Resource-Constrained Scheduling and Graph Multicoloring with Min-sum Objectives

M. M. Halldórsson*  G. Kortsarz†  H. Shachnai‡

Introduction

Consider the following scenario that exemplifies our problem. Suppose that a set of jobs $J_1, \ldots, J_6$ is initiated in a distributed computing environment. Each of the jobs is available to run on a different processor, however, the jobs share accesses to a set of files $f_1, \ldots, f_5$, stored in the network file system. In particular, $J_1, J_2, J_5$ and $J_6$ require read/write operations in the files $f_1, f_2, f_3$ and $f_4$ respectively; $J_3$ needs to copy data from $f_1$ to $f_2$ and to $f_5$; $J_4$ requires simultaneous reading of the contents of the files $f_3, f_4$ into $f_5$. The execution time of $J_1, J_2, J_5, J_6$ is one unit, while $J_3, J_4$ require 2 time units each. Note, that each of the jobs requires an exclusive access to the corresponding file/s throughout its execution. While we can schedule the jobs such that the latest completion time is 4, the only schedule that minimizes average completion time (or, equivalently the sum of completion times) is the one that schedules the unit-length jobs first, using a total of 5 time units.

Resource-constrained scheduling: We are given a collection of $n$ jobs of integral lengths and a collection of resources. Each job requires an exclusive access to a particular subset of the resources to execute. Two jobs that require the same resource cannot run at the same time. We seek a schedule of the jobs that minimizes the average time that a job spends before completed. We model this problem in terms of multi-coloring graphs.

Sum multicoloring Given a graph $G = (V, E)$, and associated color requirements of the vertices $x : V \mapsto \mathbb{N}$, find a multicoloring $A : V \mapsto 2^\mathbb{N}$ such that $|A(v)| = x(v)$ for each vertex $v$ and $A(v) \cap A(u) = \emptyset$ if $uv \in E$. Our objective is to minimize $\sum_{v \in V} f_A(v)$, where $f_A(v) = \max_{w \in A(v)} A(v)$ is the final color of $v$. This specifies the unrestricted sum multicoloring (SMC) problem (denoted as $pSMC$). In the contiguous SMC problem (npSMC), the colors assigned to a vertex must form a proper interval of integers. When all jobs are of unit length, we obtain the sum coloring (SC) problem.

In the opening scenario, the resources are shared files. When the resources are dedicated processors, we get multiprocessor task scheduling (see, e.g., [K96]). When the resources are paths in a communication network, we get path coloring problems, recently popularized in network design. Other examples abound, with the literature containing applications in compiler design, VLSI routing, traffic intersection control, and frequency assignment (see [BKH+99]).

One reason why multicoloring problems are relatively infrequent is that they can often be transformed into an ordinary coloring problem on a derived graph. Each node $v$ with color requirement $x(v)$ in a multicoloring instance is replaced by a clique with $x(v)$ vertices, with each clique vertex having the set of neighbors of the original node. Then, we have one-to-one mappings between

---

*Science Institute, University of Iceland, IS-107 Reykjavik, Iceland. E-mail: mmh@hi.is.
†Dept. of Computer Science, Open University, Ramat Aviv, Israel. E-mail: guyk@avror.openu.ac.il.
‡Dept. of Computer Science, Technion, Haifa 32000, Israel. E-mail: hadas@cs.technion.ac.il.
colorings of the derived graph and multicolorings of the original graphs, that preserve the number of colors needed. However, this reduction breaks down when attention turns to the sum coloring objective function. We note that while the sum-of-completion-times measure is extremely common in the scheduling literature, it is a very recent concept in graph theory.

**Known Results**

Min-sum multicolorings are generally at least as hard as classic coloring, while they are in some sense no harder than the independent set (IS) problem. Namely, for classes of graphs on which IS is $\rho$-approximable we get approximations of $O(\rho)$ for $pSMC$ and $O(\rho \log n)$ for $npSMC$. Yet, we see that for several classes for which coloring and IS are solvable sum coloring becomes hard.

<table>
<thead>
<tr>
<th>SC</th>
<th>pSMC</th>
<th>npSMC</th>
</tr>
</thead>
<tbody>
<tr>
<td>General graphs</td>
<td>$n^{1-\epsilon}$ [BBH+98]</td>
<td>$n/\log^2 n$ [BK+99]</td>
</tr>
<tr>
<td>Perfect graphs</td>
<td>4 [BBH+98]</td>
<td>APX [BK98]</td>
</tr>
<tr>
<td>Interval graphs</td>
<td>5 [NSS99]</td>
<td>NPX [S99]</td>
</tr>
<tr>
<td>Bipartite graphs</td>
<td>10/9 [BK98]</td>
<td>APX [BK98]</td>
</tr>
<tr>
<td>Line graphs</td>
<td>2 [BBH+98]</td>
<td>NPC [BBH+98]</td>
</tr>
<tr>
<td>Partial $k$-trees</td>
<td>2 [J97]</td>
<td>PTAS [HK99]</td>
</tr>
<tr>
<td>Trees</td>
<td>1 [K89]</td>
<td>PTAS [HK+99]</td>
</tr>
</tbody>
</table>

The table above lists known results on approximating sum multicolorings for some fundamental classes of graphs. The first two columns regard sum coloring, both upper and lower bounds on approximation. The last two columns give known upper bounds for preemptive and non-preemptive $SMC$, respectively. Entries marked with $\cdot$ follow by inference, either using containment of graph classes (bipartite and interval graphs are perfect), or by SC being a special case of $SMC$. Boldfaced entries indicate improvements from published work, that are discussed below.

**Recent Work**

**Line graphs:** $SMC$ of line graphs corresponds to the case where each job requires at most two resources. It has previously been studied for the purpose of biprocessor task scheduling [K96]. The first constant factor approximation was recently given in [HKN2000] for non-preemptive scheduling. It generalizes to a constant factor for the case of $k + 1$-claw free graphs, which correspond to the case where each task requires the exclusive use of up to $k$-resources.

The 2-approximate algorithm for $SMC$ in the preemptive case [BK+99] generalizing the 2-approximate algorithm for SC [BBH+98], worked by simply scheduling a job as long as one of its neighbors of lower color requirement was not being scheduled. This fails in the non-preemptive case, since once a long (i.e. large color requirement) job is initiated, it may later delay many short jobs. The solution is *patience*: even when free to execute, a job idles for some period proportional to its length before starting. The idling gives its shorter neighbors a fair chance to execute before the job starts.

**Trees and Partial $k$-trees:** In [HKP+99], we give a polynomial-time approximation scheme (PTAS) for $pSMC$ on trees, i.e. $1 + \epsilon$-approximation running in time $n^{O(1/\epsilon)}$, for any $\epsilon > 0$. Using a technique of [HK99], we could improve the time-approximation tradeoffs to obtain $1 + O(\log \log n / \log n)$-approximation in polynomial time. The technique allows one to partition the vertices (i.e. jobs) carefully according to lengths, coloring the different classes separately, and then
pasting the solutions sequentially, while paying only small overhead. This can be applied to any class of graphs of small chromatic number.

Additionally, for the non-preemptive case of partial $k$-trees, we stated a PTAS in [HK99], while it is more properly stated as a fully polynomial time approximation scheme, FPTAS.

**Online variants:** We have studied online versions of resource-constrained scheduling from two viewpoints. First, we considered the case of the sum coloring problem on interval graphs. For this case, a tight $\theta(\sqrt{n})$-approximation algorithm is given in [HKN00].

In the other variant, *online scheduling dependent jobs*, the execution clock is synchronized with the online arrival clock. Jobs are presented in the order of their release time, and stay around until completed. The objective now is the average flow time of the jobs, where the flow time of a job is the difference between its completion time and release time. It can be shown that already for unit length jobs, any deterministic algorithm has a competitive ratio $\Omega(\sqrt{n})$. A ratio of $\chi(G)$ can be achieved on any conflict graph $G$ by a greedy algorithm. For arbitrary length jobs, a randomized algorithm attains a bound of $(\Delta + 2)/2$.

**New directions**

A number of new directions are suggested by the link of multicoloring to scheduling.

**Objective functions** Our focus has been on the second-most studied objective function, the sum of completion times, with the makespan (i.e. the number of colors) being well studied. A variant is the sum of waiting times, with the waiting time of a job being the difference between the completion time and its execution time. For optimal algorithm, this makes no difference, but waiting time is harder to approximate, and currently no results are known.

Another interesting function is *sum of flow times* (or average), discussed for the online problems. Still another is the sum of squares of completion times. Both of these measures can be handled by the optimal non-preemptive algorithm of [HKP99] for trees, but no results are known for approximations. Additionally, various options arise when due dates are introduced.

**Resource capacities** In our basic model, each resource is indivisible and has exclusive access.

A more general case has upper bounds on each resource. For instance, there can be a few identical dedicated processors, or a fixed number of paths can go through each node in a communication network (e.g. by different wavelengths in an optical network). This appears to lead to hypergraph coloring problems.

**Machines** Traditional scheduling has fixed number of machines executing the jobs. This introduces an upper bound on the number of jobs that can be executed simultaneously. So far, this aspect has not yet been studied in the context of sum multicoloring.

**Open issues in sum multicoloring:** There remain numerous unresolved questions about sum multicoloring. A major outstanding problem is to give a constant factor approximation for the non-preemptive case on interval graphs. In the preemptive case, polynomial algorithms are known only for fairly trivial cases. An important open issue is to resolve the solvability of paths and trees.

Finally, recent results give a clear evidence to the poor performance of online algorithms for sum multicoloring and scheduling dependent jobs (answering a question posed in [MPT94]). Yet, the online variants deserve a more thorough treatment.
References


