Batch Coloring of Graphs

Joan Boyar[1], Leah Epstein[2], Lene M. Favrholdt[1],
Kim S. Larsen[1], and Asaf Levin[3]

[1] Dept. of Mathematics and Computer Science, University of Southern Denmark, Odense, Denmark, {joan,lenem,kslarsen}@imada.sdu.dk *
[2] Dept. of Mathematics, University of Haifa, Haifa, Israel, lea@math.haifa.ac.il
[3] Faculty of IE&M, The Technion, Haifa, Israel, levinas@ie.technion.ac.il

Abstract. In graph coloring problems, the goal is to assign a positive integer color to each vertex of an input graph such that adjacent vertices do not receive the same color assignment. For classic graph coloring, the goal is to minimize the maximum color used, and for the sum coloring problem, the goal is to minimize the sum of colors assigned to all input vertices. In the offline variant, the entire graph is presented at once, and in online problems, one vertex is presented for coloring at each time, and the only information is the identity of its neighbors among previously known vertices. In batched graph coloring, vertices are presented in k batches, for a fixed integer $k \geq 2$, such that the vertices of a batch are presented as a set, and must be colored before the vertices of the next batch are presented. This last model is an intermediate model, which bridges between the two extreme scenarios of the online and offline models. We provide several results, including a general result for sum coloring and results for the classic graph coloring problem on restricted graph classes: We show tight bounds for any graph class containing trees as a subclass (e.g., forests, bipartite graphs, planar graphs, and perfect graphs), and a surprising result for interval graphs and $k = 2$, where the value of the (strict and asymptotic) competitive ratio depends on whether the graph is presented with its interval representation or not.

1 Introduction

We study three different graph coloring problems in a model where the input is given in *batches*. In this model of computation an adversary reveals the input graph one batch at a time. Each batch is a subset of the vertex set together with its edges to the vertices revealed in the current batch or in previous batches. After a batch is revealed the algorithm is asked to color the vertices of this batch with colors which are positive integers, the coloring must be valid or *proper*, i.e., neighbors are colored using distinct colors, and this coloring cannot be modified later.

The batch scenario is somewhere between online and offline. In an *offline* problem, there is only one batch, while for an *online* problem, the requests arrive

* Supported in part by the Danish Council for Independent Research, Natural Sciences, grant DFF-1323-00247, and the Villum Foundation, grant VKR023219.

one at a time and have to be handled as they arrive without any knowledge of future events, so each request is a separate batch. Many applications might fall between these two extremes of online and offline. For example, a situation where there are two (or more) deadlines, an early one with a lower price and a later one with a higher price can lead to batches.

When considering a combinatorial problem using batches, we assume that the requests arrive grouped into a constant number k of batches. Each batch must be handled without any knowledge of the requests in future batches. As with online problems, we do not consider the execution times of the algorithms used within one batch; the focus is on the performance ratios attainable. Therefore, our goal is to quantify the extent to which the performance of the solution deteriorates due to the lack of information regarding the requests of future batches. We also investigate how much advance knowledge of the number of batches can help.

The quality of the algorithms is evaluated using competitive analysis. Let $A(\sigma)$ denote the cost of the solution returned by algorithm A on request sequence σ, and let $\mathrm{OPT}(\sigma)$ denote the cost of an optimal (offline) solution. Note that for standard coloring problems, $\mathrm{OPT}(G) = \chi(G)$, where $\chi(G)$ is the chromatic number of the graph G. An online coloring algorithm A is *ρ-competitive* if there exists a constant b such that, for all finite request sequences σ, $A(\sigma) \leq \rho \cdot \mathrm{OPT}(\sigma) + b$. The *competitive ratio* of algorithm A is $\inf\{\rho \mid A \text{ is } \rho\text{-competitive}\}$. If the inequality holds with $b = 0$, the algorithm is *strictly ρ-competitive* and the *strict competitive ratio* is $\inf\{\rho \mid A \text{ is strictly } \rho\text{-competitive}\}$.

The First-Fit algorithm for coloring a graph traverses the list of vertices given in an arbitrary order or in the order they are presented, and assigns each vertex the minimal color not assigned to its neighbors that appear before it in the list of vertices.

Other combinatorial problems have been studied previously using batches. The study of bin packing with batches was motivated by the property that all known lower bound instances have the form that items are presented in batches. The case of two batches was first considered in [9], an algorithm for this case was presented in [6], and better lower bounds were found in [2]. A study of the more general case of k batches was done in [7], and recently, a new lower bound on the competitive ratio of bin packing with three batches was presented in [1]. The scheduling problem of minimizing makespan on identical machines where jobs are presented using two batches was considered in [20].

Graph classes containing trees. The first coloring problem we consider using batches is that of coloring graph classes containing trees as a subclass (e.g., forests, bipartite graphs, planar graphs, perfect graphs, and graphs in general), minimizing the number of colors used. Offline, finding a proper coloring of bipartite graphs is elementary and only (at most) two colors are needed. However, there is no online algorithm with a constant competitive ratio, even for trees. Gyárfás and Lehel [10] show that for any online tree coloring algorithm A and any $n \geq 1$, there is a tree on n vertices for which A uses at least $\lfloor \log n \rfloor + 1$ colors. The lower bound is matched exactly by First-Fit [11], and hence, the

optimal competitive ratio on trees is $\Theta(\log n)$. For general graphs, Halldórsson and Szegedy [13] have shown that the competitive ratio is $\Omega(n/\log n)$.

We show that any algorithm for coloring trees in k batches uses at least $2k$ colors in the worst case, even if the number of batches is known in advance. This gives a lower bound of k on the competitive ratio of any algorithm coloring trees in k batches. The lower bound is tight, since (on any graph, not only trees), a k-competitive algorithm can be obtained by coloring each batch optimally with colors not used in previous batches. Thus, for graph classes containing trees as a subclass, k is the optimal competitive ratio.

Coloring interval graphs in two batches. Next we consider coloring interval graphs in two batches, minimizing the number of colors used. An interval graph is a graph which can be defined as follows: The vertices represent intervals on the real line, and two vertices are adjacent if and only if their intervals overlap (have a nonempty intersection). If the maximum clique size of an interval graph is ω, it can be colored optimally using ω colors by using First-Fit on the interval representation of the graph, with the intervals sorted by nondecreasing left endpoints. For the online version of the problem, Kierstead and Trotter [15] provided an algorithm which uses at most $3\omega - 2$ colors and proved a matching lower bound for any online algorithm.

The algorithm presented in [15] does not depend on the interval representation of the graph, but the lower bound does, so in the online case the optimal competitive ratio is the same for these two representations (see [14, 18] for the current best results regarding the strict competitive ratio of First-Fit for coloring interval graphs). In contrast, when there are two batches, there is a difference. We show tight upper and lower bounds of 2 for the case when the interval representation is unknown and 3/2 when it is known, respectively. Our results apply to both the asymptotic and the strict competitive ratio.

Note that when the interval representation of the graph is used, the batches consist of intervals on the real line (it is not necessary to give the edges explicitly).

Sum coloring. The sum coloring problem (also called chromatic sum) was introduced in [17] (see [16] for a survey of results on this problem). The problem is to give a proper coloring to the vertices of a graph, where the colors are positive integers, so as to minimize the sum of these colors over all vertices (that is, if the coloring is defined by a function C, the objective is to minimize $\sum_{v \in V} C(v)$).

Bar-Noy et al. [3] study the problem, motivated by the following application: Consider a scheduling problem on an infinite capacity batched machine where all jobs have unit processing time, but some jobs cannot be run simultaneously due to conflicts for resources. If the conflicts are given by a graph where the jobs are vertices and an edge exists between two vertices, if the corresponding jobs cannot be executed simultaneously (and thus each batch of jobs corresponds to an independent set of this graph), the value s of the optimal sum coloring of the graph gives the sum of the completion times of all jobs in an optimal schedule. Dividing by the number of jobs gives the average response time. The problem

when restricted to interval graphs is also motivated by VLSI routing [19]. The first problem seems more likely to come in batches than the second.

The sum coloring problem is NP-hard for general graphs [17] and cannot be approximated within $n^{1-\varepsilon}$ for any $\varepsilon > 0$ unless ZPP = NP [3]. Interestingly, there is a linear time algorithm for trees, even though there is no constant upper bound on the number of different colors needed for the minimum sum coloring of trees [17]. For online algorithms, there is a lower bound of $\Omega(n/\log^2 n)$ for general graphs with n vertices [12].

We show tight upper and lower bounds of k on the competitive ratio when there are k batches and k is known in advance to the algorithm. The competitive ratio is higher if k is unknown in advance to the algorithm. We do not give a closed form expression for the competitive ratio in this case, but give tight upper and lower bounds on the order of growth of the competitive ratio and the strict competitive ratio. For any nondecreasing function f, with $f(1) \geq 1$, the optimal competitive ratio for k batches is $O(f(k))$ if the series $\sum_{i=1}^{\infty} \frac{1}{f(i)}$ converges, and it is $\Omega(f(k))$ if the series diverges. Thus, for example, it is $O(k \log k (\log\log k)^2)$ and $\Omega(k \log k \log\log k)$.

Restricting to trees, First-Fit is strictly 2-competitive for the online problem. Thus, First-Fit gives a (strict) competitive ratio of 2 regardless of the number of batches. See for example [4] for results on the strict competitive ratio of First-Fit for other graph classes.

Omitted proofs and details appear in the full paper [5].

2 Graph Classes Containing Trees

In this section, we study the problem of coloring trees in k batches. The results hold for any graph class that contains trees as a special case, including bipartite graphs, planar graphs, perfect graphs, and the class of all graphs. If we want the algorithm to be polynomial time, then we are restricted to graph classes where optimal offline coloring is possible in polynomial time (e.g., perfect graphs [8]).

The construction proving the following lemma resembles that of the lower bound of $\Omega(\log n)$ for the competitive ratio for online coloring of trees [10].

Lemma 1. *For any integer $k \geq 1$, any algorithm for k-batch coloring of trees can be forced to use at least $2k$ colors, even if k is known in advance.*

The following lemma holds for any graph, not only trees.

Lemma 2. *There is a strictly k-competitive algorithm for k-batch coloring, even if k is not known in advance.*

Theorem 1 below follows directly from Lemmas 1 and 2.

Theorem 1. *For any graph class containing trees as a special case, the optimal (strict) competitive ratio for k-batch coloring is k, regardless of whether or not k is known in advance.*

3 Interval Coloring in Two Batches

Since not all trees are interval graphs, the lower bound from the previous section does not apply here. For the case of interval graphs we show the surprising result that while coloring in two batches has a tight bound of 2, the problem becomes easier if we assume that the vertices of the graph are revealed together with their interval representation (and this interval representation of vertices of the first batch cannot be modified in the second batch). The standard results for online coloring of interval graphs do not make this distinction: The lower bound is obtained for the (a priori easier) case where the interval representation of a vertex is revealed to the algorithm when the vertex is revealed, while the upper bound holds even if such a representation is not revealed (the online algorithm only computes a maximum clique size containing the new vertex and applies the First-Fit algorithm on a subset of the vertices). Throughout this section, our lower bounds are with respect to the asymptotic competitive ratio while our upper bounds are for the strict competitive ratio, and thus the results are tight for both measures.

Unknown interval representation. We start with a study of the case where the algorithm is guarantied that the resulting graph (at the end of every batch) will be an interval graph, but the interval representation of the vertices of the first batch is not revealed to the algorithm (and may depend on the actions of the algorithm). We show that in this case 2 is the best competitive ratio that can be achieved by an online algorithm.

Theorem 2. *For the problem of 2-batch coloring of interval graphs with unknown interval representation, the optimal (strict) competitive ratio is* 2*.*

Proof. The upper bound follows from Lemma 2. Each of the two induced subgraphs is an interval graph, and it can be colored optimally in polynomial time even if the interval representation is not given.

Next, we show a matching lower bound. For a given $q \in \mathbb{N}$, let $N_1 = \binom{4q}{q} + 1$ and $N_2 = \binom{4q}{2q} + 1$. In the first batch, the adversary gives $N_1 + N_2$ pairwise nonoverlapping cliques: N_1 cliques of size q and N_2 cliques of size $2q$.

Assume that an algorithm uses at most $4q$ colors for the first batch. By the pigeon hole principle, there are two cliques of size q that are colored with the same set $\mathcal{C}_1$ of colors. The vertices of these two cliques will correspond to the intervals $[5, 6]$ and $[9, 10]$, respectively. Similarly, there are two cliques of size $2q$ that are colored with the same set $\mathcal{C}_2$ of colors. For one of these cliques, q vertices will correspond to the interval $[0, 1]$ and the remaining q vertices will correspond to the interval $[0, 3]$. If any of these $2q$ vertices are colored with colors from $\mathcal{C}_1$, they will correspond to the interval $[0, 1]$. We let $\mathcal{C}_2'$ denote the set of colors used on the vertices corresponding to the interval $[0, 3]$. Note that $\mathcal{C}_1 \cap \mathcal{C}_2' = \emptyset$, and hence, $|\mathcal{C}_1 \cup \mathcal{C}_2'| = 2q$. For the other of these two cliques, the q vertices colored with $\mathcal{C}_2'$ will correspond to the interval $[12, 15]$ and the remaining q vertices will correspond to $[14, 15]$. All other intervals are placed to the right of the point 15 so that they do not overlap with any of the four cliques just described.

The second batch consists of q vertices corresponding to the interval $[2, 8]$ and q vertices corresponding to the interval $[7, 13]$. All of these $2q$ intervals overlap with each other and with intervals of all colors in $\mathcal{C}_1 \cup \mathcal{C}'_2$. Thus, the algorithm uses at least $4q$ colors.

No clique is larger than $2q$ vertices, so OPT uses $2q$ colors. Since q can be arbitrarily large, no deterministic online algorithm can be better than 2-competitive, even when considering the asymptotic competitive ratio. □

Known interval representation. We now assume that the vertices are revealed to the algorithm together with their interval representation. For this case, we show an improved competitive ratio of $\frac{3}{2}$. The proof of the following lower bound is a special case of the lower bound proof of Kierstead and Trotter [15].

Lemma 3. *For the problem of 2-batch coloring of interval graphs with known interval representation, no algorithm can achieve a competitive ratio strictly smaller than* $\frac{3}{2}$.

For the matching upper bound, we give a strictly $\frac{3}{2}$-competitive algorithm, called TwoBatches, using Algorithm FB to color the first batch of intervals and Algorithm SB to color the second batch. Intervals can be open, closed, or semi-closed. Let ω denote the maximum clique size in the full graph consisting of intervals from both batches. For any interval I, let color(I) denote the color assigned to I by TwoBatches. Similarly, for a set $\mathcal{I}$ of intervals, color($\mathcal{I}$) denotes the set of colors used to color the intervals in $\mathcal{I}$.

Each endpoint of a first batch interval I is called an *event point*, and this event point is associated with I. If there is a point that is an endpoint of several intervals, we have multiple copies of this point as event points each of which is associated with a different interval. We define a total order, T, on the event points. If $p < p'$, then p appears before p' in T. For the case $p = p'$, there are several cases; see the full paper [5] for details.

First batch. It is well-known that one can color an interval graph with a maximum clique size of ω using ω colors, by maintaining a set of available colors, and traversing the event points according to the total order T: Each time a left endpoint is considered, we color its interval with a color in the set of available colors (removing it from this set); each time a right endpoint is considered, its interval's color is returned to the set of available colors. One often considers the First-Fit rule of using the minimum color in the set of available colors as a tie-breaking rule when the set of available colors contains more than one color. However, in order to establish the improved bound of $\frac{3}{2}$ on the strict competitive ratio (or even for the competitive ratio) of the algorithm for two batches, we need to use a different tie-breaking rule, the one defined by using a stack.

Algorithm FB processes the event points in the order given by T, using a stack ordering for the colors. When a right endpoint is processed, we say that the color of the associated interval is *released* and *available* until it is used again. When processing a left endpoint, the associated interval is colored with the most recently released available color (or a new color, if necessary).

For ease of presentation, we insert 2ω dummy intervals into the first batch: one clique of size ω *before* all input intervals and one clique of size ω *after* all input intervals. Since these dummy cliques do not overlap with any other intervals, each will be colored with the colors $1, 2, \ldots, \omega$, and they will not influence the behavior of Algorithm FB on the rest of the first-batch intervals.

In the following, *Maximal cliques* always refer only to first-batch intervals. For each maximal clique, we choose a point, called a *clique point*, contained in all intervals of the clique. If a clique point p appears to the right of another clique point q, we say that the clique corresponding to p appears to the right of the clique corresponding to q, and vice versa.

For each maximal clique, $\mathcal{I}$, we order the intervals of the clique by left and right endpoints, respectively, resulting in two orderings, $L_{\mathcal{I}}(\cdot)$ and $R_{\mathcal{I}}(\cdot)$. The further an endpoint is from the clique point of $\mathcal{I}$, the earlier the interval appears in the ordering. More precisely, for each interval $I \in \mathcal{I}$, $L_{\mathcal{I}}(I) = i$, if the left endpoint of I appears as the ith in T among the endpoints associated with intervals in $\mathcal{I}$. Similarly, $R_{\mathcal{I}}(I) = j$, if the right endpoint of I appears as the jth last in T among the endpoints associated with intervals in $\mathcal{I}$. As an example, consider the clique $\mathcal{I}$ consisting of the three intervals $a = [1, 6]$, $b = [2, 4]$, and $c = [3, 5]$. For this clique, we have $L_{\mathcal{I}}(a) = 1$, $L_{\mathcal{I}}(b) = 2$, $L_{\mathcal{I}}(c) = 3$ and $R_{\mathcal{I}}(a) = 1$, $R_{\mathcal{I}}(b) = 3$, $R_{\mathcal{I}}(c) = 2$.

Lemma 4. *Consider a maximal clique, $\mathcal{I}_\ell$, of size m and an interval $I_\ell \in \mathcal{I}_\ell$ such that $R_{\mathcal{I}_\ell}(I_\ell) = h$. Let $\mathcal{I}_\ell^h = \{I \in \mathcal{I}_\ell \mid R_{\mathcal{I}_\ell}(I) < h\}$ be the $h - 1$ intervals in $\mathcal{I}_\ell$ with the rightmost right endpoints. Let $\mathcal{I}_r$ be the first maximal clique of size at least h to the right of $\mathcal{I}_\ell$ and let $I_r \in \mathcal{I}_r$ be such that $L_{\mathcal{I}_r}(I_r) = h$. Finally, let p_ℓ be the right endpoint of I_ℓ, let p_r be the left endpoint of I_r, and consider the set $\mathcal{I}'$ of first-batch intervals containing a point p with $p_\ell < p < p_r$ or an endpoint p with $p_\ell <_T p <_T p_r$. Then, color$(\mathcal{I}') \subseteq$ color$(\mathcal{I}_\ell^h)$.*

Second batch. We now describe the algorithm, Algorithm SB, given in pseudocode below, for coloring the second batch intervals.

A *chain* is a set of nonoverlapping second batch intervals. The algorithm starts with partitioning the second-batch intervals into ω chains (some of which may be empty). This is clearly possible, since the graph is ω-colorable.

The second batch intervals are colored in iterations, two chains per iteration. The algorithm keeps a counter, i, which is incremented once in each iteration, and maintains the set BATCH$_2$-COLORED of second batch intervals that the algorithm has already colored. In each iteration, a set of nonoverlapping first-batch intervals is *processed*. The algorithm maintains the invariant that, at the beginning of each iteration, any maximal first-batch clique of size h contains exactly $\min\{h, \omega - i\}$ unprocessed intervals.

A first-batch maximal clique of size at least $\omega - i + 1$ as well as its clique point is said to be *active*. The part of the real line between two neighboring active clique points is called a *region*. Throughout the execution of Algorithm SB, the number of regions is nondecreasing, and whenever a region is split, the chains of the region are also split by a simple projection onto each region and each

resulting region will contain its boundary active clique points (in particular, this means that active clique points may belong to two regions). In each iteration, each region and its chains are treated separately.

The algorithm maintains the invariant that no uncolored second batch interval overlaps with more than one region. This is the key property, allowing the algorithm to consider one region at a time in a given iteration of the algorithm. First-batch intervals overlapping with more than one region will be cut into more intervals, with a cutting point at each active clique point contained in the interval. Thus, by cutting the intervals of an active clique of size h, the clique is replaced by two cliques of size h in neighboring regions. When a first-batch interval is cut into parts, the different parts of the interval may be processed in different iterations, but no new event points are introduced.

In the ith iteration, one chain in each region is colored with the color of a first-batch interval in the region being processed in this iteration, and another chain of the region will be colored with the color $\omega + i$, which has not been used in the region before. For any point p, let d_p be the number of second batch intervals containing p. We say that p is *covered* by a set S of intervals, if there are $\min\{d_p, i\}$ second batch intervals in S containing p.

Next, we define a set $\mathcal{P}$ of *representative points*, such that each interval between two neighboring clique points is represented by one point; see the full paper [5] for details. For a region R, we denote by $\mathcal{P}_R$ the set of representative points contained in region R (that is, $\mathcal{P}_R = R \cap \mathcal{P}$).

We use the following loop invariant for each region to establish that the algorithm TwoBatches is correct and strictly 3/2-competitive. The proof of the invariant I is based on induction on the value of i.

Invariant *I*:

(I1) All points p are covered by the set Batch$_2$-Colored.
(I2) No color used for an unprocessed first-batch interval contained in a region R has been used for a second batch interval intersecting region R so far.
(I3) Each active clique has exactly $\omega - i$ unprocessed intervals.
(I4) For each region R, Chain$_R$ has at most $\omega - 2i$ chains.

We use the invariant I to prove that for any input σ, TwoBatches produces a proper coloring using at most $\left\lfloor \frac{3}{2}\mathrm{OPT}(\sigma) \right\rfloor$ colors (see the full paper [5]). Combining this result with Lemma 3 shows that the optimal (strict) competitive ratio for the problem is $\frac{3}{2}$:

Theorem 3. TwoBatches *has a strict competitive ratio of* $\frac{3}{2}$.

4 Sum Coloring of Graphs in Multiple Batches

We study two cases separately: the case where the number of batches is known to the algorithm from the beginning, and the case where it is not. Once again, our lower bounds are for the competitive ratio and our upper bounds are for the strict competitive ratio.

Algorithm SB: Coloring the second batch intervals.

Mark all first-batch intervals as unprocessed
Create an optimal coloring of the second-batch intervals, using a set $\mathcal{C}$ of ω colors
$R \leftarrow (-\infty, \infty)$ // *Initially, there is only one region*
$\textsc{Chains}_R \leftarrow \emptyset$
$\mathcal{P}_R \leftarrow$ the set of representative points in region R
for each color $c \in \mathcal{C}$ **do**
  $\textsc{Chains}_R \leftarrow \textsc{Chains}_R \cup \{\{I \mid I$ is a second batch interval with color $c\}\}$
$\textsc{Batch}_2\text{-}\textsc{Colored} \leftarrow \emptyset$ // *Set of colored second batch intervals*
$i \leftarrow 0$
while $i < \lfloor \omega/2 \rfloor$ **do** // *Invariant I*
  // Color two chains:
  $i \leftarrow i + 1$
  Split all regions (incl. the assoc. chains and sets of repr. points) at all active clique points
  for *each region R containing at least one nonempty chain* ***do***
    $(\textsc{Chain}_1, \textsc{Chain}_2) \leftarrow \textsc{CreateChains}(R)$ // *See Algorithm* CreateChains
    // Color intervals in $\textsc{Chain}_1$ and $\textsc{Chain}_2$ using a first batch color and a new color:
    $I_\ell \leftarrow$ *the unprocessed first-batch interval of the earliest event point in R*
    $I_r \leftarrow$ *the unprocessed first-batch interval of the latest event point in R*
    Mark I_ℓ and I_r as processed
    Give all intervals in $\textsc{Chain}_1$ *the color of I_ℓ*
    Give all intervals in $\textsc{Chain}_2$ *the color $\omega + i$*
    $\textsc{Batch}_2\text{-}\textsc{Colored} \leftarrow \textsc{Batch}_2\text{-}\textsc{Colored} \cup \textsc{Chain}_1 \cup \textsc{Chain}_2$
    $\textsc{Chain}_R \leftarrow \textsc{Chain}_R \setminus \{\textsc{Chain}_1, \textsc{Chain}_2\}$
// If ω is odd, each region may have one chain left to color:
for *each region R where* $\textsc{Chains}_R$ *contains a nonempty chain* Chain ***do***
  $I \leftarrow$ *the unprocessed first-batch interval with the earliest event point in R*
  Give the intervals of Chain *the color of I*

Number of batches known in advance. We start our study of sum coloring by examining the case where the algorithm knows the number of batches k in advance. Recall that we do not require that algorithms used within one batch be polynomial time.

Lemma 6. *There is a strictly k-competitive algorithm for sum coloring in k batches, if k is known in advance.*

Proof. For each batch, the algorithm, k-BatchColor, applies an optimal procedure, Color, to compute an optimal sum coloring for the subgraph induced by the set of vertices of batch i, separately from previous batches. In order to construct the solution of the input graph, k-BatchColor applies the following transformation: For every vertex v of batch i, if Color colors v with color c, then k-BatchColor colors v using color $f(i,c) = k \cdot (c-1) + i$. This function f satisfies $f(i,c) \equiv i \pmod{k}$, so if $f(i,c) = f(i',c')$, for some $1 \le i, i' \le k$, then $i = i'$. Moreover, if $f(i,c) = f(i,c')$, then $k(c-c') = 0$, and therefore $c = c'$. Thus, vertices of different batches have different colors, and two vertices of the

Algorithm CreateChains(R)

```
1: Chain1 ← a chain in Chains_R containing the leftmost left endpoint
2: Chain2 ← any other chain from Chains_R
3: while some point in P_R is not covered by Batch2-Colored ∪ Chain1 ∪ Chain2 do
4:   p ← the leftmost point in P_R not covered by Batch2-Colored∪Chain1∪Chain2
5:   Chain3 ← a chain from Chains_R containing p
6:   if for all points q < p in P_R, q is contained in Chain3 or in both Chain1 and Chain2 then
7:     Chain2 ← Chain3 // Chain2 now refers to the chain in Chains_R that Chain3 refers to
8:   else
9:     q ← the rightmost point in P_R left of p violating the condition
10:    Chain ← one of Chain1 or Chain2 not containing q // Chain now refers to a chain in Chains_R
11:    // Do a crossover of Chain and Chain3 at the point q, modifying Chain and Chain3 in Chains_R:
12:    Tail ← {I ∈ Chain | I starts to the right of q}
13:    Tail3 ← {I ∈ Chain3 | I starts to the right of q}
14:    Chain ← (Chain \ Tail) ∪ Tail3
15:    Chain3 ← (Chain3 \ Tail3) ∪ Tail
16: return (Chain1, Chain2)
```

same batch have the same color after the transformation if and only if they had the same color in the solution returned by Color. As any proper coloring of the graph provides proper colorings for the k induced subgraphs, the total cost of the k outputs of Color does not exceed the cost of an optimal coloring of the entire graph. For any color c and batch i, $f(i,c) \leq k \cdot c$. Thus, the cost of the output is at most k times the total cost of the k solutions returned by Color (for the k vertex disjoint induced subgraphs). □

We prove a matching lower bound for this case, which holds even for the asymptotic competitive ratio (see the full paper [5]). Combining that result and Lemma 6 gives the following result:

Theorem 4. *For sum coloring in k batches, with k known in advance, the optimal (strict) competitive ratio is k.*

Theorem 5. *For sum coloring of trees in k batches, First-Fit is strictly 2-competitive, and this is the best possible competitive ratio, even if k is known in advance.*

Number of batches unknown in advance. Next, we consider the case where the number of batches k is not known in advance. Thus, to obtain a given competitive ratio, this ratio must be obtained after each batch. Note that the algorithm described in the proof of Lemma 6 cannot be used in this case. While the algorithm is not well defined if k is unknown in advance to the algorithm, it may

seem that modifying the value of k by doubling would result in a competitive ratio of $O(k)$, but no such algorithm exists. We prove that for any positive nondecreasing sequence $f(i)$, which is defined for integer values of i (where $f(i) \geq 1$ for $i \geq 1$), no algorithm with competitive ratio $O(f(k))$ can be given if the series $S_f = \sum_{i=1}^{\infty} \frac{1}{f(i)}$ is divergent. On the other hand, we show that if this series is convergent, then such an algorithm can be given. This shows, in particular, that the best possible competitive ratio is $O(k \log k (\log \log k)^2)$ (since the series for this function converges according to the Cauchy condensation test), and it is $\Omega(k \log k \log \log k)$ (since the series for this function diverges according to the Cauchy condensation test). In fact it is $O(k \log k \log \log k \cdots (\log^{(x)} k)^2)$ and $\Omega(k \log k \log \log k \cdots \log^{(x)} k)$, for any positive integer x.

Consider a sequence $f(i)$ for which S_f is convergent, and let c_f be its limit. We present an algorithm, BatchColor$_f$, for this variant of sum coloring. Initially, all colors are declared *available*. When coloring the ith batch, its induced subgraph is first colored using an optimal procedure, Color. Let t_i denote the maximum color used by Color for batch i. For each $j = 1, 2, \ldots, t_i$ in increasing order, vertices that Color gives color j will be colored using the largest available color among the colors $1, 2, \ldots, \lfloor j \cdot c_f \cdot f(i) \rfloor$. Then, this color is declared *taken*. This color is now unavailable for vertices of future batches and for vertices of the current batch that were assigned a color larger than j by Color. If this process is successful (there always exists an available color), then we say that batch i is *feasible*.

Assuming that all batches are feasible, using arguments similar to those used for Lemma 6, we obtain an upper bound on the competitive ratio of BatchColor$_f$ as follows. Since a color used by Color in a particular batch is assigned to an available color by BatchColor$_f$, if all batches are feasible, each pair, (i, j), where i is a batch number and j is a color assigned by Color in batch i, is given a different color. Since Color produces a proper coloring, BatchColor$_f$ does too. The function f is nondecreasing, so the color assigned to a given vertex by BatchColor$_f$ is at most $c_f \cdot f(k)$ times the color assigned by Color.

Lemma 8. *Consider sum coloring in k batches, where the value of k is not known in advance. If for all $1 \leq i \leq k$, batch i is feasible, then the competitive ratio of* BatchColor$_f$ *is at most $c_f \cdot f(k)$.*

Lemma 9. *All batches for the algorithm* BatchColor$_f$ *are feasible.*

By Lemmas 8 and 9, we obtain:

Theorem 6. *Consider sum coloring in at most k batches and let f be any nondecreasing function with $f(i) \geq 1$ for all $i \geq 1$, whose series S_f converges to c_f. Then, the algorithm* BatchColor$_f$ *is $(c_f \cdot f(k))$-competitive, even if the value k is not known in advance.*

Now, we provide the lower bound.

Theorem 7. *Consider sum coloring in k batches, where the value of k is not known in advance. Let $f(i)$ be a nondecreasing sequence with $f(i) \geq 1$ for all $i \geq 1$, whose series S_f is divergent. Then, there is no constant c such that a competitive ratio of at most $c \cdot f(k)$ can be obtained for all $k \geq 1$.*

Proof. Assume for the sake of contradiction that there exists a constant $c > 1$ and an algorithm A, such that A is $(c \cdot f(k))$-competitive, for any number $k \geq 1$ of batches. Let $C = \max\{2c, 10\}$. Let k be such that $\sum_{i=1}^{k} 1/f(i) > 11C$ (where k must exist as the series S_f is divergent). Fix a large integer M, such that $M > 130 \cdot C^2 \cdot f(k)^2$. We say that a color a is *small* if $a \leq 10CM$.

We now describe an adversarial input. Batch i of the input consists of M^{i-1} cliques of size $3\lfloor M/f(i) \rfloor$. There are no edges between vertices in different cliques of the same batch. A vertex that A colors with a small color is called a *cheap* vertex. For each batch i, if there is at least one clique containing at least $M/f(i)$ cheap vertices, then one such clique is chosen, and the cheap vertices of this clique are called *special* vertices. In each batch, all vertices are connected to all special vertices of previous batches and to no other vertices in previous batches. Thus, no colors used for special vertices can be used in later batches, and there is at most one special vertex for each small color.

The input will contain at most k batches. If, after some batch $i < k$, the sum of colors used by A is larger than $c \cdot f(i)$ times the optimal sum of colors, there will be no more batches. Otherwise, all k batches are given. Thus, if there are fewer than k batches, the theorem trivially follows. Below, we consider the case where there are exactly k batches.

We first give an upper bound on the optimal sum of colors for the first i batches, for $1 \leq i \leq k$.

Claim 1. For every value of i (such that $1 \leq i \leq k$), the optimal sum of colors for the first i batches is at most $19M^{i+1}/(f(i))^2$.

We now show that, by the assumption that A is $(c \cdot f(i))$-competitive on i batches, $1 \leq i \leq k$, each batch i must have a clique with at least $M/f(i)$ cheap vertices. Assume for the sake of contradiction that some batch i does not contain a clique with at least $M/f(i)$ cheap vertices. Then, each clique in the batch contains at most $\lfloor M/f(i) \rfloor$ cheap vertices and hence at least $2\lfloor M/f(i) \rfloor$ vertices with colors larger than $10CM$. Thus, the sum of colors used for this batch is more than $M^{i-1} \cdot 2\lfloor M/f(i) \rfloor \cdot 10CM > 10CM^{i+1}/f(i) \geq 20cM^{i+1}/f(i)$. By Claim 1, this gives a ratio of more than

$$\frac{20cM^{i+1}/f(i)}{19M^{i+1}/(f(i))^2} > c \cdot f(i)\,.$$

Thus, the total number of special vertices is at least $\sum_{i=1}^{k} M/f(i) > 11CM$, contradicting the fact that there is at most one special vertex for each of the small colors. □

References

1. J. Balogh, J. Békési, G. Dósa, G. Galambos, and Z. Tan. Lower bound for 3-batched bin packing. *Discrete Optimization*, 21:14–24, 2016.
2. J. Balogh, J. Békési, G. Galambos, and M. C. Markót. Improved lower bounds for semi-online bin packing problems. *Computing*, 84(1–2):139–148, 2009.
3. A. Bar-Noy, M. Bellare, M. Halldorsson, H. Shachnai, and T. Tamir. On chromatic sums and distributed resource allocation. *Information and Computation*, 140:183–202, 1998.
4. A. Borodin, I. Ivan, Y. Ye, and B. Zimny. On sum coloring and sum multi-coloring for restricted families of graphs. *Theoretical Computer Science*, 418:1–13, 2012.
5. J. Boyar, L. Epstein, L. M. Favrholdt, K. S. Larsen, and A. Levin. Batch coloring of graphs. Technical Report arXiv:1610.02997 [cs.DS], arXiv, 2016.
6. G. Dósa. Batched bin packing revisited. *Journal of Scheduling*, 2015. In press.
7. L. Epstein. More on batched bin packing. *Operations Research Letters*, 44:273–277, 2016.
8. M. Grötschel, L. Lovász, and A. Schrijver. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1(2):169–197, 1981.
9. G. Gutin, T. Jensen, and A. Yeo. Batched bin packing. *Discrete Optimization*, 2:71–82, 2005.
10. A. Gyárfás and J. Lehel. On-line and First-Fit colorings of graphs. *Journal of Graph Theory*, 12:217–227, 1988.
11. A. Gyárfás and J. Lehel. First fit and on-line chromatic number of families of graphs. *Ars Combinatoria*, 29(C):168–176, 1990.
12. M. M. Halldórsson. Online coloring known graphs. *The Electronic Journal of Combinatorics*, 7, 2000.
13. M. M. Halldórsson and M. Szegedy. Lower bounds for on-line graph coloring. *Theoretical Computer Science*, 130(1):163–174, 1994.
14. H. A. Kierstead, D. A. Smith, and W. T. Trotter. First-fit coloring on interval graphs has performance ratio at least 5. *European Journal of Combinatorics*, 51:236–254, 2016.
15. H. A. Kierstead and W. T. Trotter. An extremal problem in recursive combinatorics. *Congressus Numerantium*, 33:143–153, 1981.
16. E. Kubicka. The chromatic sum of a graph: History and recent developments. *International Journal of Mathematics and Mathematical Sciences*, 2004(30):1563–1573, 2004.
17. E. Kubicka and A. J. Schwenk. An introduction to chromatic sums. In *17th ACM Computer Science Conference*, pages 39–45. ACM Press, 1989.
18. N. S. Narayanaswamy and R. S. Babu. A note on First-Fit coloring of interval graphs. *Order*, 25(1):49–53, 2008.
19. S. Nicolosoi, M. Sarrafzadeh, and X. Song. On the sum coloring problem on interval graphs. *Algorithmica*, 23(2):109–126, 1999.
20. G. Zhang, X. Cai, and C. K. Wong. Scheduling two groups of jobs with incomplete information. *Journal of Systems Science and Systems Engineering*, 12:73–81, 2003.