Algorithmica manuscript No.
(will be inserted by the editor)

Batch Coloring of Graphs

Joan Boyar · **Leah Epstein** · **Lene M. Favrholdt** ·
Kim S. Larsen · **Asaf Levin**

Received: date / Accepted: date

Abstract We study two versions of graph coloring, where 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 vertex 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 (forests, bipartite graphs, planar graphs, and perfect graphs, for example), and an interesting 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.

Keywords Online algorithms · Graph coloring · Batches · Interval graphs · Sum coloring

Supported in part by the Danish Council for Independent Research, Natural Sciences, grant DFF-1323-00247, and the Villum Foundation, grant VKR023219. A preliminary version of this paper was presented at the *Fourteenth Workshop on Approximation and Online Algorithms (WAOA)*, Lecture Notes in Computer Science, Springer.

J. Boyar
Department of Mathematics and Computer Science, University of Southern Denmark, Odense, Denmark
E-mail: joan@imada.sdu.dk

L. Epstein
Department of Mathematics, University of Haifa, Haifa, Israel
E-mail: lea@math.haifa.ac.il

L. M. Favrholdt
Department of Mathematics and Computer Science, University of Southern Denmark, Odense, Denmark
E-mail: lenem@imada.sdu.dk

K. S. Larsen
Department of Mathematics and Computer Science, University of Southern Denmark, Odense, Denmark
Tel.: +45 6550 2328
E-mail: kslarsen@imada.sdu.dk

A. Levin
Faculty of IE&M, The Technion, Haifa, Israel
E-mail: levinas@ie.technion.ac.il

1 Introduction

We study several 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, with the batches constituting a partition of the vertex set. Each batch is revealed together with edges connecting vertices of the batch to other vertices of the same or 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. To the best of our knowledge, this is the first time the batch model has been applied to graph coloring.

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 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 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. The number k of batches may be known or unknown to the algorithm. 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 the classic coloring problem, $\mathrm{OPT}(G) = \chi(G)$, 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* (also called the *asymptotic 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, and assigns each vertex the minimal color not assigned to its neighbors that appear before it in the list of vertices. When used as an online algorithm, the order chosen is the order in which the vertices are presented.

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 [11], an algorithm for this case was presented in [7], and better lower bounds were found in [3]. A study of the more general case of k batches was done in [8], and recently, a new lower bound on the competitive ratio of bin packing with three batches was presented in [2]. The scheduling problem of minimizing makespan on identical machines where jobs are presented using two batches was considered in [23].

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 (strict or asymptotic) competitive ratio, even for trees. Gyárfás and Lehel [12] 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 [13], and hence, the optimal competitive ratio on trees is $\Theta(\log n)$. Albers and Schraink [1] show that for chordal, planar, bipartite, inductive, bounded treewidth, and disk graphs, the optimal competitive ratio is also $\Theta(\log n)$, even if the algorithm is randomized and a look-ahead of $O(n/\log n)$ vertices is allowed, and this holds for the asymptotic competitive ratio. For general graphs, Halldórsson and Szegedy [15] have shown that the competitive ratio is $\Omega(n/\log^2 n)$, even for randomized graphs and with various relaxations of the online model such as a look-ahead of $\log^2 n$ vertices.

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. Since trees are 2-colorable, this gives a lower bound of k on the strict competitive ratio of any algorithm coloring trees in k batches. The lower bound is tight, since (on any graph, not only trees), a strictly 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 strict competitive ratio. For graph classes that are colorable in polynomial time, this algorithm is polynomial.

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, breaking ties arbitrarily. For the online version of the problem, Kierstead and Trotter [17] provided an algorithm which uses at most $3\omega - 2$ colors and proved a matching lower bound for any online algorithm. Thus, the optimal (strict as well as asymptotic) competitive ratio is 3.

The algorithm presented in [17] does not depend on the interval representation of the graph, and the lower bound applies even if the interval representation is known to the algorithm, so in the online case the optimal competitive ratio is the same for these two representations (see [16, 20] 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 (and it is not necessary to give the edges explicitly).

Sum coloring. The sum coloring problem (also called chromatic sum) was introduced in [19] (see [18] 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. [4] 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. The time when a job finishes is called its completion time, and the goal is to minimize the average completion time of the jobs. Note that the conflicts can be given by a graph where the jobs are vertices and an edge exists between two vertices, if the corresponding jobs cannot be executed simultaneously. A schedule without idle time is then feasible if and only if there is a coloring of the conflict graph such that the completion time of each job is given by the color of the corresponding vertex. Hence, 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 completion time (response time). The problem when restricted to interval graphs is also motivated by VLSI routing [21]. The first problem seems more likely to come in batches than the second.

The sum coloring problem is NP-hard for general graphs [19] and cannot be approximated within $n^{1-\varepsilon}$ for any $\varepsilon > 0$ unless ZPP = NP [4]. It is also NP-hard for interval graphs [22]. 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 [19]. For online algorithms, there is a lower bound of $\Omega(n/\log^2 n)$ for general graphs with n vertices [14].

We show tight upper and lower bounds of k on the (strict and asymptotic) competitive ratio when there are k batches and k is known in advance to the algorithm. The (strict as well as the asymptotic) 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 (strict as well as asymptotic) 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 $\Omega(k \log k \log\log k)$ and $O(k \log k (\log\log k)^{\delta})$, for any $\delta > 1$.

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 [6] for results on the strict competitive ratio of First-Fit for other graph classes.

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 [10]).

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 [12].

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.*

Proof After each batch, the graph will be a forest. If at some point, at least $2k+1$ distinct colors are used, no further batches will be introduced (that is, any remaining batches will be empty). Thus, in the discussion for each batch, we assume that the vertices of the batch are colored with at most $2k$ colors.

Batch i, $1 \leq i \leq k$, contains $a_i = 2 \cdot (8k^3)^{k-i}$ new vertices. After batch i, the graph will contain $a_i/2$ disjoint *level i trees*. The level of a tree is simply an index and the number of vertices grows quickly with the level. Each level i tree consists of an edge, called the *base edge*, with each endpoint connected to two *good* level j trees, for $1 \leq j \leq i-1$. Note that the endpoints of the base edge belong to batch i (and the other vertices do not). A good level j tree is a level j tree with at least one vertex of each color $1,2,\ldots,2j$. Thus, a good level 1 tree is just one edge, with colors 1 and 2 on its endpoints.

We now explain how the batches are constructed. In particular, we prove that there are enough good level j trees, for each j, $1 \leq j \leq k$. Once this is in place, we have proven that, after the kth batch, each color $1,2,\ldots,2k$ has been used.

Note that the more batches, the more vertices are needed in the first batches. In particular, if $k=2$, then $a_1 = 128$ and $a_2 = 2$, and if $k=3$, then $a_1 = 93,312$, $a_2 = 432$, and $a_3 = 2$.

The first batch is a matching over its vertices. The vertices incident to each edge of the first batch must be colored with two different colors. Since we are assuming that at most $2k$ colors are used, there are less than $2k^2$ distinct pairs of colors. Thus, there are at least $g_1 = a_1/(4k^2)$ edges having the same pair of colors assigned to its vertices. We rename the colors such that these are colors 1 and 2. These edges are then the good level 1 trees and these are the base edges.

In batch $i \geq 2$, there are $a_i/2$ new base edges. Furthermore, for each j, $1 \leq j \leq i-1$, each of the a_i new vertices is connected to a vertex of color $2j-1$ in a good level j tree and to a vertex of color $2j$ in another good level j tree (see Figure 1 for an illustration). Each good level j tree is connected to at most one new vertex, and if it is connected to a new vertex it stops being a good level j tree. Thus, the graph remains a forest. Therefore, the endpoints of each batch i base edge must be colored with two different colors not in $\{1,2\ldots,2i-2\}$. With at most $2k$ colors used, more than $g_i = a_i/(4k^2)$ of these base edges have the same pair c,c' of colors on their endpoints. We rename the colors larger than $2i-2$ such that these two colors are called $2i-1$ and $2i$.

To complete the proof, we now show that there are enough good level j trees, for $1 \leq j \leq k$. For $k=1$, this is clearly true, since $a_k = 2$ and one base edge suffices to force the algorithm to use two colors. For $k \geq 2$, we note that, for $1 \leq j < k$, good level j trees are used to construct trees in all later batches. In each batch $i > j$, exactly $2a_i$ such trees are used as every base edge is connected to four good level j trees (two such trees are connected to each of the endpoints of a base edge). Thus, using that $k \geq 2$, and counting over all values of i, the total number of good level j trees needed is

$$\sum_{i=j+1}^{k} 2a_i = \sum_{i=j+1}^{k} 4 \cdot (8k^3)^{k-i} = 4 \cdot \sum_{i=0}^{k-j-1} (8k^3)^i = 4 \cdot \frac{(8k^3)^{k-j}-1}{8k^3-1} < \frac{(8k^3)^{k-j}}{k^3} = \frac{a_j}{2k^3} \leq \frac{a_j}{4k^2} = g_j,$$

where g_j is the lower bound that we calculated on the number of good level j trees.

If a connected graph is required, one vertex can be added to the last batch, connecting all trees remaining in the forest. □

The construction in Lemma 1 can be changed to use fewer vertices when considering bipartite graphs, rather than restricting to trees (in fact, even for trees it is possible to reduce the size of the construction, making the proof more complicated).

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

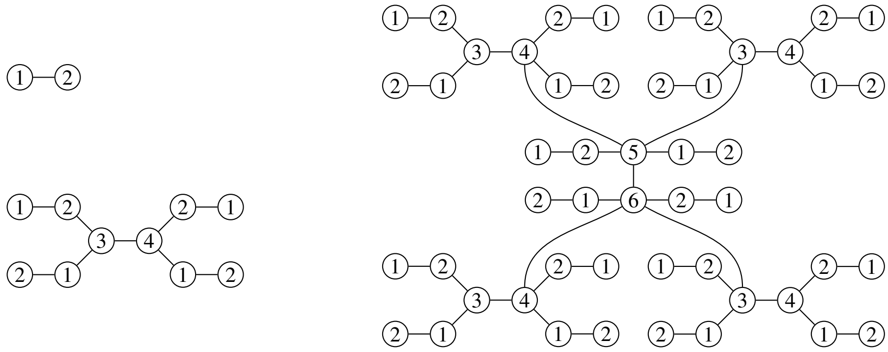

Fig. 1 The single edge on the left hand side (top) is a good level 1 tree. The tree on the left hand side (bottom) is a good level 2 tree. The tree on the right hand side is a good level 3 tree. For $k = 3$, there are $46,656$ level 1 trees, 216 level 2 trees, and one level 3 tree. Moreover, there are at least $1,296$ good level 1 trees, at least six good level 2 trees, and one good level 3 tree.

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

Proof Consider a graph G presented in k batches and let $\chi = \chi(G)$. Since each batch of vertices induces a subgraph of G, each batch can clearly be colored with at most χ colors. Thus, using the colors $(i-1)\chi+1, (i-1)\chi+2, \ldots, (i-1)\chi+\chi$ for batch i yields a strictly k-competitive algorithm. □

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. In particular, the trees constructed for $k \geq 2$ in the proof of Lemma 1 are not interval graphs. For the case of interval graphs we show that while coloring in two batches has a tight bound of 2 on the (asymptotic and strict) competitive ratio, 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.

3.1 Unknown interval representation

We start with a study of the case where the algorithm is guaranteed 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 (strict or asymptotic) 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 and asymptotic) 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 pigeonhole principle, there are two cliques of size q that are colored with the same set $\mathscr{C}_1$ 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 $\mathscr{C}_2$ 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 $\mathscr{C}_1$, they will correspond to the interval $[0,1]$. We let $\mathscr{C}_2'$ denote the set of colors used on the vertices corresponding to the interval $[0,3]$. Restricting our attention to the selected cliques in this manner, $\mathscr{C}_1 \cap \mathscr{C}_2' = \emptyset$, and hence, $|\mathscr{C}_1 \cup \mathscr{C}_2'| = 2q$. For the other of these two cliques, the q vertices colored with $\mathscr{C}_2'$ will correspond to the interval $[12,15]$ and the remaining q vertices will correspond to the interval $[14,15]$. In the interval representation, all other intervals are placed to the right of the point 15 (on the real axis) 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 $\mathscr{C}_1 \cup \mathscr{C}_2'$. 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. □

3.2 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 (strict and asymptotic) competitive ratio of $\frac{3}{2}$. We first state the following lower bound whose proof is a special case of the lower bound proof of Kierstead and Trotter [17].

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

Proof This is a special case of the lower bound of Kierstead and Trotter [17]. The construction is as follows. For a given $q \in \mathbb{N}$, let $N = \binom{3q}{2q}$. In the first batch, the adversary gives $2q$ vertices corresponding to the interval $[4i, 4i+1]$, for $i = 0, 1, \ldots, N$. Thus, the first batch consists of $N+1$ pairwise nonoverlapping cliques. The clique of intervals $[4i, 4i+1]$ is called clique i.

Assume that an algorithm colors the first batch using at most $3q$ colors. Since there are more than N cliques, there must be two cliques, clique k and clique ℓ, colored with the same $2q$ colors. Assume that the cliques are named such that $k < \ell$.

In the second batch, the adversary gives $2q$ vertices corresponding to each of the intervals $[4k, 4k+3]$ and $[4k+2, 4\ell+1]$. To color the second batch vertices, the algorithm will need $4q$ colors different from the $2q$ colors used on clique k and clique ℓ. Thus, the algorithm uses at least $6q$ colors in total.

Since there is no clique larger than $4q$ vertices, OPT uses only $4q$ colors. Since q can be chosen arbitrarily large, this shows that no deterministic online algorithm can be better than $\frac{3}{2}$-competitive. □

For the matching upper bound, we define a polynomial time algorithm, TWOBATCHES, which is strictly $\frac{3}{2}$-competitive, using Algorithm 1 to color the first batch of intervals and Algorithm 2 to color the second batch. Algorithm 2 can be replaced by an optimal extension of the coloring of the first batch intervals to a coloring of all intervals. Though the precoloring extension problem is NP-hard, even when restricted to interval graphs [5], our Algorithm 2 runs in polynomial time as it is based on an optimal coloring of the subgraph induced by the second batch intervals. Since interval graphs can be colored optimally in polynomial time, it is easily verified that both algorithms we suggest are polynomial.

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 $\mathscr{I}$ of intervals, color($\mathscr{I}$) denotes the set of colors used to color the intervals in $\mathscr{I}$.

Intervals can be open, closed, or semi-closed. 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. For two event points p and p', if $p < p'$ (this being the standard ordering on the reals), then p appears before p' in the total order. For the case $p = p'$, let I and I' be the two intervals such that p and p' are associated with I and I', respectively. We consider a total order satisfying the following properties.

1. If p and p' are both right endpoints, $p \notin I$, and $p' \in I'$, then p appears before p' in the total order of the event points.
2. Otherwise, if p is a left endpoint and p' is an right endpoint, then:
   - If $p \notin I$, then p' appears before p in the total order of the event points.
   - If $p \in I$ and $p' \notin I'$, then p' appears before p in the total order of the event points.
   - If $p \in I$ and $p' \in I'$, then p appears before p' in the total order of the event points.
3. Otherwise (that is, p and p' are both left endpoints), then if $p \in I$ and $p' \notin I'$, then p appears before p' in the total order of the event points.

We fix one particular total order, T, on the event points satisfying all these conditions. Observe that this total order is a refinement of the standard order "$\leq$" on the real numbers. If an event point p appears before another event point q in T, we write $p <_T q$. If p, q, and p' are event points, we say that q is *between* p and p', if and only if $p <_T q <_T p'$. In this case, we will also sometimes say that q is to the right of p and to the left of p'.

For any three points p, q, and p', where q is *not* an event point, we say that q is between p and p', if and only if $p < q < p'$. In this case, we will also sometimes say that q is to the right of p and to the left of p'.

3.2.1 First batch.

It is well-known that one can color an interval graph with a maximum clique size of ω using ω colors [9]. If an interval representation is known, this can be done 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 asymptotic competitive ratio) of the algorithm for two batches, we need to use a different tie-breaking rule, namely the rule defined by using a stack, as we describe below. An example of why one cannot use an arbitrary optimal coloring for the first batch is built using a similar structure of intervals to that of the proof of Theorem 2. For a given value of t, consider t intervals of each of the forms $[0,2]$, $[1,4]$, $[6,7]$, $[10,11]$, $[13,16]$, and $[15,17]$. Consider an algorithm that colors the sets of intervals of the forms $[0,2]$, $[6,7]$, $[10,11]$, and $[15,17]$ with colors $1,2,\dots,t$ and colors the sets of intervals of the forms $[1,4]$ and $[13,16]$ with colors $t+1,t+2,\dots,2t$. The second batch contains t intervals of each of the forms $[3,9]$ and $[8,14]$. The clique size in this second batch is $2t$, and each interval in the second batch has an intersection with one interval of the first batch of each color. Thus, an additional $2t$ colors will be used in any coloring of the second batch intervals.

Algorithm 1 processes the event points in the order given by T, that is, from left to right. When a right endpoint is processed, we say that the color of the associated interval is *released*. It is then *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). Thus, a stack ordering is used for the colors. Pseudo-code for the algorithm is given in Algorithm 1. The operations Push() and Pop() operate on the stack in the standard manner.

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 $1,2,\dots,\omega$, and they will not influence the behavior of Algorithm 1 on the rest of the first-batch intervals.

Before analyzing the algorithm, we introduce some additional terminology. *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.

Algorithm 1 Coloring the first-batch intervals.

$\omega_1 \leftarrow$ maximum clique size in the first batch
Initialize an empty stack of colors // *The stack will contain available colors*
for $i \leftarrow \omega_1$ down to 1 **do**
    Push(i)
for each event point, p, in the order given by T **do**
    if p is a *left* endpoint of an interval I **then**
        color(I) $\leftarrow$ Pop()
    else if p is a *right* endpoint of an interval I **then**
        Push(color(I))

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

The following lemma is illustrated in Figure 2.

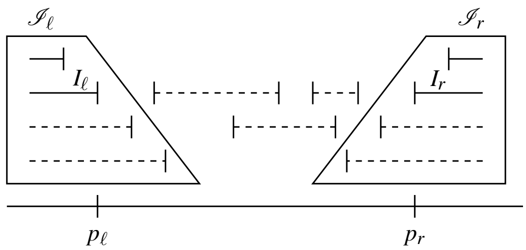

Fig. 2 Illustration of Lemma 4, with $h = 3$. The intervals of $\mathscr{I}'$ are drawn with dashed lines.

Lemma 4 *Consider a maximal clique, $\mathscr{I}_\ell$, of size m and an interval $I_\ell \in \mathscr{I}_\ell$ such that $R_{\mathscr{I}_\ell}(I_\ell) = h$. Let $\mathscr{I}_\ell^h = \{I \in \mathscr{I}_\ell \mid R_{\mathscr{I}_\ell}(I) < h\}$ be the $h-1$ intervals in $\mathscr{I}_\ell$ with the rightmost right endpoints. Let $\mathscr{I}_r$ be the first maximal clique of size at least h to the right of $\mathscr{I}_\ell$ and let $I_r \in \mathscr{I}_r$ be such that $L_{\mathscr{I}_r}(I_r) = h$. 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 $\mathscr{I}'$ 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$(\mathscr{I}') \subseteq$ color$(\mathscr{I}_\ell^h)$.*

Proof For the intervals in $\mathscr{I}_\ell^h$, the lemma follows trivially.

We now consider the intervals in $\mathscr{I}' \setminus \mathscr{I}_\ell^h$. Note that we can assume $p_\ell <_T p_r$, since otherwise $\mathscr{I}'$ is empty, and the lemma follows trivially.

Assume for the sake of contradiction that some interval $I' \in \mathscr{I}' \setminus \mathscr{I}_\ell^h$ has an endpoint to the left of p_ℓ. This interval would overlap with all intervals in $\mathscr{I}_\ell^h$, contradicting the assumption that $p_\ell <_T p_r$. Hence, we only need to consider intervals with a left endpoint between p_ℓ and p_r.

Consider any interval with a left endpoint p such that $p_\ell <_T p <_T p_r$. It follows from the definition of $\mathscr{I}_r$ and I_r that there are more right endpoints than left endpoints between p_ℓ and p (note that $h > 1$ in this case). Thus, at p, the most recently released available color is a color in color($\mathscr{I}_\ell^h$), and therefore, the interval associated with p is colored with a color in color($\mathscr{I}_\ell^h$). □

3.2.2 Second batch.

We now describe the algorithm, Algorithm 2, given in pseudo-code 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). Since the graph is ω-colorable, we can simply choose any optimal coloring of the subgraph induced by the second batch intervals and define our chains to be all intervals of the same color.

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 2, 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 $\mathscr{P}$ of *representative points*, such that each interval between two neighboring clique points is represented by one point. To this end, we define the following equivalence relation between points on the real line. For a pair of points p and p', we say that *p is equivalent to p'* if the following conditions hold:

- For every clique point q, either both $p \leq q$ and $p' \leq q$ or both $p \geq q$ and $p' \geq q$ (this in particular means that p and p' belong to a common region).
- For every interval I of either the first batch or the second batch, we have that either $p, p' \in I$ or the two points p and p' are to a common side of I.

Observe that the number of equivalence classes of this relation is linear in the number of intervals of the input. The set of representative points $\mathscr{P}$ is defined such that each equivalence class has exactly one member in $\mathscr{P}$, chosen arbitrarily. We use the following observation.

Observation 3 *For a given set S of intervals, at any point after line 9 in Algorithm 2, we have that all points are covered by S if and only if all points in $\mathscr{P}$ are covered by S.*

For a region R, we denote by $\mathscr{P}_R$ the set of representative points contained in region R.

We use the following loop invariant for each region to establish that TWOBATCHES is correct and strictly $3/2$-competitive. The proof shown below 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.

Lemma 5 *I is an invariant for the **while**-loop in Algorithm 2.*

Proof We prove by induction on i that the invariant holds at the start of every iteration of the **while**-loop.

Algorithm 2 Coloring the second batch intervals.

1: Mark all first-batch intervals as unprocessed
2: Create an optimal coloring of the subgraph induced by the second-batch intervals, using a set $\mathscr{C}$ of ω colors
3: $R \leftarrow (-\infty, \infty)$ *// Initially, there is only one region*
4: $\text{CHAINS}_R \leftarrow \emptyset$
5: $\mathscr{P}_R \leftarrow$ the set of representative points in region R
6: **for** each color $c \in \mathscr{C}$ **do**
7: $\quad \text{CHAINS}_R \leftarrow \text{CHAINS}_R \cup \{\{I \mid I$ is a second batch interval with color $c\}\}$
8: $\text{BATCH}_2\text{-COLORED} \leftarrow \emptyset$ *// Set of colored second batch intervals*
9: $i \leftarrow 0$
10: **while** $i < \lfloor \omega/2 \rfloor$ **do** *// Invariant I*
11: $\quad$ *// Color two chains:*
12: $\quad i \leftarrow i+1$
13: $\quad$ *Split all regions (incl. the assoc. chains and sets of repr. points) at all active clique points*
14: $\quad$ ***for*** *each region R containing at least one nonempty chain* ***do***
15: $\quad\quad (\text{CHAIN}_1, \text{CHAIN}_2) \leftarrow \text{CREATECHAINS}(R)$ *// See Algorithm 3*
16: $\quad\quad$ *// Color intervals in* CHAIN_1 *and* CHAIN_2 *using a first batch color and a new color:*
17: $\quad\quad I_\ell \leftarrow$ *the unprocessed first-batch interval of the earliest event point in R*
18: $\quad\quad I_r \leftarrow$ *the unprocessed first-batch interval of the latest event point in R*
19: $\quad\quad$ *Mark* I_ℓ *and* I_r *as processed*
20: $\quad\quad$ *Give all intervals in* CHAIN_1 *the color of* I_ℓ
21: $\quad\quad$ *Give all intervals in* CHAIN_2 *the color* $\omega + i$
22: $\quad\quad \text{BATCH}_2\text{-COLORED} \leftarrow \text{BATCH}_2\text{-COLORED} \cup \text{CHAIN}_1 \cup \text{CHAIN}_2$
23: $\quad\quad \text{CHAIN}_R \leftarrow \text{CHAIN}_R \setminus \{\text{CHAIN}_1, \text{CHAIN}_2\}$
24: *// If* ω *is odd, each region may have one chain left to color:*
25: ***for*** *each region R where* CHAINS_R *contains a nonempty chain* CHAIN ***do***
26: $\quad I \leftarrow$ *the unprocessed first-batch interval with the earliest event point in R*
27: $\quad$ *Give the intervals of* CHAIN *the color of I*

Algorithm 3 $\text{CREATECHAINS}(R)$

$\text{CHAIN}_1 \leftarrow$ a chain in CHAINS_R containing the leftmost left endpoint
$\text{CHAIN}_2 \leftarrow$ any other chain from CHAINS_R
while some point in $\mathscr{P}_R$ is *not* covered by $\text{BATCH}_2\text{-COLORED} \cup \text{CHAIN}_1 \cup \text{CHAIN}_2$ **do**
$\quad p \leftarrow$ the leftmost point in $\mathscr{P}_R$ *not* covered by $\text{BATCH}_2\text{-COLORED} \cup \text{CHAIN}_1 \cup \text{CHAIN}_2$
$\quad \text{CHAIN}_3 \leftarrow$ a chain from CHAINS_R containing p
$\quad$ **if** for all points $q < p$ in $\mathscr{P}_R$, q is contained in CHAIN_3 **or** in both CHAIN_1 and CHAIN_2 **then**
$\quad\quad \text{CHAIN}_2 \leftarrow \text{CHAIN}_3$
$\quad$ **else**
$\quad\quad q \leftarrow$ the rightmost point in $\mathscr{P}_R$ left of p violating the condition
$\quad\quad \text{CHAIN} \leftarrow$ one of CHAIN_1 or CHAIN_2 not containing q
$\quad\quad$ *// Do a crossover of* CHAIN *and* CHAIN_3 *at the point q, modifying* CHAIN *and* CHAIN_3 *in* CHAINS_R*:*
$\quad\quad \text{TAIL} \leftarrow \{I \in \text{CHAIN} \mid I$ starts to the right of $q\}$
$\quad\quad \text{TAIL}_3 \leftarrow \{I \in \text{CHAIN}_3 \mid I$ starts to the right of $q\}$
$\quad\quad \text{CHAIN} \leftarrow (\text{CHAIN} \setminus \text{TAIL}) \cup \text{TAIL}_3$
$\quad\quad \text{CHAIN}_3 \leftarrow (\text{CHAIN}_3 \setminus \text{TAIL}_3) \cup \text{TAIL}$
return $(\text{CHAIN}_1, \text{CHAIN}_2)$

(I1) By Observation 3, it suffices to show that the set $\text{BATCH}_2\text{-COLORED}$ covers all points in $\mathscr{P}$.
At the beginning of the first iteration, $i = 0$, so (I1) is trivially true. At the beginning of each of the following iterations, it follows from (I1) that each point p is contained in at least $\min\{d_p, i\}$ intervals in $\text{BATCH}_2\text{-COLORED}$. In line 22, all intervals of $\text{CHAIN}_1 \cup \text{CHAIN}_2$ are added to $\text{BATCH}_2\text{-COLORED}$. Thus,

we only need to prove that, if p is not covered after the increment of i in line 12, the while loop in lines 3–15 of Algorithm 3 will add at least one interval containing p to $\text{CHAIN}_1 \cup \text{CHAIN}_2$.

Consider a region R and let p_ℓ and p_r be defined as in Lemma 4, with $h = \omega - i + 1$. Any point in R to the left of p_ℓ or to the right of p_r is contained in at least h first-batch intervals. Hence, by the induction assumption (I1), there cannot be any uncovered points in R to the left of p_ℓ or to the right of p_r.

As long as some point p in $\mathscr{P}_R$ is not covered by $\text{BATCH}_2\text{-COLORED} \cup \text{CHAIN}_1 \cup \text{CHAIN}_2$, it follows from (I1) and the definition of covered that CHAINS_R contains p and that $\text{CHAIN}_1 \cup \text{CHAIN}_2$ does not contain p. Thus, CHAIN_3 of line 5 of Algorithm 3 exists.

If for all points $q < p$ in $\mathscr{P}_R$, CHAIN_3 contains q or both CHAIN_1 and CHAIN_2 contain q, swapping any of CHAIN_1 or CHAIN_2 with CHAIN_3 will ensure that $\text{CHAIN}_1 \cup \text{CHAIN}_2$ contains all points $q \leq p$ in $\mathscr{P}_R$.

Otherwise, there is a point $q < p$ in $\mathscr{P}_R$ not contained in CHAIN_3 and not contained in both CHAIN_1 and CHAIN_2. The algorithm chooses q as the rightmost such point among the points in $\mathscr{P}_R$. The algorithm then chooses a $\text{CHAIN} \in \{\text{CHAIN}_1, \text{CHAIN}_2\}$ such that CHAIN does not contain q. Since neither CHAIN nor CHAIN_3 contains q, all intervals in $\text{CHAIN} \cup \text{CHAIN}_3$ appear strictly before or after q. Thus it is possible to cut each of CHAIN and CHAIN_3 into two sets, "head" and "tail" consisting of the intervals ending before q or starting after q, respectively, and let the two chains swap tails, while maintaining the property that no two intervals within a chain overlap. We refer to this tails swapping operation as a *crossover*.

After this crossover operation, p is contained in CHAIN. Neither CHAIN_1 nor CHAIN_2 is changed to the left of q. Since all points between q and p were contained in CHAIN_3 or both CHAIN_1 and CHAIN_2 before the crossover, all such points are still contained in $\text{CHAIN}_1 \cup \text{CHAIN}_2$ after the crossover. Thus, the leftmost point in $\mathscr{P}_R$ not covered by $\text{BATCH}_2\text{-COLORED} \cup \text{CHAIN}_1 \cup \text{CHAIN}_2$ now occurs at or to the right of the leftmost point in $\mathscr{P}_R$ to the right of p. This means that, after $O(n)$ iterations of the while loop of lines 3–15 of Algorithm 3, all points in $\mathscr{P}$ are covered by $\text{BATCH}_2\text{-COLORED} \cup \text{CHAIN}_1 \cup \text{CHAIN}_2$.

(I2) At the beginning, the statement is trivially true, since no second-batch interval has been colored.
Since no unprocessed first-batch intervals in a region are colored with color(I_ℓ), according to Lemma 4, and since no first-batch intervals are colored with $\omega + i$, (I2) is maintained.

(I3) At the beginning of the first iteration, (I3) is trivially true, since every active clique has ω first-batch intervals, and all first-batch intervals are unprocessed.
In each iteration, the cliques of size $\omega - i + 1$ are added to the set of active cliques, and one interval of each active clique is processed. Hence, (I3) is maintained.

(I4) At the beginning, the statement holds as an optimal coloring of the subgraph induced by the second batch intervals uses at most ω colors and the number of chains in CHAIN_R after line 7 is the number of colors used.
In each iteration of the while loop in lines 3–15 of Algorithm 3, the number of chains in CHAIN_R is not modified, as in each such iteration we replace two chains by another pair of chains covering the same set of second batch intervals. Invariant (I4) is maintained because the number of chains in CHAIN_R is modified once in every iteration in line 22, where it is decreased by two.

□

We use the invariant I to prove that for any input σ, TWOBATCHES produces a proper coloring using at most $\left\lfloor \frac{3}{2}\text{OPT}(\sigma) \right\rfloor$ colors.

Lemma 6 *For any input σ, the algorithm* TWOBATCHES *produces a proper coloring using at most $\left\lfloor \frac{3}{2}\text{OPT}(\sigma) \right\rfloor$ colors.*

Proof We first note that, by (I1), no chain in CHAINS_R can contain an active clique point. Thus, the splitting of chains done in line 13 is possible.

In each of the $\lfloor \omega/2 \rfloor$ iterations of the while loop of lines 10–23 of Algorithm 2, two chains are colored and the number of chains in CHAIN_R is decreased by two. If ω is odd, one additional chain may be colored in line 27 (and at this point CHAIN_R consists of a single chain by invariant (I4)). Thus, all of the ω chains containing all second batch intervals are colored.

The actual coloring happens in lines 20 and 21, and possibly in line 27. In line 20, the color used is the color, c, of the earliest event point associated with the unprocessed first-batch interval, I_ℓ, in the region R. By Lemma 4 and (I3), I_ℓ and I_r are the only first-batch intervals overlapping with R that are colored with c. By invariant (I1) and the definition of I_ℓ and I_r, no interval in CHAINS_R overlaps with I_ℓ and I_r in R. By invariant (I2), no second batch interval in R has been colored with c in earlier iterations. Thus, coloring the intervals of CHAIN_1 results in a legal coloring of these intervals. The same arguments hold for the possible coloring done in line 27. Since the color $\omega + i$ has never been used before, the coloring of CHAIN_2 is also legal.

In summary, in Algorithm 1, ω colors are used. In Algorithm 2, the colors are used again for some intervals and only $\lfloor \omega/2 \rfloor$ new colors are used. □

Combining Lemmas 3 and 6 shows that the optimal (strict and asymptotic) competitive ratio for the problem is $\frac{3}{2}$:

Theorem 4 TWOBATCHES *has a strict competitive ratio of* $\frac{3}{2}$.

4 Sum Coloring of Graphs in Multiple Batches

Now, we turn our attention to the study of the sum coloring problem. 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 asymptotic competitive ratio and our upper bounds are for the strict competitive ratio.

4.1 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 7 *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 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, $f(i,c) \le k \cdot c$. 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.

Lemma 8 *No algorithm for sum coloring in k batches has an asymptotic competitive ratio strictly smaller than k, even if k is known in advance.*

Proof Assume for the sake of contradiction that there is a value of k and an online algorithm A for sum coloring of graphs in k batches whose asymptotic competitive ratio ρ' is strictly smaller than k. Let ρ be such that $\rho' < \rho < k$. Then there exists an integer M such that for all instances with at least M vertices, the ratio between the cost of the solution returned by A and the cost of an optimal solution is at most ρ. Without loss of generality, we assume that $M > \max\{2k^2, \frac{2\rho}{k-\rho}\}$.

The algorithm will be presented with k batches, such that after every batch i ($1 \le i \le k$) the input either stops (the remaining $k-i$ batches will be empty), or one vertex of batch i (currently the last batch), which will be denoted by v_i, is selected as a designated vertex, and it will be used for constructing the other batches.

Batch i (for $i = 1, 2, \ldots, k$) is constructed as follows: The batch consists of a set V_i of M^i vertices, each of which has $i-1$ neighbors that are $v_1, v_2, \ldots, v_{i-1}$ (thus, the vertices of V_i form an independent set and the vertices $v_1, v_2, \ldots, v_i$ form a clique). For $i \le k-1$, if the algorithm colors all vertices of batch i with colors of value at least k, then the input stops. Otherwise, one vertex whose color is in $\{1,2,\ldots,k-1\}$ is selected to be v_i, and the set of vertices of the next batch, V_{i+1}, is presented. If the input consists of j batches, $V = \cup_{i=1}^{j} V_i$.

We compute an upper bound on the optimal sum of colors, if the input stops after the first j batches (we describe solutions which are not necessarily optimal). If the input consists of j batches, we next show that the set $V \setminus \{v_1, v_2, \ldots, v_{j-1}\}$ is independent. Consider a vertex v of batch i. This vertex is presented with edges to $\{v_1, v_2, \ldots, v_{i-1}\}$. If v does not become v_i, it will not have any further edges. Thus, it is possible to assign color 1 to each such vertex, and use color $i+1$ for v_i. We show that, for $1 \le j \le k$, this gives a total cost of $O_j \le M^j + 2M^{j-1}$. For $j=1$ and $j=2$, we obtain $O_1 = M$ and $O_2 = M^2 + M + 1 \le M^2 + 2M$. For $j \ge 3$, the total cost of this solution is

$$\begin{aligned} O_j &= \sum_{i=1}^{j}(M^i - 1) + 1 + \sum_{i=1}^{j-1}(i+1) = \sum_{i=1}^{j} M^i + \sum_{i=1}^{j-1} i = \frac{M^{j+1}-1}{M-1} - 1 + \frac{j(j-1)}{2} \\ &< \frac{M^{j+1}}{M-1} + k^2 < \frac{M^{j+1}}{M-1} + M^{j-2}, \text{ as } j-2 \ge 1. \end{aligned}$$

We find that

$$\begin{aligned} & \frac{M^{j+1}}{M-1} + M^{j-2} \le M^j + 2M^{j-1} \\ \Leftrightarrow \quad & \frac{M^3}{M-1} + 1 \le M^2 + 2M \\ \Leftrightarrow \quad & M^3 + M - 1 \le M^3 + 2M^2 - M^2 - 2M \\ \Leftrightarrow \quad & 3M \le M^2 + 1, \text{ which holds for any } M \ge 3. \end{aligned}$$

If the input stops after $j < k$ batches, then A has colored M^j vertices with colors of at least k, and its cost is at least $k \cdot M^j$. Otherwise, consider batch k. Each of the vertices $v_1, v_2, \ldots, v_{k-1}$ was given a color no larger than $k-1$, and since they induce a clique, each of the colors $\{1,2,\ldots,k-1\}$ is used exactly once on these $k-1$ vertices. When the set V_k is presented, each vertex of V_k is connected to each vertex in $\{v_1, v_2, \ldots, v_{k-1}\}$, so every vertex of V_k must be colored with a color that is at least k. Thus, in this batch the total cost of the algorithm is at least $k \cdot M^k$.

We showed that if the input stops after batch j (for $1 \le j \le k$), the cost of the algorithm is at least $k \cdot M^j$, while the cost of an optimal solution does not exceed $M^j + 2M^{j-1}$. The performance ratio is thus at least

$$\frac{k \cdot M^j}{M^j + 2M^{j-1}} = \frac{k}{1 + 2/M} > \frac{k}{1 + 2(k-\rho)/(2\rho)} = \frac{2\rho k}{2\rho + 2k - 2\rho} = \rho\,,$$

as $M > \frac{2\rho}{k-\rho}$. This contradicts the assumption that the ratio between the cost of A and the optimal cost is at most ρ. □

Combining Lemmas 7 and 8 gives the following result:

Theorem 5 *For sum coloring in k batches, with k known in advance, the optimal (strict and asymptotic) competitive ratio is k.*

Remark 1 Observe that the graph in the proof of Lemma 8 for the case $k = 2$ has no cycles. Thus, there is no online algorithm for sum coloring of forests in k batches with (strict or asymptotic) competitive ratio strictly smaller than 2. The statement can be strengthened to hold for trees by adding one extra vertex in the second batch which is adjacent to all the isolated vertices from the first batch.

Theorem 6 *For sum coloring of trees in k batches, First-Fit is strictly* 2*-competitive, and for* $k \ge 2$*, this is the best possible (strict and asymptotic) competitive ratio, even if k is known in advance.*

Proof The lower bound of 2 holds by Remark 1. To prove the upper bound of First-Fit, we will show by induction on t that when First-Fit is used for coloring a tree (a connected subgraph) on t vertices, the sum of the colors of the vertices is at most $2t-1$. Obviously, no algorithm can have a cost below t (in fact, if $t > 1$, then any coloring has cost at least $t+1$).

For $t = 1$, the claim follows trivially as a single vertex is assigned color 1. Assume that the claim holds for all $t' < t$ and we prove it for t. Consider the last vertex to be colored by First-Fit. This vertex will be connected to some number of existing trees (connected components of the graph prior to this iteration), and we denote this number (of components) by X. If $X = 0$, then the new vertex gets color 1 and becomes a singleton (so $t = 1$ and the sum of colors is $1 = 2t-1$). If $X > 0$, then for every $j = 1,2,\ldots,X$, the j-th tree with t_j vertices had the sum of colors $2t_j - 1$ (or less). The new vertex has a color not exceeding $X+1$ (as the new vertex is connected to one vertex of each of the X existing trees), and we have $\sum_{j=1}^{X} t_j = t-1$. We find that the total cost of the solution returned by First-Fit does not exceed $\sum_{j=1}^{X}(2t_j - 1) + X + 1 = 2(t-1) - X + (X+1) = 2t-1$. □

4.2 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 7 cannot be used in this case. While the algorithm is not well defined if k is not known to the algorithm in advance, it may seem that modifying the value of k by doubling would result in an asymptotic 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 asymptotic 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 with a strict (and asymptotic) competitive ratio of $O(f(k))$ 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 $O(k \log k \log\log k \cdots (\log^{(x)} k)^{\delta})$ for $\delta > 1$ and $\Omega(k \log k \log\log k \cdots \log^{(x)} k)$, for any positive integer x.

The claims above follow by applying the Cauchy condensation test, which states that a series $\sum_{i=1}^{\infty} f(i)$ converges if and only if $\sum_{i=1}^{\infty} 2^i f(2^i)$ converges. For example, the sum $\sum_{i=3}^{\infty} \frac{1}{i \log i \log\log i}$ converges if and only if the sum $\sum_{i=3}^{\infty} \frac{2^i}{2^i \log(2^i) \log\log(2^i)} = \sum_{i=3}^{\infty} \frac{1}{i \log i}$ converges, and by repeating this, it converges if and only if $\sum_{i=3}^{\infty} \frac{1}{i}$ converges, which it does not. Inductively, this works for any number x of repeated logarithms. In addition, if the last term is raised to the power $\delta > 1$, it does converge, since its convergence is equivalent to the convergence of $\sum_{i=3}^{\infty} \frac{1}{i^{\delta}}$, which converges.

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 7, we obtain an upper bound on the strict 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 9 *Consider sum coloring in $k \geq 2$ 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 strict competitive ratio of* BATCHCOLOR$_f$ *is at most $c_f \cdot f(k)$.*

Lemma 10 *All batches for the algorithm* BATCHCOLOR$_f$ *are feasible.*

Proof Assume that the algorithm has an infeasible batch, let i be the minimal index of a batch that is not feasible, and j be the smallest color that was used by COLOR, for which BATCHCOLOR$_f$ cannot find an available color among the first $\lfloor j \cdot c_f \cdot f(i) \rfloor$ colors. Let $t+1$ be the smallest available color at the time when BATCHCOLOR$_f$ tries to select a color for vertices that COLOR gives color j in batch i. That is, all the t smallest colors were selected earlier (during the first $i-1$ batches or earlier during batch i), and color $t+1$ is still available. By definition, $t+1 > \lfloor j \cdot c_f \cdot f(i) \rfloor$.

The color $t+1$ was available when previous colors were selected. Consider a pair j', ℓ such that $\ell \leq i$, $1 \leq j' \leq t$, and $j' < j$ if $\ell = i$. If $t+1 \leq \lfloor j' \cdot c_f \cdot f(\ell) \rfloor$, then the color selected by BATCHCOLOR$_f$ for COLOR's color j' for batch ℓ is above $t+1$, since the maximum available color no larger than $\lfloor j' \cdot c_f \cdot f(\ell) \rfloor$ was selected. Thus, all colors $1,2,\ldots,t$ were selected for pairs j', ℓ satisfying $\lfloor j' \cdot c_f \cdot f(\ell) \rfloor \leq t$, and thus $j' \cdot c_f \cdot f(\ell) < t+1$. For a given value of ℓ, the number of suitable values of j' is smaller than $\frac{t+1}{c_f f(\ell)}$. As the color $t+1$ cannot be selected for COLOR's color j for batch i, j is one such value for batch i, so for this batch the number of values of j' whose selected colors are no larger than t is smaller than $\frac{t+1}{c_f f(i)} - 1$. The total number of colors strictly below $t+1$ selected in the first i batches just before COLOR's color j for batch i is considered is strictly below $\sum_{\ell=1}^{i} \frac{t+1}{c_f f(\ell)} - 1 \leq \frac{t+1}{c_f} \sum_{\ell=1}^{\infty} \frac{1}{f(\ell)} - 1 = t$, where the last inequality holds since the series converges to c_f, contradicting the assumption that all the first t colors were already selected. □

By Lemmas 9 and 10, we obtain:

Theorem 7 *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 strictly $(c_f \cdot f(k))$-competitive, even if the value k is not known in advance.*

Now, we provide the lower bound.

Theorem 8 *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 an asymptotic 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. Fix a constant $c > c'$. 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). By definition there exists an integer M such that if a graph consisting of at least M vertices is revealed to the algorithm in $i \leq k$ batches, then the cost of the coloring returned by A on this graph is at most $c \cdot f(i)$ times the optimal cost for this graph. Without loss of generality assume 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: 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 prove the claim: Consider the following proper coloring. For each clique K, let n_K denote the number of vertices in K that are not special. These vertices are colored using the colors $1,2,\ldots,n_K$. Each special vertex v is given the color $3M+b$, where b is the color assigned to v by A. As the vertices of each clique are only connected to special vertices of previous batches, and they are not connected to vertices of other cliques of the same batch, this coloring is proper.

For $i=1$, there is only one clique, and the sum of colors in the coloring, where there is one vertex of every color in $\{1,2,\ldots 3\lfloor \frac{M}{f(i)} \rfloor\}$, is

$$\sum_{\ell=1}^{3\lfloor \frac{M}{f(i)} \rfloor} \ell < \frac{9M^2}{(f(i))^2} < \frac{19M^{i+1}}{(f(i))^2}\,.$$

For $i \geq 2$, the sum of the colors of special vertices is at most

$$\sum_{\ell=3M+1}^{3M+10CM} \ell < (10CM)(3M+10CM) < 130C^2M^2 < \frac{M^3}{(f(k))^2} \leq \frac{M^3}{(f(i))^2} \leq \frac{M^{i+1}}{(f(i))^2},$$

and the sum of the colors of the remaining vertices is at most

$$\sum_{j=1}^{i} M^{j-1} \sum_{\ell=1}^{3\lfloor \frac{M}{f(j)} \rfloor} \ell < \sum_{j=1}^{i} M^{j-1} \frac{9M^2}{(f(j))^2} = \sum_{j=1}^{i} \frac{9M^{j+1}}{(f(j))^2} < \frac{18M^{i+1}}{(f(i))^2},$$

where the last inequality follows by showing that for every $j \leq i$ we have $\frac{9M^{j+1}}{(f(j))^2} \leq \frac{9M^{i+1}}{(f(i))^2} \cdot \frac{1}{2^{i-j}}$ by induction on $i-j$. For $i-j=0$, the claim trivially holds. Assume that it holds for $i-j$ and denote $j'=j-1$, we will show it for $i-j'$. We have

$$\frac{9M^{j'+1}}{(f(j'))^2} = \frac{9M^{j+1}}{(f(j))^2} \cdot \frac{(f(j))^2}{M \cdot (f(j'))^2} \leq \frac{9M^{j+1}}{(f(j))^2} \cdot \frac{(f(i))^2}{M} \leq \frac{1}{2} \cdot \frac{9M^{j+1}}{(f(j))^2} \leq \frac{9M^{i+1}}{(f(i))^2} \cdot \frac{1}{2^{i-j'}}$$

where the first inequality holds because $1 \leq f(j') \leq f(j) \leq f(i)$, the second inequality holds by our choice of M, and the last inequality holds by the induction assumption. Thus, for this coloring, the total sum of the colors is less than $19M^{i+1}/(f(i))^2$.

This concludes the proof of the claim.

We now show that, by the assumption that the cost of the coloring returned by A is at most $c \cdot f(i)$ times the optimal cost of the graph consisting of 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 the claim, this gives a ratio between the two costs 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 small color. □

References

1. S. Albers and S. Schraink. Tight bounds for online coloring of basic graph classes. *ArXiv*, 2017. arXiv:1702.07172 [cs.DS].
2. J. Balogh, J. Békési, G. Dósa, G. Galambos, and Z. Tan. Lower bound for 3-batched bin packing. *Discrete Optimization*, 21:14–24, 2016.
3. 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.
4. 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.
5. M. Biró. Precoloring extension. I. Interval graphs. *Discrete Mathematics*, 100(1–3):267–279, 1992.
6. 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.
7. G. Dósa. Batched bin packing revisited. *Journal of Scheduling*, 20:199–209, 2015.
8. L. Epstein. More on batched bin packing. *Operations Research Letters*, 44:273–277, 2016.
9. M. C. Golumbic. *Algorithmic Graph Theory and Perfect Graphs*. Academic Press, 1980.
10. M. Grötschel, L. Lovász, and A. Schrijver. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1(2):169–197, 1981.
11. G. Gutin, T. Jensen, and A. Yeo. Batched bin packing. *Discrete Optimization*, 2:71–82, 2005.
12. A. Gyárfás and J. Lehel. On-line and First-Fit colorings of graphs. *Journal of Graph Theory*, 12:217–227, 1988.
13. A. Gyárfás and J. Lehel. First fit and on-line chromatic number of families of graphs. *Ars Combinatoria*, 29(C):168–176, 1990.
14. M. M. Halldórsson. Online coloring known graphs. *The Electronic Journal of Combinatorics*, 7, 2000.
15. M. M. Halldórsson and M. Szegedy. Lower bounds for on-line graph coloring. *Theoretical Computer Science*, 130(1):163–174, 1994.

16. 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.
17. H. A. Kierstead and W. T. Trotter. An extremal problem in recursive combinatorics. *Congressus Numerantium*, 33:143–153, 1981.
18. E. Kubicka. The chromatic sum of a graph: History and recent developments. *International Journal of Mathematics and Mathematical Sciences*, 2004(30):1563–1573, 2004.
19. E. Kubicka and A. J. Schwenk. An introduction to chromatic sums. In *17th ACM Computer Science Conference*, pages 39–45. ACM Press, 1989.
20. N. S. Narayanaswamy and R. S. Babu. A note on First-Fit coloring of interval graphs. *Order*, 25(1):49–53, 2008.
21. S. Nicolosoi, M. Sarrafzadeh, and X. Song. On the sum coloring problem on interval graphs. *Algorithmica*, 23(2):109–126, 1999.
22. T. Szkaliczki. Routing with minimum wire length in the dogleg-free Manhattan model is NP-complete. *SIAM J. Comput.*, 29(1):274–287, 1999.
23. 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.