On bin packing with conflicts

Leah Epstein[1] and Asaf Levin[2]

[1] Department of Mathematics, University of Haifa, 31905 Haifa, Israel.
Email: lea@math.haifa.ac.il
[2] Department of Statistics, The Hebrew University, Jerusalem, Israel.
levinas@mscc.huji.ac.il.

Abstract. We consider the offline and online versions of a bin packing problem called BIN PACKING WITH CONFLICTS. Given a set of items $V = \{1, 2, \ldots, n\}$ with sizes $s_1, s_2 \ldots, s_n \in [0, 1]$ and a conflict graph $G = (V, E)$, the goal is to find a partition of the items into independent sets of G, where the total size of each independent set is at most one, so that the number of independent sets in the partition is minimized. This problem is clearly a generalization of both the classical (one-dimensional) bin packing problem where $E = \emptyset$ and of the graph coloring problem where $s_i = 0$ for all $i = 1, 2, \ldots, n$. Since coloring problems on general graphs are hard to approximate, following previous work, we study the problem on specific graph classes. For the offline version we design improved approximation algorithms for perfect graphs and other special classes of graphs, these are a $\frac{5}{2} = 2.5$-approximation algorithm for perfect graphs, a $\frac{7}{3} \approx 2.33333$-approximation for a sub-class of perfect graphs, which contains interval graphs, and a $\frac{7}{4} = 1.75$-approximation for bipartite graphs. For the online problem on interval graphs, we design a 4.7-competitive algorithm and show a lower bound of $\frac{155}{36} \approx 4.30556$ on the competitive ratio of any algorithm. To derive the last lower bound, we introduce the first lower bound on the asymptotic competitive ratio of any online bin packing algorithm with known optimal value, which is $\frac{47}{36} \approx 1.30556$.

1 Introduction

We consider the following BIN PACKING WITH CONFLICTS problem (BPC) (see [15, 3] and also the information on the bin packing problem given in [4]). Given a set of items $V = \{1, 2, \ldots, n\}$ with sizes $s_1, s_2 \ldots, s_n \in [0, 1]$ and a conflict graph $G = (V, E)$, the goal is to find a partition of the items into independent sets of G where the total size of each independent set is at most one, so that the number of independent sets in the partition is minimized. This problem is clearly a generalization of both the classical (one-dimensional) bin packing problem where $E = \emptyset$ and of the graph coloring problem where $s_i = 0$ for all $i = 1, 2, \ldots, n$. In an online environment, items arrive one by one to be packed immediately and irrevocably. A new item is introduced by its size, together with all its edges in the current conflict graph (i.e., edges which connect it to previously introduced items).

This problem arises in assigning processes or tasks to processors. In this case we are given a set of tasks, where some pairs of tasks are not allowed to execute on the same processor due to efficiency or fault tolerance reasons. The goal is to assign a minimum number of processors to this set of processes given that the makespan is bounded by some constant (see Jansen [14]). Other applications of this problem arise in the area of database replicas storage, school course time tables construction, scheduling communication systems (see de Werra [5]), and finally in load balancing, the parallel solution of partial differential equations by two dimensional domain decomposition (see Irani and Leung [13]). We follow earlier work and consider the BPC on sub-classes of perfect graphs. This restriction is motivated by the theoretical hardness of approximating graph coloring on general graphs.

In order to analyze our approximation and online algorithms we use common criteria which are the approximation ratio (also called performance guarantee) and competitive analysis. For an algorithm $\mathcal{A}$, we denote its cost by $\mathcal{A}$ as well. An optimal offline algorithm that knows the complete sequence of items is denoted by OPT. We consider the (absolute) approximation (competitive) ratio that is defined as follows. The (absolute) approximation (competitive) ratio of $\mathcal{A}$ is the infimum $\mathcal{R}$ such that for any input, $\mathcal{A} \leq \mathcal{R} \cdot \mathrm{OPT}$. If the absolute approximation (competitive) ratio of an offline (online) algorithm is at most ρ we say that it is a ρ-approximation (ρ-competitive). For the offline problem, we restrict ourselves to algorithms that run in polynomial time. Our online algorithm is also a polynomial time algorithm (though this property is not always required in the competitive analysis literature). We focus on the absolute criteria and not on the criteria of asymptotic approximation ratio and asymptotic competitive ratio (these criteria are commonly used for bin packing problems) since a conflict graph can allow us to magnify small bad instances into large ones (with large enough values of OPT) with the same absolute ratio. So in general, we do not expect to have a better asymptotic approximation ratio than the corresponding absolute approximation ratio, even though this may be possible.

Since the BPC problem generalizes the classical coloring problem that is known to be extremely hard to approximate, we follow earlier studies and consider the BPC problem on the class of perfect graphs for which the coloring problem is polynomially solvable (see [25]). The best previously known approximation algorithm for BPC on perfect graphs is the algorithm of Jansen and Öhring [15] with an approximation ratio of 2.7. In Section 3.1 we improve this result and present our 2.5-approximation algorithm for BPC on perfect graphs.

Following Jansen and Öhring [15] we consider the class of graphs for which one can solve in polynomial time the PRECOLORING EXTENSION problem defined as follows. Given an undirected graph $G = (V, E)$ and k distinct vertices $v_1, v_2, \ldots, v_k$, the problem is to find a minimum coloring f of G such that $f(v_i) = i$ for $i = 1, 2, \ldots, k$. This problem is reviewed in [12, 21], and it is known to be polynomially solvable for the following graph classes: interval graphs, forests, split graphs, complements of bipartite graphs, cographs, partial

K-trees and complements of Meyniel graphs[3] (see [12] for a review of these results) and it is also polynomially solvable for chordal graphs as shown by Marx [22]. However, it is known to be NP-complete for bipartite graphs [12]. We denote by $\mathcal{C}$ the class of graphs G for which one can solve in polynomial time the precoloring extension problem for any induced subgraph of G (including G itself). I.e., $\mathcal{C}$ is closed under the operation of induced subgraph extraction. Jansen and Öhring [15] analyzed the following *algorithm with precoloring* for the case where G belongs to $\mathcal{C}$. Denote the set of large items by $L = \{j : s_j > \frac{1}{2}\}$, and denote by $\chi_I(G)$ the minimum number of colors used by an optimal solution for the precoloring extension problem defined by G. Finally, we define the set of precolored vertices to be L. Compute a feasible coloring of G using $\chi_I(G)$ colors, where for each pair of items in L they are assigned different colors. For each color class, apply a bin-packing heuristic such as the First-Fit-Decreasing algorithm. They proved that the resulting algorithm is a $\frac{5}{2}$-approximation algorithm. In Section 3.2 we improve this result by presenting a $\frac{7}{3}$-approximation algorithm.

For all $\varepsilon > 0$ Jansen and Öhring [15] also presented a $(2+\varepsilon)$-approximation algorithm for BPC on cographs and partial K-trees. Furthermore, they presented a 2-approximation algorithm for bipartite graphs. A d-inductive graph has the property that the vertices can be assigned distinct numbers $1, \ldots, n$ such that each vertex is adjacent to at most d lower numbered vertices. Jansen [14] showed an asymptotic fully polynomial time approximation scheme for BPC on d-inductive graphs where d is a constant. This result includes the cases of trees, grid graphs, planar graphs and graphs with constant treewidth. Oh and Son [24] and McCloskey and Shankar [23] considered BPC on graphs that are union of cliques, but their results are inferior to the 2.7-approximation algorithm of Jansen and Öhring [15].

The hardness of approximation of BPC follows from the hardness of standard offline bin packing (with respect to the absolute approximation ratio). It is not hard to see that unless $\mathbf{P} = \mathbf{NP}$, no algorithm can have absolute approximation ratio of less than $\frac{3}{2}$ (due to a simple reduction from the PARTITION problem, see problem SP12 in [8]). Since standard bin packing is a special case of BPC, where the conflict graph is an independent set, we get that for all graph classes studied in this paper, BPC is **APX**-hard, and unless $\mathbf{P} = \mathbf{NP}$, cannot be approximated within a factor smaller than $\frac{3}{2}$. Note that for bin packing, already the simple First-Fit-Decreasing algorithm is a $\frac{3}{2}$-approximation [27].

Our results. In Section 2 we describe the methods applied in this paper. We use weights for our analysis. The weights used throughout the paper have the unique and novel property that weights are assigned not only as a function of size of items, but also as a function of the location of items in an optimal solution or in an approximate solution. We think that this new technical approach can contribute to the analysis of algorithms for other problems as well.

We use these methods in Section 2 to give improved and tight bounds on two algorithms designed in [15]. We show that their algorithm for perfect graphs has performance guarantee of approximately 2.691 and their algorithm with pre-

[3] A graph is Meyniel if every cycle of odd length at least five has at least two chords.

coloring has performance guarantee of approximately 2.423. These tight results follow from our analysis together with bad examples for these algorithms given in [15]. Note that these bounds and their proofs resemble the analysis of the Harmonic algorithm [19] (the bounds are one unit higher than the upper bounds for Harmonic). However, neither the algorithms of [15] nor our algorithms use a partition into classes as is done in the Harmonic algorithm. Moreover, such a partition in our case would result in an arbitrarily high approximation ratios.

In Section 3 we present our improved new algorithms for the offline case of BPC. In Section 3.1 we design an improved algorithm for perfect graphs with performance guarantee of 2.5. Our algorithm is also a 2.5-approximation algorithm for BPC on all graph classes where one can solve the regular coloring problem (i.e., coloring the vertex set of a graph using a minimum number of colors) in polynomial time. In Section 3.2 we design an improved algorithm with precoloring with performance guarantee of $\frac{7}{3}$. In Section 3.3 we design a $\frac{7}{4}$-approximation algorithm for bipartite graphs.

In Section 4 we discuss *online* algorithms for BPC on interval graphs. We design a simple 4.7-competitive algorithm and show a lower bound of $\frac{155}{36} \approx 4.30556$ on the competitive ratio of any online algorithm. We derive the last lower bound by introducing the first non-trivial lower bound for online bin packing with known optimal value, which is $\frac{47}{36} \approx 1.30556$. We also show an $O(\log n)$ competitive algorithm for bipartite graphs, which is best possible. Both algorithms are adaptations of online algorithms for the standard coloring problem, see [18, 20].

2 Weighting functions and the performance of FFD based algorithms

In this section, we define weighting functions which are a major tool in the analysis of algorithms for bin packing. The weights defined in this section are later adapted and used for the analysis of our improved algorithms.

The idea of such weights is simple. An item receives a weight according to its size and its packing in some fixed solution. The weights are assigned in a way that the cost of an algorithm is close to the total sum of weights. In order to complete the analysis, it is usually necessary to consider the total weight that can be packed into a single bin of an optimal solution.

In this paper, we exploit this method in order to achieve improved algorithms for BPC. Though this method was not applied to BPC before, it was widely used for standard bin packing, and many variants on bin packing. This technique was used already in 1971 by Ullman [28] (see also [17, 19, 26]).

In this section, we define a set of weights which depends solely on the size of items. For an item x such that $s_x > \frac{1}{2}$ we define $weight(x) = 1$. We define the interval $\mathcal{I}_1$ by $\mathcal{I}_1 = (\frac{1}{2}, 1]$. For an item x such that $s_x \leq \frac{1}{2}$, let j be an integer such that $s_x \in \mathcal{I}_j = (\frac{1}{j+1}, \frac{1}{j}]$. We define $weight(x) = s_x + \frac{1}{j(j+1)}$. Note that even though this classification to intervals was used before, the weight function is non-standard. Typically either all items in an interval receive the same weight or are scaled by a common multiplicative factor (see e.g. [19, 2]). We note that

the weight function does not round up the size of an item to the next unit fraction.

We need to use this special weight function in order to make sure that the amount of weight is large enough, even if the input is partitioned into several classes, each of which is packed separately. On the other hand, we must make sure that the weights are not too large, so that the bound on the performance guarantee is not increased artificially. A similar (though different) weight function was used before by Galambos and Woeginger [6]. Their weight function can be used to prove Corollary 2 and Theorem 1 but not the other results of this paper. Therefore, we need to modify the weight function of [6] for our needs.

Given this set of weights, we note that for an item x of size $s_x \in \mathcal{I}_j$ ($j \geq 2$), the ratio between its weight and its size is bounded as follows, $\frac{j+2}{j+1} \leq \frac{weight(x)}{s_x} < \frac{j+1}{j}$.

For a set of items X, we denote the sum of weights of all items in X by $W(X)$. I.e. $W(X) = \sum_{x \in X} weight(x)$. We next show that any algorithm which first partitions the input into μ classes, and then applies the algorithm First-Fit-Decreasing on each class separately, satisfies the following condition on its cost as a function of the total weight and μ.

Lemma 1. *Consider an algorithm $\mathcal{A}$ and a subset of items J which forms an independent set and is packed using First-Fit-Decreasing (*FFD*). Let Y be the number of bins used for this packing. Then we have $Y \leq W(J) + 1$.*

Proof. Note that for the above weight function, any bin which contains an item of size in $(\frac{1}{2}, 1]$ has total weight of items at least 1. Note also that the weight of an item in $\mathcal{I}_j = (\frac{1}{j+1}, \frac{1}{j}]$ is at least $\frac{1}{j+1} + \frac{1}{j(j+1)} = \frac{1}{j}$. Therefore, any bin which contains j items of size in the interval $(\frac{1}{j+1}, \frac{1}{j}]$ has total weight of at least 1. We can remove such bins from the packing and focus on all other bins called *transition bins* (if no bins are left after the removal, we are done).

A transition bin contains only items whose size is at most $\frac{1}{2}$. Note that the last bin ever opened may result in a transition bin, and it contains at least one item. Moreover, let a transition bin be of *type* j (for some $j \geq 2$), if the first item ever packed into it has size in $\mathcal{I}_j$. Next, we argue that there can be at most one transition bin of each type. Since the items are packed using FFD, transition bins are created in a sorted order, starting with the smallest type. If there are two bins of the same type j, this means that during the time between the packing the first items in these two bins, all packed items were also of size in interval $\mathcal{I}_j$. Therefore, the first bin must be assigned j such items before the second transition bin of this type is opened, and thus the first bin is not a transition bin. Let k be the largest type of any transition bin ever opened (i.e., the transition bin with the smallest item). Remove from the packing all items of size at most $\frac{1}{k+1}$. This removal may only decrease the total weight. As stated above, the weight of all remaining items in the transition bins is at least a multiplicative factor of $\frac{k+2}{k+1}$ their size.

Let α be the size of the first item in the last transition bin. Since the last transition bin is opened, all other bins have a total size of items which is more than $1 - \alpha$. Let $i_1 < \ldots < i_t < k$ be the sorted list of types of transition bins.

We consider two cases which are $t \leq \lfloor \frac{k+2}{2} \rfloor$, and $t > \lfloor \frac{k+2}{2} \rfloor$. In both cases we need to show that the total weight in all transition bins is at least t (since there are $t+1$ transition bins).

In the first case, if $t = 0$ we are done. Assume therefore $t \geq 1$. We get a total weight of at least $t(1-\alpha)\frac{k+2}{k+1} + \alpha + \frac{1}{k(k+1)} = t\frac{k+2}{k+1} + \frac{1}{k(k+1)} - \alpha(t\frac{k+2}{k+1} - 1) \geq t\frac{k+2}{k+1} + \frac{1}{k(k+1)} - \frac{t\frac{k+2}{k+1}-1}{k} = t\frac{k+2}{k+1}\frac{k-1}{k} + \frac{k+2}{k(k+1)}$. The inequality holds since the coefficient multiplied by α is negative and $\alpha \leq \frac{1}{k}$. We need to show that the weight is at least t, i.e. that $t(\frac{k^2+k-2}{k^2+k} - 1) + \frac{k+2}{k(k+1)} = \frac{-2t}{k^2+k} + \frac{k+2}{k^2+k} \geq 0$. We get that this holds for $t \leq \frac{k+2}{2}$.

Consider the second case. The proof of the first case shows that it is enough to consider the first $f = t - \lfloor \frac{k+2}{2} \rfloor$ transition bins, and to show that the total weight of items in these bins is at least f. These bins are bins of types $i_1, \ldots, i_f$. Consider the bin of type i_{f+1}. Note that $i_{f+1} \leq k - \lfloor \frac{k+2}{2} \rfloor$ since no two transition bins are of the same type, and $i_{f+1} \geq 3$, since $i_f \geq 2$. Let β be the size of the first item in the bin of type i_{f+1}. Let $m = i_{f+1}$. Considering only items of sizes in $(\frac{1}{m}, \frac{1}{2}]$, we have that each bin out of the first f transition bins has total size of such items of at least $1 - \beta$. However, they also have a total size of items in $(\frac{1}{k+1}, \frac{1}{2}]$ of at least $1 - \alpha$. Therefore the weight of items in each such bin is at least $(1-\beta)\frac{m+2}{m+1} + (\beta-\alpha)\frac{k+2}{k+1} = \frac{m+2}{m+1} + \beta(\frac{1}{k+1} - \frac{1}{m+1}) - \alpha\frac{k+2}{k+1}$. We will show that this amount is never smaller than 1. This expression is minimized for maximum values of α, β and thus we need to show, $(1 - \frac{1}{m}) \cdot \frac{m+2}{m+1} + (\frac{1}{m} - \frac{1}{k}) \cdot \frac{k+2}{k+1} - 1 \geq 0$, which is equivalent to $\frac{k-m}{km} \cdot \frac{k+2}{k+1} \geq \frac{2}{m(m+1)}$. Note that $k - m \geq \frac{k+1}{2}$, $m + 1 \geq 4$ and thus $\frac{k-m}{km}\frac{k+2}{k+1} \cdot \frac{m(m+1)}{2} \geq \frac{k+2}{k} > 1$. This completes the proof.

In the sequel, we consider algorithms for an input I of the following structure. The set I is partitioned into ν independent sets. Out of these sets $\mu \leq \nu$ are packed using FFD. Each other independent set J is packed into a single bin and is assigned a total weight of at least 1.

Corollary 1. *An algorithm $\mathcal{B}$ as above satisfies $\mathcal{B} \leq W(I) + \mu$.*

We now give a tight analysis of the FFD based algorithm given in [15] for perfect graphs. That algorithm finds a coloring of all items with a minimum number of colors, and then uses FFD to pack each color class. It was shown in [15] that the performance guarantee of this algorithm is at most 2.7 and at least $1 + \Pi_\infty \approx 2.69103$. The value Π_∞ is the sum of a series and is computed using the well known sequence π_i, $i \geq 1$, which often occurs in bin packing. Let $\pi_1 = 2, \pi_{i+1} = \pi_i(\pi_i - 1) + 1$. Then $\Pi_\infty = \sum_{i=1}^{\infty} \frac{1}{\pi_i - 1}$. This sequence is presented e.g. in [2, 19].

We are now ready to prove a matching upper bound of $1 + \Pi_\infty = 2.691$ for this algorithm. In order to do so, we need to find an upper bound on the total weight which can reside in one bin. The proof is similar to those of [2, 19], however our weights are defined differently since these proofs do not hold in our case. We assume that the weight of an item s_x of size in $\mathcal{I}_1$ is $x + \frac{1}{2}$, which may only increase the total weight, since we assigned weight 1 to these items.

Lemma 2. *Consider a set of items J packed into one bin in* OPT. *Then* $W(J) \leq \Pi_\infty \approx 1.69103$.

Proof. We define the "increase" of an item by its weight minus its size, i.e., $weight'(x) = weight(x) - s_x$. Let $W'(X)$, for a set X, be the sum of the increases of items in X. We need to show that $W'(J) \leq \Pi_\infty - 1$.

We show that $\Pi_\infty - 1$ is the supremum increase of a set of items that fits into a single bin.

First, consider the sequence π_i, using its definition, we get that for any value of k, and small enough $\delta > 0$, the following holds $\sum_{i=1}^{k}(\frac{1}{\pi_i} + \delta) < 1$. Taking $k \to \infty$ we get that the total increase in a bin which contains such items tends to $\sum_{i=1}^{\infty} \frac{1}{\pi_i(\pi_i - 1)} = \sum_{i=1}^{\infty} \frac{1}{\pi_{i+1} - 1} = \Pi_\infty - 1$.

Assume by contradiction that there exists a value $\varepsilon > 0$ (which is an inverse of an integer $M = \frac{1}{\varepsilon}$) and a set of items for which the sum of increases is at least $\Pi_\infty - 1 + \varepsilon$. We prove by induction on i that the bin must contain exactly one item of each interval $\mathcal{I}_{\pi_i - 1}$, for $i \geq 1$.

Assume that we already proved that the bin must contain items from the intervals $\mathcal{I}_{\pi_1 - 1}, \ldots, \mathcal{I}_{\pi_{i-1} - 1}$. We prove that it must contain an item of $\mathcal{I}_{\pi_i - 1}$. It is not difficult to show that $1 - \sum_{j=1}^{i-1} \frac{1}{\pi_j} = \frac{1}{\pi_i - 1}$, for $i \geq 1$. Thus, the largest interval from which the next item can come from is $\mathcal{I}_{\pi_i - 1}$. Assume that there is no such item. Thus, all item sizes are from the interval $(0, \frac{1}{\pi_i}]$ and the increase of an item is smaller than a multiplicative factor of $\frac{1}{\pi_i}$ of its size. The total increase is therefore smaller than $\sum_{j=1}^{i-1} \frac{1}{\pi_j(\pi_j - 1)} + \frac{1}{\pi_i - 1} \cdot \frac{1}{\pi_i} = \sum_{j=1}^{i} \frac{1}{\pi_{i+1} - 1} < \Pi_\infty - 1$.

Continue this process until the upper bound on the remaining space in the bin is small enough, namely, $\frac{1}{\pi_i - 1} < \varepsilon = \frac{1}{M}$. The rest of the items in the bin, no matter what they are have an increase of at most $\frac{1}{\pi_{i-1}}$. This means that the total increase is at most $\Pi_\infty - 1 + \frac{1}{\pi_{i-1}} < \Pi_\infty - 1 + \varepsilon$. Contradiction.

Corollary 2. *The performance guarantee of the* FFD *based algorithm $\mathcal{A}$ of [15] for perfect graphs is $\Pi_\infty + 1 \approx 2.69103$.*

Proof. As [15] supplies an example which achieves this bound (asymptotically), we prove the upper bound. Since the input is colored optimally, the number of independent sets is exactly $\mu = \chi(G) \leq$ OPT. We have $\mathcal{A} \leq W(I) +$ OPT. However $W(I) \leq \Pi_\infty \cdot$ OPT and thus $\mathcal{A} \leq (\Pi_\infty + 1) \cdot$ OPT.

Note that Lemma 2 can be generalized as follows. For an integer $z \geq 1$, let $\pi_1(z) = z + 1, \pi_{i+1}(z) = \pi_i(z)(\pi_i(z) - 1) + 1$. Then $\Pi_\infty(z) = \sum_{i=1}^{\infty} \frac{1}{\pi_i(z) - 1}$.

Lemma 3. *Consider a set of items J which consists of the contents of a sub-bin of size $\frac{1}{z}$. Then $W(J) \leq \Pi_\infty(z)$.*

The proof is very similar to the proof of Lemma 2. We have an initial bin of size $\frac{1}{\pi_1-1}$. Since we use the same recursive definition, the property $1-\sum_{j=1}^{i-1}\frac{1}{\pi_j(z)} = \frac{1}{\pi_i(z)-1}$, for $i \geq 1$ holds in this case.

In order to analyze the *algorithm with precoloring*, we need to define a set of weights which does not give very high weights to items in $\mathcal{I}_1 = (\frac{1}{2}, 1]$. We define the weight for $s_x \in \mathcal{I}_1$ to be $weight(x) = s_x + \frac{1}{6}$. This unique definition it possible due to the special treatment of items in $\mathcal{I}_1$.

In order to establish a lemma regarding the sum of weights in an independent set, we modify the type of algorithms we allow to use. Once again, the set I is partitioned into ν independent sets. Each independent set has at most one item of size in $\mathcal{I}_1$. Out of these sets $\mu \leq \nu$ are packed using FFD. Each other independent set J is packed into a single bin, and is assigned a total weight of at least 1.

Lemma 4. *An algorithm $\mathcal{B}$ as above satisfies $\mathcal{B} \leq W(I) + \mu$.*

Proof. In order to complete the proof, we note that independent sets where no item of size in $\mathcal{I}_1$ exists do not have a change in weights and thus the previous proof holds. Consider therefore an independent set with a single item in $\mathcal{I}_1$. This item is the first one to be packed by FFD. In this proof we consider this single bin to be a transition bin as well. Let k be the type of the last transition bin as in the proof of Lemma 1. If $k \geq 6$, we have that the total size y of items in the first transition bin is at least $\frac{5}{6}$ and thus the weight is at least $y + \frac{1}{6} \geq 1$ (since the weight of the large item is its size plus $\frac{1}{6}$, and no weight of an item is smaller than its size). The proof for all other transition bins is the same as in Lemma 1. We are left with five cases $k = 1, 2, 3, 4, 5$. In the first case there is a single transition bin and we are done. In the second case, there are two transition bins. The sum of items in the two bins is more than 1 and so is their weight. If $k = 3$ there are at most three transition bins. The case of two transition bins is covered by the case $k = 2$ thus we may assume that three transition bins exist. As in the proof of Lemma 1, we lower bound the sum of weights in bins that are not the first or last transition bin using $\frac{k+2}{k+1}(1-\alpha)$. Since $\frac{1}{k+1} \leq \alpha \leq \frac{1}{k}$, this value is strictly smaller than 1. The weight of items in the first transition bin is at least $1-\alpha+\frac{1}{6}$, and in the last one $\alpha + \frac{k}{k+1}$. For $k = 3$ this gives a total weight of at least $1-\alpha+\frac{1}{6}+\frac{5}{4}(1-\alpha)+\alpha+\frac{1}{12} = \frac{5}{2}-\frac{5\alpha}{4} \geq \frac{25}{12} > 2$. For $k = 4$ there are three or four transition bins, let t denote this amount. We get a total weight of at least $1-\alpha+\frac{1}{6}+(t-2)\frac{6}{5}(1-\alpha)+\alpha+\frac{1}{20} = -\frac{6}{5}\alpha(t-2)+\frac{6}{5}(t-2)+\frac{73}{60} \geq \frac{9}{10}(t-2)+\frac{73}{60} > t-1$ which holds for $t \leq 4$. Finally, for $k = 5$ we denote again by t the number of transition bins (which is at least three and at most five) and get $1-\alpha+\frac{1}{6}+(t-2)\frac{7}{6}(1-\alpha)+\alpha+\frac{1}{30} = -\frac{7}{6}\alpha(t-2)+\frac{7}{6}(t-2)+\frac{6}{5} \geq \frac{14}{15}(t-2)+\frac{6}{5} \geq t-1$ which holds for $t \leq 5$.

We can now show a tight analysis of the FFD based *algorithm with precoloring* given in [15]. That algorithm finds a coloring of all items with a minimum number of colors, with the restriction that items of size in $\mathcal{I}_1$ receive

distinct colors, and then uses FFD to pack each color class. It was shown in [15] that the performance guarantee of this algorithm is at most 2.5 and at least $\Pi_\infty(3) + 2 \approx 2.4231$.

Theorem 1. *The performance guarantee of the* FFD *based algorithm with precoloring $\mathcal{B}$ of [15] is $\Pi_\infty(3) + 2 \approx 2.4231$.*

Proof. Since in this case $\mu = \chi_I(G) \leq \mathrm{OPT}$, it is left to upper bound the amount of weight that can fit into a single bin, and show that it is at most $\Pi_\infty(3)+1 \approx 1.4231$. Given a packed bin in OPT, we may assume that all items have size at most $\frac{1}{2}$. Otherwise, there is a single item of size $y > \frac{1}{2}$, replace this item with two items of size $\frac{y}{2}$. If $\frac{1}{4} < \frac{y}{2} \leq \frac{1}{3}$, then the weight of each of these two items is $\frac{y}{2} + \frac{1}{12}$, and their total weight equals the weight of the original item. If $\frac{1}{3} < \frac{y}{2} \leq \frac{1}{2}$, then the weight of each of these two items is $\frac{y}{2} + \frac{1}{6}$, and their total weight is even larger than the weight of the original item. Hence, we can assume that all items have size at most $\frac{1}{2}$.

If the bin contains zero or one items of size in $\mathcal{I}_2$, then since for smaller items, the ratio between weight and size is at most $\frac{4}{3}$, we conclude that the total weight is at most $\frac{4}{3}$ in the case of zero items and at most $\frac{25}{18} \approx 1.38889$ in the case of one item. If it contains two such items, then by Lemma 3, the remainder of the bin is of size smaller than $\frac{1}{3}$, and we get a total weight of at most $\Pi_\infty(3)+1 \approx 1.4231$.

3 Improved algorithms

In the previous section we showed better bounds for two variants of the problem, based on previously known algorithms from [15]. Though this already gives an improvement over the previously known bounds, the bounds we have shown are tight bounds, and thus further improvement is possible only using new algorithms, which we now design. To analyze these algorithms we use weighting in a more complex way.

3.1 Perfect conflict graphs

We design an algorithm which uses a preprocessing phase.
Algorithm Matching Preprocessing:

1. Define the following bipartite graph. One set of vertices consists of all items of size in $\mathcal{I}_1$. The other set of vertices consists of all other items. An edge (a, b) between vertices of items of sizes $s_a > \frac{1}{2}$ and $s_b \leq \frac{1}{2}$ occurs if the two following conditions hold.
   (a) $s_a + s_b \leq 1$.
   (b) $(a, b) \notin E(G)$.
   That is, if these two items can be placed in a bin together. If this edge occurs, we give it the cost $c(a, b) = \mathit{weight}(b)$, where $\mathit{weight}(b)$ is defined as above to be $s_b + \frac{1}{j(j+1)}$, for the integer j such that $s_b \in (\frac{1}{j+1}, \frac{1}{j}]$.
2. Find a maximum cost matching in the bipartite graph.

3. Each pair of matched vertices are removed from G and packed into a bin together.
4. Let G' denote the induced subgraph over the items that were not packed in the preprocessing.
5. Compute a feasible coloring of G' using $\chi(G')$ colors.
6. For each color class, apply the First-Fit-Decreasing algorithm.

We next analyze this algorithm.

Theorem 2. *The above algorithm is a* $\frac{5}{2} = 2.5$*-approximation algorithm.*

Proof. The outline of the proof is as follows. We assign weights according to an optimal packing. Afterwards, we take the total weight and re-assign it to items so that the total weight does not grow and the conditions of Corollary 1 hold.

Fix an optimal packing, OPT. For a bin with no items of size in $\mathcal{I}_1$, weights are defined as before. For an item of size in $\mathcal{I}_1$ the weight is always 1. Given a bin with an item of size in $\mathcal{I}_1$ which contains additional items, pick an item of largest size in the bin among the items in the bin with size at most $\frac{1}{2}$, and give it weight zero. All other items in the bin receive weights as before. Note that the items which received zero weight, together with the items of size in $\mathcal{I}_1$ placed together with them in the same bins of OPT form a valid matching in the bipartite graph, whose cost is exactly the total reduction in the weights of items (compared to the weights used for perfect graphs in Section 2). We use the notation $weight_1$ for this reduced weight function, and $weight$ for the regular weight function (as used in the proofs for perfect graphs in Section 2). Let ω be the cost of the matching removed by the algorithm. Then by the optimality of the removed matching, we conclude that $\sum_{x \in I} (weight(x) - weight_1(x)) \leq \omega$. We re-assign weights to items so that an item of size in $(0, \frac{1}{2}]$ that was removed in the matching receives weight zero, and any other item receives a weight as usual (as defined by the function $weight$). This weight function (after the re-assignment) is called $weight_2$. We have $\sum_{x \in I} weight_2(x) + \omega = \sum_{x \in I} weight(x) \leq \omega + \sum_{x \in I} weight_1(x)$. Therefore, the total weight does not grow, and we may analyze the algorithm (but not OPT) using the weights $weight_2$. Clearly, each of the bins removed by the algorithm in the matching has weight of at least 1 since each of these contains an item of unit weight. Therefore, we can use Corollary 1 since the weights of items that are packed using FFD are the same as before.

Finally, we need to analyze the largest amount of weight that can be packed into a single bin of OPT. Using Theorem 1, we can see that if all item sizes are no larger than $\frac{1}{2}$, then this amount is smaller than $\frac{3}{2}$. We can use this as the weights of all the items considered here, are the same as in that proof. Consider now a bin with an item of size in $\mathcal{I}_1$. If this is the only item in the bin, then the total weight is 1. Otherwise let $x_1 \geq \ldots \geq x_t$ be the sorted list of other items in the bin, where x_1 is the item which was assigned a zero weight in $weight_1$. Let $j \geq 2$ be an integer such that $x_1 \in \mathcal{I}_j$. The total weight of the large item and

items $x_1, \ldots, x_t$ is therefore at most $\frac{j+1}{j}(\sum_{i=2}^{t} s_{x_i}) + 1 \leq 1 + \frac{j+1}{j}(1 - \frac{1}{2} - \frac{1}{j+1}) = 1 + \frac{j+1}{2j} - \frac{1}{j} = 1 + \frac{j-1}{2j} < \frac{3}{2}$.

We next show that our analysis of Algorithm Matching Preprocessing is tight.

Proposition 1. *The approximation ratio of Algorithm Matching Preprocessing is at least* 2.5.

Proof. Let M and ℓ be large constants and $\varepsilon = \frac{1}{2M}$. To construct the set of items we do as follows. We use one sequence of ℓ items $a_1, \ldots, a_\ell$ each with size $\frac{1}{2} + \varepsilon$. Furthermore, we have $(M-1)\ell$ additional items $b_{i,j}$, $1 \leq i \leq \ell$, $1 \leq j \leq M-1$, each of size ε. The conflict graph induces a clique with the ℓ items $b_{i,M-1}$, and contains no further edges.

An optimal solution is given by ℓ independent sets $U_i = \{a_i, b_{i,1}, \ldots, b_{i,M-1}\}$ with total size of exactly one for each set. Thus $\mathrm{OPT} = \ell$.

The preprocessing step finds ℓ sets of pairs, which are $\{a_i, b_{i,1}\}$. Next, a coloring with ℓ colors which is found for the remaining items consists of one independent set which contains all items $b_{i,j}$ for $1 \leq i \leq \ell$, $2 \leq j \leq M-2$, and additionally contains $b_{1,M-1}$. Each other independent set contains a single item $b_{i,M-1}$ for $2 \leq i \leq \ell$. The number of bins used to color the first independent set is $\lceil \frac{\ell(M-3)+1}{2M} \rceil$, since this is the total size of items. Each other independent set consumes one additional bin, thus in total we get at least $2\ell - 1 + \frac{\ell}{2} - \frac{3\ell}{2M}$ bins. It can be seen that for $M = \ell^2$ and $\ell >> 1$, the ratio becomes arbitrarily close to $\frac{5}{2}$.

Remark 1. Algorithm Matching Preprocessing is a 2.5-approximation algorithm for BPC on any hereditary class of graphs for which one can find in polynomial time a coloring that uses a minimum number of colors.

3.2 Conflict graphs that belong to $\mathcal{C}$

In this section we study an approximation algorithm for the case where the conflict graph G belongs to $\mathcal{C}$. I.e., given an induced subgraph of G, $G' = (V', E')$ and a set of vertices $L' \subseteq V'$, we can find a coloring of G using a minimum number of colors such that each pair of vertices from L' are assigned distinct colors.

We analyze the following algorithm. The weight function *weight* is defined as in Section 2 for items with size at most $\frac{1}{2}$ and for an item x such that $s_x \in \mathcal{I}_1$, $weight(x) = s_x + \frac{1}{6}$. We can use Lemma 4 since our algorithm will follow its conditions.

Algorithm Greedy Preprocessing:

1. **While** there is a set of three items $\{a, b, c\}$ that can fit into one bin (i.e., $s_a + s_b + s_c \leq 1$ and $\{a, b, c\}$ is an independent set of G) such that $weight(a) + weight(b) + weight(c) > 1$ and $s_c \leq s_b \leq s_a \leq \frac{1}{2}$, or two items $\{a, b\}$ that can fit into one bin (i.e., $s_a + s_b \leq 1$ and $\{a, b\}$ is an independent set of G)

such that $weight(a) + weight(b) > 1$ **do** as follows.
Choose such a set A of maximal total weight. Delete A from G, and assign a new bin for the items of A that is dedicated to this set of items.
Denote by $G' = (V', E')$ the resulting conflict graph induced by the remaining items.

2. Denote the set of large items by $L = \{j \in V' : s_j > \frac{1}{2}\}$, and denote by $\chi_I(G')$ the minimum number of colors used by the optimal solution for the precoloring extension problem defined by G' and the set of precolored vertices L. Compute a feasible coloring of G' using $\chi_I(G')$ colors, where any two items in L are assigned different colors.
3. For each color class, apply the First-Fit-Decreasing algorithm.

Theorem 3. *The approximation ratio of the above algorithm is exactly* $\frac{7}{3} \approx 2.33333$.

Proof. Fix an optimal solution OPT. Let $weight$ be the weight function as used in the algorithm. We assign weights according to OPT, and denote this weight function by $weight_1$. For an optimal bin which contains no items of size in $\mathcal{I}_1$, and contains no triple of items of total weight strictly larger than 1 with respect to $weight$, we use $weight_1 = weight$ to define weights of items for all items in the bin. For a bin which contains an item x of size in $\mathcal{I}_1$, but contains no other item y such that $weight(x) + weight(y) > 1$, we again use $weight_1 = weight$ for every item in the bin.

For a bin which contains no items of size in $\mathcal{I}_1$, but contains a triple of items of total weight strictly larger than 1 with respect to $weight$, let a_1, a_2, a_3 be three items with largest weights in the bin ordered according to their weight. Note that $s_{a_1} \in \mathcal{I}_2 \cup \mathcal{I}_3$, since otherwise the sum of weights of the three items cannot exceed 1. We define the reduction value for this bin to be

$$\Delta = \frac{weight(a_1) + weight(a_2) + weight(a_3) - 1}{3} .$$

For any item b in this bin such that $b \neq a_i$ for $i = 1, 2, 3$, we define $weight_1(b) = weight(b)$. Note that the algorithm in the preprocessing step removes at least one of a_1, a_2 and a_3 since otherwise if all three items are not removed, then the preprocessing step cannot terminate. Let i' be the index of the item of an item a_i such that $1 \leq i' \leq 3$, and $a_{i'}$ is removed no later than a_j for all $1 \leq j \leq 3$. We define $weight_1(a_{i'}) = weight(a_{i'}) - \Delta$, and for $i \neq i'$, $weight_1(a_i) = weight(a_i)$.

For a bin which contains an item x of size in $\mathcal{I}_1$ and contains another item y such that $weight(x) + weight(y) > 1$, let y be such an item with maximum weight according to $weight$. We define the reduction value for this bin to be $\Delta = \frac{weight(x)+weight(y)-1}{3}$. For any item b in the bin for $b \neq x, y$, we define $weight_1(b) = weight(b)$. Note that at least one of x and y is removed in the preprocessing step. If y is removed no later than x, we define $weight_1(y) = weight(y) - \Delta$ and $weight_1(x) = weight(x)$, and otherwise $weight_1(y) = weight(y)$ and $weight_1(x) = weight(x) - \Delta$.

Consider a bin which is removed in the greedy preprocessing step. We next argue that the total weight of the items in this bin according to $weight_1$ is greater

than 1. First note that the total weight of the items according to $weight$ is at least one. Therefore, if for every item a in this bin, $weight_1(a) = weight(a)$, we get that the sum of weights in this bin is strictly larger than 1. We will show that a possible reduction in the weights does not decrease the sum of weights below 1. Let A be the set of items in this bin and $\Gamma = \frac{\left(\sum_{a \in A} weight(a)\right) - 1}{3}$. For an item $a \in A$, we have $weight_1(a) < w(a)$ if the following conditions hold. Consider the bin to which a belongs in OPT. Then a value $\Delta(a)$ was computed for this bin such that $weight_1(a) = weight(a) - \Delta(a)$. We get that a is removed in the preprocessing, and at the time of removal of a, it belongs to a set of items $\mathbf{A}$ of largest weight that is valid for removal in the preprocessing step. Moreover, no item of $\mathbf{A}$ has been already removed at the time that a is being removed. This means that in the greedy process, we have $\Gamma \geq \Delta(a)$. Thus we have $\sum_{a \in A} weight_1(a) = \sum_{a \in A} [weight(a) - \Delta(a)] \geq \sum_{a \in A} weight(a) - 3\Gamma = 1$. Therefore, each of the bins that were removed by the algorithm in the greedy preprocessing step, has weight of at least 1. Therefore we can use Lemma 4 since the weights of items that are packed using FFD are the same as in Section 2.

Finally, we need to analyze the largest amount of weight that can be packed into a single bin of OPT. This analysis is done with respect to w_1. Consider the set of items A in a given bin of OPT.

If all items in A have size at most $\frac{1}{3}$, then for all $a \in A$, $weight_1(a) \leq \frac{4}{3} \cdot s_a$, and thus the total weight of the items in A is at most $\frac{4}{3}$. This covers both the case where there is no reduction in the weight of items in $weight_1$ compared to $weight$ and the case where there is such a reduction for some items.

Next, assume that A has an item x of size in $\mathcal{I}_2$, but all weights in this bin were assigned according to $weight$ (i.e., for all $a \in A$ $weight_1(a) = weight(a)$). This can happen in two cases. If A contains an additional item y of size in $\mathcal{I}_2$, then $A = \{x, y\}$. This is so as a third item in the bin would imply a triple whose total weight is strictly more than 1 and hence we will have $weight_1(x) \neq weight(x)$. Therefore, in this case where $A = \{x, y\}$ we get a total weight of $s_x + \frac{1}{6} + s_y + \frac{1}{6} \leq 1 + \frac{1}{3} \leq \frac{4}{3}$. Otherwise, for all $y \in A \setminus \{x\}$, we conclude that $s_y \in (0, \frac{1}{3}]$. If all $y \in A \setminus \{x\}$ actually have size in $(0, \frac{1}{4}]$, then $weight_1(y) \leq \frac{5}{4} \cdot s_y$, and the total size of all items in $A \setminus \{x\}$ is at most $1 - s_x$. Together this gives a total weight of at most $s_x + \frac{1}{6} + \frac{5}{4} \cdot (1 - s_x) = \frac{17}{12} - \frac{s_x}{4}$. This value is maximized when s_x is minimized, and therefore the total weight of the items in A is at most $\frac{16}{12} = \frac{4}{3}$. Finally, if there is $y \in A$ such that $s_y \in \mathcal{I}_3$, then we conclude that all items of $A \setminus \{x, y\}$ have size in $(0, \frac{1}{6}]$, and thus their weights are at most $\frac{7}{6}$ times their sizes. A third item of size in $(\frac{1}{6}, \frac{1}{4}]$ in the bin would imply a triple whose total weight is at least $\frac{1}{2} + \frac{1}{3} + \frac{1}{5} > 1$. This triple will force $weight_1(x) < weight(x)$ contradicting our assumption. Therefore, in this case the total weight of the items in A is at most $s_x + \frac{1}{6} + s_y + \frac{1}{12} + (1 - s_x - s_y) \cdot \frac{7}{6} = \frac{17}{12} - \frac{s_x + s_y}{6} \leq \frac{17}{12} - \frac{7}{72} = \frac{95}{72} < \frac{4}{3}$.

Suppose that A has an item y of size in $\mathcal{I}_1$. If $A = \{y\}$, then the total weight is at most $\frac{7}{6}$. Otherwise, let $A = \{y, x_1, x_2, \ldots, x_t\}$ where $s_{x_1} \geq \ldots \geq s_{x_t}$ is the sorted list of other items in the bin. Then, x_1 is an item of largest weight according to $weight$ among $A \setminus \{y\}$. Let $j \geq 2$ be an integer such that $x_1 \in \mathcal{I}_j$. Let

$\Delta = \frac{weight(y)+weight(x_1)-1}{3} = \frac{s_y+\frac{1}{6}+s_{x_1}+\frac{1}{j(j+1)}-1}{3}$ be the reduction value of this bin. We have $weight(y) + weight(x_1) = 3\Delta + 1$. The total weight (according to $weight_1$) of the items $y, x_1, \ldots, x_t$ is therefore at most $\frac{j+1}{j}(\sum_{i=2}^{t} s_{x_i}) + weight(y) + weight(x_1) - \Delta \leq \frac{j+1}{j}(1 - s_y - s_{x_1}) + 1 + \frac{2}{3}(s_y + \frac{1}{6} + s_{x_1} + \frac{1}{j(j+1)} - 1) = -\frac{(j+3)(s_y+s_x)}{3j} + \frac{j+1}{j} + \frac{4}{9} + \frac{2}{3j(j+1)}$. We use $s_y \geq \frac{1}{2}$ and $s_{x_1} \geq \frac{1}{j+1}$, and get total weight of at most $-\frac{3(j+3)^2}{18j(j+1)} + \frac{18(j+1)^2+8j(j+1)+12}{18j(j+1)} = \frac{23j^2+26j+3}{18j(j+1)} = \frac{23}{18} + \frac{1}{6j}$. If $j \geq 3$ we are done. However, if $j = 2$, then all items but y and x_1 are of size strictly smaller than $\frac{1}{6}$, and thus their weights are at most $\frac{7}{6}$ times their sizes. We get a weight of at most $\frac{7}{6}(1 - s_y - s_{x_1}) + 1 + \frac{2}{3}(s_y + \frac{1}{6} + s_{x_1} + \frac{1}{6} - 1) = -\frac{s_{x_1}+s_y}{2} + \frac{31}{18} \leq -\frac{5}{12} + \frac{31}{18} = \frac{47}{36} < \frac{4}{3}$.

It is left to consider the case where all items in A are no larger than $\frac{1}{2}$, but $weight_1$ is not equal to $weight$ for all items. Such a bin must contain at least one item of size in $\mathcal{I}_2$ (otherwise, this case is already covered). There are two types of such bins. One option is that A has two items of size in $\mathcal{I}_2$. The other option is that A has a single item of size in $\mathcal{I}_2$. In the second option A must contain at least one item of size in $\mathcal{I}_3$, otherwise this case is already covered by the proof where $weight = weight_1$ for all items (since in our case, the weight according to $weight_1$ can only be smaller, and for that proof we made no assumption on whether any reductions of weight were applied to this bin).

Consider the first option. Let $y_1, y_2 \in A$ be the items of size in the interval $\mathcal{I}_2$ and assume that $A = \{y_1, y_2, x_1, \ldots, x_t\}$ where $s_{x_1} \geq \cdots \geq s_{x_t}$, so x_1 is an item of largest weight according to $weight$ in $A \setminus \{y_1, y_2\}$. Let $j \geq 3$ be an integer such that $s_{x_1} \in \mathcal{I}_j$. Let $\Delta = \frac{weight(y_1)+weight(y_2)+weight(x_1)-1}{3} = \frac{s_{y_1}+\frac{1}{6}+s_{y_1}+\frac{1}{6}+s_{x_1}+\frac{1}{j(j+1)}-1}{3}$ be the reduction value of this bin. We have $weight(y_1) + weight(y_2) + weight(x_1) = 3\Delta + 1$. The total weight (according to $weight_1$) of the items in A is therefore at most $\frac{j+1}{j}(\sum_{i=2}^{t} s_{x_i}) + weight(y_1) + weight(y_2) + weight(x_1) - \Delta \leq \frac{j+1}{j}(1 - s_{y_1} - s_{y_2} - s_{x_1}) + 1 + \frac{2}{3}(s_{y_1} + \frac{1}{6} + s_{y_2} + \frac{1}{6} + s_{x_1} + \frac{1}{j(j+1)} - 1) = -\frac{(j+3)(s_{y_1}+s_{y_2}+s_x)}{3j} + \frac{j+1}{j} + \frac{5}{9} + \frac{2}{3j(j+1)}$. We use $s_{y_1}, s_{y_2} \geq \frac{1}{3}$ and $s_{x_1} \geq \frac{1}{j+1}$, and get a total weight of at most $-\frac{(j+3)(2j+5)}{9j(j+1)} + \frac{9(j+1)^2+5j(j+1)+6}{9j(j+1)} = -\frac{2j^2+11j+15}{9j(j+1)} + \frac{9j^2+18j+9+5j^2+5j+6}{9j(j+1)} = \frac{4}{3}$.

Consider the second option. Let y_1, y_2 be the items of sizes in $\mathcal{I}_2$,$\mathcal{I}_3$ (respectively), and assume that $A = \{y_1, y_2, x_1, \ldots, x_t\}$ where $s_{x_1} \geq \cdots \geq s_{x_t}$, so x_1 is an item of largest weight according to $weight$ in $A \setminus \{y_1, y_2\}$. Let $j \geq 3$ be an integer such that $s_{x_1} \in \mathcal{I}_j$. Let $\Delta = \frac{weight(y_1)+weight(y_2)+weight(x_1)-1}{3} = \frac{s_{y_1}+\frac{1}{6}+s_{y_2}+\frac{1}{12}+s_{x_1}+\frac{1}{j(j+1)}-1}{3}$ be the reduction value of this bin. We have $weight(y_1) + weight(y_2) + weight(x_1) = 3\Delta + 1$. The total weight (according to $weight_1$) of the items in A is therefore at most $\frac{j+1}{j}(\sum_{i=2}^{t} s_{x_i}) + weight(y_1) + weight(y_2) + weight(x_1) - \Delta \leq \frac{j+1}{j}(1 - s_{y_1} - s_{y_2} - s_{x_1}) + 1 + \frac{2}{3}(s_{y_1} + \frac{1}{6} + s_{y_2} + \frac{1}{12} + s_{x_1} + \frac{1}{j(j+1)} -$

$1) = -\frac{(j+3)(s_{y_1}+s_{y_2}+s_x)}{3j} + \frac{j+1}{j} + \frac{1}{2} + \frac{2}{3j(j+1)}$. We use $s_{y_1} \geq \frac{1}{3}$, $s_{y_2} \geq \frac{1}{4}$ and $s_{x_1} \geq \frac{1}{j+1}$, and get a total weight of at most $-\frac{(j+3)(7j+19)}{36j(j+1)} + \frac{36(j+1)^2+18j(j+1)+24}{36j(j+1)} = -\frac{7j^2+40j+57}{36j(j+1)} + \frac{36j^2+72j+36+18j^2+18j+24}{36j(j+1)} = \frac{47}{36} + \frac{1}{12j} \leq \frac{4}{3}$.

We next show that our analysis of Algorithm Greedy Preprocessing is tight. Let M and ℓ be large constants and $\varepsilon = \frac{1}{3M}$. To construct the set of items we do as follows. We use two sequences of ℓ items each, $a_1, \ldots, a_\ell, b_1, \ldots, b_\ell$, where each of these items has size $\frac{1}{3} + \varepsilon$. Furthermore, we have $(M-2)\ell$ additional items $c_{i,j}$, $1 \leq i \leq \ell$, $1 \leq j \leq M-2$, each of size ε. The conflict graph induces a clique on the ℓ items $c_{i,M-2}$, and contains no further edges.

An optimal solution is given by ℓ independent sets $U_i = \{a_i, b_i, c_{i,1}, \ldots, c_{i,M-2}\}$ with total size of exactly one for each set. Thus $\mathrm{OPT} = \ell$.

The preprocessing step finds ℓ sets of triples, which are $\{a_i, b_i, c_{i,1}\}$. Next, a coloring with ℓ colors which is found for the remaining items consists of one independent set which contains all items $c_{i,j}$ for $1 \leq i \leq \ell$, $2 \leq j \leq M-3$, and additionally contains $c_{1,M-2}$. Each other independent set contains a single item $c_{i,M-2}$ for $2 \leq i \leq \ell$. The number of bins used to color the first independent set is $\lceil \frac{\ell(M-4)+1}{3M} \rceil$, since $\frac{\ell(M-4)+1}{3M}$ is the total size of these items. Each other independent set consumes one additional bin, thus in total we get at least $2\ell - 1 + \frac{\ell}{3} - \frac{4\ell}{3M}$ bins. It can be seen that for $M = \ell^2$ and $\ell >> 1$, the ratio becomes arbitrarily close to $\frac{7}{3}$.

Note that this example shows that even if the triples are removed optimally, the performance cannot be improved. Note also that we apply a greedy removal step on triples. Another way to handle the preprocessing is to use a clever removal method such as local search. Such a method for weighted elements can be found in [1]. It allows the removal of almost half the weight which can be removed by an optimal algorithm (for removal of triples), rather than one third of the weight as we achieve using a naïve greedy algorithm.

3.3 Bipartite graphs

In [15] it was shown that a simple algorithm which finds some coloring of the graph with two colors, and packs each color class using Next-Fit, is a 2-approximation. It was shown there that even if Next-Fit is replaced by FFD, still this algorithm does not have a better approximation ratio.

We design an algorithm which gives special treatment to some of the problematic cases and thus get a $\frac{7}{4}$-approximation.

We start with an analysis of the algorithm above (with FFD), which we call TWO-SET (TS), as a function of the value OPT. Let A and B denote the sets of items of the two colors. Let $\ell(A)$ and $\ell(B)$ denote the numbers of bins packed by FFD for each of the two sets, let $s(X)$ denote the sum of item sizes in a set X, i.e., $s(X) = \sum_{x \in X} s_x$, and let $\mathrm{OPT}(X)$ denote the cost of an optimal solution for a set X. Clearly, we have $s(X) \leq \mathrm{OPT}(X) \leq \mathrm{OPT}$ for $X = A, B$, and also $\mathrm{OPT} \geq s(A) + s(B)$.

Simchi-Levi [27] proved that for any input Y, the solution of FFD on this output satisfies $\mathrm{FFD}(Y) \leq \frac{3}{2}\mathrm{OPT}(Y)$. Therefore, if the size of one of the sets

(without loss of generality, the set A) is small enough, namely, this set fits into one bin $s(A) \leq 1$, we get $TS \leq \mathrm{FFD}(B) + 1 \leq \frac{3}{2}\mathrm{OPT} + 1$.

Otherwise, if for both sets, the output of FFD created at least one bin where the smallest item that opens a new bin is in the interval $(0, \frac{1}{3}]$. Then, for each set A and B, all bins but the last one are occupied by more than $\frac{2}{3}$, and the sum of items in the two last bins together is more than 1. We get for $X = A, B$, $s(X) > \frac{2}{3}(\ell(X) - 2) + 1$. Thus $TS \leq \ell(A) + \ell(B) < \frac{3}{2}\mathrm{OPT} + 1$.

Suppose next that both sets A and B do not have a bin opened by an item with size in the interval $(0, \frac{1}{3}]$. Then, we remove all items smaller than $\frac{1}{3}$ from the input. Clearly, the output does not change. Each bin contains an item of size in $(\frac{1}{2}, 1]$ (and possibly one smaller item as well) or two items in the interval $(\frac{1}{3}, \frac{1}{2}]$, except possibly the last bin for each set, that may contain a single item of this last interval. Let Z denote the number of items of size in $(\frac{1}{2}, 1]$ in $A \cup B$ and let V denote the number of items from $A \cup B$ with size in the interval $(\frac{1}{3}, \frac{1}{2}]$. Therefore, $TS \leq Z + \frac{V-2}{2} + 2 = Z + \frac{V}{2} + 1$. However, for any packing and thus for an optimal one we have that each bin contains at most one item with size larger than $\frac{1}{2}$, and at most two items with size larger than $\frac{1}{3}$, thus we have $\mathrm{OPT} \geq Z$ and $\mathrm{OPT} \geq \frac{Z+V}{2}$. We get $TS \leq \frac{Z+V}{2} + \frac{Z}{2} + 1 \leq \frac{3}{2}\mathrm{OPT} + 1$.

We are left with the case where (without loss of generality) the set A contains a bin opened by an item in $(0, \frac{1}{3}]$, and B does not. If A does not contain a bin opened by an item of size in $(0, \frac{1}{4}]$, we can remove all items smaller than $\frac{1}{4}$ from the input, and get the same output. Let Z denote again the number of items in $(\frac{1}{2}, 1]$ and V denote the number of items in $(\frac{1}{4}, \frac{1}{2}]$. We now argue that $V \leq 3(\mathrm{OPT} - Z) + Z = 3\mathrm{OPT} - 2Z$. this last inequality holds, since a bin with an item larger than $\frac{1}{2}$, can contain at most one item larger than $\frac{1}{4}$, and any other bin can contain at most three such items. Therefore, $TS \leq Z + \frac{V-2}{2} + 2 \leq Z + \frac{3}{2}\mathrm{OPT} - Z + 1 \leq \frac{3}{2}\mathrm{OPT} + 1$.

Finally, we need to consider the case that A contains at least one bin opened by an item of size in $(0, \frac{1}{4}]$, and B does not have a bin opened by an item whose size is at most $\frac{1}{3}$. Thus all bins of A but the last one are occupied by more than $\frac{3}{4}$. We get $s(A) > \frac{3}{4}(\ell(A) - 2) + 1$ and $s(B) > \frac{1}{2}\ell(B)$. The last inequality holds for any Any-Fit type algorithm, and for FFD in particular. Moreover, note that the packing of B is an optimal one. This can be proved using simple exchange arguments (see [27]). Thus we have $\ell(B) \leq \mathrm{OPT}$. We get $\mathrm{OPT} \geq s(A) + s(B) > \frac{3}{4}\ell(A) + \frac{1}{2}\ell(B) - \frac{1}{2}$. Thus $SL < \frac{4}{3}\mathrm{OPT} + \frac{2}{3} + \frac{1}{3}\mathrm{OPT} = \frac{5}{3}\mathrm{OPT} + \frac{2}{3}$. Since both OPT and SL are integers, we get $SL \leq \frac{5}{3}\mathrm{OPT} + \frac{1}{3}$.

We can prove the following lemma.

Lemma 5. *If* $\mathrm{OPT} \geq 3$ *then the algorithm above satisfies* $SL \leq \frac{7}{4}\mathrm{OPT}$ *and this bound is tight when* $\mathrm{OPT} = 4$.

Proof. We obtained two bounds and since SL is integer we conclude that $SL \leq \max\{\lfloor \frac{3}{2}\mathrm{OPT} + 1 \rfloor, \lfloor \frac{5}{3}\mathrm{OPT} + \frac{1}{3} \rfloor\}$. If $\mathrm{OPT} \geq 4$ we get $\frac{3}{2}\mathrm{OPT} + 1 \leq \frac{7}{4}\mathrm{OPT}$ and $\lfloor \frac{5}{3}\mathrm{OPT} + \frac{1}{3} \rfloor \leq \frac{7}{4}\mathrm{OPT}$. For $\mathrm{OPT} = 3$ we get $SL \leq 5 \leq \frac{5}{3}\mathrm{OPT}$.

To see that this bound is tight consider the following example. Let $\varepsilon > 0$ be a small number and define the following set of item sizes $A = \{\frac{1}{4} + \varepsilon, \frac{1}{4} + \varepsilon, \frac{1}{4} +$

$\varepsilon, \frac{1}{4}+\varepsilon, \frac{1}{4}-2\varepsilon, \frac{1}{4}-2\varepsilon, \frac{1}{4}-2\varepsilon, \frac{1}{4}-2\varepsilon\}$, and $B = \{\frac{1}{2}+\varepsilon, \frac{1}{2}+\varepsilon, \frac{1}{2}+\varepsilon, \frac{1}{2}+\varepsilon\}$. Assume that the conflict graph has a single edge between one item of size $\frac{1}{2}+\varepsilon$ and one of the items of size $\frac{1}{4}+\varepsilon$. Then an optimal solution has four bins each of which has one item of size $\frac{1}{2}+\varepsilon$, one item of size $\frac{1}{4}+\varepsilon$, and one item of size $\frac{1}{4}-2\varepsilon$. Clearly the two items which have a conflict do not share a bin. However, assume that the coloring into two colors partitions the items into A and B. Then, $\mathrm{FFD}(A) = 3$ and $\mathrm{FFD}(B) = 4$. Therefore, for this example $\mathrm{OPT} = 4$ and the algorithm returns a solution that uses seven bins.

As we can see, the only case which is left is $\mathrm{OPT} = 2$ which requires a special treatment. This case can be identified by a solution of cost 4. Clearly, such solutions can be achieved also for $\mathrm{OPT} = 3$ and $\mathrm{OPT} = 4$. We define an algorithm and prove that it succeeds if $\mathrm{OPT} = 2$. Thus, if it fails, then $\mathrm{OPT} \geq 3$ which means that the original solution already does not violate the approximation ratio $\frac{7}{4}$ which we would like to prove. We call this algorithm MODIFIED TWO-SET.

If $\mathrm{OPT} = 2$, this means that it is possible to color the input using two colors, and pack each independent set into a single bin. If the conflict graph is connected, there is a unique way to color the items, and thus this optimal packing can be achieved. However, a bipartite disconnected graph has more than one possible coloring with two colors, since the roles of the two colors in each connected component can be swapped. As a first step, we color each connected component using two colors. Let z be the number of components, and denote the items of component i by V_i. For each $1 \leq i \leq z$, we get two sets A_i and B_i, such that $A_i \cup B_i = V_i$ and $A_i \cap B_i = \emptyset$. Each set contains the vertices of V_i that share a color. We define $p_i = |s(A_i) - s(B_i)|$. Let $q_i = \max\{s(A_i), s(B_i)\} - p_i$. The sizes p_i define a scheduling problem on two machines. We run LPT (Longest Processing Time First) on this input. This means that we initialize two empty sets, A and B. Sort the sizes p_i in non-increasing order. Then starting from the largest size, we assign each size to the set whose total sum is minimal. Graham [10] defined and analyzed this algorithm for an arbitrary number of machines (subsets). It is not difficult to see that when the algorithm terminates, we have $|s(A) - s(B)| \leq p_k$, where k is the last index of size assigned to the set with larger sum. For $1 \leq i \leq z$, we define a coloring using two colors (which are defined by the sets C and D) as follows. If $s(A_i) \geq s(B_i)$, and p_i is in A or if $s(A_i) < s(B_i)$, and p_i is in B, assign the items in A_i to C and the items in B_i to D. Otherwise assign the items in B_i to C and the items in A_i to D. This assignment means that $s(C) = \sum_{i \in A} p_i + \sum_{i=1}^{z} q_i$ and $s(D) = \sum_{i \in B} p_i + \sum_{i=1}^{z} q_i$. Thus we have $|s(C) - s(D)| \leq p_k$ as well. Assume (without loss of generality) that $s(C) \geq s(D)$. Since $\mathrm{OPT} = 2$, $S(C) + S(D) \leq 2$. Thus $s(D) \leq 1$ and all the items assigned to D fit into a single bin. Now remove $p_k + q_k$ from $s(C)$. We get a total of less than $s(D) \leq 1$, and thus the remaining items of C fit into one bin. Finally the items of the larger set among $s(A_k)$ and $s(B_k)$ must be packed together in a solution with two bins only, and since $\mathrm{OPT} = 2$, we get that these items also fit into one bin.

We proved the following theorem.

Theorem 4. *Algorithm* MODIFIED TWO-SET *has an approximation ratio of exactly* $\frac{7}{4}$.

Proof. We showed that if OPT = 2, the process above succeeds to pack the input into two bins. Otherwise, the theorem follows from Lemma 5.

4 Online algorithms

In this section we discuss online algorithms for interval graphs. For many classes of graphs, the online problem is hard to approximate. The coloring problem is a special case of BPC, where all item sizes are zero.

Consider e.g. the problem on trees. Gyárfás and Lehel [11] proved a deterministic lower bound of $\Omega(\log n)$ on the online coloring of bipartite graphs on n vertices, which holds already for trees. Lovász, Saks and Trotter [20] showed an online coloring algorithm which colors such a graph (which is 2 colorable) using $O(\log n)$ colors. This immediately implies an online coloring algorithm for BPC on bipartite graphs, which is optimal up to a constant multiplicative factor on the competitive ratio. This algorithm $\mathcal{A}$ uses the algorithm of [20] to color the conflict graph using C colors. Then each color class is colored by some reasonable algorithm, e.g. Next-Fit. We get that for each color class i, which contains ℓ_i bins, the total size of items S_i is more than $\frac{\ell_i - 1}{2}$ (since no two consecutive bins can be combined). We get that $\mathcal{A} \leq \sum_{i=1}^{C} \ell_i < \sum_{i=1}^{C} (2S_i + 1) \leq 2\mathrm{OPT} + C \leq O(\log n)\mathrm{OPT}$.

Since the same can be applied for any graph class for which no constant competitive algorithm exists, we focus on a graph class for which such an algorithm exists, namely, interval graphs. Kierstead and Trotter [18] constructed an online coloring algorithm for interval graphs which uses at most $3\omega - 2$ colors where ω is the maximum clique size of the graph. They also presented a matching lower bound of $3\omega - 2$ on the number of colors in a coloring of an arbitrary online coloring algorithm. Note that the chromatic number of interval graphs equals to the size of a maximum clique, which is equivalent in the case of interval graphs to the largest number of intervals that intersect any point (see [16, 9]). The technique above implies a 5-competitive algorithm. We can show that using First-Fit (FF) instead of Next-Fit for coloring each class slightly improves this bound.

Theorem 5. *The algorithm of [18] combined with* FF *for coloring each class has competitive ratio* 4.7.

Proof. The proof is similar to the proof of [15] for perfect graphs. We can use the well known weight function $\hat{w}$ defined for FF already in [7]. It was shown that for a set of items J on which FF is applied, we have $\ell(J) \leq W(J) + 1$, where $W(J)$ denotes the total weight of items in J, and $\ell(J)$ is the number of bins in the packing of J by FF. On the other hand if we remove the incompatibility constraint (i.e., assume that the conflict graph has no edges), and let OPT' denote an optimal solution for this instance, we have $W(I) \leq 1.7\mathrm{OPT}' \leq 1.7\mathrm{OPT}$. Let

I_j denote the items colored by the algorithm of [18] by color j. This algorithm colors the items with at most $3\chi(G) - 2 \leq 3\text{OPT} - 2$ colors, and thus we get $\mathcal{A} \leq \sum_{j=1}^{C} \ell(I_j) < \sum_{i=1}^{C}(W(I_j)+1) \leq W(I) + C \leq 4.7\text{OPT}$.

Based on the examples of [17, 15] and [18], we can show a sequence of instances whose competitive ratio is arbitrarily close to 4.7. We fix a value of $\text{OPT} = k+2$ and a value of $\varepsilon >> \delta > 0$ that is small enough, and we denote by $\ell = \frac{k}{10}$. The construction is composed of two phases. In the first phase we repeat the construction of [18] with a maximum clique size $k+1$ where all sizes defined for the vertices which the intervals represent all have zero size. We repeat the construction to have $3k$ copies of this construction that use the same set of $3k+1$ colors. Since the intervals are colored by the algorithm of [18], all structures are all colored in the same way. We call these $3k+1$ colors $1, 2, \ldots, 3k+1$.

At the end of the first phase, for each sub-interval of the real line that contains a copy of the construction of [18] that uses the color set $\{1, 2, \ldots, 3k+1\}$ we shrink this sub-interval and replace it with a *point of interest*. We do it only to simplify the construction, and in order to implement this we replace each interval $[a, b]$ that contains such point of interest with an interval that is adjacent to all the vertices in the corresponding copy of the first phase construction.

We next present $3k$ disjoint intervals one by one (i.e., this set of intervals are independent set of G). Each of these intervals contains exactly one point of interest from the first phase, and therefore these intervals cannot be colored using colors $1, 2, \ldots, 3k+1$. The first k intervals are denoted by $a_{i,p}$ for $i = 1, \ldots, 10$ and $p = 1, \ldots, \ell$ have sizes according to the following. $a_{i,p}$ has size $\frac{1}{6} + \frac{\varepsilon}{3^p} - \delta$ for $1 \leq i \leq 5$ and otherwise it has size $\frac{1}{6} - \frac{\varepsilon}{3^{p+1}} - \delta$. These k intervals are introduced according to the following order:

$$a_{1,1}, a_{2,1}, a_{3,1}, a_{6,1}, a_{7,1}, a_{4,1}, a_{5,1}, a_{8,1}, a_{9,1}, a_{10,1}, a_{1,2}, \ldots \ .$$

The next k intervals are denoted by $b_{i,p}$ for $i = 1, \ldots, 10$ and $p = 1, \ldots, \ell$. Their sizes are defined according to the following. The size of $b_{i,p}$ for $1 \leq i \leq 5$ is $\frac{1}{3} + \frac{\varepsilon}{3^{p-1}} - \delta$ and otherwise the size of $b_{i,p}$ is $\frac{1}{3} - \frac{\varepsilon}{3^p} - \delta$. These k intervals are introduced in the following order $b_{1,1}, b_{6,1}, b_{2,1}, b_{7,1}, b_{3,1}, b_{8,1}, b_{4,1}, b_{9,1}, b_{5,1}, b_{10,1}, b_{1,2}, \ldots$. The last k intervals are denoted by c_i for $i = 1, 2, \ldots, k$ and each of these has size of $\frac{1}{2} + \delta$.

Note that the coloring algorithm of [18] will color all the intervals of the second phase using color $3k+2$ and therefore we need to compute the number of bins that the First-Fit algorithm uses in order to pack all these items. Applying First-Fit will open ℓ bins of the following type $\{a_{1,p}, a_{2,p}, a_{3,p}, a_{6,p}, a_{7,p}\}$ for $p = 1, 2, \ldots, \ell$. It will use another ℓ bins of the following type $\{a_{4,p}, a_{5,p}, a_{8,p}, a_{9,p}, a_{10,p}\}$. There will be another 5ℓ bins each containing $\{b_{i,p}, b_{i+5,p}\}$ for $i = 1, 2, \ldots, 5$ and $p = 1, 2, \ldots, \ell$. Last, the algorithm will pack each of the items c_i separately using another 10ℓ bins. The total number of bins used to pack the second phase intervals is $17\ell = 1.7k$. By adding the $3k+1$ colors that are used to color the first phase intervals, we get a solution whose cost is $4.7k + 1$.

It remains to show that the optimal solution costs $k+2$. We first show a packing of the second phase intervals using exactly $k+2$ bins. We use 5ℓ bins each containing $\{a_{i,p}, b_{5+i,p}, c_{5(p-1)+i}\}$ for $i = 1, \ldots, 5$ and $p = 1, \ldots, \ell$. We use another $5\ell - 10$ bins each containing $\{a_{5+i,p-2}, b_{i,p}, c_{5(p+\ell-3)+i}\}$. We use another five bins containing $\{c_{10(\ell-1)+i}, b_{i,1}\}$ for $i = 1, \ldots, 5$, and another five bins containing $\{c_{10(\ell-1)+5+i}, b_{i,2}\}$ for $i = 1, \ldots, 5$. We have another two bins where the first one consists of $\{a_{6,\ell-1}, a_{7,\ell-1}, a_{8,\ell-1}, a_{9,\ell-1}, a_{10,\ell-1}\}$ and the second bin consists of $\{a_{6,\ell}, a_{7,\ell}, a_{8,\ell}, a_{9,\ell}, a_{10,\ell}\}$. Then for each construction of the first phase we have a list of $k+2$ colors such that one of them is used in the coloring of the second phase interval that is adjacent to it. Therefore, we have a set of $k+1$ colors that can be used to color the intervals of the first phase construction. However, this can be done as the maximum clique size in each such construction is $k+1$. Therefore, we are able to color the entire set of intervals using $k+2$ colors each of them containing items of total size at most one. Hence, $\mathrm{OPT} = k+2$.

We can show that an algorithm of much smaller competitive ratio does not exist.

Theorem 6. *The competitive ratio of any online algorithm for* BPC *on interval graphs has competitive ratio of at least* $\frac{155}{36} \approx 4.30556$.

In order to prove this theorem, we prove the following two lemmas.

Lemma 6. *Let c be a lower bound on the asymptotic competitive ratio of any online algorithm for standard bin packing, which knows the value* OPT *in advance. Then the competitive ratio for any online algorithm for* BPC *on interval graphs has competitive ratio of at least* $3 + c$.

Proof. We use a construction similar to the lower bound given in [18]. In this construction we assume that we know the optimal value $\mathrm{OPT} = k$, and thus we are allowed to use a set of at most $(3+c)k$ colors. The construction is composed of two phases. In the first phase we introduce intervals such that the corresponding vertices have size zero (we call them zero sized intervals, where the size of an interval is unrelated to its length). The maximum cardinality clique among these intervals is $k-1$ and the algorithm is forced to use at least $3k-5$ colors, denoted by $1, 2, \ldots, 3k-5$. We next describe this phase in detail.

During the first phase, we shrink some parts of the line into single points. Given a point p, that is a result of shrinking an interval $[a, b]$. Every interval presented in the past which is contained in $[a, b]$ is also shrunk into p and therefore such a point inherits a list of colors that such intervals received. These colors cannot be assigned to any interval that contains the point p. The shrinking is done only for simplification purposes. In practice it means that for a given point p that is the result of shrinking, every future interval either contains this point or not, *i.e.*, it either contains all intervals that were shrunk into this point, or has no overlap with any of them.

If an algorithm uses more than $U = (3+c)k$ colors, we can stop the construction. Therefore, we assume that the algorithm is initially given a palette

of U colors. As soon as all these colors are used, the proof is complete. This is just one stopping condition, we may stop the sequence earlier as well, after the algorithm used $3k-5$ colors from the set of colors $\{1,2,\ldots,(3+c)k\}$. In this case a second phase is used.

The first phase is composed of at most $k-1$ steps that we define as follows. At the first step of the first phase we introduce S disjoint intervals where $S = U^{3k}$. Since the algorithm is using at most U colors, this means that there exists a set of $\frac{S}{U}$ intervals that share the exact same color **c**. We shrink all intervals into single points. Later steps result in additional points.

We now define step j of the first phase. The steps are constructed in a way that in the beginning of step j there is a set of at least U^{3k-j+1} points that contain a given subset of the U colors. These points are called *points of interest*.

There exist some other points containing other subsets of colors. All these points are called *void points*. At this time, we partition the points of interest into consecutive sets of four. At most three points of interest that do not participate in this become void points.

We next define additional intervals, increasing the size of the largest cardinality clique (with respect to the number of intervals, *i.e.*, ignoring sizes) by exactly one. Given a set of four points listed from left to right a_1, a_2, a_3, a_4, let b be the leftmost void point on the right hand side of a_1, between a_1 and a_2. If no such point exists, then let $b = \frac{a_1+a_2}{2}$, *i.e.*, the point which is halfway between a_1 and a_2. Similarly, let d be the rightmost void point between a_3 and a_4, and if no such point exists then $d = \frac{a_3+a_4}{2}$. Let f be a point between a_2 and a_3 that is not a void point. We introduce the intervals $I_1 = [a_1, \frac{a_1+b}{2}]$ and $I_2 = [\frac{d+a_4}{2}, a_4]$.

If they both receive the same color, we introduce the intervals $I_3 = [\frac{a_1+b}{2}, f]$ and $I_4 = [f, \frac{d+a_4}{2}]$. The interval I_3 intersects with a_2, and with I_1. The second interval I_4 intersects I_3, a_3 and I_2, therefore two new colors must be used. In total, three new colors were used.

If I_1, I_2 receive distinct colors, we introduce the interval $I_5 = [\frac{a_1+b}{2}, \frac{d+a_4}{2}]$. Interval I_5 intersects with I_1, I_2, a_2, a_3, and thus gets a new color. In total, three new colors were used.

We shrink every such interval $[a_1, a_4]$ into a single point. Each of the new shrunk points received three new colors.

Note that we do not use more than U colors, and each new shrunk point receives three new colors. Four intervals are introduced only if the first two received the same color. There are less than $\frac{U^3}{6}$ options to choose from the set of three new colors. We can choose at least $6 \cdot U^{3k(i-j)}$ points having the same set of used colors. The points containing these exact sets of colors become the points of interest of the next step, and the others become void points of the next step. Points that are void points of previous steps and are not contained in shrunk intervals remain void points. Note that the points where the new intervals intersect are points with no previous intervals, and therefore the clique size increases by exactly 1.

At the end of the $k-1$-th step, the first phase where we have a set of zero sized intervals ends, and online algorithm used at least $3k-5$ colors to color these intervals.

Assume that c is a lower bound on the asymptotic competitive ratio of any online algorithm for standard bin packing, which knows the value $\mathrm{OPT} = k$ in advance. I.e., let $\varepsilon > 0$ be a small value, then there is a sequence of at most $f(k)$ items such that $\mathrm{OPT} = k$ and the algorithm is forced to use at least $(c-\varepsilon)k$ bins. E.g., in the proof of Lemma 7, $f(k) = 3k$. (Note that even if the lower bound construction uses an infinite number of items, we can use a subsequence of the construction of finite length which gives a lower bound of $c-\varepsilon$.)

We repeat the first phase construction $f(k) \cdot \binom{U}{3k-5}$ times (where each time is totally disjoint to the other times). In this way we create $f(k)$ sub-intervals of the real line, each has a set of zero sized intervals and these intervals are colored with colors $\{1, 2, \ldots, 3k-5\}$. In the second phase we will introduce intervals which match these sub-intervals of the real line. In this way the vertices of the new intervals will be adjacent to vertices of intervals with colors $1, 2, \ldots, 3k-5$ and therefore must be colored using a higher color.

In the second phase we consider the lower bound instance of the bin packing problem where OPT is known to the algorithm. If the lower bound construction asks to present an item of size s_i we present an interval with size s_i that overlaps exactly one sub-interval of the real line defined by the first phase (and therefore it cannot be colored with a color from $\{1, 2, \ldots, 3k-5\}$), and it does not intersect to any preceding interval of the second phase. In this way all intervals of the second phase are colored with colors greater than $3k-5$, and since they cannot be packed by the online algorithm using less than $(c-\varepsilon)k$ bins, they use colors $3k-4, \ldots, (3+c-\varepsilon)k-5$ (this is w.l.o.g. after renaming the colors).

To prove the claim it suffices to show that $\mathrm{OPT} = k$. To see this note that each of the first phase construction can be colored using $k-1$ colors (as the maximum clique size in it is $k-1$). Therefore, we consider the optimal solution for the bin packing instance that uses k bins. Then we traverse the first phase constructions one by one and allocate the intervals in the first phase constructions to $k-1$ colors among the existing k colors so that the overlapping interval of the second phase (if there is such one) has a different color. In this way the total size of items that are allocated to a color is at most one, and we obtain a coloring using k colors that satisfies the conflicts constraints such that the total size of each color class is at most one.

Lemma 7. *Any online algorithm for standard bin packing, which knows the value* OPT *in advance, has competitive ratio of at least* $\frac{47}{36} \approx 1.30556$.

Proof. We use a construction similar to the lower bound given by Yao in [30] (see also [29]). The difference is that since we commit on a given value of OPT in advance, we need to pad the sequence with items of size 1 in cases where we would otherwise simply stop the sequence.

Let N be a large integer which is divisible by 6. The input consists of one of the following three inputs.

1. N items of size 0.15, followed by $\frac{5}{6}N$ items of size 1.
2. N items of size 0.15, followed by N items of size 0.34, followed by $\frac{N}{2}$ items of size 1.
3. N items of size 0.15, followed by N items of size 0.34, followed by N items of size 0.51.

It is not difficult to verify that in all three cases, $\mathrm{OPT} = N$. We use the following variables. For $i = 1, \ldots, 6$, X_i denotes the number of bins with exactly i items of size 0.15, after only these items have arrived. For $i = 0, \ldots, 6$, $0 \leq j \leq 2$, $i + j > 0$, $X_{i,j}$ denotes the number of bins with exactly i items of size 0.15 and j items of size 0.34, after these two sets of items have arrived. Clearly, if $i \geq 3$ then $X_{i,2} = 0$ and if $i \geq 5$ then $X_{i,j} = 0$ for $j \neq 0$. For convenience we also let $X_{0,0} = 0$. We define $X_0 = X_{0,1} + X_{0,2}$ to be the number of bins with only (one or two) items of size 0.34. Moreover we have for $1 \leq i \leq 6$, $X_i = X_{i,0} + X_{i,1} + X_{i,2}$.

The following equalities must hold due to amounts of items. $\sum_{i=1}^{6} iX_i = N$ and $\sum_{i=0}^{6} \sum_{j=0}^{2} jX_{i,j} = N$.

Let $\mathcal{R}$ be the competitive ratio of an algorithm. We can compute the cost of the algorithm for each of the three inputs. This cost is at most $\mathcal{R} \cdot N$. The costs are $\sum_{i=1}^{6} X_i + \frac{5}{6}N$, $\sum_{i=0}^{6} X_i + \frac{N}{2}$, since in these cases, the algorithm must put the large items into new bins, and $X_{0,2}+X_{1,2}+X_{2,1}+X_{2,2}+X_{3,1}+X_4+X_5+X_6+N$. This is true since the following bins can accommodate an item of size 0.51; bins with no items of size 0.34 and at most three items of size 0.15, bins with one item of each of these sizes, and bins with only one item, which is of size 0.34.

We have three inequalities, which we multiply by the coefficients $1, 2, 3$, respectively, and get the following.

$$2X_{0,1} + 5X_{0,2} + 3X_{1,0} + 3X_{1,1} + 6X_{1,2} + 3X_{2,0} + 6X_{2,1} + 6X_{2,2} + 3X_{3,0} + 6X_{3,1} + 6X_{4,0} + 6X_{4,1} + 6X_{5,0} + 6X_{6,0} + \frac{29}{6}N \leq 6\mathcal{R}N$$

We have established the following two equalities $\sum_{i=1}^{6} iX_i = N$ and $\sum_{i=0}^{6} \sum_{j=0}^{2} jX_{i,j} = N$, which we multiply by the coefficients $1, 2$, respectively. Hence, we get the following.

$$2X_{0,1} + 4X_{0,2} + X_{1,0} + 3X_{1,1} + 5X_{1,2} + 2X_{2,0} + 4X_{2,1} + 6X_{2,2} + 3X_{3,0} + 5X_{3,1} + 4X_{4,0} + 6X_{4,1} + 5X_{5,0} + 6X_{6,0} = 3N$$

Since all variables are non-negative, we substitute and get $\frac{29}{6}N + 3N \leq 6\mathcal{R}N$, and thus $\mathcal{R} \geq \frac{47}{36}$.

5 Conclusion

We have improved the upper bounds for BPC on perfect graphs, interval graphs (and a few related classes) and bipartite graphs. Most our results follow from

adaptation of weighting systems to enable analysis of algorithms for BPC, and new algorithms which carefully remove small subgraphs of items which cause problematic instances. There is still a gap between the inapproximability which follows from bin packing, and the upper bounds. An open problem would be to close this gap.

Another open question is the following. As in [15], we used the absolute approximation ratio to analyze the performance of our algorithms. It can be seen that using the asymptotic approximation ratio, we can achieve a slightly better upper bound for bipartite graphs. It is unclear whether the same is true for other graph classes, i.e., whether the asymptotic approximation ratio for BPC is strictly lower than the absolute one for some cases.

References

1. E. Arkin and R. Hassin. On local search for weighted packing problems. *Mathematics of Operations Research*, 23:640–648, 1998.
2. B. S. Baker and E. G. Coffman, Jr. A tight asymptotic bound for next-fit-decreasing bin-packing. *SIAM J. on Algebraic and Discrete Methods*, 2(2):147–152, 1981.
3. E. G. Coffman, Jr., J. Csirik, and J. Leung. Variants of classical bin packing. In T. F. Gonzalez, editor, *Approximation algorithms and metaheuristics*. Chapman and Hall/CRC. To appear.
4. P. Crescenzi, V. Kann, M. M. Halldórsson, M. Karpinski, and G. J. Woeginger. A compendium of NP optimization problems. *http://www.nada.kth.se/ viggo/problemlist/compendium.html*.
5. D. de Werra. An introduction to timetabling. *European Journal of Operational Research*, 19:151–162, 1985.
6. G. Galambos and G. J. Woeginger. Repacking helps in bounded space online bin packing. *Computing*, 49:329–338, 1993.
7. M. R. Garey, R. L. Graham, D. S. Johnson, and A. C. C. Yao. Resource constrained scheduling as generalized bin packing. *Journal of Combinatorial Theory (Series A)*, 21:257–298, 1976.
8. M. R. Garey and D. S. Johnson. *Computers and intractability*. W. H. Freeman and Company, New York, 1979.
9. M. C. Golumbic. *Algorithmic Graph Theory and Perfect Graphs*. Academic Press, 1980.
10. R. L. Graham. Bounds on multiprocessing timing anomalies. *SIAM Journal on Applied Mathematics*, 17:263–269, 1969.
11. A. Gyárfás and J. Lehel. On-line and first-fit colorings of graphs. *Journal of Graph Theory*, 12:217–227, 1988.
12. M. Hujter and Z. Tuza. Precoloring extension, III: Classes of perfect graphs. *Combinatorics, Probability and Computing*, 5:35–56, 1996.
13. S. Irani and V. J. Leung. Scheduling with conflicts, and applications to traffic signal control. In *Proc. of 7th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'96)*, pages 85–94, 1996.
14. K. Jansen. An approximation scheme for bin packing with conflicts. *Journal of Combinatorial Optimization*, 3(4):363–377, 1999.
15. K. Jansen and S. Öhring. Approximation algorithms for time constrained scheduling. *Information and Computation*, 132:85–108, 1997.

16. T. R. Jensen and B. Toft. *Graph coloring problems*. Wiley, 1995.
17. D. S. Johnson, A. Demers, J. D. Ullman, Michael R. Garey, and Ronald L. Graham. Worst-case performance bounds for simple one-dimensional packing algorithms. *SIAM Journal on Computing*, 3:256–278, 1974.
18. H. A. Kierstead and W. T. Trotter. An extremal problem in recursive combinatorics. *Congressus Numerantium*, 33:143–153, 1981.
19. C. C. Lee and D. T. Lee. A simple online bin packing algorithm. *Journal of the ACM*, 32(3):562–572, 1985.
20. L. Lovász, M. Saks, and W. T. Trotter. An on-line graph coloring algorithm with sublinear performance ratio. *Discrete Math.*, 75:319–325, 1989.
21. D. Marx. Precoloring extension. *http://www.cs.bme.hu/ dmarx/prext.html*.
22. D. Marx. Precoloring extension on chordal graphs. manuscript, 2004.
23. B. McCloskey and A. Shankar. Approaches to bin packing with clique-graph conflicts. Technical Report UCB/CSD-05-1378, EECS Department, University of California, Berkeley, 2005.
24. Y. Oh and S. H. Son. On a constrained bin-packing problem. Technical Report CS-95-14, Department of Computer Science, University of Virginia, 1995.
25. A. Schrijver. *Combinatorial optimization polyhedra and efficiency*. Springer-Verlag, 2003.
26. S. S. Seiden. On the online bin packing problem. *Journal of the ACM*, 49(5):640–671, 2002.
27. D. Simchi-Levi. New worst-case results for the bin-packing problem. *Naval Res. Logist.*, 41(4):579–585, 1994.
28. J. D. Ullman. The performance of a memory allocation algorithm. Technical Report 100, Princeton University, Princeton, NJ, 1971.
29. A. van Vliet. An improved lower bound for online bin packing algorithms. *Information Processing Letters*, 43(5):277–284, 1992.
30. A. C. C. Yao. New algorithms for bin packing. *Journal of the ACM*, 27:207–227, 1980.