Multicoloring: Problems and Techniques

Magnús M. Halldórsson[1] and Guy Kortsarz[2]

[1] Department of Computer Science, University of Iceland, IS-107 Reykjavik, Iceland.
mmh@hi.is
[2] Department of Computer Science, Rutgers University, Camden, NJ 08102.
guyk@camden.rutgers.edu

1 Multicoloring Graphs: Problems, Measures, Applications

A *multicoloring* is an assignment where each vertex is assigned not just a single number (a "color") but a set of numbers. The number of colors assigned to the vertex is specified by the *length* (or *color requirement*) parameter of that vertex in the input. As usual, adjacent vertices cannot receive the same color; thus here, the sets of colors they receive must be disjoint. Multicolorings are therefore proper generalizations of ordinary graph colorings. The purpose of this paper is to summarize some of the techniques that have been developed specifically for obtaining good approximate multicolorings in different classes of graphs.

The multicoloring problem is reserved for the case where there is no restriction on the set of colors that each vertex can receive (except its size) and the objective is to minimize the number of colors used. A different problem is obtained when we require the colors assigned to a vertex to form a contiguous interval – we refer to such an assignment as a *non-preemptive multicoloring*. Yet a different family of problems occurs when we change the objective function; in particular, we will be interested in the minimizing the *sum* of the multicolorings, or the sum of the largest color assigned to each vertex (assuming the colors correspond to the natural numbers).

Applications. Graph coloring has great many applications. One of the classic examples are in timetabling, where we want to assign courses (nodes) to time slots (colors) so that classes that cannot be taught simultaneously (e.g., that share a student/teacher population) are assigned different time slots. If courses are all of the same length, we have an ordinary coloring problem (let us ignore the issue of the number of classrooms). Only in special cases do we have an ordinary graph coloring problem; with lectures of different length, we have a non-preemptive multicoloring problem, while with lectures of identical length but several occurrences within the scheduling time frame, we have a preemptive problem.

Another application is frequency allocation or channel assignment in wireless communication. In a cellular network, communication between mobiles and a base station in a cell is across a narrow frequency channel. Two base stations cannot use the same frequency if it causes interference due to geographic locality.

This is modeled by a graph where the nodes correspond to the base stations and edges represent geographic adjacency [42]. Each node needs to be allocated as many channels, or colors, as there are calls connecting to that base station, resulting in a multicoloring problem.

For many classes of graphs, the multicoloring problem can be translated to the ordinary coloring problem. A vertex v of length $x(v)$ is replaced by a clique of $x(v)$ vertices (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. This is one reason why multicolorings appear less often in the literature.

As the timetabling example indicates, practical applications of graph coloring often relate to scheduling. The graph then represents some constraints or conflicts between the jobs that disallow simultaneous execution. One difference with typical scheduling problems is that they tend to involve a fixed number of "machines", rather than allowing for an unbounded number of vertices of the same color. Another difference is that constraints on jobs in scheduling tend to be either non-existent or based on precedence instead of conflicts. Yet, there are several exceptions to these restrictions/differences.

Viewing a problem as a scheduling or as a graph theory problem is not as trivial a issue as it may seem; these are two (overlapping but) different communities with widely different vocabulary and different perspective. It may even be frowned upon to mix metaphors or borrow different concepts. We, however, advocate freedom from denominational canons in order to benefit from the best of both worlds. As we shall see, this will allow us to map a technique from one area to the other and back. We shall intermix the vocabulary, talking equally of vertices and jobs, colors and rounds (or steps), schedules and colorings.

Measures. The possibility of considering different objective functions is one eye-opening product of the scheduling perspective. The second most common objective function is the *sum of completion times*, or its weighted version. This has been considered for (unit-length) graph coloring as the *sum coloring problem*: the colors are positive integers, and the objective is to minimize the sum of the colors assigned to the vertices. In the multicoloring versions, we sum over the vertices the *finish times*, or the last color assigned to that vertex. In the context of dependent jobs in a system, the sum measure has been seen to favor the users (that own the jobs), while the makespan measure is the favorite of the system (that wants to get done quickly). We use the following notation for the different problems:

`SC`	Minimum sum coloring
`pMC`	Multichromatic number of G, or preemptive makespan multicoloring
`npMC`	Non-preemptive makespan, or fewest colors in a contiguous multicoloring
`npSMC`	Non-preemptive sum multicoloring
`pSMC`	Preemptive sum multicoloring

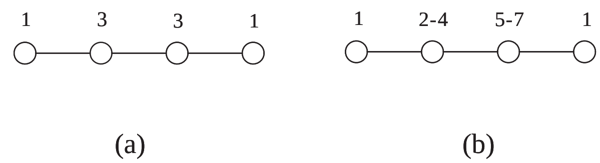

Fig. 1. An example of a path whose optimal sum coloring uses more than the minimum number of colors. (a) Graph with vertex lengths; (b) A minimum sum coloring.

Multicoloring vs. ordinary coloring problems. How different is multicoloring from ordinary graph coloring? We mention some classes of graphs where the difference becomes significant.

Paths Almost anything is trivial for paths, including all makespan problems and all unit-length coloring problems. However, it is not at all easy to derive a polynomial time algorithm for preemptive sum multicoloring (as can be attested by many false starts by the authors). The current best algorithm due to Kovács runs in time $O(n^3 p)$ [31]; a strongly polynomial algorithm is yet to be found.

Trees Preemptive sum multicoloring has been shown to be strongly NP-hard for trees, even binary trees with polynomially bounded weights [38]. On the other hand, sum coloring yields an easy (but not greedy) linear time algorithm.

Interval graphs Non-preemptive makespan multicoloring on interval graphs is the Dynamic Storage Allocation problem, which is NP-hard and APX-hard (i.e., hard to approximate within some $c > 1$). The ordinary coloring problem is, however, easily solvable by a greedy method. Sum coloring and sum multicoloring problems are all approximable within some constant; however, the best ratio differs by a factor of as much as 4 (see table).

Perfect graphs For this class of graphs, we see a difference between preemptive and non-preemptive problems. Preemptive sum multicoloring is approximable within a small constant factor, while no constant factor approximation is known for the non-preemptive case.

1.1 Known Results

One of the most celebrated conjectures of mathematics for a long time was whether all planar graphs could be colored with at most 4 colors. This was finally proved by Appel and Haken [2] with a computer-aided proof, more recently refined in [47]. However, determining whether a planar graph requires 4 colors or not is NP-complete [13]. Exact coloring algorithms have been derived for numerous classes of graphs, one of the most general is that of perfect graphs, due to Grötschel, Lovász, and Schrijver [22]. Another important class of graphs is that of line graphs; coloring line graphs is equivalent to finding an edge coloring of the underlying graph. This is NP-hard [23] but can be done within an additive one of the trivial lower bound of $\Delta(G)$, the maximum degree of the graph [49].

Minimum preemptive multicoloring is NP-hard to approximate on planar graphs within better than $4/3$; this follows from the fact that it is NP-hard to tell if a planar graph is 3-colorable. The problem is known to be hard even on the special class of *hexagon graphs* [40], which are of particular importance for their applications for cellular networks. A $4/3$-approximation for minimum multicoloring of hexagon graphs is given in [43]. The coloring algorithm of [22] for perfect graphs extends to multicoloring, and hence it is solvable on all of its subclasses. For line graphs, minimum multicoloring is equivalent to edge coloring multigraphs, which is approximable within a factor of 1.1 [45].

Non-preemptive makespan problems have been considered for different classes of graphs, but under names unrelated to colorings. The `npMC` problem for interval graphs is better known as *dynamic storage allocation*. Gergov gave an algorithm that uses at most $3\omega(G)$ colors [17]. Buchsbaum et al. [8] recently gave an algorithm with a performance ratio of $2+\epsilon$, for any $\epsilon > 0$. Non-preemptive makespan of line graphs was studied by Coffman et al. [11] under the name *file transfer problem*, with applications to efficient movement and migration of data on a network.. They showed that a class of greedy algorithms yields a 2-approximation and gave a $(2+\epsilon)$-approximation for a version with more general resource constraints.

The sum coloring problem was first studied by Kubicka [33]. Efficient algorithms have been given for trees [33], partial k-trees [29], and regular bipartite graphs [37]. NP-hardness has been shown for general graphs [35], interval graphs [48], bipartite [6], line [4], planar [25], and cubic planar graphs [37]. Approximation algorithms were studied for sparse graphs [34, 4], bounded-degree graphs [4], bipartite graphs [6, 18], interval graphs [44, 27], comparability graphs [27], perfect graphs [4], planar graphs [25], line graphs [4], while results on hardness of approximation have been shown for general [12, 4], bipartite [6], and interval graphs [20]. See Table 1 for a summary of best results known.

Exact and approximate algorithms for multicoloring sum problems have been given for various classes of graphs, as indicated in Table 1. There are hardness results specific to sum multicoloring; the case to date is a recent NP-hardness result of Marx [38] of `pSMC` on trees.

Results on sum multicoloring problems are all fairly recent and in many cases there are large gaps between the best upper and lower bounds on approximability. Several successes are however prominent:

- Approximation preserving reductions to the maximum independent set problem on any hereditary graph class [5]: within a factor of 4 for sum coloring, and factor 16 for preemptive sum multicoloring.
- Polynomial time approximation schemes (PTAS) for planar graphs (`pSMC` and `npSMC`),
- Constant factor approximations for `npSMC` of line graphs and interval graphs [27], [16].
- Very small factor approximations of sum coloring bipartite graphs [18].

Table 1. Known results for sum (multi-)coloring problems

	SC		**SMC**	
	u.b.	*l.b.*	`pSMC`	`npSMC`
General graphs	$n/\log^2 n$ [4]	$n^{1-\epsilon}$ [4]	$n/\log^2 n$ [5]	$n/\log n$ [5]
Perfect graphs	3.591 [16]	$c > 1$ [6]	5.436 [16]	$O(\log n)$ [5]
Interval graphs	1.796 [27]	$c > 1$ [20]	[16]	$7.682 + \epsilon$ [16]
Bipartite graphs	27/26 [18]	$c > 1$ [6]	1.5 [5]	2.8 [5]
Partial k-trees	1 [29]		PTAS [25]	FPTAS [25]
Planar graphs	PTAS [25]	NPC [25]	PTAS [25]	PTAS [25]
Trees	1 [33]		PTAS [26]	1 [26]
Intersection of k-sets	k [4]		k [5]	$3.591k+.5$ [16]
Line graphs	2 [4]	NPC	2 [5]	7.682 [16]
Line graphs of trees	1		PTAS [39]	

Notation

We use the following symbols in the rest of the text:

$x(v)$	Length (or color requirements) of vertex v
$p = p(G)$	Maximum vertex length
$\chi(G)$	Chromatic number of graph G, ignoring vertex lengths

2 Length Partitioning Technique and `npSMC` of Planar Graphs

We will consider in this section the `npSMC` problem for planar graphs, in order to illustrate several of the techniques applicable to multicoloring problems. Unless where otherwise stated, the results are from [25]. We will be aiming towards a polynomial time approximation scheme (PTAS), but in order to get there, we shall be looking at progressively more general special cases. First, however, let us consider some of the more basic approaches.

The first approach might be to ignore the lengths to begin with, apply the quadratic algorithm behind the 4-color theorem [47], and then expand each color class as needed to fit the lengths of the vertices. This can lead to a multicoloring whose sum is arbitrarily worse than optimal. Consider the graph in Fig. 2. The only valid two coloring mixes the white vertices in color classes with the long dark vertices; then, at least half of the white vertices have to wait very long in order to start.

We see that we must give short vertices precedence over long vertices. A reasonable approach would be to color the vertices in groups, shortest-first.

> **Grouping by length:** Divide the vertices into groups of geometrically increasing lengths, and fully color the groups in order of length.

The most natural version is to use powers-of-two as breakpoints between groups, i.e., assign each vertex v to group $\lfloor \lg x(v) \rfloor + 1$. Each group is then colored into

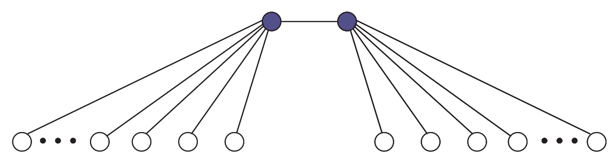

Fig. 2. Example of a planar graph (a tree) whose 2-coloring can lead to an arbitrarily poor sum multicoloring. The many white vertices are short, while the two dark vertices are very long.

$\chi(G)$ sets, each using at most $2^{\lceil \lg x(v) \rceil} - 1$ colors. For instance, vertices of lengths $4, 5, 6, 7$ are in group 3, and each of the $\chi(G)$ sets in that group are assigned 7 colors (the largest length of a vertex in the group).

This approach works reasonably well. Observe that group i will be fully colored after at most $\chi(G)[1 + 3 + 7 + \ldots + 2^i - 1]$ colors have been used. This amounts to less than $\chi(G)(2^{i+1} - 1)$. On the other hand, each vertex in group i is of length at least 2^{i-1}. The performance ratio is therefore at most $4\chi(G)$. A closer look can actually reduce this to $2\chi(G)$ [5]. A further improvement is obtained by selecting the base of the geometric sequence *randomly*; this gives the best ratio known of e for non-preemptive sum multicoloring bipartite graphs [5].

A planar graphs are 4-colorable, this length grouping approach gives us a constant factor approximation. We have, however, higher expectations for planarity. We now turn our attention to the unit-length case, the `SC` problem, as a first step on the road to an approximation ratio arbitrarily close to 1.

Sum Coloring Planar Graphs The primary technique for approximately solving optimization problems, especially subgraph and partitioning problems, on planar graphs is the *decomposition* technique of Baker [3]. The decomposition theorem says that for any integer k, we can partition the vertices of a planar graph with n vertices into two sets inducing subgraph H_1 and H_2, where H_1 is *k-outerplanar* and H_2 is *outerplanar* with at most n/k vertices. A plane graph is said to be outerplanar if all the vertices lie on the outer (i.e., infinite) face. Outerplanar graphs are also the 1-outerplanar graphs, while a graph is said to be *k-outerplanar* if after removing all vertices on the outer face the graph is $k-1$-outerplanar.

Figure 3 illustrates a planar graph with the vertices on the outer face being emphasized.

The advantage with this decomposition is that outerplanar and k-outerplanar graphs are frequently easy to solve optimally. Baker gave explicit dynamic programming algorithms for many optimization problems on k-outerplanar graphs [3]; e.g., the algorithm to find maximum independent sets runs in time $O(8^k n)$. A more general indirect approach is to use the observation of Bodlaender that

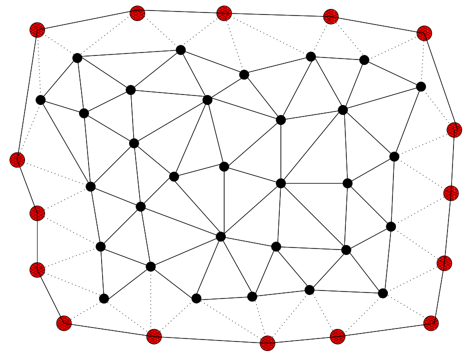

Fig. 3. Planar graph and the outerplanar graph induced by its outer face.

k-outerplanar graphs have treewidth at most $3k-2$ [7], tapping into the vast resource of algorithms on partial k-trees.

Baker's decomposition proceeds as follows. Let V_1 be the set of vertices on the outer face of the graph; remove this set from the graph. Now recursively apply this rule to obtain sets $V_2, V_3, \ldots$, each inducing an outerplanar graph. Figure 4 illustrates this "peeling of onion skins". It is easy to see that a vertex in a set V_i is adjacent only to vertices in its current set, previous set, or the following set. We now form set $U_i, i = 1, 2, \ldots, k$, as $\cup_t V_{tk+i}$, the union of the layers modulo k. Each of the U_i also induces an outerplanar graph, and at least one of them, say U_p, contains at most n/k vertices. The remaining vertices, $V - U_p$, then form a k-outerplanar graph.

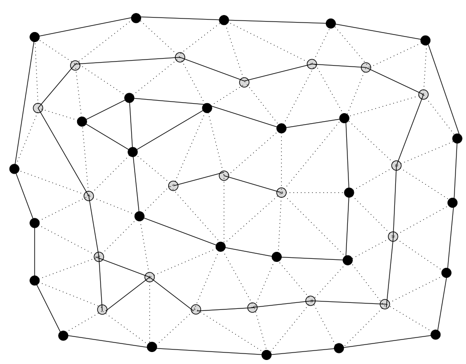

Fig. 4. Partition of a planar graph into a sequence of outerplanar graphs.

We can fairly easily solve sum coloring problems on partial k-trees, using traditional dynamic programming on the tree decomposition, as shown by

Jansen [29]. When processing a supernode (a node in the tree decomposition), we want to compute for each possible coloring of the up to k vertices in the supernode, the minimum cost coloring of the subtree of the tree decomposition. If the maximum color value of a vertex is c, then the total time complexity will be $O(kc^k n)$. We see that it is crucial to bound the number of colors needed.

Kubicka and Schwenk [35] observed that an optimum sum coloring of a tree may require $\Omega(\log n)$ colors, but $O(\log n)$ colors also suffice. A similar bound of $\theta(k \log n)$ was obtained for partial k-trees by Jansen [29]. We shall prove later a more general result for multicolorings. This graph measure, the minimum number of colors in a minimum sum coloring, has been studied more extensively recently as the *strength* of a graph.

We now see that it is easy to apply Baker's decomposition and solve each part optimally in quasi-linear time. The problem is: How do we combine these solutions into a single coloring with of low sum? Intuitively, since H_1 contains the great majority of the vertices, we would want to color those vertices first, and the vertices of H_2 afterwards. However, consider what happens if we wait until H_1 is fully colored for coloring H_2. The cost of coloring H_1 is the minimum chromatic sum of H_1, which is at most the minimum sum of the whole graph G, which is good. However, H_2 may now start to be colored at color $k \log n + 1$. Thus, the cost of coloring H_2 is as much as $n/k \cdot (k \log n + 1) \geq n \log n$. This is certainly much more than the optimal sum (for one thing, 5-coloring G has sum of at most $3n$).

Lemma 1. *After $t\chi(G)$ colors, an optimal sum coloring of graph G has colored all but at most $n/2^t$ vertices.*

> **Truncating a coloring:** If a good sum coloring uses too many colors, it may be preferable to stop that coloring earlier, and revert to a minimum-makespan coloring for the remainder.

The combined strategy is then the following:

1. Apply Baker's decomposition on input graph G, obtaining a k-outerplanar graph H_1 and a smaller graph H_2 with at most n/k vertices.
2. Solve H_1 optimally, using dynamic programming.
3. Use the first $4 \lg k$ colors of the optimal coloring of H_1, leaving at most $n/2^{\lg k} = n/k$ vertices uncolored.
4. Color the remaining at most $n/k + n/k$ vertices using the 5 colors $4 \lg k + 1, \ldots, 4 \lg k + 5$.

The cost of the vertices colored in the first $4 \lg k$ colors is at most the optimal value for G. The cost of the vertices colored later is at most $(4 \lg k + 3)2n/k$. For any given ϵ, choose $k = \theta(\epsilon^{-1} \lg \epsilon^{-1})$ such that this is at most $\epsilon n \leq \epsilon \cdot \texttt{npSMC}(G)$.

Improved time complexity of sum coloring planar graphs. We indicate how we can improve slightly the time complexity of the PTAS for sum coloring planar graphs of [25].

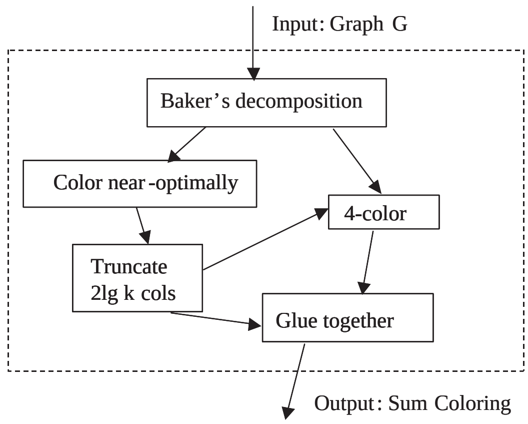

Fig. 5. The schema for sum coloring planar graphs.

Instead of finding an optimal sum coloring of H_1, we might as well take into account that we shall only be using the first $4 \lg k$ color classes. Thus, instead, we can search for an optimal *truncated pseudocoloring* with $4 \lg k + 1$ colors; this is a proper coloring of all of the vertices except those in the last class. Lemma 1 applies as before, for any $t \leq \lg k$, so the last color class contains at most n/k vertices. The cost of such a coloring is at most the optimum chromatic sum of H_1, since any coloring is also a valid truncated pseudocoloring.

The advantage of using a truncated coloring is immediate from our observation of the time complexity of the DP approach being $O(nc^k)$, where c is the value of the largest color used. Here, $c = O(\lg k)$, so the resulting complexity is $O(n 2^{k \lg \lg k})$.

Multicoloring with small lengths. What hinders us from applying the same strategy to the multicoloring case? Let us see how far the techniques used so far will take us.

First, we need to bound the number of colors used. A straightforward extension of Lemma 1 shows that after $O(tp\chi(G))$ colors, at most $n/2^t$ vertices remain, and the total number of colors used in an optimal multisum coloring is at most $O(p\chi(G) \log n)$.

The dynamic programming solution of partial k-trees can be applied with minimal changes to non-preemptive multicolorings. The only change needed is to assign each vertex an *interval* of colors instead of a single color. The primary effect is on the complexity, since the number of possible colors is larger. Thus, we can handle combinations of p and k such that $(p \log n)^k$ is polynomial.

This gives us a PTAS for `npSMC` of planar graphs with polynomially bounded lengths (although not very efficient).

Multicoloring with "almost identical" lengths. What if all the vertex lengths are the same? Is that the same as the unit-length case? In the non-preemptive case, that is indeed true; this can be shown in various ways, e.g., by taking a valid solution, and turning into one where jobs never overlap. (In the preemptive case, it is not true, although it holds for several special cases like bipartite graphs and cliques.)

More generally, if the lengths are all multiple of a common factor q, we can *scale* the instance by this factor q, i.e. reduce the problem to the instance where all lengths are smaller by a factor of q.

Lemma 2 (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, $q \cdot \mathtt{npSMC}(I/q) = \mathtt{npSMC}(I)$.*

We can argue by induction that in any optimal coloring of I, all changes happen at colors that are multiples of q. This shows that optimal colorings of I is equivalent to *stretching* an optimal coloring of I/q by a factor of q by repeating each color class q times in order.

What if the vertex lengths are fairly similar? We can then turn to a classical approximation technique from scheduling:

Rounding-and-scaling: If all lengths are greater than r and we round them upwards to a multiple of q, then the increase in the objective function is at most a factor of $1 + q/r$.

This holds independent of the graph and for any convex objective function of the lengths (including multisum and makespan).

Partitioning by length. We have now seen how to handle unit-length instances, and certain restricted kind of multicoloring instances, most generally the case when the ratio between the minimum and maximum length is bounded (by a term that could be a small polynomial in n).

This suggests that we would want to divide the instance into groups of according to length, coloring the "short" vertices before the "long" vertices. We place the vertices on the scale according to length, divide the instance into groups of similar length, color each of them separately, and then "paste" them together in order of length. In order for this to work, we need to ensure that earlier groups do not "delay" the later groups. Basically, if a group starts receiving colors late, it may not matter how efficiently we color it; the resulting coloring will already have become too expensive.

[An aside: One may suggest that instead of coloring the groups in sequence – thus effectively delaying all the vertices in a group until all previous groups have been completed – that we try to color the vertices in the group as early as possible, intermixed with the colorings of the earlier groups. This may well be a good heuristic, but can be hard to analyze; in particular, it would destroy the independence of the solutions of the individual groups. We shall not attempt to pursue that direction here.]

Consider what could happen to a naive partition. A group of maximum length x, requires $\Omega(x)$ colors; indeed, if even if it is 3-colorable, if it contains a triangle with each vertex of length x, we will have to use at least $3x$ colors. This may prove too much for the next group, which may have most of its vertices with lengths $x+1$. If we start that group at color $3x+1$ – not even accounting for the still earlier groups – that effectively precludes the possibility of a PTAS. Thus, what we need to ensure is that the cost of coloring the earlier groups is small in comparison with the *average length* of vertices in the current group.

We want to find a sequence of breakpoints $b_0 = 1, b_1, \ldots,$, that induce subsets $V_1, V_2, \ldots, V_t$ by $V_i = \{v \in V : x(v) \in (b_{i-1}, b_i]\}$. We find that we can save a logarithmic factor on any arbitrary choices of breakpoints.

Lemma 3 (Length partitioning). *For any $q = q(n)$, we can partition the vertex set into length groups so that the average length of vertices in V_i is at least $(\ln\sqrt{q})b_i$. Further, the groups differ by a factor of at most q, i.e. $b_i \le b_{i-1} \cdot q$.*

This lemma has an interesting relationship with the classic inequality of Markov from probability theory. Consider any collection of positive numbers $x_1, \ldots, x_n$. Markov inequality shows that at most $1/\ell$ fraction of the elements of the set $X = \{x_1, x_2, \ldots, x_n\}$ are greater than ℓ times the average value $\overline{x}$ (cf. [41]). It is easy to show that this is tight for any fixed value of t; but it cannot be tight for more than one value of t simultaneously. If we are free to choose t from a range of values, the resulting bound on the tail is better; as our lemma shows, it is improved by a logarithmic factor.

Putting together the pieces. This becomes the missing puzzle in our quest for a PTAS.

The combined strategy is illustrated in Figure 6. The length partitioning lemma breaks the vertex set into groups with lengths in a compact interval; the groups are processed independently using a variation of the sum coloring approximation scheme. Finally, the individual solutions are pasted together, in order of the group lengths.

The cost of the multicoloring is derived from two parts: the sum of the costs of the subproblems, and the delays incurred by the colorings of the earlier subproblems. The former is at most $1+\epsilon$ times the optimal cost of coloring each of the subgraphs separately, which is at most $(1+\epsilon)\texttt{npSMC}(G)$. The main issue is therefore accounting for the contribution of the delays. The number of colors used in each subproblem G_j is at most σb_j, where $\sigma = O(\log k) = O(\log \epsilon^{-1})$. The sum of these is dominated by a geometric series with base of $\sqrt{q}$; thus, the sum of the colors used on the subproblems preceding G_i is at most $(1+1/(\sqrt{q}-1))\sigma b_i$. Here it becomes crucial that the average weight of vertices in each subproblem G_i, and thus the multicolor sum as well, is at least $\ln\sqrt{q}b_i$. Thus the cost incurred by the delays of earlier subproblems are at most $O(\texttt{npSMC}(G) \cdot \sigma / \ln\sqrt{q})$. In our case we choose $q = e^{(\epsilon^{-1}\ln\epsilon^{-1})^2}$, to make this quantity at most $\texttt{npSMC}(G) \cdot O(\epsilon)$. The total cost of the solution is therefore $1 + O(\epsilon)$ times optimal, which can be made arbitrarily close to 1.

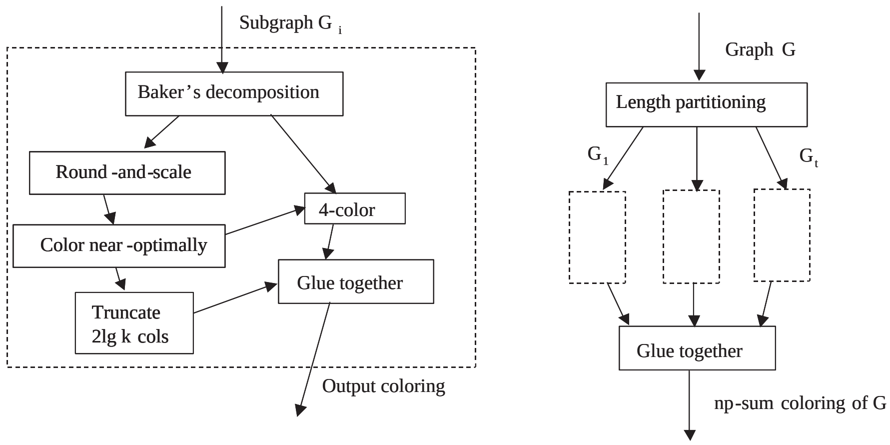

Fig. 6. The schema for non-preemptive sum multicoloring planar graphs. Each small dotted box on the right is an instance of the schema on the right applied to a length-constrained subgraph.

3 The Delay Technique, `npSMC` on Line Graphs and Open-shop Scheduling

We now describe another general technique applicable in several scenarios. In particular, it is used to approximate the `npSMC` problem on line graphs, and for approximating open-shop scheduling.

Let $L(v)$ and $S(v)$ denote the sets of shorter and longer neighbors of v, respectively. Let $x(U) = \sum_{u \in U} x(u)$ be the sum of the lengths of vertices in a set U.

Note that for every edge $e = (u, v) \in E$ and in any legal schedule the jobs corresponding to u and v are performed in disjoint rounds. In particular, if the job of v starts (and thus ends) before the job u starts, u has to "wait" $x(v)$ time units before it can start being executed. Say that $x(u) \leq x(v)$. In such a case, a "perfect" algorithm will "manage" to schedule u before v incurring $x(u)$ delay (rather than $x(v)$ delay). Admittedly, some of the delays can happen "together, namely, at the same round several of the neighbors of v are active together. On the other hand, these active neighbors of v must form an independent set. At most two neighbors of any node in a line graph can be active at the same time. This intuition is converted into the following claim [27]. Let $Q(G) = \sum_v x(S(v))$.

Proposition 1. *[27] For a line graph G,*

$$Q(G) \leq 2 \cdot (\mathtt{npSMC}(G) - S(G)).$$

As it turns out, Claim 1 does not suffice to give a "good" approximation. The problem is with the longer neighbors $L(v)$ of a vertex v. Say that $u \in L(v)$

and $x(u) \gg x(v)$. In the non-preemptive scenario, if u executes before v then v has to wait for u to end causing a large delay.

If we disallow $u \in L(v)$ to be executed before v this may cause the independent sets executed at given rounds not to be even maximal independent sets. We adopt an intermediate approach that can be summarized as follows.

1. Before a vertex v can become "active", namely, is executed non-preemptively for $x(v)$ rounds, it has to "pay" $\beta \cdot x(v)$ rounds, where β is some carefully chosen constant.
2. A vertex can only pay if it not executed and has no active neighbor.

Thus the paying of $\beta \cdot x(v)$ rounds is a way of disallowing long jobs to have early process starting times. In fact, the delay "paid" by a job is proportional to its length, hence long jobs wait more.

The algorithm used in [27] is implied by the following additional rules:

– At each round, the union of the active and paying vertices is a *maximal* independent set. Thus, two vertices paying at the same round cannot be neighbors.
– **The length rule:** A vertex v pays in a round if and only if it has neither an active neighbor or a shorter paying neighbor in that round.

Our goal is to prove that the application of this algorithm on a line graph gives a $O(x(S(v))$ finish time for a vertex v. Then, by Proposition 1 an $O(1)$-approximation algorithm for `npSMC` is implied. In given round, a vertex v can either be

1. active,
2. paying, or
3. neither paying nor active, in which case it is *delayed*.

It is easy to account for the contribution of rounds where v is active or paying. It remains to check rounds where v is delayed. By definition, v can be delayed because either the round contains a paying $S(v)$ vertex or the round contains an active neighbor of v.

Again, it is easy to account for rounds where v is delayed by a shorter neighbor (paying or active); see Proposition 1. Hence the only "problematic" rounds are rounds that contain a longer active neighbor.

Let $L'(v)$ be the vertices of $L(v)$ that became active before v. Observe that by definition, before becoming active, these $L'(v)$ vertices paid a total of $\beta \cdot x(L'(v))$ units (while some of those units were simultaneously paid). The following claim bounds $x(L'(v))$ by $O(x(S(v))$. Thus, we can use Claim 1 to get a constant factor approximation for `npSMC`.

Proposition 2. $x(L'(v)) = O(k \cdot x(S(v))$

Proof. Since the vertices of $L'(v)$ needed to pay for a total of $\beta \cdot x(L'(v))$ rounds before becoming active, and since at most 2 neighbors of a vertex in a line graph are active at the same time, there were at least $\beta \cdot x(L'(v))/k$ rounds in which the $L'(v)$ vertices were paying. Call such a round an "important" round.

Consider an important "problematic" round for v, namely, an important round for v in which no $S(v)$ vertex is paying or active. Since v was not chosen to be the paying vertex in an important round, it follows that v must have an active neighbor in such a round; otherwise, by the length rule, it has to be paying. Thus we get:

$$\frac{\beta \cdot x(L'(v))}{k} \leq x(L'(v)) + x(S(v)).$$

Remark: The main property used here is that vertices in $L(v) \setminus L'(v)$ cannot become active before v becomes active.

Hence, we get that $x(L'(v)) = O(k \cdot x(S(v)))$ as required.

A more detailed proof along these lines implies a 12-ratio for `npSMC` on line graphs.

Open-shop and simultaneous delay. In our context, it is best to describe the non-preemptive open-shop scheduling problem as follows. We are given a bipartite graph $G(M, J, E)$, (M for machines and J for jobs). Each edge e corresponds to a task and has length $x(e)$. A subset of the tasks (edges) has to be scheduled at every round. The edges scheduled at a given round must be "independent", namely, must induce a matching. We need to schedule non-preemptively all tasks (edges) so that every e is executed non-preemptively for $x(e)$ time units.

In this scenario, vertices $m \in M$ correspond to jobs. A job is completed if all its tasks (edges) complete. Formally, let $m \in M$. The finish time $f(m)$ is the maximum finish time of an edge touching m. The objective function is to minimize $\sum_{m \in M} f(m)$.

The open shop scheduling problem resembles the `npSMC` problem on line graphs (because a round is an independent set of edges). The main difference is that we sum the finish times of vertices (of the underlying graph, of which we take the line graph) and not of edges. In that respect, the open-shop problem resembles more the data migration problem (see [30]). In [15] the delay method is used combined with LP techniques to give improved approximation. The problem is relaxed to a fractional linear program. The fractional values are used in the delay function. Namely, if an edge e has fractional starting time $\mu(e)$, it is delayed by a function of $\mu(e)$ of rounds. Instead of a "combinatorial" lower bound lemma 1, the fractional LP value is used to prove a lower bound.

One important new idea is used here. In the line graph algorithm, adjacent vertices cannot simultaneously pay at a round (the paying vertices are an independent set). In [15] a simultaneous pay method is used: Adjacent vertices can pay at the same round. While this simultaneous pay method fails to give a good approximation for general line graphs, it succeeds for open shop scheduling, in part because open shop scheduling essentially corresponds to line graphs of bipartite graphs.

Theorem 1. [15] The non-preemptive open-shop scheduling problem admits a 5.055-ratio approximation.

This improves the previous best 5.83-approximation of [46].

References

1. F. Afrati, E. Bampis, A. Fishkin, K. Jansen, and C. Kenyon. Scheduling to minimize the average completion time of dedicated tasks. In *FSTTCS 2000*, LNCS, Delhi.
2. K. Appel, W. Haken. Every planar map is 4-colorable. *Contemporary Mathematics*, Volume 98, 1989.
3. B. S. Baker. Approximation algorithms for NP-complete problems on planar graphs. *J. ACM* **41**:153–180, Jan. 1994.
4. A. Bar-Noy, M. Bellare, M. M. Halldórsson, H. Shachnai, T. Tamir. On chromatic sums and distributed resource allocation. *Inf. Comp.* **140**:183–202, 1998.
5. A. Bar-Noy, M. M. Halldórsson, G. Kortsarz, H. Shachnai, and R. Salman. Sum multicoloring of graphs. *J. Algorithms* **37**(2):422–450, 2000.
6. A. Bar-Noy and G. Kortsarz. The minimum color-sum of bipartite graphs. *J. Algorithms* **28**:339–365, 1998.
7. H. L. Bodlaender. A partial k-arboretum of graphs with bounded treewidth. *Theoretical Computer Science*, **209**:1–45, 1998.
8. A. L. Buchsbaum, H. Karloff, C. Kenyon, N. Reingold and M. Thorup. OPT versus LOAD in Dynamic Storage Allocation. *STOC'03*.
9. W. J. Cook, W. H. Cunningham, W. R. Pulleyblank, and A. Schrijver. *Combinatorial Optimization.* Wiley, 1998.
10. S. Chakrabarti, C. A. Phillips, A. S. Schulz, D. B. Shmoys, C. Stein, and J. Wein. Improved Scheduling Problems For Minsum Criteria. In *Proc. 23rd International Colloquium on Automata, Languages, and Programming (ICALP)*, LNCS 1099, 646-657, 1996.
11. E. G. Coffman, M. R. Garey, D. S. Johnson, and A. S. LaPaugh. Scheduling File Transfers. *SIAM Journal on Computing*, **14**(3):744-780, 1985.
12. U. Feige and J. Kilian. Zero Knowledge and the Chromatic number. *Journal of Computer and System Sciences*, **57**(2):187-199, October 1998.
13. M. R. Garey, D. Johnson. *Computers and intractability: A guide to the theory of NP-completeness.* W. H. Freeman, 1979.
14. H. Gabow and O. Kariv. Algorithms for edge coloring bipartite graphs and multigraphs. *SIAM Journal of Computing*, **11**(1), February 1992.
15. R. Gandhi, M. M. Halldórsson, G. Kortsarz and H. Shachnai, Approximating non-preemptive open-shop scheduling and related problems. *ICALP '04*.
16. R. Gandhi, M. M. Halldórsson, G. Kortsarz, and H. Shachnai. Improved Bounds for Sum Multicoloring and Weighted Completion Time of Dependent Jobs. Unpublished manuscript, 2004.
17. J. Gergov. Algorithms for compile-time memory allocation. *SODA'99.*
18. K. Giaro, R. Janczewski, M. Kubale and M. Malafiejski. A 27/26-approximation algorithm for the chromatic sum coloring of bipartite graphs. *APPROX*, 131–145, 2002.
19. M. K. Goldberg. Edge Coloring of Multigraphs: Recoloring Technique. *J. Graph Theory*, **8**:121-137, 1984.

20. M. Gonen. Coloring Problems on Interval Graphs and Trees. M.Sc. Thesis, School of Computer Science, The Open Univ., Tel-Aviv, 2001.
21. R. Graham. Bounds for certain multiprocessing anomalies. *Bell System Technical Journal*, 45:1563–1581, 1966.
22. M. Grötschel, L. Lovász, and A. Schrijver. *Geometric Algorithms and Combinatorial Optimization*. Springer-Verlag, 1993.
23. I. Holyer. The NP-completeness of Edge-Colouring. *SIAM J. Comput.*, **10**(4), 1981.
24. L. Hall, A. S. Schulz, D. B. Shmoys, and J. Wein. Scheduling to Minimize Average Completion Time: Off-line and On-line Approximation Algorithms. *Mathematics of Operations Research*, **22**:513–544, 1997.
25. M. M. Halldórsson and G. Kortsarz. Tools for multicoloring with applications to planar graphs and partial k-trees. *J. Algorithms*, **42**(2), 334–366, 2002.
26. M. M. Halldórsson, G. Kortsarz, A. Proskurowski, R. Salman, H. Shachnai, and J. A. Telle. Multicoloring trees. *Inf. Computation*, **180**(2):113–129, 2003.
27. M. M. Halldórsson, G. Kortsarz, H. Shachnai. Sum coloring interval and k-claw free graphs with application to scheduling dependent jobs. *Algorithmica* **37**:187–209, 2003.
28. H. Hoogeveen, P. Schuurman, and G. Woeginger. Non-approximability results for scheduling problems with minsum criteria. In *Proc. of the 6th International Conference on Integer Programming and Combinatorial Optimization (IPCO)*, LNCS 1412, 353–366, 1998.
29. K. Jansen. The optimum cost chromatic partition problem. In *Proc. Third Italian Conference on Algorithms and Complexity (CIAC '97)*, LNCS # 1203, pages 25–36, 1997.
30. Y. Kim, Data migration to minimize average completion time. SODA 2003. In *Proc. 14th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 97–98, 2003.
31. A. Kovács. Sum-multicoloring on paths. In STACS 2004.
32. M. Kubale. Preemptive versus non preemptive scheduling of biprocessor tasks on dedicated processors. *European J. Operational Research*, **94**:242–251, 1996.
33. E. Kubicka. The chromatic sum of a graph. PhD thesis, Western Michigan University, 1989.
34. E. Kubicka, G. Kubicki, and D. Kountanis. Approximation Algorithms for the Chromatic Sum. In *Proc. First Great Lakes Computer Science Conference*, LNCS 1203, pages 15–21, 1989.
35. E. Kubicka and A. J Schwenk. An Introduction to Chromatic Sums. In *Proc. 17th Annual ACM Computer Science Conference, "Computing trends in the 1990's"*, pages 39–45, 1989.
36. E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy-Kan, and D. B. Shmoys. Sequencing and Scheduling: Algorithms and Complexity. In S. C. Graves, A. H. G. Rinnooy-Kan, and P. Zipkin, eds., *Handbooks in Operations Research and Management Science, Vol. 4: Logistics of Production and Inventory*, 445–522, 1993.
37. M. Malafiejski. The complexity of the chromatic sum problem on cubic planar graphs and regular graphs. *Electronic Notes in Discrete Mathematics*, **8**, May 2001.
38. D. Marx. The complexity of tree multicolorings. In *Proc. Mathematical Foundations of Computer Science (MFCS)*, 532–542, LNCS 2420, Springer, 2002.

39. D. Marx. Minimum sum multicoloring on the edges of trees. In *Proc. 1st Workshop on Approximation and Online Algorithms (WAOA)*, 214–226, LNCS 2909, Springer, 2004.
40. C. McDiarmid and B. Reed. Channel assignment and weighted coloring. *Networks*, **36**(2):114–117, 2000.
41. R. Motwani and R. Ragavan *Randomized algorithms.* Cambridge University Press, 1995.
42. L. Narayan. Channel assignment and graph multicoloring. In *Handbook of wireless networks and mobile computing*, pages 71 – 94, Wiley Series On Parallel And Distributed Computing, 2002.
43. L. Narayanan and S. Shende. Static frequency assignment in cellular networks. *Algorithmica*, **29**(3), 396–401, 2001.
44. S. Nicoloso, M. Sarrafzadeh and X. Song. On the sum coloring problem on interval graphs. *Algorithmica*, **23**:109–126,1999.
45. T. Nishizeki and K. Kashiwagi. On the 1.1 edge-coloring of multigraphs. *SIAM Journal on Discrete Mathematics*, **3**(3):391–410, 1990.
46. M. Queyranne and M. Sviridenko. Approximation Algorithms for Shop Scheduling Problems with Minsum Objective. *Journal of Scheduling*, **5**:287–305, 2002.
47. N. Robertson, D.P. Sanders, P. D. Seymour, and R. Thomas. A new proof of the four color theorem. *Electron Res. Announc. Amer. Math. Soc.*, **2**:17-25, 1996.
48. 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.
49. V. G. Vizing. On the estimate of the chromatic class of p-graphs. *Diskret. Analiz.*, **3**:23-30,1964.