segments used, $b-a+1$, is at most $\log_{1+\epsilon_4}(2/\epsilon')\frac{x(v)}{\epsilon' x(v)} = \log_{1+\epsilon_4}(2/\epsilon')^2$, which is $O(1/\epsilon \cdot \log 1/\epsilon)$. In each of these segments a vertex v has $O(1/\epsilon^3 \cdot \log n)$ preemptions, by Lemma 4.3. Hence, the total number of preemptions for each vertex is $O(1/\epsilon^4 \cdot \log 1/\epsilon \cdot \log n)$. Let $\mathcal{F}$ denote the family of all possible legal colorings in the above restricted form. Thus, in a way similar to Lemma 4.3, $D(\mathcal{F}, v)$ is polynomially bounded in n. □

5 Sum multicoloring planar graphs

We give a PTAS for both preemptive and non-preemptive cases on planar graphs, starting with the easier preemptive case. For this, we rely heavily on the result obtained in the previous section for partial k-trees. We match them with an NP-hardness result for the unit-length Sum Coloring problem.

The results hold also for other classes of graphs that are constant colorable and can be partitioned into partial k-trees, such as $K_{3,3}$-free graphs [C98].

5.1 NP-hardness of sum coloring planar graphs

It is clear that `p-makespan` and `np-makespan` are NP-hard on planar graphs, as they extend the NP-hard minimum coloring problem on planar graphs (cf. [GJ79]). We prove that already the sum-coloring problem, `SC`, is NP-complete for planar graphs.

Theorem 5.1 *The* `SC` *problem and* `SMC` *problem are NP-complete on planar graphs.*

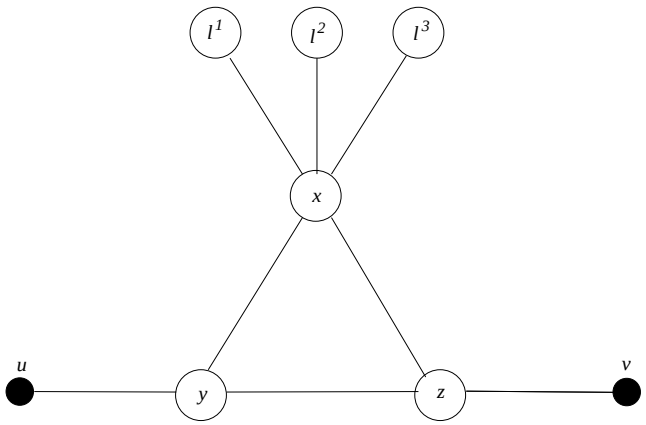

Figure 2: Gadget for the edge uv.

The reduction is from the maximum independent set (`MIS`) problem on planar graphs, which is NP-complete (cf. [GJ79]). Given a planar graph $G(V, E)$, we construct a graph $\tilde{G}(\tilde{V}, \tilde{E})$ by replacing each edge of G by the gadget shown in Fig. 2. For edge $e = uv$ in G we add to $\tilde{G}$ the vertices $x_e, y_e, z_e, l_e^1, l_e^2, l_e^3$ and the edges $x_e y_e, x_e z_e, y_e z_e, x_e l_e^1, x_e l_e^2, x_e l_e^3$ as well as uy_e and vz_e. Clearly $\tilde{G}$ is planar when G is planar.

Let $\alpha(G)$ be the size of the maximum independent set in G. Theorem 5.1 follows from the following lemma.

Lemma 5.2 $\mathtt{SC}(\tilde{G}) = 9 \cdot |E| + 2 \cdot |V| - \alpha(G)$.

Proof. We first show the upper bound by constructing a coloring for $\tilde{G}$ given a maximum independent set I^* of G. Color the vertices of I^* with 1 and other vertices coming from G with 2. Color the gadget of each edge uv as follows. Color x with 2, and the leaves l^i with 1. If u is colored 1,

then by the choice we made v is not colored 1. Color y with 3 and z with 1; otherwise, color y with 1 and z with 3. This forms a valid coloring of $\tilde{G}$ where the cost of coloring each edge-gadget is 9, and the cost of coloring the vertices of G is $2|V| - \alpha(G)$.

We now argue a matching lower bound on $\texttt{SC}(\tilde{G})$, claiming that the coloring constructed above is in a sense canonical. Let ψ^* be an optimal sum coloring of $\tilde{G}$. We claim that vertices of G colored with 1 in ψ^* must form an independent set in G. Assuming that claim, the minimum cost of coloring the vertices of G is $2|V| - \alpha(G)$. Since the the minimum cost of coloring each gadget is 9, the minimum cost of coloring $\tilde{G}$ is at least $9|E| + 2|V| - \alpha(G)$. Note that u and v are themselves not part of the gadget for the edge uv.

To prove the claim, suppose for the sake of obtaining a contradiction that for some edge uv in $\tilde{G}$, both u and v are colored 1 in ψ^*. We then form another coloring ψ' of $\tilde{G}$ by changing the color of u to 2, and recoloring the gadget of each incident edge uv' as needed. Namely, the leaf nodes of the gadgets are colored 1, x-nodes 2, y-nodes 1, and z-nodes 3 (unless v' was colored 3, in which case we color z with 2 and x with 3). It can be verified that the recoloring never increases the cost of a gadget. If both u and v' were colored 1, then the cost of coloring the gadget compatibly was at least 12 (as the "triangle" x_e, y_e, z_e has colors 2, 3, and 4 or larger colors, since color 1 is prohibited). Our transformation thus decreased the cost of the gadget by at least 3, while increasing the cost of u by only 1. Hence, we have obtained a coloring of lower cost, which is a contradiction. Hence, the claim and the lemma follow. □

The lemma implies a linear relationship between the approximability of MIS and SC on planar graphs. This uses the fact that for a planar graph G, $\alpha(G) \geq |V|/4$ and that by Euler's theorem, $|E| \leq 3|V|$ (see [H69]).

Corollary 5.3 *Let $f(n)$ be a monotone non-increasing function. If* SC *can be approximated on planar graphs in polynomial time within a ratio of $1 + f(n)$, then* MIS *can be approximated on planar graphs in polynomial time within a ratio of $1 + 30f(n)$.*

Proof. Assume that we are given a procedure A that can approximate the SC problem within $1+f(n)$. Let G be a graph for which we want to approximate the size of the maximum independent set. Consider the graph $\tilde{G}$ defined above. Let ψ^* be the minimum sum coloring for $\tilde{G}$. By Lemma 5.2, we may assume that this coloring assigns color 1 to a maximum independent set and that it completes the coloring as described in the proof of Lemma 5.2. Let I_A denote the vertices colored 1 by A. Observe that $\texttt{SC}(\tilde{G}) = \texttt{SC}(\psi^*) = 9 \cdot |E| + 2 \cdot |V| - \alpha(G)$. By our assumption, $9 \cdot |E| + 2 \cdot |V| - |I_A| \leq (1 + f(n)) \cdot (9 \cdot |E| + 2 \cdot |V| - \alpha(G))$. Rearranging the terms, we get

$$|I_A| \geq f(n)(9|E| + 2|V|) + (1 + f(n))\alpha(G) \geq 29f(n)|V| + (1 + f(n))\alpha \geq (1 + 30f(n))\alpha(G).$$

The claim now follows. □

5.2 PTAS for p-sum

The following well-known decomposition lemma of Baker [B94] will be used for both np-sum as well as p-sum.

A class of plane graphs are *outerplanar* if all vertices are on the exterior face. More generally, the class of *t-outerplanar* graphs(cf. [B94]) are defined to be the outerplanar graphs when $t = 1$, and inductively, when $t > 1$, graphs such that the graph induced by vertices not on the exterior face is $t - 1$-outerplanar. The only property of t-outerplanar graphs that is relevant here is that they are of treewidth at most $3t - 1$ [B98]. The weight of a graph is the sum of the weights of the vertices. We view color requirements as vertex weights.

Lemma 5.4 (Planar decomposition) *Let G be a planar graph, and t be a positive integer. Then G can be decomposed into two vertex-disjoint graphs: G_b, which is t-outerplanar, and G_a, which is outerplanar with at most $2n/t$ vertices and at most $2\,\mathcal{S}(G)/t$ weight.*

We briefly recall how this decomposition is done. Given a planar embedding of the graph, let L_0 be the set of vertices on the exterior face, and inductively let L_i be the exterior vertices of the graphs induced by $V(G) - \cup_{j=0}^{i-1} L_j$, $i = 1, \ldots, t$.

For a given j, $0 \leq j < t$, let $U_j = \cup_{i=0} L_{it+j}$. Namely, U_j consists of all the layers whose index is congruent to j modulo t. By a simple averaging argument, there must be some value j, $0 \leq j < t+1$ such that $|U_j| \leq 2n/t$ and $\mathcal{S}(U_j) \leq 2\,\mathcal{S}(G)/t$ (because fewer than $k/2$ of the U_j fail on either one of these two properties). For this value of j, let $V_a = U_j$, let $V_b = V - V_a$, and let G_a (G_b) be the graph induced by V_a (V_b). Then G_a consists of disjoint outerplanar graphs, and thus is outerplanar, and similarly G_b consists of disjoint t-outerplanar graphs, and thus is t-outerplanar. □

The following lemma relates approximations of planar graphs to those of partial k-trees.

Lemma 5.5 *A ρ-approximation for* `p-sum` *on partial k-trees for any fixed k, implies a $\rho(1+\epsilon)$-approximation for planar graphs, for any $\epsilon > 0$.*

Proof. Let t be a constant to be determined. Decompose G into G_1 and G_2, with G_1 t^2-outerplanar, and G_2 outerplanar, following Lemma 5.4. Then, $\mathcal{S}(G_2) \leq 2\,\mathcal{S}(G)/t^2$. Use the assumed approximation of `p-sum` on partial k-trees to get solutions ψ_1 and ψ_2 whose sums are bounded by $\rho \cdot OPT(G_1)$ and $\rho \cdot OPT(G_2)$. Then, use a biased round-robin as follows: after each group of $t-1$ color classes of ψ_1, insert the next color class of ψ_2. Clearly, the finish times of each of the vertices in G_1 is multiplied by at most $1 + 1/t$, and that of a vertex in G_2 by t. Note, that since G_2 is 4-colorable, $OPT(G_2) \leq 4\,\mathcal{S}(G_2)$. Hence, the four-colorability of G_2 gives that

$$OPT(G_2) \leq 4\,\mathcal{S}(G_2) \leq 8\,OPT(G)/t^2.$$

Therefore, the cost of the sum coloring of G is bounded above by

$$\rho((1+1/t)OPT(G_1) + t \cdot OPT(G_2)) \leq \rho(1+9/t) \cdot OPT(G).$$

Choosing $t = 1/9\epsilon$ yields the lemma. □

The following theorem now follows from Lemma 5.5 and Theorem 4.4.

Theorem 5.6 *The* `p-sum` *problem on planar graphs admits a PTAS whose running time is $n^{O(1/\epsilon^5)}$.*

5.3 PTAS for `np-sum`

We now turn to the non-preemptive case. Given Theorem 2.10, the missing link is in solving planar graphs with small ratios $\tau(G)$ between maximum and minimum color requirements. First, we need a variation on the number of colors used. Let $OPT(G)$ be an optimal multicoloring sum of G.

Corollary 5.7 *At most $OPT(G)/(c \cdot p)$ vertices remain to be completed in an optimal sum multicoloring of G by step $p + 2\mu(G)\lg c$, for any positive $c > 1$.*

Proof. By step p, at most $OPT(G)/p$ vertices remain to be completed (for otherwise the delay is more than the optimum). By Lemma 2.1, after additional $2\mu(G)\lg c$ rounds, the number of remaining vertices is down to at most $OPT(G)/(cp)$. □

Lemma 5.8 (Compact lengths) *Let (G, x) be a planar instance with $\tau(G) = O(\log n/(\log\log n)^3)$, and let $\epsilon = \epsilon(n)$. Then,* `np-sum`(G) *admits a $1+\epsilon$-approximation using $O(p \cdot \log \epsilon^{-1})$ colors in time $(\log n)^{O(\tau(G)\epsilon^{-1}\log\epsilon^{-1})} n$.*

Proof. Let $h = h(n)$ be determined later, and let $d = d(n) = hp/(OPT(G)/n)$ and $b = b(n) = 1 + 8\lg h$. Note that

$$d = \frac{hp}{OPT(G)/n} \leq \frac{hp}{\mathcal{S}(G)/n} \leq \frac{hp}{p_{min}} = h\tau(G).$$

We apply the following approach.

1. Partition V via Lemma 5.4 into V_1 and V_2, where V_1 induces a d-outerplanar graph G_1 while $|V_2| \leq 2n/d$.

2. Sum multicolor G_1 nearly-optimally, using the rounding lemma 2.9 with CompSum on the reduced instance of maximum color requirement $\tau(G)/\epsilon$. Use the first $b \cdot p = (1 + 8 \log h)p$ colors in this solution (discarding the remaining color-classes), and let $\hat{V}$ be the set of vertices not fully colored by these colors.

3. Color $V_2 \cup \hat{V}$ using a graph 4-coloring algorithm, yielding a multicoloring with at most $4p$ colors.

The cost of coloring V_1, and thus that of coloring $V_1 - \hat{V}$, is at most $(1+\epsilon)OPT$. By Lemma 5.7, $\hat{V}$ contains at most $OPT(G)/(hp)$ vertices. Also, V_2 contains at most $n/d = OPT(G)/(hp)$ vertices. Hence, the cost of coloring $V_2 \cup \hat{V}$ is at most

$$(b+4)p \cdot 3\, OPT(G)/(hp) = \frac{15 + 24 \lg h}{h} OPT(G).$$

The $4p$ term here reflects a bound on the cost per each vertex in a four-coloring, while the $b \cdot p$ term reflects the delay of $V_2 \cup \hat{V}$. Now set h to make the above expression at most $\epsilon\, OPT$. Thus $h = O(\epsilon^{-1} \cdot \log \epsilon^{-1})$. Then, the total cost of the coloring is at most $(1 + 2\epsilon)OPT$.

The complexity of our algorithm depends primarily on CompSum. Recall that d is at most $h \cdot \tau(G)$, and without loss of generality $\epsilon^{-1} = O(\log n)$. By Corollary 3.2, the scaled instance is solved in time

$$(\epsilon^{-1}\tau(G) \log n)^{O(d)} n = (\log n)^{O(h\tau(G))} n = (\log n)^{O(\tau(G)\epsilon^{-1} \log \epsilon^{-1})} n.$$

The number of colors used is $(b+4)p = (5 + 8 \lg h)p = O(p \log \epsilon^{-1})$. □

Theorem 5.9 *Let $f_\epsilon = \epsilon^{-1} \log \epsilon^{-1}$. The* `np-sum` *problem on planar graphs admits a PTAS using $O(p \cdot \log \epsilon^{-1})$ colors whose running time is $n \cdot exp(\ln \ln n \cdot exp(O(f_\epsilon^2)))$.*

Proof. By Lemma 5.8, `np-sum` on short instances (with $\tau(G)$ at most a given q) admits a $1+\epsilon$-approximation in time $n \cdot (\log n)^{O(q \cdot f_\epsilon)}$. The number of colors used is σp, for $\sigma = c \log \epsilon^{-1}$ for some constant c. Let q be such that $\sigma/\sqrt{\ln q} = \epsilon$, or $q = exp((cf_\epsilon)^2)$. Applying Theorem 2.10 we obtain an approximation of `np-sum` within a $1 + 2\epsilon$ factor. The time complexity is

$$n \cdot (\log n)^{O(q \cdot f_\epsilon)} = n \cdot exp(\ln \ln n \cdot O(exp((cf_\epsilon)^2) \cdot f_\epsilon)) = n \cdot exp(\ln \ln n \cdot exp(O(f_\epsilon)^2)).$$

In particular, we can obtain a $1 + \sqrt{\ln \ln n}/\ln \ln \ln n$-approximation in sub-quadratic time $n^{1+O(\log \log \log n / \log \log n)}$. □

For Sum Coloring, we obtain better tradeoffs, since we can solve exactly partial k-trees for $k = O(\log n / \log \log n)$, by directly applying the Compact Lengths Lemma 5.8.

Theorem 5.10 `SC` *on planar graphs admits a PTAS with running time of $exp(O(\ln \ln n \cdot f_\epsilon)) \cdot n$.* □

6 Open problems

Our research leaves some open problems, of which we mention one. The fact that `p-makespan` and `np-makespan` are solvable on bipartite graphs easily yields a 2-approximation for these problems on planar graphs. Further, an approximation better than 4/3 does not exist, unless $P = NP$. What then is the approximation threshold of these problems on planar graphs? Can the reduction of the maximum color to $O(\log n)$ help in designing an $4/3 + \epsilon$ approximation for any ϵ?

Acknowledgements

This paper owes its existence to Amotz Bar-Noy, for bringing the authors together. We thank him for his support and many early discussions. We also thank Hadas Shachnai and Hermann Thorisson for helpful discussions and Barun Chandra for important comments.

References

[B98] H. L. Bodlaender. A partial k-arboretum of graphs with bounded treewidth. *Theoretical Computer Science*, 209:1–45, 1998.

[B92] M. Bell. Future directions in traffic signal control. *Transportation Research Part A*, 26:303–313, 1992.

[B94] B. S. Baker. Approximation algorithms for NP-complete problems on planar graphs. *J. ACM*, 41:153–180, Jan. 1994.

[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.

[BH94] D. Bullock and C. Hendrickson. Roadway traffic control software. *IEEE Transactions on Control Systems Technology*, 2:255–264, 1994.

[BK98] A. Bar-Noy and G. Kortsarz. The minimum color-sum of bipartite graphs. *Journal of Algorithms,* 28:339–365, 1998.

[BHK+98] A. Bar-Noy, M. M. Halldórsson, G. Kortsarz, H. Shachnai, and R. Salman. Sum Multi-Coloring of Graphs. In *Proc. Seventh European Symposium on Algorithms* (ESA '99), Lecture Notes in Computer Science Vol. 1643, Springer-Verlag, July 1999. To appear in *Journal of Algorithms.*

[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.

[C98] Z.-Z. Chen. Efficient Approximation Schemes for Maximization Problems on $K_{3,3}$-Free Graphs. *J. Algorithms* 26(1):166–187, 1998.

[Cof85] E. G. Coffman, M. R. Garey, D. S. Johnson, and A. S. Lapaugh. Scheduling file transfers. *SIAM J. Comp.* 14:744-780, 1985.

[FK98] U. Feige and J. Kilian. Zero Knowledge and the Chromatic number. *Journal of Computer and System Sciences*, 57(2):187-199, October 1998.

[GJ79] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-completeness.* W. H. Freeman, 1979.

[Ger96] J. Gergov. Approximation algorithms for dynamic storage allocation. In *Proc. 4th Ann. European Symp. on Algorithms*, Lecture Notes in Comput. Sci. 1136, Springer-Verlag, 52-61, 1996.

[GLS88] M. Grötschel, L. Lovász and A. Schrijver. *Geometric Algorithms and Combinatorial Optimization*, Springer-Verlag, 1988.

[H69] F. Harary, Graph Theory. Addison-Wesley, 1969

[H94] C. T. Hoang, Efficient algorithms for minimum weighted coloring of some classes of perfect graphs. *Discrete Applied Math*, 55:133-143, 1994.

[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.

[HK+99] M. M. Halldórsson, G. Kortsarz, A. Proskurowski, H. Shachnai, R. Salman, and J. A. Telle. Multi-Coloring Trees. In *Proceedings of the Fifth International Computing and Combinatorics Conference*, Tokyo, Japan, Lecture Notes in Computer Science Vol. 1627, Springer-Verlag, pages 271–280, July 1999.

[HKS01] M. M. Halldórsson, G. Kortsarz, and H. Shachnai. Minimizing Average Completion of Dedicated Tasks and Interval Graphs. In *Proceedings of the Fourth International Workshop on Approximation algorithms* (APPROX '01). Lecture Notes in Computer Science, Springer-Verlag, August 2001.

[J97] K. Jansen. The Optimum Cost Chromatic Partition Problem. In *Proceedings of the Third Italian Conference on Algorithms and Complexity (CIAC '97)*. Lecture Notes in Computer Science Vol. 1203, pp. 25–36, Springer-Verlag, July 1997.

[KKK89] E. Kubicka and G. Kubicki, and D. Kountanis. Approximation Algorithms for the Chromatic Sum. *Proceedings of the First Great Lakes Computer Science Conf.*, Lecture Notes in Computer Science Vol. 507, pp. 15–21, Springer-Verlag, July 1989.

[KS89] E. Kubicka and A. J Schwenk. An Introduction to Chromatic Sums. In *Proceedings of the Seventeenth Annual ACM Computer Science Conference, "Computing trends in the 1990's"*, pages 39–45, 1989.

[L81] N. Lynch. Upper Bounds for Static Resource Allocation in a Distributed System. *J. of Computer and System Sciences*, 23:254–278, 1981.

[MR97] C. McDiarmid and B. Reed. Channel assignment and weighted coloring. *Networks*, 36:114–117, 2000.

[MR95] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.

[NS93] M. Naor and L. Stockmeyer. What Can be Computed Locally? *Proceedings of the Twenty Fifth Annual Symposium on the Theory of Computing*, pp. 184–193, 1993.

[NS97] L. Narayanan and S. Shende. Static Frequency Assignment in Cellular Networks. In *Proceedings of the Fourth Colloquium on Structural Information and Communication Complexity*, July 1997. To appear in *Algorithmica*.

[NSS99] S. Nicoloso, M. Sarrafzadeh, and X. Song. On the Sum Coloring Problem on Interval Graphs. *Algorithmica*, 23:109–126, 1999.

[S-99] T. Szkaliczki. Routing with Minimum Wire Length in the Dogleg-Free Manhattan Model is NP-complete. *SIAM Journal on Computing*, 29(1):274-287, 1999.

[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.

[T95] A. S. Tanenbaum, *Distributed Operating Systems.* Prentice-Hall, 1995.

[X98] J. Xue. Solving the minimum weighted integer coloring problem. *Computational Optimization and Application* 11(1):53–64, 1998.

A General cost functions

We consider here a general class of cost functions, to which the CompSum algorithm is extended.

To consider first some concrete examples of cost functions, one example is the sum of squares of completion times, $\sum_v f_\psi(v)^2$. This could be viewed as the L_2-metric, with L_1 being the multicolor sum and L_∞ the makespan. Any L_p-metric can be handled within our framework.

Another example is the $d(v)$**-coloring** problem which comes with edge lengths $d : E \to Z^+$ and asks for an ordinary coloring where the colors of adjacent vertices are further constrained to satisfy $|f_\psi(v) - f_\psi(w)| \geq d(vw)$. 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.

Each cost function $\mathcal{C}$ associates a positive real number with each legal multicoloring. We need the function $\mathcal{C}$ to be also defined for partial colorings, i.e., legal multicolorings ψ of only a subset $S \subseteq V$ of the vertices. In this case, the value of the coloring is defined by the graph induced by S. We denote this cost by $\mathcal{C}(\psi, S)$. The goal is to compute a legal multicoloring (of the entire graph) with minimum $\mathcal{C}$ value. We show that CompSum can be generalized to the following family of functions.

Let g_1 and g_2 be two multicolorings of subsets $A \subseteq V$ and $B \subseteq V$, respectively, with g_1 and g_2 agreeing on $A \cap B$. Then the extension function $\mathcal{G}_x(g_1, g_2)$ of g_1 and g_2 is the function coloring $A \cup B$ according to g_1 and g_2. The functions we deal with obey the following properties.

- **Monotonicity property:** A cost function is monotonic if it is preferable to color as many vertices as possible with small colors. Formally, a cost function is monotonic if the following holds. Say that we take a coloring ψ, and "move" a vertex v colored j, into a smaller color class i, $i < j$, resulting in a new (legal) coloring ψ'. Then, $\mathcal{C}(\psi') \leq \mathcal{C}(\psi)$.

- **Composition property:** A function obeys the Composition property if the following holds. Given A and B and a coloring g of $A \cap B$, the optimum way of extending g to $A \cup B$ is by choosing the optimum coloring g_A of A among the colorings agreeing with g on $A \cap B$, and similarly taking the optimum coloring g_B of B agreeing with g on $A \cap B$ and choosing $\mathcal{G}_x(g_A, g_B)$. Further, if we have a coloring g of A, the optimum way to extend g to $A \cup B$ is by choosing an optimum coloring of B, among the colorings agreeing with g on $A \cap B$. Finally, the cost of $\mathcal{C}(\mathcal{G}_x(g_A, g_B), A \cup B)$ can be computed in polynomial time from $\mathcal{C}(g_A, A)$ and $\mathcal{C}(g_B, B)$.

Beside the examples mentioned above, another previously studied monotonic cost function appears in the Optimum Chromatic Cost Problem (OCCP) (see [J97]). OCCP generalizes the Sum Coloring problem in that the color classes come equipped with a cost function $c : Z^+ \to Z^+$ and we assign a single color $f(v)$ to each vertex minimizing $\sum_{v \in V} c(f(v))$. Without loss of generality,

we may assume that the color costs are non-decreasing. We can generalize this to multicolorings, where the cost incurred for vertex v is $c(f_\psi(v))$.

We first show that any monotonic cost function $\mathcal{C}$ the optimum uses few colors on certain graphs, that include planar graphs and partial k-trees. We then derive a generalization of the CompSum algorithm of Section 3 to monotonic cost functions.

A.1 Number of colors

The following discussion shows that on certain graphs, the number of colors an optimum algorithm (under a monotonic cost function) needs to optimally multicolor the graph is "small". This applies both to the preemptive and non-preemptive cases. The following lemmas extend a similar lemma of the work in [J97] that applies to the unweighted case (with $x(v) = 1$ for all v). It is interesting to note that this lemma applies to a family of graphs that contains among others planar graphs and partial k-trees: the family of inductive graphs.

We say that a graph $G(V, E)$ is *t-inductive* if for any subgraph $G'(V', E')$ of G, $|E'| \leq t \cdot |V'|$. Also, G is *inductive*, if there exist a constant t such that G is t-inductive. In particular, planar graphs are 5-inductive. Also partial k-trees are k-inductive since any subgraph of a partial k-tree is also a partial k-tree.

Lemma A.1 *An optimal sum multicoloring of an inductive graph (under a general monotonic cost function) uses at most $O(p \cdot \log n)$ colors.*

We prove the lemma in two claims below.

Claim A.2 *Let G be t-inductive, and let (G_i, x_i) be the instance induced by the colors $i, i+1, \ldots$ of some optimal preemptive solution, under a monotonic cost function. Then, the number of vertices in $G_{i'}$ is at most half that of G_i, when $i' = i + (8t + 2)p$.*

Proof. It suffices to consider each connected component of G_i independently, thus for simplicity assume G_i is connected. Let n_i denote the number of vertices of G_i. Let t be the largest integer such that $n_{i+t} \geq n_i/2$. In each iteration, a vertex is either colored or adjacent to a vertex that gets colored, by maximality of the independent set colored. Thus, in each iteration $j = i, i+1, \ldots, i+t$, either at least $n_i/4$ vertices are colored or at least $n_i/4$ vertices are dominated by colored vertices. There can be at most $4p$ iterations of the former type before the sum of the color requirements becomes zero. Let us now concentrate on the latter type of iterations, each involving at least $n_i/4$ edges incident on colored vertices. Each edge (u, v) can dominate, or delay, a vertex at most $x(u) + x(v) \leq 2p$ times. Since there are at most tn_i edges by inductiveness, total counts of dominations is at most $2tpn_i$. Hence, the number of such iterations is at most $8tp$. Thus, $t \leq (8t + 2)p$. □

We now prove a similar claim for the non-preemptive case. Again we deal with a monotonic cost function $\mathcal{C}$ and some optimum non-preemptive coloring under $\mathcal{C}$. Note that in the non-preemptive case the color-classes are not necessarily maximal. Hence, a different proof is needed. However, we may assume that for each vertex v, there is no smaller color-class in the optimum in which we can (non-preemptively) insert v (and keep the coloring legal).

Claim A.3 *Each $O(p)$ iterations in an optimum solution on inductive graphs, halves the maximum size of any connected component remaining.*

Proof. Take a component with n' vertices, and consider all following iterations having a subcomponent of at least $n'/2$ vertices. Consider the first p iterations. Let N be the number of vertices chosen in one of these first p iterations. If a vertex v is chosen in one of these iterations, then v is deleted in at most $2 \cdot p$ iterations (recall that we are dealing with the non-preemptive case now.) If v is not chosen in any of the first p iterations, it must have a neighbor chosen in one of these first p iterations. Thus, at time $2 \cdot p$ a neighbor of v is deleted. This means that after $2 \cdot p$

iterations, at least $n'/2 - N$ edges of the graph are deleted (the total number of vertices is at least $n'/2$, and an edge is deleted for each non-chosen vertex.) Since $\max\{N, n'/2 - N\} = \Omega(n)$ a proof similar to the proof of Claim A.2 gives the desired result. □

A.2 Adapting CompSum to deal with monotonic functions

It is now immediate to adapt CompSum for partial k-trees to deal with monotonic functions. Again, for any supervertex X_i, we let $M_i(g)$ denote the value of the best assignment of colors that extends g from X_i to U_i. As in the case with multisum objective, we need to compute $M_i(g)$ going exhaustively over all possible functions g. Again, we use the fact that few colors are used to bound the number of functions g to be considered. By the composition property, we only need to "guess" (search exhaustively) the coloring of the root in the optimum. Once we get the "correct" coloring g of the root we know from the composition property that we get an optimum coloring of U_i extending the coloring optimally on the subtrees. The value of the optimal extensions is computed by the recursive calculation.

We now verify the running time of the procedure. By Lemma A.1, the number of possible colorings of X_i is $O((p \cdot \log n)^k)$. By the composition property, the optimum coloring of $\mathcal{T}_i$ (the subtree rooted at X_i) can be computed in polynomial time $P(n)$. It follows that in time $O(P(n) \cdot n \cdot (p \cdot \log n)^k)$ the optimum assignment $\mathcal{C}$ is computed.

As in Section 3.1 two corollaries apply.

Corollary A.4 *The non-preemptive coloring problem under any monotonic cost function admits an exact algorithm in $O(P(n) \cdot n \cdot (p \cdot \log n)^{k+1})$ time.* □

Corollary A.5 *The preemptive problem under any monotonic cost function admits a polynomial solution in the case $p = O(\log n / \log\log n)$, k fixed.* □