Class constrained bin packing revisited*

Leah Epstein† Csanád Imreh‡ Asaf Levin§

Abstract

We study the following variant of the bin packing problem. We are given a set of items, where each item has a (non-negative) size and a color. We are also given an integer parameter k, and the goal is to partition the items into a minimum number of subsets such that for each subset S in the solution, the total size of the items in S is at most 1 (as in the classical bin packing problem) and the total number of colors of the items in S is at most k (which distinguishes our problem from the classical version). We follow earlier work on this problem and study the problem in both offline and online scenarios.

1 Introduction

In the CLASS CONSTRAINED BIN PACKING PROBLEM (CCBP), we are given a set of items $I = \{1, 2, \ldots, n\}$, where each item has a size and a color associated with it. The size of item i is denoted by s_i, and we assume that $s_i \in [0, 1]$. The color of item i is denoted by c_i (so if i and j have the same color then $c_i = c_j$). The set of items of one color is also called a *color class*. We assume that each color has a positive integer associated with it, that is, $c_i \in \mathbb{N}$. We are also given a (non-negative) integer parameter k. A feasible solution is a partition of I into subsets $S_1, \ldots, S_m$ such that for each $i = 1, 2, \ldots, m$, the following two conditions hold: $\sum_{j \in S_i} s_j \leq 1$ and S_i has items from at most k color classes (i.e., $|\bigcup_{j \in S_i} \{c_j\}| \leq k$). The goal of CCBP is to find a feasible solution that minimizes the number of subsets in the partition. We denote by q the total number of color classes in the instance. Note that if $q = n$, the resulting problem is equivalent to the bin packing problem with cardinality constraints [9, 10, 1, 2, 8, 5, 6]. If all items are of at most k color classes, i.e., $q \leq k$, we get the classical bin packing problem [3]. We refer the reader to the previous work on CCBP [14, 16, 19] for details on the applications of this packing problem in Video on Demand, storage management and other fields.

For an algorithm $\mathcal{A}$, we denote its cost on an input X by $\mathcal{A}(X)$, and if X is clear from the context, we simply use $\mathcal{A}$. An optimal offline algorithm (that in the case of comparison to online algorithms, knows the complete sequence of items) is denoted by OPT. For minimization problems, the (asymptotic) approximation ratio (competitive ratio for online algorithms) of an algorithm $\mathcal{A}$ is the infimum $\mathcal{R} \geq 1$ such that for any input X, $\mathcal{A}(X) \leq \mathcal{R} \cdot \text{OPT}(X) + c$ holds, where c is a constant independent of the input. An (asymptotic) polynomial time approximation scheme is a family of approximation algorithms such that for every $\varepsilon > 0$ the family contains a polynomial time algorithm with an (asymptotic) approximation ratio of

*This research has been partially supported by the Hungarian National Foundation for Scientific Research, Grant F048587.

†Department of Mathematics, University of Haifa, 31905 Haifa, Israel. lea@math.haifa.ac.il

‡Department of Informatics, University of Szeged, 6720 Szeged, Hungary. cimreh@inf.u-szeged.hu

§Chaya fellow. Faculty of Industrial Engineering and Management, The Technion, 32000 Haifa, Israel. levinas@ie.technion.ac.il

$1+\varepsilon$. We abbreviate *polynomial time approximation scheme* by PTAS and *asymptotic polynomial time approximation scheme* by APTAS, which is also called an asymptotic PTAS. A fully polynomial time approximation scheme (FPTAS) is a PTAS whose time complexity is polynomial not only in the number of items n but also in $\frac{1}{\varepsilon}$. Similarly, an AFPTAS is an APTAS whose time complexity is polynomial not only in the number of items n but also in $\frac{1}{\varepsilon}$. An algorithm which has an approximation ratio of at most R is called an R-approximation, or an R-approximate solution. For online algorithms, such an algorithm (of competitive ratio at most R) is called R-competitive.

In this paper, we consider both the offline version of CCBP and its online version. It is known that the classical bin packing problem admits an APTAS [4] and an AFPTAS [7]. Furthermore, bin packing with cardinality constraints, admits an APTAS [2] and an AFPTAS [6]. A natural question is whether CCBP, which is a generalization of both these problems, admits an APTAS (and possibly an AFPTAS as well). The problem is clearly NP-hard by the hardness of classical bin packing. Moreover, Shachnai and Tamir showed that already the problem with identical sized items is NP-hard in the strong sense [14, 15]. In [14], they designed an algorithm which uses the smallest possible number of bins, but allows to use slightly larger bins of size $1+\varepsilon$. Such an algorithm is called a dual PTAS. The time complexity of this dual PTAS is polynomial in n for constant values of q. Xavier and Miyazawa [19] designed an APTAS for constant values of q. This raises the question of whether seeing q as a parameter rather than as a constant changes the complexity of the problem. We answer this question affirmatively and show that this more general case does not admit an APTAS for any value of k. We close the offline problem by designing an AFPTAS for the case of constant q; due to our hardness result, there is no APTAS for the case of arbitrary values of q (unless P=NP), and hence our scheme is best possible.

In Section 2, we consider this special case of the offline problem, where q is a constant, and present our AFPTAS for it, improving upon the APTAS (for this special case) of Xavier and Miyazawa [19]. To do so, we present a different way to handle the small items. Whereas the scheme of [19] tries all possible packing of the large items, and for each of them solves a linear program for the packing of the small items, we construct one linear program that considers both large and small items. We use methods that are similar to the ones recently developed in [6], that were used to develop an AFPTAS for bin packing with cardinality constraints.

In Section 3 we show our hardness result on the approximability of CCBP for every constant value of k, namely, that the asymptotic approximation ratio of any algorithm for CCBP is at least $1+\frac{1}{10k}$ for any value of k (if q is seen as a parameter of the problem).

Regarding the online version of the problem, previous studies [16, 19] analyze two variants of the FIRST FIT (FF) heuristic. The first one is simply called FF. Whenever a new item arrives, FF tries to pack it in an existing bin (if it fits both with respect to the number of colors in the bin and its size), and if this is indeed possible, it packs the item in the first such bin. The second variant is called COLOR SETS FIRST FIT (CSFF). In this variant, color classes are partitioned online into sets of k colors (where the first k colors that ever appear are the first color set, the next k colors that ever appear are the second color set, and so forth), and each such color set has its own dedicated bins. When a new item arrives we apply FF, considering only the bins of the color set that contains the color of the new item. Another natural algorithm is called COLOR SETS NEXT FIT (CSNF). This algorithm partitions the input into color sets exactly as CSFF, but each color set is packed using NEXT FIT rather than first fit, that is, each color set uses a single active bin, and whenever a new item cannot be packed there, the bin is closed and a new active bin is opened.

The online version of the problem was first studied in [16], where the case of identical sized items was considered. They showed that the competitive ratio of both CSFF and FF is at most 2. A matching lower

bound for some cases was presented. Note that the value of the lower bound for $k = 2$ is $\frac{3}{2}$. Since items have identical sizes, the case $k = 1$ can be solved trivially by a NEXT FIT (NF) approach on each color. In Sections 4.1 and 4.2 we show that the competitive ratio of both FF and CSFF for $k = 2$ is exactly 2, for identical items. We further show that it is strictly above 2, namely, at least $\frac{9}{4}$, for items of arbitrary sizes. We present a lower bound of approximately 1.5652 on the competitive ratio of any online algorithm for $k = 2$ (acting on items of arbitrary size) in Section 4.3. Note that the lower bound of van Vliet [18] on the competitive ratio of any algorithm for classical bin packing is 1.5401.

The online problem was studied further in [19] where algorithms for arbitrary sizes of items were studied. It was shown that the competitive ratio of CSFF it at most 3. Xavier and Miyazawa [19] designed a different algorithm that is based on a partition into three classes according to size, and showed that its competitive ratio is at most 2.75. It was shown that the competitive ratios of these two algorithms cannot be below 2.7 and $\frac{8}{3}$, respectively, for large enough values of k.

In Section 5.1 we analyze CSFF further and show that its competitive ratio is at most $3 - \frac{1}{k}$. Thus, we conclude that the competitive ratio of CSFF for $k = 2$ lies in the interval $[\frac{9}{4}, \frac{5}{2}]$. In Section 5.2, we design an improved online algorithm for $k = 3$ (of competitive ratio $\frac{107}{42} \approx 2.547619$), that is based on a partition of items into three sets as in [19], but allows combining items of different sets. Moreover, our algorithm uses an unusual rule, where tiny items are sometimes combined with a very large item in a bin.

In Section 5.3 we show a general reduction to online (classical) bin packing algorithms under some conditions on these algorithms, that allows to convert such an algorithm into an algorithm for CCBP, with a loss of at most 1 in the asymptotic competitive ratio. This, together with the algorithm in Section 5.2 (that can be applied to any value of k) allows us to find improved algorithms for all values of k, giving an overall upper bound of 2.63492.

The next table summarizes our results.

	Upper bound	Lower bound
Offline algorithms, constant q	AFPTAS	Strongly NP-hard (generalizes bin packing)
Offline algorithms, arbitrary q	2.6349	$1 + \frac{1}{10k}$ (unless P=NP)
Online algorithms, $k = 2$	2.5	1.5652
Online algorithms, arbitrary k	2.6349 (2.54762 for $k = 3$)	2 [16]
CSFF, $k = 2$	2.5	2.25
CSFF, FF $k = 2$, equal sized items	2 [16]	2
CSFF, arbitrary k	$3 - \frac{1}{k}$	2 [16]

2 An AFPTAS for the offline problem with constant q

In this section we improve the APTAS of Xavier and Miyazawa [19] for constant values of q, by incorporating the column-generation technique of Karmarkar and Karp [7] into the scheme of [19], together with simplified version of the methods of [6] for dealing with small items. Note that the assumption of a constant q also means that k is a constant, since we may assume $k \leq q$ (otherwise, the problem reduces to classical bin packing). Let $\varepsilon > 0$ be such that $\frac{1}{\varepsilon}$ is an integer and $\varepsilon \leq 1/3$. Our scheme is valid for any $k \geq 1$. A scheme for $k = 1$ can be constructed also from applying the scheme of [7] for every color class separately. Since any solution must pack every color class independently, and there is a constant number of color

classes, this immediately results in an AFPTAS.

Scheme overview. The general structure of the AFPTAS is as follows. We separate the items into large and small ones, and use linear grouping ([4]) on each color class separately. After the grouping, there is a constant number of rounded sizes of items of each color. The pair consisting of the rounded size of the item and its color, is called the type of the item. We define packing patterns on the types of items, taking into account the properties that a bin may contain items of at most k different colors, and that the total rounded size of packed items of a bin is at most 1. If the items of the pattern have less than k colors, still the pattern would have exactly k colors associated with it. We define a packing of the large rounded items, where the small items are seen as fractional items, that is, we only keep track of the total size of the small items of each color. However, the linear program has a variable for every pair of a color p and a subset T of k colors, which corresponds to the total size of small items of color p which are to be packed into bins with patterns whose colors are T. The packing is defined via a linear program which determines the number of copies of each pattern. The linear program is solved approximately using the column generation technique of [7]. For that, we apply an FPTAS for the knapsack problem, which tests the approximate feasibility of a dual solution. Given a solution to the primal linear program, we transform it into a basic solution which costs no more than the given solution, and then in order to give an output solution, we round up all fractional components, and pack small items greedily, while the small remaining items are packed into dedicated bins. We next describe the details of our scheme and its analysis.

Linear Grouping. We say that an item is *large* if its size is at least ε and otherwise it is *small*. We assume that the set of colors in the instance is denoted by $Q = \{1, 2, \ldots, q\}$. We denote by L^p the set of large items of color p, and by S the set of small items. We first apply linear grouping (originally introduced in [4]) on the large items of each color class separately. That is, for every $p = 1, 2, \ldots, q$ we partition L^p into $\frac{1}{\varepsilon^2}$ parts $L^p_1, \ldots, L^p_{1/\varepsilon^2}$ such that the following two conditions hold: $\lceil |L^p|\varepsilon^2 \rceil = |L^p_1| \geq |L^p_2| \geq \cdots \geq |L^p_{1/\varepsilon^2}| = \lfloor |L^p|\varepsilon^2 \rfloor$, and moreover L^p_1 contains the $|L^p_1|$ largest items of L^p, and for every $t = 2, 3, \ldots, 1/\varepsilon^2$, L^p_t contains the $|L^p_t|$ largest items of $L^p \setminus (\bigcup_{j=1}^{t-1} L^p_j)$. Note that these two conditions uniquely define the partition of the large items up to the allocation of equal sized items of a common color. In the case $|L^p| < 1/\varepsilon^2$, we modify the partition so that each L^p_j has up to one item, and L^p_1 is empty.

Lemma 1 *For every $p = 1, 2, \ldots, q$, $|L^p_1| \leq 3\varepsilon^2 |L^p \setminus L^p_1|$ holds.*

Proof If L^p_1 is empty, it is clearly true. Otherwise, $|L^p_1| = \lceil |L^p|\varepsilon^2 \rceil \leq |L^p|\varepsilon^2 + 1$, and $|L^p \setminus L^p_1| \geq |L^p| - |L^p|\varepsilon^2 - 1$. It is enough to prove that for a value $X \geq \frac{1}{\varepsilon^2}$, we have $X\varepsilon^2 + 1 \leq 3\varepsilon^2(X(1-\varepsilon^2) - 1)$. This is equivalent to $X \geq \frac{1+3\varepsilon^2}{2\varepsilon^2 - 3\varepsilon^4}$ (since $2\varepsilon^2 > 3\varepsilon^4$ for any $\varepsilon \leq \frac{1}{3}$) and holds since $\frac{1}{\varepsilon^2} \geq \frac{1+3\varepsilon^2}{2\varepsilon^2-3\varepsilon^4}$ for any $\varepsilon \leq \frac{1}{3}$. □

Next we round the size of the large items. For every $p = 1, 2, \ldots, q$ and every $j \geq 2$ we let the *rounded up size* of the items of L^p_j to be $\max_{i \in L^p_j} s_i$, and we let s'_i denote the rounded up size of item i (where for an item i such that $i \notin \bigcup_{p=1}^{q} \bigcup_{j=2}^{1/\varepsilon^2} L^p_j$ we let $s'_i = s_i$). Note that if $|L^p| < \frac{1}{\varepsilon^2}$, then we have $s'_i = s_i$ for every $i \in L^p$. We let $L' = \bigcup_{p=1}^{q} \bigcup_{j=2}^{1/\varepsilon^2} L^p_j$ and $I' = L' \cup S$, where the size of item $i \in I'$ is the rounded up size s'_i. The set L' can be seen as a multi-set of items, where all items of L' are of at most $q(\frac{1}{\varepsilon^2} - 1)$ distinct types,

where a *type* is specified by a pair of a size and a color, (v, p). Let H denote the set of distinct types of items in L'. We enumerate H by $\{\psi_1, \psi_2, \ldots\}$. It is not difficult to see that $\text{OPT}(I') \leq \text{OPT}(I)$, since for any item in a set L_j^p for which rounding was applied (for $j \geq 2$), its rounded up size is no larger than the original size of any item in L_{j-1}^p, and I' does not contain the sets L_1^p. Let Γ denote the set of all subset of k colors, that is $\Gamma = \{Q' \subseteq Q : |Q'| = k\}$. We enumerate Γ by $\{\kappa_1, \kappa_2, \ldots\}$, and note that $|\Gamma| = O(q^k)$, that is a constant.

Bin configuration. A configuration of a bin represents a possible packing of a subset of items of L' into a bin. It is a $|H| + 1$-tuple, where the i-th component, for $1 \leq i \leq |H|$ states the number of items of type ψ_i which are packed into this bin, and the $|H| + 1$-th component is a member of Γ, κ_j. Every positive component i must satisfy that the color of ψ_i is in κ_j. A configuration C is therefore a subset of items of L' of at most k colors, whose total size is at most 1. The set of colors κ_j, which C has associated with it, is denoted by $Col(C)$ and according to the above definition, the colors of the large items of C belong to $Col(C)$ (but $Col(C)$ may possibly contain additional colors). These are the k allowed colors for a bin with configuration C. We denote the set of all configurations by $\mathcal{C}$. Note that two configurations C_1 and C_2, that have the same configuration of large items, but $Col(C_1) \neq Col(C_2)$, are seen as two distinct configurations. We note that $|\mathcal{C}| = O(|H|^{|H|} \cdot |\Gamma|) = O((\frac{q}{\varepsilon}^2)^{(\frac{q}{\varepsilon}^2)} \cdot q^k)$, that is an exponential function of $\frac{1}{\varepsilon}$, and therefore we cannot enumerate $\mathcal{C}$ in polynomial time.

Constructing the linear program. For each $h = (v, p) \in H$ and a configuration C, we denote by $n(h, C)$ the number of items with type h in C, and we denote by $n(h)$ the number of items of type h in L'. We (approximately) solve the following linear program where for each configuration C, we have a variable x_C indicating the number of bins that we pack using configuration C. Moreover, for any subset $T \subseteq \{1, 2, \ldots, q\}$ of exactly k colors, and a color $p \in T$, we have a variable $Y_{p,T}$ indicating the total size of the small items of color p that we pack into bins with some configuration C such that $Col(C) = T$. We implicitly set $Y_{p,T} = 0$ if $p \notin T$. We denote the set of small items of color p by S_p.

$$
\begin{aligned}
\min \quad & \sum_{C \in \mathcal{C}} x_C & & \\
s.t. \quad & \sum_{C \in \mathcal{C}} n(h, C) x_C \geq n(h) & & \forall h \in H \\
& \sum_{C : Col(C) = T} \left(1 - \sum_{h=(v,p) \in H} n(h, C) \cdot v\right) x_C \geq \sum_{p \in T} Y_{p,T} & & \forall T \subseteq Q : |T| = k \\
& \sum_{T \subseteq Q : |T| = k} Y_{p,T} \geq \sum_{i \in S_p} s'_i & & \forall p \in Q \\
& x_C \geq 0 & & \forall C \in \mathcal{C} \\
& Y_{p,T} \geq 0 & & \forall p \in Q,\ \forall T \subseteq Q : |T| = k.
\end{aligned}
$$

Note that this linear program has an exponential number of variables (exponential as a function of $\frac{1}{\varepsilon}$), and hence we will not write it down explicitly, however we will be able to solve it approximately within a factor of $1 + \varepsilon$. Denote by (x^*, y^*) an approximate (within a factor of $1 + \varepsilon$) basic solution to this linear program, and let $\tilde{x}_C = \lceil x_C^* \rceil$ for all C. Our scheme returns a solution that packs $\tilde{x}_C$ bins with configuration C. Each item of the rounded up instance is later replaced by the corresponding item of I. We can clearly pack the items of $\bigcup_{p=1}^{q} \bigcup_{j=2}^{1/\varepsilon^2} L_j^p$ in these bins (some slots reserved to such items may remain empty). Note that for every p, the total size assigned to small items of color p is at least the total size of these small items.

The column generation technique. To solve the above linear program approximately we invoke the column generation technique of Karmarkar and Karp [7]. We next elaborate on this technique. The linear program may have an exponential number of variables but it has a polynomial number of constraints (neglecting the non-negativity constraints). Instead of solving the linear program we solve its dual program (that has a polynomial number of variables but possibly an exponential number of constraints). The variables α_h correspond to the item types in H, their intuitive meaning can be seen as weights of these items. The variables β_T correspond to subsets of k colors that are packed in a common bin, i.e., subsets that can act as a set $Col(C)$ of some configuration C. The variables γ_p correspond to colors, and their intuitive meaning can be seen as weights per unit of size of the small items of this color.

$$\begin{aligned}
\max \quad & \sum_{h\in H} n(h)\alpha_h + \sum_{p=1}^{q}\Big(\sum_{i\in S_p} s'_i\Big)\gamma_p & \\
s.t. \quad & \sum_{h\in H} n(h,C)\alpha_h + \left(1 - \sum_{h=(v,p)\in H} n(h,C)v\right)\beta_{Col(C)} \le 1 & \forall C \in \mathcal{C} \\
& -\beta_T + \gamma_p \le 0 & \forall p \in Q,\ \forall T \subseteq Q : |T| = k, p \in T \\
& \alpha_h \ge 0 & \forall h \in H \\
& \beta_T \ge 0 & \forall T \subseteq Q : |T| = k \\
& \gamma_p \ge 0 & \forall p \in Q.
\end{aligned}$$

To be able to apply the ellipsoid algorithm, in order to solve the above dual problem within a factor of $1+\varepsilon$, it suffices to show that there exists a polynomial time algorithm (polynomial in n and $\frac{1}{\varepsilon}$) such that for a given solution $a^* = (\alpha^*, \beta^*, \gamma^*)$, which is a vector of length at most $\frac{q}{\varepsilon^2} + q^k + q$ (since there are $|H| = \frac{q}{\varepsilon^2}$ variables of type α_h, less than q^k variables of type β_T and $|Q| = q$ variables of type γ_p), decides whether a^* is close enough to a feasible dual solution. More precisely, it should either provides a configuration $C \in \mathcal{C}$ such that $\sum_{h\in H} n(h,C)\alpha^*_h + \left(1 - \sum_{h=(v,p)\in H} n(h,C)v\right)\beta^*_{Col(C)} > 1$ or outputs that such an approximate infeasibility evidence does not exist, that is, for all configurations $C \in \mathcal{C}$, $\sum_{h\in H} n(h,C)\alpha^*_h + \left(1 - \sum_{h=(v,p)\in H} n(h,C)v\right)\beta^*_{Col(C)} \le 1+\varepsilon$ holds. In such a case, $\frac{a^*}{1+\varepsilon}$ satisfies all the constraints of the first family in the dual program, and there is a polynomial number of other constraints that can be checked efficiently.

Approximated separation oracle for the dual linear program. Such a configuration C can be found using an FPTAS for the KNAPSACK problem. This is so because for each $T \subseteq Q$ such that $|T| = k$, we need to solve the following problem: given items H where for each $h = (v,p) \in H$ there is a volume $\alpha^*_h - v\beta^*_T$ and a size v, the goal is to pack a multiset of the items of the types of H (where an item can appear multiple times but at most a given number of times) whose total size is at most 1, so that the total volume is maximized. If our FPTAS to the KNAPSACK problem finds a solution with total volume greater than $1-\beta^*_T$ then this solution is a configuration whose constraint in the dual linear program is not satisfied, and we can continue with the application of the ellipsoid algorithm. Otherwise, since the FPTAS is an approximation within a factor of $1+\varepsilon$, we get that the maximum volume is at most $(1+\varepsilon)(1-\beta^*_T)$. We show that in this case, all the constraints of the dual linear program are satisfied by the solution $\frac{a^*}{1+\varepsilon}$. This clearly holds for the second type of constraints. To show this for the first type of constraints, consider a configuration $C \in \mathcal{C}$. We have $\sum_{h=(v,p)\in H} (\alpha^*_h - v\beta^*_{Col(C)})n(h,C) \le (1+\varepsilon)(1-\beta^*_{Col(C)})$, therefore

$$\sum_{h\in H} n(h,C)\frac{\alpha^*_h}{1+\varepsilon} + \left(1 - \sum_{h=(v,p)\in H} n(h,C)v\right)\frac{\beta^*_{Col(C)}}{1+\varepsilon} \leq 1 - \beta^*_{Col(C)} + \frac{\beta^*_{Col(C)}}{1+\varepsilon} < 1.$$

Bounding the cost of (x^*, y^*). We note that $\text{OPT}(I')$ induces a feasible solution to the primal linear program (x_C is the number of bins with configuration C in $\text{OPT}(I')$, and $Y_{p,T}$ is the total size of the items in S_p such that $\text{OPT}(I')$ packs in bins with color set T). Therefore, since (x^*, y^*) is an $(1+\varepsilon)$-approximated solution to the primal linear program, we conclude that $\sum_{C\in\mathcal{C}} x^*_C \leq (1+\varepsilon)\text{OPT}(I')$.

Rounding the primal solution. Given the $(1+\varepsilon)$-approximated solution to the primal linear program, we find a basic feasible solution to this linear program, which is not worse than the approximated solution we obtained (in terms of their objective function values). Hence, without loss of generality we assume that (x^*, y^*) is a basic feasible solution which is an $(1+\varepsilon)$-approximated solution. We note that in the primal linear program there are at most $|H| + q + q^k$ inequality constraints (in addition to the non-negativity constraints), and therefore in a basic solution such as (x^*, y^*), there are at most $|H| + q + q^k$ basic variables. Since all non-basic variables are set to zero, we conclude that the number of fractional components in (x^*, y^*) is at most $|H|+q+q^k$. Therefore, $\sum_{C\in\mathcal{C}} \tilde{x}_C \leq \sum_{C\in\mathcal{C}} x^*_C + |H|+q+q^k \leq (1+\varepsilon)\text{OPT}(I') + |H|+q+q^k$.

Packing $L_1 = \bigcup_{p=1}^{q} L_1^p$. We next bound the increase of the cost caused by the largest items in each color class in dedicated bins. Recall that $L' = L \setminus L_1$. Then, the solution defined by $\tilde{x}$ packs the items of L'. We pack each item of L_1 in a separate (dedicated) bin. We note that $\text{OPT}(I') \geq \sum_{i\in L'} s'_i$, since the size of a large item is at least ε, using Lemma 1, we get that for every p, $|L_1^p| \leq 3\varepsilon^2 |L^p \setminus L_1^p| \leq 3\varepsilon \cdot \sum_{i\in L^p\setminus L_1^p} s'_i$. Summing the last inequality for all p we get that $|L_1| \leq 3\varepsilon \cdot \sum_{i\in L'} s'_i \leq 3\varepsilon\text{OPT}(I')$. Therefore, packing the items in L_1 in separate bins adds at most $3\varepsilon\text{OPT}(I')$ to the cost of $\tilde{x}$. So the resulting solution costs at most $(1+4\varepsilon)\text{OPT}(I') + |H| + q + q^k$.

Packing the small items. Consider a color set T ($T \subseteq Q$ and $|T| = k$). Then the solution $(\tilde{x}, Y^*)$ allocates space for small items. More precisely, for a bin that is packed according to configuration C, whose color set is T and its available space is $\sigma(C) = \left(1 - \sum_{h=(v,p)\in H} n(h,C)v\right)$ (after packing all the large items), we define a space for small items of color p to be $z_p(C) = \sigma(C) \cdot \frac{Y_{p,T}}{\sum_{p'\in T} Y_{p',T}}$. Note that for $p \notin T$ this implies $z_p(C) = 0$. By the second constraint of the primal linear program, the sum of values $\sigma(C)$ over all bins of the solution that are according to a configuration whose set of colors is T, is at least $\sum_{p'\in T} Y_{p',T}$. Thus the total size allocated in such bins for small items of color p is at least $Y_{p,T}$.

We pack the items of S_p into the available spaces allocated for them using Next-Fit. Specifically, for every bin packed by some configuration C, we place items from S_p until we exceed a total of $z_p(C)$. This packing is possibly invalid, and at most one item of S_p needs to be removed from the bin. We apply this to all colors in T and all bins (with color set T). The process is stopped if no items or no spaces are left. Recall that (x^*, y^*) is a feasible solution to the primal linear program, and all allocated spaces for color p are filled completely, unless all items are assigned. Thus, at the end of this procedure, we are left with no small items that needs to be packed. This means that for every bin, and each color that is allocated space

in this bin, there is at most one item that needs to be removed from the bin to allow the packing to become valid. Therefore, since the size of each small item is at most ε, the total size of the removed small items is at most $k\varepsilon \cdot \sum_{C \in \mathcal{C}} x^*_C \leq k\varepsilon(1+\varepsilon)\text{OPT}(I') \leq 2k\varepsilon\text{OPT}(I')$, where the last inequality holds because $\varepsilon \leq 1$.

For every $p = 1, 2, \ldots, q$, we pack the remaining small items of color p in dedicated bins (dedicated only to this color), using Next-Fit. We note that the number of bins that are not full up to a level of $1-\varepsilon$ is at most q (by the area guarantee of the Next-Fit algorithm when applied to items with sizes less than ε). Therefore, the number of additional dedicated bins is at most $\frac{2k\varepsilon\text{OPT}(I')}{1-\varepsilon} + q \leq 3k\varepsilon\text{OPT}(I') + q$. This concludes the presentation of the AFPTAS for fixed values of q.

Theorem 1 *If the number of colors in the instance is a fixed constant, the above scheme is an AFPTAS for* CCBP.

Proof The number of bins used by our solution is at most $(1+4\varepsilon)\text{OPT}(I')+|H|+q+q^k+3k\varepsilon\text{OPT}(I')+q \leq (1+(3k+4)\varepsilon)\text{OPT}(I')+|H|+2q+q^k$. Since $|H| \leq \frac{q}{\varepsilon^2}$ and k, q are constant, we conclude that the additive error term $|H|+2q+q^k$ is a constant, and hence by scaling ε by a factor of $3k+4$ we obtain an AFPTAS as required.

The time complexity of our scheme is dominated by the linear grouping and rounding of the sizes of large items, which can be done in $O(\frac{n}{\varepsilon^2})$. This is because the application of the ellipsoid algorithm on the dual problem takes a polynomial number of iterations in the number of variables of the dual linear program, and the encoding of the coefficients (that is, a polynomial function of $\frac{q}{\varepsilon^2}+q+q^k$, $\log n$ and $\max_{s_i>0} \log \frac{1}{s_i}$), which is a polylogarithmic number of iterations. Each iteration consists of at most q^k applications of the FPTAS for the knapsack problem where the number of items is $|H| = \frac{q}{\varepsilon^2}$ which is a constant. Hence, it takes a constant time for each iteration of the ellipsoid algorithm, and the resulting primal solution is obtained in polylogarithmic time. □

In the next section we show that our result is best possible in the sense that without the assumption of a fixed constant number of colors in the instance, CCBP does not have an asymptotic approximation scheme, that is, an AFPTAS or even an APTAS (already for fixed values of k).

3 Hardness of approximation when q is not fixed

In this section we show that for each constant value of k, it is NP-hard to approximate CCBP with an asymptotic approximation ratio strictly smaller than $1+\frac{1}{10k}$, and therefore without the assumption that q is a constant, CCBP does not have an APTAS (or AFPTAS).

Theorem 2 *Fix a value of k. If $k \geq 2$, then the offline* CCBP *problem does not have an approximation algorithm with an asymptotic approximation ratio strictly smaller than $1+\frac{1}{10k}$ unless $P = NP$. If $k = 1$, then the offline* CCBP *problem does not have an approximation algorithm with an asymptotic approximation ratio strictly smaller than $\frac{3}{2}$ unless $P = NP$.*

Proof We will show this claim via a reduction from the PARTITION problem defined as follows. We are given n non-negative rational numbers $a_1, a_2 \ldots, a_n$ such that $\sum_{i=1}^{n} a_i = 1$. The goal is to check whether there is a subset $S \subseteq \{1, 2, \ldots, n\}$ such that $\sum_{i \in S} a_i = \frac{1}{2}$. We construct the following instance of CCBP. The bin size is scaled to be $k - \frac{1}{2}$. Sizes of items are defined according to this bin size. There are $2n(k-1)+n$

color classes denoted by $A_1, \ldots, A_{2n(k-1)}, B_1, \ldots, B_n$. For every $i = 1, 2, \ldots, 2n(k-1)$, the color class A_i consists of a single item of size 1. For every $i = 1, 2, \ldots, n$, the color class B_i has n items, where the r-th item of this color class denoted by b_i^r has size a_r. To prove the claim it suffices to show that if the PARTITION instance is feasible, then the optimal solution to the instance of CCBP costs at most $2n$, whereas if the PARTITION instance is infeasible, then the cost of the optimal solution to CCBP is at least $2n + \frac{n}{5k}$. For $k = 1$, a stronger result can be proved, namely, the cost of the optimal solution to CCBP is at least $3n$.

We first prove the claim for $k = 1$. In this case, only the sets B_i exist. If the PARTITION instance is feasible, then each color class requires two bins. However, if the instance is infeasible, since each bin can contain items of a single color, at least three bins are necessary for each color. Therefore, at least $3n$ bins are used.

Next, we prove the claim for $k \geq 2$. First, assume that the PARTITION instance is feasible. That is, we assume that there is a subset $S \subseteq \{1, 2, \ldots, n\}$ such that $\sum_{i \in S} a_i = \frac{1}{2}$. We construct a solution to the CCBP instance as follows. For every $p = 1, 2, \ldots, n$ we pack in bin $2p - 1$ the items of color classes $A_{(2p-2)(k-1)+1}, A_{(2p-2)(k-1)+2}, \ldots, A_{(2p-1)(k-1)}$ (altogether these are items of $k-1$ color classes, with a total size of $k-1$), and in addition, the set $\{b_p^r : r \in S\}$ of items of one color class (with total size $\frac{1}{2}$) is packed into the same bin, which gives a total of k color classes. For $p = 1, 2, \ldots, n$, we pack in bin $2p$ the items of color classes $A_{(2p-1)(k-1)+1}, A_{(2p-1)(k-1)+2}, \ldots, A_{2p(k-1)}$ and also the items $\{b_p^r : r \notin S\}$. We conclude that all items can be packed in $2n$ bins. Informally, in this case every bin receives $k-1$ of the unit sized items, and every set B_i is partitioned into two parts of equal size, to be split into two bins.

We next assume that the PARTITION instance is infeasible. We fix an optimal solution OPT to CCBP. If the color class B_i is partitioned into at least two bins, then we call it *partial color class*, and all other color classes (including the A_i classes) are called *full color classes*. Note that each bin in OPT has at most $k-1$ full color classes, since the total size of items of each full class is 1, and the bin size is $k - \frac{1}{2}$. The remainder of the proof is based on the property that since the PARTITION instance is infeasible, a color class B_i cannot be split into two bins, where each bin contains, in addition to the items of B_i, exactly $k-1$ full color classes. So one of the following three options must occur, where each one of the options leads to an increased number of bins compared to the case where the PARTITION instance is feasible. The first option is that many such sets are full, leading to a large number of bins with a total size of items of only $k-1$, rather than $k - \frac{1}{2}$. The second option is that many such sets B_i are split into two parts of different size. We show later that this leads to partially occupied bins as well. The third option is that many sets B_i are spread over at least three bins. In this last case, since the number of colors in a bin is limited to k, a large number of color classes with multiple parts would result in a large number of bins. We next split the set of possible solutions into these three options, we consider each option, and prove the claim that each option leads to a large number of packed bins.

First, assume that there are at least $\frac{n}{5}$ full color classes among $B_1, \ldots, B_n$. Then, there are at least $2n(k-1) + \frac{n}{5}$ full color classes in total (as A_i is always a full color class), and therefore OPT uses at least $\frac{2n(k-1)+\frac{n}{5}}{k-1} = 2n + \frac{n}{5(k-1)} \geq 2n + \frac{n}{5k}$ bins. Therefore, in the remainder of the proof we can assume without loss of generality that the number of full color classes among $B_1, \ldots, B_n$ is at most $\frac{n}{5}$.

We next note that if there exists a partial color class B_i such that the elements of B_i are packed in exactly two bins of OPT, and each of these two bins has $k-1$ full color classes (in addition to the items of B_i), then the PARTITION instance is feasible, since the space left for items of B_i in each one of the two bins is exactly $\frac{1}{2}$. Therefore, if we have a partial color class B_i, then its elements are packed in at least three bins or at least one of the bins (that contains at least one element of B_i) contains at most $k-2$ full color classes.

We next consider the case where there are at least $\frac{2n}{5}$ partial color classes such that each of these classes is partitioned into at least three bins of OPT (recall that we assume that there are at most $\frac{n}{5}$ full color classes among $B_1, \ldots, B_n$ and therefore there are at least $\frac{4n}{5}$ partial colors). We let a *part* be a maximal subset of a color class that is packed by OPT into a common bin. Then, the number of parts is at least $2n(k-1) + \frac{n}{5} + 2\frac{2n}{5} + 3\frac{2n}{5} = 2nk + \frac{n}{5}$. Since each bin in OPT has at most k parts, we conclude that the number of bins is at least $2n + \frac{n}{5k}$. Therefore, in the remainder of the proof we can assume that there are at most $\frac{2n}{5}$ partial color classes such that each of them is partitioned into at least three bins in OPT. By our assumptions we conclude that there are at least $\frac{2n}{5}$ color classes among $B_1, B_2, \ldots, B_n$ that OPT packs each of them into exactly two bins. Without loss of generality, assume that OPT packs each one of the color classes $B_1, B_2 \ldots, B_{2n/5}$ into exactly two bins. We define the *main bin* of a color class B_i for $i \in \{1, \ldots, \frac{2n}{5}\}$ as a bin that contains a total size more than $\frac{1}{2}$ of the items of this color class. Such a bin must exist since the items of B_i are partitioned into exactly two bins, and cannot be shared equally due to the infeasibility of the PARTITION instance. The subset of items of a color B_i which are packed in the main bin of this color class are called the *main part* of this color class.

For the i-th bin packed by OPT, let t_i denote the number of color classes for which their main part is packed into this bin. We have $\sum_{i=1}^{\text{OPT}} t_i \geq \frac{2n}{5}$.

If a bin has t main parts of partial colors, then in the case $t \geq 2$ it has at most $k - t$ full colors, since it cannot have items of more than k color classes. If $t = 1$, it can have at most $k - 2$ full colors, due to space constraints. In both cases, the number of full colors is at most $k - 1 - \frac{t}{2}$. If $t = 0$, then the number of full colors is again at most $k - 1 = k - 1 - \frac{t}{2}$. The number of full colors is at least $2n(k-1)$, so we have $2n(k-1) \leq \sum_{i=1}^{\text{OPT}} \left(k - 1 - \frac{t_i}{2}\right) = \text{OPT}(k-1) - \frac{n}{5}$. Rearranging the last inequality gives $\text{OPT} \geq 2n + \frac{n}{5(k-1)} \geq 2n + \frac{n}{5k}$. □

4 Lower bounds for online algorithms and $\mathbf{k = 2}$

In this section we provide lower bounds on the performance guarantees of specific algorithms as well as a lower bound on the performance of any online algorithm. We focus on the case $k = 2$.

4.1 Lower bound of 2 on the competitive ratio of FF when applied to equal size items

We note that Shachnai and Tamir [16] proved a lower bound of $\frac{3}{2}$ on the competitive ratio of any online algorithm for the case of equal size items and $k = 2$. Their upper bound of 2 for the competitive ratio of FF applies for all values of k (for equal sized items). We show in what follows that their upper bound is tight already for the case $k = 2$.

Theorem 3 *The competitive ratio of* FF *for the case $k = 2$ and equal sized items is exactly 2.*

Proof The upper bound follows from [16], and we next prove the lower bound. Let N be a large integer. The (common) size of the items is $\varepsilon = \frac{1}{32N}$.

In the instance there are $2N$ color classes. For $n = 1, 2, \ldots, N$, the total size of the elements of color class X_n is $\frac{1}{2} + 2n\varepsilon$, and the total size of elements of color class Y_n is $\frac{1}{2} - 2(n-1)\varepsilon$. The items arrive according to the following order. First the items of $X_1 \cup Y_1$, then the items of $X_2 \cup Y_2$ and so on. The order of the items satisfies also the following condition. For every n, the last pair of items of $X_n \cup Y_n$ has one

item from each of these two color classes. This order of the items ensures that for each value of n, the items of $X_n \cup Y_n$ will be packed in exactly two bins (that will not be used by other items). Therefore, FF uses two bins for every value of n, and in total, a set of $2N$ bins.

An optimal solution packs the items of $X_n \cup Y_{n+1}$ in a bin (for $n = 1, 2, \ldots, N-1$), since the total size of these items, for the relevant values of n, is exactly one. One other bin is used for the items of Y_1, and one additional bin for the items of X_N. Therefore, the total cost of the optimal solution is $N+1$. Hence, the competitive ratio of FF for the case of $k = 2$ and equal sized items is at least $\frac{2N}{N+1}$, and this lower bound approaches 2 as N tends to infinity. □

Remark 1 *The same construction of lower bound of 2 holds also for* CSFF *(for the case of $k = 2$ and equal sized items).*

Remark 2 *Similar constructions give a lower bound of 2 for* FF *and* CSFF *for every fixed value of $k \geq 2$ and equal sized items.*

4.2 Lower bound of $9/4$ on the competitive ratio of CSFF for the case $k = 2$

In this section we show that application of the same algorithm on non-identical sized items increases the competitive ratio strictly above 2. We show that the lower bound of 2 shown in the previous section on the competitive ratio of CSFF when $k = 2$ can be increased above 2, if items are not necessarily all of the same size.

Theorem 4 *The competitive ratio of* CSFF *when applied to* CCBP *with $k = 2$ is at least $\frac{9}{4}$.*

Proof Let N be a large even integer. Let $\varepsilon = \frac{1}{2^{2N+3}}$. The instance is defined as follows. The first N items are of different colors and each of them has size ε. These N items are denoted by $A_1, \ldots, A_N$. The colors of these items are never used again. For $n = 1, 2, \ldots, N$ (starting with $n = 1$) the next items are as follows: there are items of size ε from two color classes B_n and C_n such that the total size of the items in B_n is $\frac{1}{2} - (2^n + 1)\varepsilon$ and the items of color C_n have total size $(2^{n+1} + 1)\varepsilon$. Then, there are the following additional items: item X_n of size $\frac{1}{2} + 2^n\varepsilon$ and of color B_n, item Y_n of size $\frac{1}{2} + 2^n\varepsilon$ and of color C_n, and item Z_n of size $\frac{1}{2} - 2^{n-1}\varepsilon$ and color C_n. This completes the subsequence corresponding to the value of n. All items for a given value of n are given consecutively, and after that n is increased by 1. The process is repeated until $n = N + 1$ and stops after the items defined for $n = N$.

CSFF uses $\frac{N}{2}$ bins to pack the items $A_1, A_2, \ldots, A_N$. Afterwards, for each value of n, the color classes B_n and C_n form a color set, and CSFF uses four bins to pack the items of such a color set. Therefore, the total cost of CSFF is $\frac{9N}{2}$.

To prove the claim it suffices to show a feasible solution which uses $2N+4$ bins. This is done as follows. For each value of $n = 1, 2, \ldots, N$ we pack the color class B_n together with the item A_n using one bin. This is a feasible packing because the total size of the items of color B_n is exactly $\frac{1}{2} - (2^n + 1)\varepsilon + \frac{1}{2} + 2^n\varepsilon = 1 - \varepsilon$ and therefore the total size of the items that we pack into this bin is exactly 1. For each value of $n = 1, 2, \ldots, N-4$ we pack in one bin the items of color C_n with size ε together with Y_n (which is of color C_n as well) and item Z_{n+4} (that has a different color, so in total there are items of exactly two colors in the bin). This bin is feasible with respect to total size, since the total size of the items in this bin is exactly $(2^{n+1} + 1)\varepsilon + \frac{1}{2} + 2^n\varepsilon + \frac{1}{2} - 2^{(n+4)-1}\varepsilon < 1$. We pack items Z_1, Z_2, Z_3, Z_4 in four dedicated bins, and we pack the remaining items of color classes $C_N, C_{N-1}, C_{N-2}, C_{N-3}$ (the small items and the Y_i items) using four additional bins (one bin for each such color class, that would contain one larger item and all smaller items of the same color), clearly these bins are feasible, and the claim follows. □

4.3 Lower bound against any online algorithm for the case of $\mathrm{k} = 2$

For arbitrary item sizes and $k = 2$ there are two previously known lower bounds. The first one follows from classical bin packing. Since CCBP is its generalization, the lower bound for the former problem holds also for CCBP. This gives a lower bound of 1.54014 due to van Vliet [18]. The other lower bound is of $\frac{3}{2}$ because CCBP with arbitrary sizes generalizes the problem with identical sizes, and therefore the lower bound of [16] holds also for this problem. We next show how to combine the methods of the two lower bounds to obtain an improved result for CCBP with arbitrary sized items and $k = 2$.

Theorem 5 *Any online algorithm for* CCBP *with* $k = 2$ *has a competitive ratio of at least 1.5652.*

Proof Let $\varepsilon > 0$ be a small enough number ($\varepsilon \leq \frac{1}{1000}$ is sufficiently small), and let N be a large integer. The sequence consists of at most four steps (sub-sequences), where a set of items is introduced at each step, and then depending on the output of the algorithm at this time, the adversary decides whether it would continue to the next step (otherwise, the sequence ends). The first step has $2N$ items each of size ε. Moreover, each of them has a different color from the set $\{1, 2, \ldots, 2N\}$. All additional items, of the second, third and fourth steps will share a new color, $2N + 1$. The second step consists of $12N$ items, each of which has size $\frac{1}{7} + \varepsilon$. These last items are called items of type A. The third step consists of $12N$ items, each of which has size $\frac{1}{3} + \varepsilon$, and these items are called items of type B. The fourth step consists of $12N$ items, each of which has a size $\frac{1}{2} + \varepsilon$, and these items are called items of type C.

We next find the cost of the optimal solution at the end of each step. After the first step, we have $\mathrm{OPT} = N$, since any pair of items can be packed in one bin. After the second step, each set of six items of type A can be packed together in a bin. Moreover, since all items of type A have the same color, any such set can be combined with an additional item of size ε. So it is possible to pack all items in $2N$ bins, and so $\mathrm{OPT} = 2N$. After the third step we have $\mathrm{OPT} = 6N$ because we can pack two items of type B with two items of type A and an additional item of size ε in one bin (such an additional item is packed only in $2N$ of the $6N$ bins). At the end of the fourth step $\mathrm{OPT} = 12N$ because a bin can accommodate three items of the three types, A, B and C, and an additional item of size ε in $2N$ of the bins.

We denote by X_0 the number of bins with two items at the end of the first step (so at the end of this step there are exactly $2N - 2X_0$ bins with one item in each, and these are the bins that can be used by the next steps). We use patterns in the analysis. A *pattern* is a vector, consisting of three components, which corresponds to a way that a bin is packed, with respect to the number of items of types A, B and C, which it contains. A bin corresponds to a *pattern* $p = (p_1, p_2, p_3)$, if it has p_1 items of type A, p_2 items of type B and p_3 items of type C. A pattern may contain non-zero components corresponding to numbers of items of types B and C. In such a case, the spaces allocated to such items (that do not necessarily arrive) remain empty if the sequence is stopped before their arrival. We say that a pattern $p = (p_1, p_2, p_3)$ dominates a pattern $q = (q_1, q_2, q_3)$ if they correspond to bins opened for the same step (i.e., the smallest non-zero component is the same component in the two patterns) and for every $i = 1, 2, 3$, $q_i \leq p_i$ holds. A pattern p which no pattern dominates it (except for p), is called dominant. Since we only use inequalities in the linear program which counts the number of items (i.e., it is possible for an optimal packing to add items in order to get a sufficient number of items for a given pattern), it is never profitable for an offline algorithm to use a pattern which is not dominant, so only the dominant patterns are listed in the sequel. In addition, we remove the pattern $(4, 1, 0)$, since the variables are not necessarily integral, and a bin packed according to this pattern can be replaced by two halves of bins packed according to $(6, 0, 0)$ and $(2, 2, 0)$.

The following variables count the number of bins with a given packing pattern of items of types A, B and C. We denote by X_1 the number of bins that the online algorithm packs six items of type A, i.e., bins

packed according to the pattern $(6,0,0)$. We denote by X_2 the number of bins that the online algorithm packs three items of type A and an item of type C, i.e., bins packed according to the pattern $(3,0,1)$. We denote by X_3 the number of bins that the online algorithm packs with two items of type A and two items of type B, i.e., bins packed according to the pattern $(2,2,0)$. We denote by X_4 the number of bins that the online algorithm packs with one item of type A, one item of type B and one item of type C, i.e., bins packed according to the pattern $(1,1,1)$. We denote by X_5 the number of bins that the online algorithm packs with two items of type B, i.e., bins packed according to the pattern $(0,2,0)$. We denote by X_6 the number of bins that the online algorithm packs with one item of type B and one item of type C, i.e., bins packed according to the pattern $(0,1,1)$. Finally, we denote by X_7 the number of bins that the online algorithm packs with one item of type C, i.e., bins packed according to the pattern $(0,0,1)$. The four first packing patterns correspond to bins that are used already after the type A items arrive. The next two patterns correspond to bins that are used if the type B items arrive. The last pattern corresponds to bins that are used only after the type C items arrive.

By a counting argument, the following three constraints must hold: $6X_1 + 3X_2 + 2X_3 + X_4 \geq 12N$ (counting the number of items of type A), $2X_3+X_4+2X_5+X_6 \geq 12N$ (type B), and $X_2+X_4+X_6+X_7 \geq 12N$ (type C). We denote by R the competitive ratio of the online algorithm. Then, the following four additional constraints must hold: $2N - X_0 \leq RN$ (comparing the cost of the algorithm and the optimal algorithm at the end of the first step), $X_0 + X_1 + X_2 + X_3 + X_4 \leq R \cdot 2N$ (comparing the cost of the algorithm and the optimal algorithm at the end of the second step), $X_0+X_1+X_2+X_3+X_4+X_5+X_6 \leq R \cdot 6N$ (comparing the cost of the algorithm and the optimal algorithm at the end of the third step), and $X_0+X_1+X_2+X_3+X_4+X_5+X_6+X_7 \leq R\cdot 12N$ (comparing the cost of the algorithm and the optimal algorithm at the end of the fourth step). In addition to these constraints all variables $(X_0, X_1, \ldots, X_7, R)$ need to be non-negative and we would like to minimize R. Letting $x_i = \frac{X_i}{N}$, and $x_8 = \mathcal{R}$, We get the following linear program such that its optimum is clearly a lower bound on the competitive ratio of any online algorithm.

$$
\begin{array}{lrcll}
\min & x_8 & & & \\
s.t. & x_i & \geq & 0 & 0 \leq i \leq 8 \\
& 6x_1 + 3x_2 + 2x_3 + x_4 & \geq & 12 & \\
& 2x_3 + x_4 + 2x_5 + x_6 & \geq & 12 & \\
& x_2 + x_4 + x_6 + x_7 & \geq & 12 & \\
& x_0 + x_8 & \geq & 2 & \\
& -x_0 - x_1 - x_2 - x_3 - x_4 + 2x_8 & \geq & 0 & \\
& -x_0 - x_1 - x_2 - x_3 - x_4 - x_5 - x_6 + 6x_8 & \geq & 0 & \\
& -x_0 - x_1 - x_2 - x_3 - x_4 - x_5 - x_6 - x_7 + 12x_8 & \geq & 0 &
\end{array}
$$

Using a standard LP-solver we get that the optimum is approximately 1.565217. □

5 Online algorithms

In this section, we analyze online algorithms for CCBP. For large values of k, we would like to use modifications of the algorithm HARMONIC [11]. This algorithm partitions online the input into independent streams,

and packs each such stream in separate bins. The streams are of items of size in intervals of the type $(\frac{1}{i+1}, \frac{1}{i}]$ for $1 \leq i \leq M$, and there is one last interval of items no larger than $\frac{1}{M+1}$. We refer to this type of packing as harmonic packing. For intervals of relatively large items, the constraint on the number of colors in a bin is satisfied immediately. This property can be exploited for the design of algorithms for CCBP. Modifications of HARMONIC [11, 12, 13], allow to combine several types of large items. As long as such a bin, where different types of items are combined, contains a small number of items, it can also satisfy the constraint on the number of colors, therefore, such algorithms are useful for our purposes as well. One major difference with previous algorithms is that we sometimes combine items of the smallest class of items, i.e., of the last interval, with very large items. The reason for this is that unlike standard bin packing, where a very small item does not occupy much space, and small items are packed very densely using NF, here any packing of small items (including optimal packings of such items) can result in very empty bins, containing k items of different colors. To deal with this, we allow to combine small items with a large item, of a color in the same color set, but only if it has a very specific size.

Our plan is to consider separately the small values of k, and then to consider larger values of k. We first show in Section 5.1 that CSFF is $(2 + \frac{k-1}{k})$-competitive algorithm. This ratio is small for $k = 2$ and it is in fact the best algorithm we present for the case $k = 2$. For $k = 3$ we present an improved algorithm in Section 5.2, which is shown to be $\frac{107}{42} \approx 2.547619$-competitive. Afterwards we show a general reduction from online bin packing algorithms which results a good competitive ratio for large values of k.

To prove upper bounds on the competitive ratio, we use the technique of weighting functions. This technique was originally introduced by Ullman [17]. We use the following theorem, see Seiden [13].

Theorem 6 *Consider a bin packing algorithm. Let w_1, w_2 be two weight measures $w_i : (0, 1] \rightarrow \mathbb{R}_0^+$. Assume that for every input, there exists a value i ($i = 1$ or $i = 2$) such that the number of bins used by the algorithm ALG is at most $X_i(\sigma) + c$ for some constant c, where $X_i(\sigma)$ is the sum of weights of all items in the sequence according to weight measure w_i. Denote by $W_i > 0$ the supremum amount of weight that can be packed into a single bin according to measure w_i ($i = 1, 2$). Then the competitive ratio of the algorithm is at most $\max(W_1, W_2)$.*

Proof Given an input, let i be the value that satisfies the theorem for this input. Clearly $OPT(\sigma) \geq \frac{X_i(\sigma)}{W_i}$. We get $ALG \leq X_i(\sigma) + c \leq W_i OPT + c$. □

5.1 Analysis of CSFF

Theorem 7 CSFF *is a $(2 + \frac{k-1}{k})$-competitive algorithm.*

Proof We assign weights to items as follows. Consider a color class C, where the total size of items in this class is S_C, then the total weight that we assign to C is $\max\{2S_C, \frac{1}{k}\}$. This total weight is split among the items in C according to the specific size of each item, and in proportion to its size. That is, an item with size a in C has a weight of $\frac{a}{S_C} \cdot \max\{2S_C, \frac{1}{k}\}$.

We first argue that the total weight of the items plus 1 is an upper bound on the cost of CSFF. To see this, consider a color set which is not the last color set ever defined, for which the total size of items is $\mathcal{S}$. If this color set results in a single bin, then still each color class in this color set has a total weight of at least $\frac{1}{k}$, and in total it is at least 1. Now assume that at least two bins are used for this color set. By definition of weight, the total weight of the items in this color set is at least $2\mathcal{S}$. If FF, executed on a set of items, results in at least two bins, then the sum of item sizes in every (not necessarily consecutive) pair of bins is larger than 1. Let λ be the cost of FF on the color set. Consider all pairs of bins, and let η_i denote

the total size packed into the i-th bin. Then $\sum_{1\leq i\leq \lambda-1}\sum_{i<j\leq\lambda}(\eta_i+\eta_j) > \frac{\lambda(\lambda-1)}{2}$. On the other hand, $\sum_{1\leq i\leq \lambda-1}\sum_{i<j\leq\lambda}(\eta_i+\eta_j) = (\lambda-1)\sum_{1\leq i\leq\lambda}\eta_i = (\lambda-1)\mathcal{S}$. Thus $\lambda < 2\mathcal{S}$ and so $2\mathcal{S}$ is a an upper bound on the total number of bins used by FF when applied to item set with items of total size $\mathcal{S}$. The very last color set is the only one that may contain less than k colors, and for that color set the total weight may be smaller than 1, if it results in a single bin. If it results in at least two bins, then the proof above hold for this color set as well. We conclude that the total weight of the items plus 1 is an upper bound on the cost of CSFF.

To prove the theorem, it suffices to show that when we consider one bin of OPT, the total weight of the items in this bin is at most $2+\frac{k-1}{k}$. To see this we note that for a color class C whose total size S is at most $\frac{1}{2k}$ (called a small color class), the total weight of the items in C is $\frac{1}{k} \leq 2S+\frac{1}{k}$, and for a color class C whose total size S is larger than $\frac{1}{2k}$, the total weight of the items in C is at most $2S$. When we fix a bin of OPT, the total weight of the items in this bin is at most twice their total size plus $\frac{1}{k}$ times the number of small color classes in this bin. There are two cases: if all color classes of items in this bin are small, then the total weight of the items in these color classes is at most 1, and the claim holds. Otherwise, there is at least one large color class used by OPT in this bin, and in this case the total weight is at most $2+\frac{k-1}{k}$, as we claimed. □

5.2 An improved algorithm for the case $k \geq 3$

Our algorithm is based on online partitioning of the items according to size. The algorithm is defined for any $k \geq 3$. The algorithm is based on the idea of the algorithm REFINED HARMONIC of Lee and Lee [11] and MODIFIED HARMONIC of Ramanan et al. [12] for the packing of relatively large items, combined with CSFF for the packing of small items. That is, roughly speaking, items are partitioned into large items, of size larger than $\frac{1}{4}$, and small items, of size at most $\frac{1}{4}$. The large items are partitioned into medium items, of size at most $\frac{11}{18}$, which are packed using methods similar to those of REFINED HARMONIC [11]. The largest large items, of size more than $\frac{2}{3}$, are packed in dedicated bins. Small items are usually packed using CSFF, with some exceptions. A small item which was not assigned to a color set yet can be combined with an item of the same color, and size in $(\frac{11}{18},\frac{2}{3}]$, in a bin, possibly with some additional small items of the same color. In such a case, the color of the small item is not assigned to a color set at this time. The algorithm tries to combine such small items with a large item even if their color was assigned to a color set, and in addition, if the large item is the one arriving later, the algorithm tries to push it into the very first bin of the color set, to be combined with the small items there. Naturally, this is possible only if the bin contains a small total size of items. In what follows, we give a complete definition of the algorithm.

Unlike the standard analysis of such algorithms, since we use CSFF for some items, we define weights that in some cases depend on the specific packing rather than depending just on the size of items. Moreover, our algorithm tries to pack together the smallest items and the largest items in common bins. This last property of the algorithm is non-standard for online bin packing algorithms.

We define several intervals, where the input items are partitioned into several independent streams according to size. Items of size in an interval I will be called I-items.

There are six intervals, J_1, J_2, J_3, J_4, J_a, and J_b, where $J_1 \cup J_2 \cup J_3 \cup J_4 \cup J_a \cup J_b = [0,1]$.

The definition of the intervals is as follows:

- $J_1 = (\frac{11}{18} \approx 0.6111, 1]$.
- $J_a = (\frac{1}{2}, \frac{11}{18}]$.

- $J_2 = (\frac{7}{18} \approx 0.38889, \frac{1}{2}]$.
- $J_b = (\frac{1}{3}, \frac{7}{18}]$.
- $J_3 = (\frac{1}{4}, \frac{1}{3}]$.
- $J_4 = [0, \frac{1}{4}]$.

The bins created by the algorithm will be of the following types.

- J_1-bins, each of which will contain a single J_1-item, possibly with some J_4-items, which belong to the color set of the color of J_1, in the packing of the J_4-items.
- J_2-bins, each of which will contain two J_2-items, except for possibly the last such bin, which may contain one item.
- J_3-bins, each of which will contain three J_3-items, except for possibly the last such bin, which may contain one or two items.
- J_{bb}-bins, each of which contains two J_b-items, except for possibly the last such bin, which may contain one item.
- J_a bins, each of which contains a single J_a-item.
- J_{ab}-bins, each of which contains a J_a-item and a J_b-item.
- J_{a3}-bins, each of which contains a J_a-item and a J_3-item.
- J_{bs}-bins, each of which contains a single J_b-item.
- J_{3s}-bins, each of which contains a single J_3-item.
- J_4-bins, which are packed using CSFF.

We next define algorithm A_3. The algorithm uses variables N_x for $x = b$ and $x = 3$, to count the number of items which arrived so far with size in the interval J_x. We initialize $N_b = N_3 = 0$. We let i denote the index of the new item to be packed, and initialize $i = 0$.

If all items were packed, then stop. Otherwise, apply $i = i + 1$. Let i be the next item to be packed of size s_i. If i is a J_b-item then $N_b = N_b + 1$. If i is a J_3-item then $N_3 = N_3 + 1$.

- If $s_i \in J_4$, then act as follows.
  - If there exists a J_1-bin with an item of the same color as item i, of size no larger than $\frac{2}{3}$, such that item i can be packed there without violating the constraint on the total size, then pack item i into such a bin.
  - Otherwise, pack item i using CSFF into a J_4-bin.
- If $s_i \in J_3$, then act as follows.
  - First assume that N_3 is not divisible by 16. If there exists a J_3-bin with less than three items, then pack item i in this bin. Otherwise (i.e., there is no such J_3-bin), open a new J_3-bin and pack the item there.

  - Now assume that N_3 is divisible by 16. If there exists a J_a-bin, then pack item i in this bin (which becomes a J_{a3}-bin). Otherwise, open a new J_{3s}-bin and pack the item there.

- If $s_i \in J_b$, act as follows.
  - First assume that N_b is not divisible by 7. If there exists a J_{bb}-bin with only one item, then pack item i in this bin. Otherwise, open a new J_{bb}-bin and pack the item there.
  - Now assume that N_b is divisible by 7. If there exists a J_a-bin, then pack item i in this bin (which becomes a J_{ab}-bin). Otherwise, open a new J_{bs}-bin and pack the item there.

- If $s_i \in J_2$, if there exists a J_2-bin with exactly one item, then pack item i in this bin. Otherwise, open a new J_2-bin and pack the item there.

- If $s_i \in J_a$, if there exists a J_{bs}-bin or a J_{3s}-bin, choose such a bin and pack the item. In this case the bin where the item is packed becomes a J_{ab}-bin, if it was a J_{bs}-bin, and otherwise a J_{a3}-bin. Otherwise, open a new J_a-bin and pack the item there.

- If $s_i \in J_1$, then act as follows.
  - If all the following conditions hold: $s_i \leq \frac{2}{3}$, the color of item i already belongs to a color set of J_4-items, and item i can be packed into the first bin ever created for this color set, then pack it there. This bin becomes a J_1-bin.
  - Otherwise, open a new J_1-bin and pack the item there.

Note that in the packing of J_4-items, if a J_4-item is combined with a J_1-item in a bin (possibly containing additional J_4-items of the same color), then the J_4-item is not assigned to a color set, unless it already belongs to a color set. Thus, either it is never assigned to a color set, in which case all the J_4-items of this color are combined with J_1-items in the same bins, or it is assigned to a color set later, which means that for at least one J_4-item of this color, it was no longer possible to combine it into a bin containing a J_1-item of the same color.

On the other hand, if it is the J_1-item which is inserted into a J_4-bin, then it must be the case that its color was previously assigned to a color set.

It is not difficult to see that the properties above regarding the contents of each type of bin are satisfied. In addition, if there is at least one J_{3s}-bins or a J_{bs}-bin, then there is no J_a-bin, and vice versa, a J_a-bin excludes the existence of J_{3s}-bins and of J_{bs}-bins. The reason for this is that a J_{3s}-bin or a J_{bs}-bin is created only if no J_a-bin exists, and on the other hand, a J_a-bin is created if no J_{3s}-bins exist and no J_{bs} bins exist.

Theorem 8 *Algorithm A_3 is $\frac{107}{42} \approx 2.547619$-competitive for the case $k = 3$, and algorithm A_3 has a competitive ratio of at most $\frac{37}{14} - \frac{2}{7k}$ for larger values of k.*

Proof We first note that the algorithm returns a feasible solution. This is so because each bin dedicated to items, which are not J_4-items, has at most three items and therefore it satisfies the constraint on the number of color classes represented in the bin, and its total size is at most 1 (by our partitioning of the large items, and the definitions of bin types). The small items are packed in feasible bins by the feasibility of CSFF, and since we pack a J_1-item in a bin with small items only after checking that the total size will not exceed 1, and that its color is already assigned to this color set.

It remains to prove the competitive ratio. To do so we design two weight functions. By the definition of the algorithm, there are several types of bins in the solution returned by the algorithm. In addition to the first nine types defined above, we consider separately bins packed by CSFF which contain total size more than $\frac{3}{4}$, and bins whose total size of items is at most $\frac{3}{4}$, which are two sub-types of bins of the tenth type (i.e., of J_4-bins).

Recall that there is at most one bin of each of the following types of bins: J_{bb}-bins, J_2-bins, and J_3-bins, which does not contain the complete number of items that this bin should contain (which is two for the first two types, and three for the third type). Furthermore, since CSFF is applied on items of size at most $\frac{1}{4}$, there is at most one bin with a total size of items at most $\frac{3}{4}$, given as an output of CSFF, for each color set. Finally, we need to consider separately the case where there is at least one J_a-bin (but no J_{3s}-bins or J_{bs}-bins), and the case where there are no J_a-bins.

We use one weight function, w_1, for the case that there are no J_a-bins, and a second weight function, w_2, is for the case that such bins may exist.

We assign weights to items as follows. For an item i of size s_i, we have the following cases. Note that the weights are a function of the sizes of items. The case of J_4-items is discussed below, and in that case, the weight is based not only on the size of an item, but also on the resulting packing (unlike the analysis of the algorithms REFINED HARMONIC [11], MODIFIED HARMONIC [12], and HARMONIC++ [13]).

- If $s_i \in J_1$, then $w_1(s_i) = w_2(s_i) = 1$.
- If $s_i \in J_a$, then $w_1(s_i) = 0$ and $w_2(s_i) = 1$.
- If $s_i \in J_2$, then $w_1(s_i) = w_2(s_i) = \frac{1}{2}$.
- If $s_i \in J_b$, then $w_1(s_i) = \frac{4}{7}$ and $w_2(s_i) = \frac{3}{7}$.
- If $s_i \in J_3$, then $w_1(s_i) = \frac{3}{8}$ and $w_2(s_i) = \frac{5}{16}$.

Let M_x denote the final number of J_x-bins for all types of bins, and let N_x denote the final number of J_x-items for all types of items.

Lemma 2 *If the function w_1 is used, i.e., $M_a = 0$, then we have,*

$$M_1 = \sum_{i \in J_1} w_1(s_i),$$

$$M_2 \leq \sum_{i \in J_2} w_1(s_i) + \frac{1}{2}$$

$$M_3 + M_{a3} + M_{3s} \leq \sum_{i \in J_3} w_1(s_i) + \frac{2}{3}$$

$$M_{bs} + M_{bb} + M_{ab} \leq \sum_{i \in J_b} w_1(s_i) + \frac{1}{2}$$

Proof The first claim holds since $\sum_{i \in J_1} w_1(s_i) = N_1$.

The second claim holds since $\sum_{i \in J_2} w_1(s_i) = \frac{N_2}{2}$, and $M_2 = \lceil \frac{N_2}{2} \rceil$.

To prove the third claim, note that we have $M_3 = \lceil \frac{N_3 - \lfloor \frac{N_3}{16} \rfloor}{3} \rceil \leq \frac{N_3 - \lfloor \frac{N_3}{16} \rfloor}{3} + \frac{2}{3} \leq \frac{5}{16}N_3 + \frac{2}{3}$, and $M_{a3} + M_{3s} = \lfloor \frac{N_3}{16} \rfloor \leq \frac{N_3}{16}$. Since $\sum_{i \in J_3} w_1(s_i) = \frac{3}{8}N_3$, the claim holds.

To prove the fourth claim, note that we have $M_{bb} = \lceil \frac{N_b - \lfloor \frac{N_b}{7} \rfloor}{2} \rceil \leq \frac{N_b - \lfloor \frac{N_b}{7} \rfloor}{2} + \frac{1}{2} \leq \frac{3}{7}N_b + \frac{1}{2}$, and $M_{bs} + M_{ab} = \lfloor \frac{N_b}{7} \rfloor \leq \frac{N_b}{7}$. Since $\sum_{i \in J_b} w_1(s_i) = \frac{4}{7}N_b$, the claim holds. □

Lemma 3 *If the function w_2 is used, i.e., $M_{3s} = M_{bs} = 0$, then we have,*

$$M_1 = \sum_{i \in J_1} w_2(s_i),$$

$$M_2 \leq \sum_{i \in J_2} w_2(s_i) + \frac{1}{2},$$

$$M_3 \leq \sum_{i \in J_3} w_2(s_i) + \frac{2}{3}$$

$$M_{bb} \leq \sum_{i \in J_b} w_2(s_i) + \frac{1}{2},$$

$$M_{ab} + M_{a3} + M_a \leq \sum_{i \in J_a} w_2(s_i),$$

Proof The first two claims are identical to those of the previous lemma, since w_1 and w_2 are defined identically for these two cases.

To prove the third claim, we use again $M_3 \leq \frac{5}{16}N_3 + \frac{2}{3}$. Since $\sum_{i \in J_3} w_2(s_i) = \frac{5}{16}N_3$, the claim holds.

To prove the fourth claim, we use again $M_{bb} \leq \frac{3}{7}N_b + \frac{1}{2}$. Since $\sum_{i \in J_b} w_2(s_i) = \frac{3}{7}N_b$, the claim holds.

To prove the fifth claim, we have $M_{ab} + M_{a3} + M_a = N_a$. Since $\sum_{i \in J_a} w_2(s_i) = N_a$, the claim holds. □

We next define weights for J_4-items. These items receive weights according to a function w defined on the items, and we let $w_1^i = w(i)$ and $w_2^i = w(i)$ for such items, unlike other items for which $w_1^i = w_1(s_i)$ and $w_2^i = w_2(s_i)$.

We use two non-negative constants $\gamma = \frac{9}{7}$ and $\delta = \frac{5}{7k}$. Consider an item of size a (where $a \leq \frac{1}{4}$) that belongs to a color class which contains ℓ items. Recall that CSFF partitions color classes into color sets, each of which is packed using FF. There may be some color classes, for which there exists at least one J_4-item, but they are never assigned to color sets, since all their items are combined with J_1-items in bins. In this case we say that the color set contains a single color.

Consider all J_4-items of colors which belong to one color set. If there are no J_4-bins of this color set (which can happen if the color set only has one color, but can also happen if a J_4-bin becomes a J_1-bin, and no additional bins are opened for this color set), then all weights of J_4-items of this color set are defined to be zero. Otherwise, there are two options. If the total size of these J_4-items, including items of this color set which were packed into J_1-bins, is at least $\frac{1}{3}$, then the weight of this item is defined as $\gamma \cdot a + \frac{\delta}{\ell}$. This type of weights of J_4-items are called *size based weights*. Otherwise, it is defined as $\frac{1}{k\ell}$. This type of weights of J_4-items are called *size independent weights*. The following claim follows directly from the last definition.

Claim 1 *Consider a color class, where the total size of items is S. Then the total weight of items in this class is $\gamma \cdot S + \delta$ if the item weights are defined to be size based weights, the total weight is $\frac{1}{k}$, if the item weights are size independent, and zero, if all weights were defined to be zero.*

Lemma 4 *Consider a color set of k colors, for which* CSFF *uses m J_4-bins. Then the total weight of items in this color set is at least m. For a color set containing less than k colors, the total weight is at least $m-1$.*

Proof We first consider a color set of k colors. If $m = 0$, the claim is trivial since the weights are always non-negative.

If the weight of every item of the color set was defined to be $\frac{1}{k\ell}$, where ℓ is the number of items of its color class, then the total size of the items of the color set is less than $\frac{1}{3}$, and hence $m = 1$. The total weight of items of each class is $\frac{1}{k}$, and since there are k color classes in the color set, the claim follows.

If $m = 1$ but the weight of some item of size a of this color set was set to $\gamma \cdot a + \frac{\delta}{\ell}$, then the total size of all J_4-items of colors in this color set is at least $\frac{1}{3}$ (possibly some such items are not packed in J_4-bins but in J_1-bins). The total weight of all these items is at least $\gamma \cdot \frac{1}{3} + k\delta = \frac{8}{7} > 1$.

If $m \geq 2$, consider the m J_4-bins used for the color set, and all J_4-items of colors in this color set, including items which are packed in J_1-bins, if exist. We next argue that the total size of these items is at least $\frac{4}{5}(m-2)+1$. If each bin, possibly except for the last one, contains total size of items at least $\frac{4}{5}$, then the total size of items is at least $\frac{4}{5}(m-2)+1$, since the total size of items in the last two bins together is at least 1 (where the last property is a property of FF). Otherwise, consider the first bin that has total size less than $\frac{4}{5}$. By definition, this is not the last bin. Every additional item which is packed in later bins has size more than $\frac{1}{5}$ (but no more than $\frac{1}{4}$, since it is a J_4-item), thus each bin (except for possibly the last one) contains exactly four such items, and thus contains total size more than $\frac{4}{5}$. This means again that all bins, except for two bins, have a total size of items which is at least $\frac{4}{5}$, and the sum of items sizes in the remaining two bins (the one with a total size smaller than $\frac{4}{5}$ and the last bin) together is at least 1. Hence, we showed that the total size of the J_4-items in this color set is at least $\frac{4}{5}(m-2)+1$. Since $m \geq 2$, the total weight of all items of the color set is γ times their total size, plus $k\delta$.

We thus need to find a lower bound on $\gamma\mathcal{S} + k\delta$, where $\mathcal{S}$ is the sum of items sizes in the color set (i.e., the total size of all items of all color classes of this color set) and to show that it is at least m. Indeed we have $\gamma\mathcal{S} + k\delta \geq \gamma((m-2)\frac{4}{5}+1) + k\delta = \frac{9}{7}(\frac{4}{5}m - \frac{3}{5}) + \frac{5}{7} \geq m$, which is equivalent to $\frac{m}{35} \geq \frac{2}{35}$ and thus holds for $m \geq 2$.

If the color set has less than k colors, then if $m = 1$ the claim holds. Otherwise, the total size of J_4-items of colors in this color set is at least 1, so we need to find a lower bound on $\gamma\mathcal{S} + \delta > \frac{9}{7}(\frac{4}{5}m - \frac{3}{5}) \geq m - \frac{5}{7} > m - 1$, by the previous case where $m > 1$. □

Note that there is at most one color set with less than k color classes. By Lemmas 2, 3 and 4, we can summarize that in the first case where $M_a = 0$, we have for an input of n items $A_3 < \sum_{i=1}^{n} w_1^i + 3$, and in the second case $A_3 < \sum_{i=1}^{n} w_2^i + 3$. Before we proceed, we show the following lemma.

Lemma 5 *Consider a color set of J_4-items, resulting by at least one J_4-bin. If there is at least one J_1-item of a color in the color set, of size in $(\frac{11}{18}, \frac{2}{3}]$, then the weights of items of this color are defined to be size based.*

Proof Since there is at least one J_4-bin, then the color class contains colors which were assigned to it actively (and not just one color class which was never assigned to a color set).

If there exists a J_1-item which was packed into a J_4-bin of this color set and thus it turned into a J_1-bin, then since an additional J_4-bin of the same color set was opened, then the total size of items of J_4-items of colors in this color set exceeds $\frac{1}{3}$, since the size of the J_1-item does not exceed $\frac{1}{3}$, so a total size of at most $\frac{1}{3}$ would result in a situation where there are no J_4-bins for this color set.

If there exists a J_1-bin which contains some J_4-item of a color of this color set, but this bin was never defined as a J_4-bin (i.e., the J_1-item was the first item packed into this bin) then the color of the J_1-item was assigned to the color set only when some J_4-item could not be packed into the J_1-bin. Since $s_i \leq \frac{2}{3}$, the total size of J_4-items of this color must have exceeded $\frac{1}{3}$.

Otherwise, every J_1-bin, containing a J_1-item of this color set, only contains this item. Consider a specific such J_1-item i. There are two options as for the arrival time of item i.

- If i arrived before the color of i joined the color set. This leads to a contradiction, since at least one J_4-item of the same color as i can fit into the J_1-bin of i, since their total size is at most $\frac{2}{3} + \frac{1}{4} < 1$.
- If i arrived after the color of i was assigned to a color set, since i could not join the first bin of the color set, the total size of items in this bin must be above $\frac{1}{3}$, since $s_i \leq \frac{2}{3}$.

In all cases, the total size of J_4-items of colors in the color set is at least $\frac{1}{3}$, so size based weights are used. □

To apply Theorem 6, we consider a bin packed in a valid manner (by the optimal solution), and compute an upper bound on the total weight of this bin, using each one of the two weight functions. We discuss the J_4-items first.

Lemma 6 *Consider a packed bin and let s be an upper bound on the total size of J_4-items in this bin. The total weight of the J_4-items is at most $\max\{1, \frac{9}{7}s + \frac{7k-2}{7k}\} \leq \frac{9s}{7} + 1$. If there are J_4-items of at most $k-1$ colors of non-zero weights, or if at least one color has sized based weights, or $s \geq \frac{2}{27}$, then the total weight of the J_4-items is at most $\frac{9}{7}s + \frac{7k-2}{7k}$.*

Proof Since the bin may contain items of at most k color classes, let $\tau_1 \geq \tau_2 \geq \cdots \geq \tau_k \geq 0$ be the total sizes of items from these color classes, such that $\sum_{j=1}^{k} \tau_j \leq s$. Our bound on the total weight of items of the i-th color class is at most $\max\{\gamma \cdot \tau_i + \delta, \frac{1}{k}\}$. A clear upper bound on the total weight is $\gamma s + j\delta + \frac{k-j}{k} = \frac{9}{7} \cdot s + \frac{5j}{7k} + \frac{7k-7j}{7k} = \frac{9}{7}s + \frac{7k-2j}{7k}$, where j is the number of color classes whose total weight is size based. If $j = 0$, then we get an alternative upper bound of 1. Otherwise, $j \geq 1$, and the upper bound is $\frac{9s}{7} + \frac{7k-2}{7k}$. If there are at most $k-1$ colors which have non-zero weights, then the bound for the case $j = 0$ becomes $\frac{k-1}{k} < \frac{9}{7}s + \frac{7k-2}{7k}$.

If the total size of J_4-items is at least $\frac{2}{27}$, then we have $\frac{9}{7}s + \frac{7k-2}{7k} = \frac{9}{7}s + 1 - \frac{2}{7k} \geq 1$ for any $k \geq 3$, so it is possible to take into account only the upper bound $\frac{9}{7}s + \frac{7k-2}{7k}$. □

We next consider the possible contents of a bin, in addition to possible J_4-items. For the J_4-items, we consider their supremum size, and calculate their total weight based on this bound. If their total size is smaller, their total weight cannot be larger.

If the bin contains only J_3-items and J_4-items, then the ratio of weight to size (for both w_1 and w_2) of every J_3-item is at most $\frac{3/8}{1/4} = \frac{3}{2}$. Let s be the total size of J_3-items in the bin, thus the total size of J_4-items is at most $1 - s$. The total weight is at most $\frac{3s}{2} + \frac{9(1-s)}{7} + 1 \leq 2.5$.

We consider next the cases in which the bin contains either a J_1-item or a J_a-item.

For w_1, since a J_a-item has a weight of zero, a J_a-item can be neglected. Therefore we assume that there is a J_1-item. This bin may contain at most one additional item of size above $\frac{1}{4}$, due to space constraints.

Specifically, such an item can be a J_b-item or a J_3-item. Therefore, there are three cases for the items of size in $(\frac{1}{4}, 1]$ in the bin, which are:

- The bin contains a J_1-item and a J_b-item. In this case the space remaining for J_4-items is less than $\frac{1}{18}$.

  We compute an upper bound on the total weight of J_4-items. If there are k colors of J_4-items, and all of them have size independent weights, then no J_1-item of size no larger than $\frac{2}{3}$, of any of these colors, could exist contradicting the assumption that a J_1-item and a J_b-item are packed together. By Lemma 5 and Lemma 6, the total weight of the items is at most $\frac{9}{7} \cdot \frac{1}{18} + \frac{7k-2}{7k} + 1 + \frac{4}{7} = \frac{37}{14} - \frac{2}{7k}$.

- The bin contains a J_1-item and a J_3-item. In this case the space remaining for J_4-items is less than $\frac{5}{36}$. This gives a total weight of at most $1 + \frac{3}{8} + \frac{9}{7} \cdot \frac{5}{36} + \frac{7k-2}{7k} < \frac{37}{14} - \frac{2}{7k}$.

- The bin contains a J_1-item. In this case the space remaining for J_4-items is less than $\frac{7}{18}$.

  This gives a total weight of at most $1 + \frac{9}{7} \cdot \frac{7}{18} + \frac{7k-2}{7k} < \frac{37}{14} - \frac{2}{7k}$.

On the other hand, for w_2, it is sufficient to consider a J_a-item, since J_1-items and J_a-items have the same weight. Here there are four cases.

- The bin contains a J_a-item and a J_2-item. In this case the space remaining for J_4-items is less than $\frac{1}{9}$.

  This gives a total weight of at most $1 + \frac{1}{2} + \frac{9}{7} \cdot \frac{1}{9} + \frac{7k-2}{7k} = \frac{37}{14} - \frac{2}{7k}$.

- The bin contains a J_a-item and a J_b-item. In this case the space remaining for J_4-items is less than $\frac{1}{6}$.

  This gives a total weight of at most $1 + \frac{3}{7} + \frac{9}{7} \cdot \frac{1}{6} + \frac{7k-2}{7k} = \frac{37}{14} - \frac{2}{7k}$.

- The bin contains a J_a-item and a J_3-item. In this case the space remaining for J_4-items is less than $\frac{1}{4}$.
  This gives a total weight of at most $1 + \frac{5}{16} + \frac{9}{7} \cdot \frac{1}{4} + \frac{7k-2}{7k} < \frac{37}{14} - \frac{2}{7k}$.

- The bin contains a J_a-item. In this case the space remaining for J_4-items is less than $\frac{1}{2}$.

  This gives a total weight of at most $1 + \frac{9}{7} \cdot \frac{1}{2} + \frac{7k-2}{7k} = \frac{37}{14} - \frac{2}{7k}$.

We are left with the case where no items have size above $\frac{1}{2}$. In this case, we only need to consider w_1, since w_1 dominates w_2 for this range of sizes. Furthermore, since we have already considered the case where all items have size of at most $\frac{1}{3}$, we only need to consider bins which contain one or two items of size in $(\frac{1}{3}, \frac{1}{2}]$. In fact, since J_b-items are smaller than J_2-items, but have larger weight, we can assume that all such items of size larger than $\frac{1}{3}$, are J_b-items. We have the following five cases.

- The bin contains two J_b-items and a J_3-item.

  In this case the space remaining for J_4-items is less than $\frac{1}{12}$.

  This gives a total weight of at most $2 \cdot \frac{4}{7} + \frac{3}{8} + \frac{9}{7} \cdot \frac{1}{12} + \frac{7k-2}{7k} < \frac{37}{14} - \frac{2}{7k}$.

- The bin contains two J_b-items.

  In this case the space remaining for J_4-items is less than $\frac{1}{3}$.

  This gives a total weight of at most $2 \cdot \frac{4}{7} + \frac{9}{7} \cdot \frac{1}{3} + \frac{7k-2}{7k} < \frac{37}{14} - \frac{2}{7k}$.

- The bin contains one J_b-item and two J_3-items.

  In this case the space remaining for J_4-items is less than $\frac{1}{6}$.

  This gives a total weight of at most $\frac{4}{7} + 2 \cdot \frac{3}{8} + \frac{9}{7} \cdot \frac{1}{6} + \frac{7k-2}{7k} < \frac{37}{14} - \frac{2}{7k}$.

- The bin contains one J_b-item and one J_3-item.

  In this case the space remaining for J_4-items is less than $\frac{5}{12}$.

  This gives a total weight of at most $\frac{4}{7} + \frac{3}{8} + \frac{9}{7} \cdot \frac{5}{12} + \frac{7k-2}{7k} < \frac{37}{14} - \frac{2}{7k}$.

- The bin contains one J_b-item.

  In this case the space remaining for J_4-items is less than $\frac{2}{3}$.

  This gives a total weight of at most $\frac{4}{7} + \frac{9}{7} \cdot \frac{2}{3} + \frac{7k-2}{7k} < \frac{37}{14} - \frac{2}{7k}$.

□

Algorithm A_3 was constructed to handle the case $k = 3$ and as a byproduct it gives improved bounds for all $k \geq 3$. Thus, the parameters were optimized only for the case $k = 3$. It is straightforward to see that for each fixed value of $k \geq 4$, it is possible to adapt the algorithm and optimize the parameters γ, δ, and the numerical parameters such as the fraction of items among the items of a given interval of sizes which are packed with a J_a-item. Specifically, it is possible to use a larger number of intervals, and to possibly combine additional classes of items in bins together with larger items.

5.3 Improved algorithms for large values of k

In this section we consider the case where k is large and adapt the algorithm HARMONIC++ of Seiden [13] for our problem, with a loss of 1 in the competitive ratio, that is, we get a competitive ratio of 2.58889 for any $k \geq 49$. We use the main part of this algorithm as a black box, namely, the packing of *large* items, which are defined to be items of size more than $\frac{1}{50}$, is performed by HARMONIC++.

We first discuss the general properties of an algorithm that can be used as a black box. Let $\mathcal{A}$ be an online (classical) bin packing algorithm, such that $\mathcal{A}$ partitions the input into *small* items, which are items in $(0, \frac{1}{t+1}]$ for some integer t and *large* items, which are all other items. The set of large items may be partitioned further. We require that $\mathcal{A}$ packs the large items in some way, independently of small items, and that it packs small items using NF. We call such an algorithm $\mathcal{A}$ a *uniform* bin packing algorithm. The integer t is seen as a property of $\mathcal{A}$, and is denoted t_A.

We next show how to adapt a uniform algorithm, $\mathcal{A}$ into an algorithm for our problem, with $k \geq t_A$. We partition items (online) into small and large items, as they are defined by $\mathcal{A}$. Large items are simply packed using $\mathcal{A}$. Since all these items are strictly larger than $\frac{1}{t_A+1}$, each bin may contain at most t_A items, and thus the number of color classes they can belong to is no larger than k. Therefore, the packing of large items is valid. The small items are packed using CSNF. We call this algorithm CS(A).

Theorem 9 *Let $\mathcal{R}$ be an upper bound on the competitive ratio of $\mathcal{A}$ (when it is applied to the classical online bin packing problem). Then the competitive ratio of* CS(A) *is at most $\mathcal{R} + 1$.*

Proof Let $T_1, \ldots, T_f$ be the color sets used by CSNF on small items. Clearly, we have $f \leq \lceil \frac{q}{k} \rceil$. Note that OPT $\geq \lceil \frac{q}{k} \rceil \geq f$. We apply two modifications on the input. Consider a given color set. If CSNF created only one bin for this color set we do nothing. Otherwise, we consider the last two bins created by CSNF for

this color set. Let x be the size of the first item packed in the last bin, and let y be the total size of items packed in the previous bin. Clearly, we have $x + y > 1$. We split the item of size x into two items of the same color of x, of sizes $x_1 = 1 - y < x$ and $x_2 = x - x_1$. The two new items arrive instead of the item of size x, first the part of size x_1 and then the other part. Note that the two parts are also small items. We perform this to every color set, and get a modified input sequence. Next, we move all small items to the beginning of the sequence, we reorder them so that the items of each color set, that are packed in all the bins used for this colors set except the last such bin, arrive consecutively, according to their order in the modified sequence. The content of the last bin of each color set arrives only after all other small items (of all color sets) have arrived. That is, first arrive the items of color set T_1 except for the last bin of the small items of T_1, then the same is applied for T_2, and so on. For every color set, small items arrive exactly in the order that they are originally packed into the bins by CSNF. Let I' denote the new input after the second modification, OPT$'$ the cost of an optimal offline algorithm for the new input and CS(A)$'$, the cost of CS(A) on the new input. Then, since large items are packed independently of small items, CS(A) acts exactly the same on these items. Moreover, the number of bins packed for each color set by CSNF does not change. This is clear for color sets that used a single bin. For those that used at least two bins, the only change in packing is that the last bin of a color set that receives an item of size $0 < x_2 < x$ instead of an item of size x. The second to last bin of a given color set must contain a total size of items of exactly 1. Any optimal solution of the original input can be adapted to pack the new input. The change of order does not influence offline packings. For every item that was split, its parts can take its place in the packing, since they have the same total size and the same color. Thus we have CS(A)$' =$ CS(A) and OPT$' \leq$ OPT. Now let $\mathcal{A}$ and OPT$''$ denote the costs of packing the new input sequence, but without the restriction on colors. That is, these are the costs of solutions for the classical bin packing problem on the new input (ignoring colors of items), where $\mathcal{A}$ denotes the cost of the packing returned by algorithm A, and OPT$''$ denotes the cost of the optimal solution to this instance of the classical bin packing problem. We have OPT$'' \leq$ OPT$'$ since any solution that takes colors into account is still a valid solution for the classical problem. If we apply $\mathcal{A}$ on the new input, the packing of the items of each color set of small items would be exactly the same as in CS(A), except for possibly the packing of the items of the last bin of each color set. Except for these items, items of distinct color sets do not get mixed, since the second to last bin of each color set (that now became the last bin) is full. Therefore, we have $\mathcal{A} \geq$ CS(A)$' - f$. Using $\mathcal{A} \leq \mathcal{R} \cdot$ OPT$'' + O(1)$, due to the competitive ratio for the standard bin packing problem, we get,

$$\text{CS(A)} = \text{CS(A)}' \leq \mathcal{A} + f \leq \mathcal{R} \cdot \text{OPT}'' + O(1) + \text{OPT} \leq \mathcal{R} \cdot \text{OPT}' + O(1) + \text{OPT} \leq (\mathcal{R}+1) \cdot \text{OPT} + O(1) \, .$$

□

We mention several algorithms in the literature that are indeed uniform. For the algorithm REFINED HARMONIC [11], the parameter t is 19, and the competitive ratio of this algorithm is at most 1.6359. For the algorithm MODIFIED HARMONIC of [12], the parameter t is 37, the competitive ratio is at most 1.61562. As mentioned above, Algorithm HARMONIC++ [13] is uniform as well.

Corollary 1 *The algorithm* CS(HARMONIC++) *has a competitive ratio of at most* 2.58889 *for any* $k \geq 49$. *The algorithm* CS(MODIFIED HARMONIC) *has a competitive ratio of at most* 2.61562 *for any* $k \geq 37$. *The algorithm* CS(REFINED HARMONIC) *has a competitive ratio of at most* 2.6359 *for any* $k \geq 19$.

To conclude this section, we note that for $k = 2$ we can use CSFF and get a competitive ratio of 2.5, for values of k such that $3 \leq k \leq 36$ we can use Theorem 8 and get a competitive ratio of $\frac{37}{14} - \frac{2}{7k} \leq$

$\frac{37}{14} - \frac{2}{7 \cdot 36} \approx 2.63492$, for values of k such that $37 \leq k \leq 48$ we get a competitive ratio of at most 2.61562 using Corollary 1, and for values of k such that $k \geq 49$ we get a competitive ratio of at most 2.58889 using Corollary 1. Therefore, for all values of k, we get a competitive ratio of at most 2.63492 and we established the following.

Theorem 10 *For all values of k, there exists an online algorithm for* CCBP*, with a competitive ratio of at most* 2.63492.

References

[1] Luitpold Babel, Bo Chen, Hans Kellerer, and Vladimir Kotov. Algorithms for on-line bin-packing problems with cardinality constraints. *Discrete Applied Mathematics*, 143(1-3):238–251, 2004.

[2] Alberto Caprara, Hans Kellerer, and Ulrich Pferschy. Approximation schemes for ordered vector packing problems. *Naval Research Logistics*, 92:58–69, 2003.

[3] Edward G. Coffman Jr. and János Csirik. Performance guarantees for one-dimensional bin packing. In T. F. Gonzalez, editor, *Handbook of Approximation Algorithms and Metaheuristics*, chapter 32, pages (32–1)–(32–18). Chapman & Hall/Crc, 2007.

[4] Wenceslas Fernandez de la Vega and George S. Lueker. Bin packing can be solved within 1+epsilon in linear time. *Combinatorica*, 1(4):349–355, 1981.

[5] Leah Epstein. Online bin packing with cardinality constraints. *SIAM Journal on Discrete Mathematics*, 20(4):1015–1030, 2006.

[6] Leah Epstein and Asaf Levin. AFPTAS results for common variants of bin packing: A new method to handle the small items. *CoRR*, abs/0906.5050, 2009.

[7] Narendra Karmarkar and Richard M. Karp. An efficient approximation scheme for the one-dimensional bin-packing problem. In *Proceedings of the 23rd Annual Symposium on Foundations of Computer Science (FOCS'82)*, pages 312–320, 1982.

[8] Hans Kellerer and Ulrich Pferschy. Cardinality constrained bin-packing problems. *Annals of Operations Research*, 92:335–348, 1999.

[9] K. L. Krause, V. Y. Shen, and Herbert D. Schwetman. Analysis of several task-scheduling algorithms for a model of multiprogramming computer systems. *Journal of the ACM*, 22(4):522–550, 1975.

[10] K. L. Krause, V. Y. Shen, and Herbert D. Schwetman. Errata: "Analysis of several task-scheduling algorithms for a model of multiprogramming computer systems". *Journal of the ACM*, 24(3):527–527, 1977.

[11] C.C. Lee and Der-Tsai Lee. A simple online bin packing algorithm. *Journal of the ACM*, 32(3):562–572, 1985.

[12] Prakash V. Ramanan, Donna J. Brown, C. C. Lee, and Der-Tsai Lee. On-line bin packing in linear time. *Journal of Algorithms*, 10(3):305–326, 1989.

[13] Steven S. Seiden. On the online bin packing problem. *Journal of the ACM*, 49(5):640–671, 2002.

[14] Hadas Shachnai and Tami Tamir. Polynomial time approximation schemes for class-constrained packing problems. *Journal of Scheduling*, 4(6):313–338, 2001.

[15] Hadas Shachnai and Tami Tamir. On two class-constrained versions of the multiple knapsack problem. *Algorithmica*, 29(3):442–467, 2001.

[16] Hadas Shachnai and Tami Tamir. Tight bounds for online class-constrained packing. *Theoretical Computer Science*, 321(1):103–123, 2004.

[17] Jeffrey D. Ullman. The performance of a memory allocation algorithm. Technical Report 100, Princeton University, Princeton, NJ, 1971.

[18] André van Vliet. An improved lower bound for online bin packing algorithms. *Information Processing Letters*, 43(5):277–284, 1992.

[19] Eduardo C. Xavier and Flávio Keidi Miyazawa. The class constrained bin packing problem with applications to video-on-demand. *Theoretical Computer Science*, 393(1-3):240–259, 2008.