Polynomial Time Approximation Schemes for Class-Constrained Packing Problems*

Hadas Shachnai † Tami Tamir‡

Department of Computer Science
The Technion, Haifa 32000, Israel

Abstract

We consider variants of the classic bin packing and multiple knapsack problems, in which sets of items of *different classes (colors)* need to be placed in bins; the items may have different sizes and values. Each bin has a limited capacity, and a bound on the number of distinct classes of items it can accommodate. In the *class-constrained multiple knapsack (CCMK)* problem, our goal is to maximize the total value of packed items, whereas in the *class-constrained bin-packing (CCBP)*, we seek to minimize the number of (identical) bins, needed for packing all the items.

We give a *polynomial time approximation scheme* (PTAS) for CCMK and a dual PTAS for CCBP. We also show that the 0-1 class-constrained knapsack admits a *fully* polynomial time approximation scheme, even when the number of distinct colors of items depends on the input size. Finally, we introduce the *generalized class-constrained packing problem* (*GCCP*), where each item may have more than one color. We show that GCCP is APX-hard, already for the case of a single knapsack, where all items have the same size and the same value.

Our optimization problems have several important applications, including storage management for multimedia systems, production planning, and multiprocessor scheduling.

Keywords: polynomial time approximation schemes, class constraints, multiple knapsack, bin packing, algorithms.

*A preliminary version of this paper appeared in the proceedings of APPROX'00, Saarbrücken, Germany, September 2000.

†hadas@cs.technion.ac.il. Author supported in part by Technion V.P.R. Fund, and by the Fund for the Promotion of Research at the Technion.

‡tami@cs.technion.ac.il

1 Introduction

1.1 Problem Statement

In the well-known *bin packing (BP)* and *multiple knapsack (MK)* problems, a set, I, of items of different sizes and values has to be packed into bins of limited capacities; a packing is legal if the total size of the items placed in a bin does not exceed its capacity. We consider the following *class-constrained* variants of these problems. Suppose that each item has a size, a value, and *a class (color)*; each bin has limited capacity, and a limited number of compartments. Items of different colors cannot be placed in the same compartment. Thus, the number of compartments in each bin bounds the number of *distinct* colors of items it can accommodate. A packing is legal if it satisfies the traditional capacity constraint, as well as the *class constraint*.

Formally, the input to our packing problems is a set of items, I, of size $|I| = n$. Each item $u \in I$ has a size $s(u) \in Z^+$, and a value $p(u) \in Z^+$. Each item in I is colored with one of M distinct colors. Thus, $I = I_1 \cup I_2 \cdots \cup I_M$, where any item $u \in I_i$ is colored with color i. The items need to be placed in bins, where each bin $j, j = 1, 2, \ldots$, has volume v_j and c_j compartments.

The output of our packing problems is a *placement*, which specifies for each bin j which items from each class are placed in j (and accordingly, the classes to which j allocates compartments). A placement is legal if for all $j \geq 1$, bin j allocates at most c_j compartments (i.e., bin j has items of at most c_j colors), and the overall size of the items placed in j does not exceed v_j. We study two optimization problems:

The class-constrained multiple knapsack problem (CCMK), in which there are N bins (to which we refer as knapsacks). A placement determines a subset $S = S_1 \cup S_2 \cdots \cup S_M$ of I, such that $S_i \subseteq I_i$ is the subset of packed items of color i. Our goal is to find a legal placement which maximizes the total value of the packed items, given by $\sum_{i=1}^{M} \sum_{u \in S_i} p(u)$;

The class-constrained bin-packing problem (CCBP), in which the bins are identical, each having size 1 and c compartments, and the items have sizes $s(u) \leq 1$. Our goal is to find a legal placement of all the items in a minimal number of bins.

We also consider a generalized version of class-constrained packing (GCCP), where each item, u, has a size $s(u)$, a value $p(u)$, and a set $C(u)$ of colors, such that it is legal to color u in any color in $C(u)$. Thus, if some bin allocates compartments to the set C_1 of colors, then any item $u \in I$ added to this bin needs to satisfy $C(u) \cap C_1 \neq \emptyset$. (In the above CCMK and CCBP problems, we assume that $\forall u \in I, |C(u)| = 1$). In another generalization of class-constrained packing considered in this paper, the color of each item depends on the knapsack in which it is placed. That is, each item $u \in I$ has a size $s(u)$, a value $p(u)$ and an N-vector, $\vec{c}_u$, where $\vec{c}_u(j)$ is the color of u when it is placed in the knapsack K_j, $1 \leq j \leq N$.

1.2 Motivation

Storage management in multimedia systems The CCMK problem is motivated by a fundamental problem in storage management for multimedia-on-demand (MOD) systems (see, e.g.,[35, 14]). In a MOD system, a large database of M video program files is kept on a centralized server. Each program file, i, is associated with a popularity parameter, given by $\alpha_i \in [0,1]$, where $\sum_{i=1}^{M} \alpha_i = 1$. The files are stored on N shared disks. Each of the disks is characterized by (i) its storage capacity, that is, the number of files that can reside on it, and (ii) its load capacity, given by the number of data streams that can be read simultaneously from that disk. Assuming that $\{\alpha_1, \ldots, \alpha_M\}$ are known, we can predict the expected load generated by each of the programs at any time. We need to allocate to each file disk space and fraction of the load capacity, such that the load generated due to access requests to that file is satisfied.

The above storage management problem can be formulated as a special case of the CCMK problem, in which $s(u) = p(u) = 1$ for all $u \in I$: a disk j, with load capacity ℓ_j and storage capacity c_j, is represented by a knapsack K_j, with capacity ℓ_j and c_j compartments, for all $j = 1, ..., N$. The ith file, $1 \leq i \leq M$, is represented by a set I_i, the size of which is proportional to the file popularity. Thus, $n = |I| = \sum_{j=1}^{N} \ell_j$ and $|I_i| = \alpha_i |I|$. [1] A solution for the CCMK problem induces a legal assignment of the files to the disks: K_j allocates a compartment to items of color i, *iff* a copy of the ith file is stored on disk j, and the number of items of color i that are packed in K_j is equal to the total load that file i can generate on disk j.

Production planning Our class-constrained packing problems capture also a production planning problem arising in computer systems and in many other areas. Consider a set of machines, the jth machine has a limited capacity, v_j, of some physical resource (e.g., storage space, quantity of production materials). In addition, hardware specifications allow machine j to produce items of only c_j different types. The system receives orders for products of M distinct types. Each order, u, is associated with a demand for $s(u)$ units of the physical resource, a profit, $p(u)$, and its type $i \in \{1, \ldots, M\}$.

We need to determine how the production work should be distributed among the machines. When the goal is to obtain maximal profit from a given set of machines, we have an instance of the CCMK. When we seek the minimal number of machines required for the completion of all orders, we have an instance of the CCBP. When each order can be processed under a few possible configurations we get an instance of GCCP.

Scheduling parallelizable tasks Consider the problem of scheduling *parallelizable* tasks on a multiprocessor, i.e., each task can run simultaneously on several machines (see, e.g., [33, 31]).

[1] For simplicity, we assume that $\alpha_i |I|$ is an integer (otherwise we can use a standard rounding technique [18]).

Suppose that we are given N parallelizable tasks, to be scheduled on M uniform machines. The ith machine, $1 \leq i \leq M$, runs at a specific rate, I_i. Each task, T_j, requires v_j processing units and can be allotted (possibly simultaneously) to at most c_j machines. In other words, T_j can "split" into at most c_j sub-tasks, and each sub-task can be processed on a different machine. The set, M_j, of machines allotted to T_j is fixed for the duration of the schedule. The work required by T_j needs to be partitioned among the machines in M_j. Note that the case where $c_j = 1 \ \forall j$, is the classic multiprocessor scheduling problem. Our objective is to find a schedule that maximizes the total work done in a given time interval of some length w. This problem can be formulated as an instance of the CCMK, in which a task T_j is represented by a knapsack K_j with capacity v_j and c_j compartments; a machine with rate I_i is represented by wI_i items, such that $s(u) = p(u) = 1$, for all $u \in I$.

1.3 Related Work and Our Results

There is a wide literature on the bin packing (BP) and the multiple knapsack (MK) problems (see, e.g., [20, 3, 17, 9, 5] and detailed surveys in [27, 6, 26]). Since these problems are NP-hard, most of the research work in this area focused on finding approximation algorithms. The special case of MK where $N = 1$, known as the classic *0-1 knapsack problem*, admits a fully polynomial time approximation scheme *(FPTAS)*. That is, for any $\varepsilon > 0$, a $(1 - \varepsilon)$-approximation for the optimal solution can be found in $O(n/\varepsilon^2)$, where n is the number of items [19, 13]. In contrast, MK was shown to be NP-hard in the strong sense [11], therefore it is unlikely to have an FPTAS, unless $P = NP$. It was unknown until recently, whether MK possessed a polynomial time approximation scheme *(PTAS)*, whose running time is polynomial in n, but can be *exponential* in $\frac{1}{\varepsilon}$. Chekuri and Khanna [5] resolved this question: They presented an elaborated PTAS for MK, and showed that with slight generalizations this problem becomes APX-hard. Independently, Kellerer developed in [23] a PTAS for the special case of the MK, where all bins are identical.

It is well known (see, e.g., [28]) that bin packing does not belong to the class of NP-hard problems that possess a PTAS. However, there exists an *asymptotic* PTAS (*APTAS*), which uses $(1+\varepsilon)OPT(I) + k$ bins for some fixed k. Vega and Lueker presented an APTAS, with $k = 1$ [34]. Alternatively, a *dual* PTAS, which uses $OPT(I)$ bins of size $(1 + \varepsilon)$ was given by Hochbaum and Shmoys [16]. Such a dual PTAS can also be derived from the recent work of Epstein and Sgall [8] on multiprocessor scheduling, since BP is dual to the minimum makespan problem.

Other related work deal with *multi-dimensional packing problems*[2]: Frieze and Clarke [10] gave a PTAS for the case $N = 1$; Chekuri and Khanna presented in [4] a PTAS for the case where $N \geq 1$ and the knapsacks have d equal-sized dimensions. Note that for the special case in which I contains *a single* item in each color, the CCMK reduces to a special case of 2-dimensional packing:

[2] also known as *vector packing*.

the knapsack K_j has size $v_j \times c_j$, and each item $u \in I$ has size $s(u) \times 1$. This *cardinality-constrained packing problem*, in which the number of items packed in each knapsack is constrained, was studied in [25] and recently in [24]. For $N = 1$, an FPTAS for the cardinality-constrained 0-1 knapsack problem is given in [2]. Indeed, an instance of our class-constrained packing problems (e.g., the CCMK) may contain many items in each color. In terms of 2-dimensional packing, it means that the jth knapsack is of size $v_j \times c_j$; however, the size of an item $u \in I_i$ is $s(u) \times 1$ if u is the first item of I_i in K_j, and $s(u) \times 0$ if K_j already holds items from I_i. Thus, we have not been able to adopt any of the approximation techniques developed for multi-dimensional packing for solving our class-constrained packing problems.

Shachnai and Tamir considered in [29] a special case of the CCMK, in which $s(u) = p(u) = 1$, for all $u \in I$, and showed that it is NP-hard. The paper presents an approximation algorithm that achieves a factor of $c/(c+1)$ to the optimal, when $c_j \geq c$, $\forall \ \ 1 \leq j \leq N$. Recently, Golubchik et al. [15] derived a tighter bound of $1 - 1/(1+\sqrt{c})^2$ for this algorithm, with a matching lower bound for *any* algorithm for this problem. They gave a PTAS for CCMK with unit sizes and values, and showed that this special case of the CCMK is strongly NP-hard, even if all the knapsacks are identical. Other hardness results are given in [30]. It is shown that the problem is strongly NP-hard even for instances where the total number of compartments can be arbitrarily large and all the sets are of the same size, that is, $\forall i, |I_i| = \frac{|I|}{M}$.

In this paper we study the approximability of the CCBP and the CCMK problems.

- We give a dual PTAS for CCBP which packs any instance, I, with a fixed number of distinct colors, into at most $OPT(I)$ bins whose size is at most $1+\varepsilon$.
- We derive a PTAS for CCMK which is suitable for instances with a fixed number of distinct colors. The running time of our PTAS depends on the number, t, of bin types in the instance[3]. Specifically, we distinguish between the case where t is some fixed constant and the general case, where t can be as large as N. In both cases, the profit is guaranteed to be at least $(1-\varepsilon)OPT(I)$.
- We show that the 0-1 class-constrained knapsack (CCKP) admits an FPTAS. For the case where all items have the same value, we give an *optimal* polynomial time algorithm. Our FPTAS is based on a two-level dynamic programming scheme. As in the MK problem [5], when we use the FPTAS for CCKP to fill the knapsacks sequentially with the remaining items, we obtain a $(2+\varepsilon)$-approximation for CCMK. Our results for the CCKP are suitable also for instances in which M, the number of distinct colors, depends on the input size.
- We show that the generalizations of class-constrained packing are APX-hard, already for instances where all items have the same size and the same value, and (for GCCP) there is

[3]Bins of the same type have the same capacity, and the same number of compartments.

only one knapsack.

When solving the CCBP, we note that even if $M > 1$ is some fixed constant, we cannot adopt the technique commonly used for packing (see, e.g., [16, 22, 34]), where we first consider the large items (of size $\geq \varepsilon$), and then add the small items. We show (in Appendix A) that in the presence of class constraints, one cannot extend even an *optimal* placement of the large items into an *almost* optimal placement of all items. The best we can achieve when packing first the large items, is an APTAS, whose absolute bin-waste depends on M. Such an APTAS is given in [7]. We show that CCBP admits also a *dual* PTAS. Our dual PTAS is obtained by using a different technique, which first *eliminates* most of the small items. Note that the CCBP problem is dual to the following problem of scheduling a set of jobs on m identical machines. We are given a set of n jobs of M different types; each machine can process at most c types of jobs (Each type requires a different *configuration* of the machine). We need to schedule all the jobs on the machines, so as to minimize the makespan.

Our results contain two technical contributions. We present (in Section 2.1) a technique for eliminating small items. This technique is suitable for any packing problem in which handling the small items is complex, and in particular, for class constrained packing. Using this technique, we transform an instance, I, to another instance, I', which contains at most *one* small item in each color. We then solve the problem (CCBP in our case) for I' and slightly larger bins, which is much simpler than solving for I with the original bins. Indeed, a similar technique of grouping and rounding small items is used in [1, 8]; however, our technique is tailored for instances with many types of items. It has the advantage that the *total* inaccuracy depends only on ε and independent of M. Our second idea is to transform any instance of CCMK to an instance which contains $O(\lg n/\varepsilon)$ distinct bin types. This reduction in the number of bin types is essential when guessing the partition of the bins to *color sets* in our problem[4], and it may be useful for solving other variants of the MK problem in future studies.

The rest of the paper is organized as follows. In Section 2 we give the dual PTAS for the CCBP problem. In Section 3 we consider the CCMK problem, and give PTASs for a fixed number of bin types (Section 3.5), and for an arbitrary number of bin types (Section 3.6). In Section 4 we consider the class-constrained 0-1 knapsack problem. Finally, in Section 5 we give the APX-hardness proofs for generalized class-constrained packing.

2 A Dual Approximation Scheme for CCBP

In this section we derive a dual-approximation scheme for the CCBP problem. Let $OPT_B(I)$ be the minimal number of bins needed for packing an instance I. We give a dual PTAS for CCBP,

[4]See in Section 3.6

that is, for a given $\varepsilon > 0$, we present an algorithm, A_B^ε, which packs I into $OPT_B(I)$ bins; the number of different colors in any bin does not exceed c, and the sum of the sizes of the items in any bin does not exceed $1+\varepsilon$. The running time of A_B^ε is polynomial in n and exponential in $M, \frac{1}{\varepsilon^2}$. We assume throughout this section that the parameter $M > 1$ is a fixed constant.

Our algorithm operates in two stages. In the first stage we eliminate small items, i.e., we transform I into an instance I' which consists of large items, and possibly one small item in each color. In the second stage we pack I'. We show that packing I' is much simpler than packing I, and that we only need to slightly increase the bin capacities, by factor $1+\varepsilon$. We show that a natural extension of known packing algorithms to the class-constrained problem, yields a complicated scheme. The reason is that without elimination, we need to handle the small items of each color *separately*. The elimination technique presented below involves some interaction between small items of different colors.

Various techniques can be used to approximate the optimal solution for I'. We adopt the technique presented by Epstein and Sgall [8]. Alternatively, we could use *interval partition* ([22, 34]).

Note that when there are no class constraints, our elimination technique can be used to convert any instance I into one which contains a *single* small item.

2.1 Eliminating Small Items

We now describe our *elimination* technique, and show that the potential 'damage' when solving the problem on the resulting instance, is small. For a given parameter, $\delta > 0$, denote by *small* the subset of items of sizes $s(u) < \delta$. Other items are considered *large*. Our scheme applies the following transformation to a given instance I.

1. Partition I into M sets by the colors of the items.
2. For each color $1 \le i \le M$, partition the set of small items of color i into groups: the total size of the items in each group is in the interval $[\delta, 2\delta)$; one group may have total size $< \delta$. This can be done, e.g., by grouping the small items greedily: we start to form a new group, when the total size of the previous one exceeds δ.

The resulting instance, I', consists of *non-grouped-items*, which are the original large items of I, large *grouped-items* of sizes in $[\delta, 2\delta)$, and at most M *small grouped-items* (one in each color).

From the way I' is constructed, we have

Fact 2.1 *For each color $1 \le i \le M$, the total size of the items of color i in I' is equal to the total size of the items of color i in I.*

Given a packing of I', we can replace each grouped item by the set of small items from which it was formed. In this process, neither the total size of the items nor the set of colors contained in each bin is changed. Hence,

Fact 2.2 *Any legal packing of I' into m bins induces a legal packing of I into m bins.*

Now, we need to bound the potential damage from solving the problem on I' (rather than the original instance I). We show that any legal packing of I induces a packing of I' into a bit larger bins. Note that in the transformation from I to I' we cannot use a simple replacement, since small items who belong to the same group in I' may be packed in different bins. Let $S(A)$ denote the total size of a set A of items.

Lemma 2.3 *Given a packing of I in m bins of size 1, we can pack I' into m bins of size $1+2\delta$.*

Proof: First, replace all the large items of I with their corresponding non-grouped-item in I'. The remaining items are small, and need to be replaced by grouped items. For these items (only) we show that such a replacement exists. In fact, we prove a stronger claim. For a bin B, let S_B denote the set of small items packed in B. For a set $C_1 \subseteq \{1, ..., M\}$ of colors, let $S_B^{C_1}$ denote the subset of S_B of items with colors in C_1. Denote by $C(I, B)$ the set of colors of small items of I placed in B, and by $C(I', B)$ the set of colors of the grouped items of I' placed in B. We show that the small items of I can be replaced by all the grouped items of I', such that

(i) for any bin B and a set of colors C_1, the items in $S_B^{C_1}$ are replaced by grouped items in I' whose colors are in C_1, and whose total size is at most $S(S_B^{C_1}) + 2\delta$;

(ii) for any bin B, $C(I', B) \subseteq C(I, B)$.

The proof is by induction on m, the number of bins used for packing I.
Base: $m = 1$, that is, all the items of I are placed in one bin, B. By Fact 2.1, for each color $1 \leq i \leq M$, the total size of the items of color i in I' is equal to the total size of the items of color i in I. Hence, we can replace all the items of I by all the items of I'.
Induction Step: Assume that the claim holds for any instance with $m' < m$ bins. Given an instance with m bins, let B_1, B_2 be two arbitrary bins, which contain small items from I of total size $S(S_{B_1}) + S(S_{B_2})$. Consider an instance with $m-1$ bins, derived from our instance by replacing B_1 and B_2 by one new bin B_{new}, which contains all the small items of B_1 and B_2. By the induction hypothesis, we can replace the small items of I by the grouped items of I', such that (i) and (ii) hold. The solution for the new instance induces a solution for the original instance: bins other than B_1 and B_2 are filled as in the new instance. The grouped items that are placed in B_{new} split between B_1 and B_2 as follows.

1. Let $C_1 = C(I, B_1) \setminus C(I, B_2)$; place all the C_1-colored grouped-items from B_{new} in B_1.
2. Let $C_2 = C(I, B_2) \setminus C(I, B_1)$; place all the C_2-colored grouped-items from B_{new} in B_2.
3. Let $C_3 = C(I, B_1) \cap C(I, B_2)$; place arbitrary C_3-colored grouped-items from B_{new} in B_1, until it first reaches the capacity $S(S_{B_1})$; place the remaining C_3-colored grouped-items from B_{new} in B_2.

By the induction hypothesis, B_{new} contains C_1-colored grouped-items of size at most $S(S_{B_1}^{C_1}) + 2\delta$, thus, the overflow in B_1 after step 1 is at most 2δ. Similarly, the overflow in B_2 after step 2 is at most 2δ. In addition, the total overflow in B_1 after step 3 is at most 2δ, since the size of each grouped-item is at most 2δ and we stop filling B_1 as soon as its contents exceed $S(S_{B_1})$. Finally, the overflow in B_2 after step 3 is at most 2δ, since by the induction hypothesis the total size of the items in B_{new} is at most $S(S_{B_1}) + S(S_{B_2}) + 2\delta$. ■

In particular, for an optimal packing of I, we have:

Corollary 2.4 *Let $OPT_B(I, v)$ be the minimal number of bins of size v needed for packing an instance I, then $OPT_B(I', 1 + 2\delta) \leq OPT_B(I, 1)$.*

Note that our proof induces a replacement of the small items of I by grouped items, such that the *total* overflow (when summing over all bins) is at most 2δ. However, we cannot predict how this overflow will split among the bins.

2.2 Packing the Grouped Instance I'

2.2.1 The PTAS of Epstein and Sgall

Epstein and Sgall [8] presented PTASs for multiprocessor scheduling problems on m uniform machines. When each job is represented by an item and each processor is represented by a bin, their general approximation schema can be modified to yield a dual PTAS for bin packing. Specifically, for a given parameter $\delta > 0$, the dual PTAS packs any instance I into the optimal number of bins, with the size of each bin increased at most by factor $1 + \delta$ (This can be done using binary search on m, the minimal number of bins, as in [16]).

The PTAS in [8] is based on partitioning the set of jobs, I, into $O(\frac{1}{\delta^2})$ subsets. The jobs in each subset have the same length (the original lengths are rounded). The short jobs are replaced by a few jobs of short, but non-negligible length. This partition can be represented as a *configuration vector* of length $O(\frac{1}{\delta^2})$. Any vector whose entries are at most the entries of this vector is a configuration vector of some set $S \subseteq I$ of jobs. The algorithm constructs a graph whose vertices are a source, a

target and one vertex for each possible configuration vector. There is an edge connecting the two configurations n' and n'' if a schedule of n' on i machines can be extended to a schedule of n'' on $i+1$ machines. The weight of an edge (n', n'') reflects the cost of moving from the configuration n' to the configuration n'', that is, the time needed for the $(i+1)$th machine to process the jobs in $n'' - n'$.

Having the weights defined in such a way, the problem of finding a good schedule is reduced to the problem of finding a good path in the graph. We seek a path of length m from the source (whose configuration represents an empty set of jobs), to the target (whose configuration represents the set of all jobs). The evaluation of a path depends on our objectives. For example, when minimizing the makespan, a good path is a one that minimizes the maximal weight of some participating edge; when minimizing the sum of the completion times, a good path is a one that minimizes the total weight of the participating edges. The time complexity of the algorithm depends on the size of the graph, which is dominated by the total number of configurations. For a given δ, the graph has size $O(n^f)$ where $f = \frac{81}{\delta^2}$.

When this scheme is used for bin-packing, each configuration represents a set of items. An edge connecting the two configurations n' and n'' implies that a placement of n' in i bins can be extended to placement of n'' in $i+1$ bins. The weight of the edge is the total size of items in $n'' - n'$, that are to be placed in the $(i+1)$th bin. For a given value of m, if there is a packing of I into m bins of size 1, then there is a path of length m from the source to the target with all edges having weight at most $1+\delta$ (Recall that we use binary search to find the minimal value of m).

2.2.2 The Algorithm A_{ES}:

A natural extension of Epstein-Sgall's algorithm for the CCBP problem is to refine the configurations to describe the number of items from each size set and *from each color*. Such extension results in a graph of size $O(n^f)$ where $f = M(2M+1)^4/\delta^2$. The value of f increases from $81/\delta^2$ to $M(2M+1)^4/\delta^2$ since the small items of each color are handled separately. In addition, the total size of small items of each color is rounded to the nearest multiple of $\frac{\delta}{9}$. Thus, if rounded items of c colors are packed into the same bin, we may get in this bin an overflow of $c\frac{\delta}{9}$. Given that there is only one small item in each color, as in I', we can use the *actual* sizes of the small items. This decreases significantly the possible number of configurations and prevents any potential overflow due to rounding.

Hence, given an instance I', consisting of large items and at most M small items, each of different color, we derive a simplified version of the PTAS in [8]. In this version, denoted by A_{ES}:

1. Each set of items $S \subseteq I'$ is represented by a unique configuration of length $O(\frac{M}{\delta^2})$, which indicates how many items in S belong to each color and to each size class. Note that there

is no need to round the total size of the small items: we only need to indicate which small items are included in the set (a binary vector of length M). In terms of [8], this means that we do not need to accumulate the size of small items in each configuration, and no precision is lost because of small items.

2. To guarantee that each bin contains items of at most c colors, we connect in the configuration graph only vertices representing configurations which differ by at most c colors. In other words, there is an edge connecting the two configurations n' and n'' only if $n'' - n'$ is positive in entries that belong to at most c colors.

We summarize in the next Lemma:

Lemma 2.5 *Let m be the minimal number of bins of size v needed for packing I', then, for a given $\delta > 0$, A_{ES} finds a packing of I' into m bins of size $v(1+\delta)$. The running time of A_{ES} is $O(n^f)$ where $f = \frac{M}{\delta^2}$.*

2.3 The Algorithm A_B^ε

Let $\delta = \frac{\varepsilon}{4}$; $v = 1 + 2\delta$. The algorithm A_B^ε proceeds as follows.

1. Construct I' from I, using the algorithm in Section 2.1.
2. Use A_{ES} for packing I' into bins of size $v(1+\delta)$. Let m be the number of bins used.
3. Ungroup the grouped items to obtain a packing of I in m bins of size $v(1+\delta)$.

Theorem 2.6 *The algorithm A_B^ε uses at most $OPT_B(I)$ bins. The sum of the sizes of the items in any bin does not exceed $1+\varepsilon$. The running time of A_B^ε is $O(n^f)$ where $f = \frac{16M}{\varepsilon^2}$.*

Proof: We first show that the sum of the sizes of the items in any bin does not exceed $1+\varepsilon$: Combining Fact 2.2 and Lemma 2.5, we conclude, that each of the m bins is filled up to capacity $v(1+\delta)$. Clearly, for any $\varepsilon \leq 2$, $\delta \leq \frac{1}{2}$ (otherwise, we can choose a smaller value for δ), thus, $v(1+\delta) \equiv (1+2\delta)(1+\delta) \leq 1 + 4\delta = 1 + \varepsilon$.

Next, we show that the number of bins used by A_B^ε is at most $OPT_B(I,1)$: By Lemma 2.5, I' is packed by A_{ES} in m bins of size $v(1+\delta)$, such that $m \leq OPT_B(I', v)$. Then, we use these m bins to pack I. From Corollary 2.4, $OPT_B(I', v) \leq OPT_B(I, 1)$. Therefore $m \leq OPT_B(I,1)$. Finally, setting $\delta = \frac{\varepsilon}{4}$, the running time of A_B^ε follows directly from the running time of A_{ES}. ∎

We note that our assumption that M is fixed is needed only in the second step, when packing I'. Hence, this assumption can be relaxed when our technique, for eliminating small items, is used for other purposes.

3 Approximation Schemes for CCMK

3.1 The PTAS of Chekuri and Khanna

Chekuri and Khanna developed in [5] a PTAS for MK. Let $P(U)$ denote the value of a set $U \subseteq I$, and let $OPT(I)$ be the value of an optimal solution when packing I. The PTAS in [5] is based on two steps: (i) *Guessing items:* identify a set of items $U \subseteq I$ such that $P(U) \geq (1-\varepsilon)OPT(I)$ and U has a feasible packing. (ii) *Packing items:* given such U, find a feasible packing of $U' \subseteq U$, such that $P(U') \geq (1-\varepsilon)P(U)$.

For the sake of completeness, we summarize the implementation of the *guessing items* step as given in [5]. For a guessed value P satisfying $(1-\varepsilon)OPT(I) \leq P \leq OPT(I)$, discard all the items $u \in I$ with $p(u) < \varepsilon P/n$; then, divide all the profits by $\varepsilon P/n$, and round each down to the nearest power of $(1+\varepsilon)$. The resulting instance, I', has $h \leq \lceil \log_{1+\varepsilon} \frac{n}{\varepsilon} \rceil = O(\frac{\ln n}{\varepsilon})$ distinct profits. Thus, we assume that the items of I are partitioned into h sets $I^1, \ldots, I^h$ with items in each set having the same profit. Let $\mathcal{D}$ be a subset of items which gives a solution with profit P, and let $\mathcal{D}^j$ be the subset of items in $\mathcal{D}$ in the jth profit class.

We proceed to guess an h-vector, $\vec{k}$, of integers, $1 \leq k_j \leq h/\varepsilon^2$; k_j reflects the contribution of $\mathcal{D}^j$ to the overall profit in some optimal packing, in multiples of $\frac{\varepsilon^2 P}{h}$, i.e.,

$$k_j \frac{\varepsilon^2 P}{h} \leq P(\mathcal{D}^j) \leq (k_j+1)\frac{\varepsilon^2 P}{h}.$$

We seek a vector $\vec{k} = (k_1, \ldots, k_h)$ satisfying $\sum_{j=1}^{h} k_j \leq \frac{h}{\varepsilon^2}$. As shown in [5], the number of such vectors is $O(n^{O(1/\varepsilon^3)})$; thus, $\vec{k}$ can be guessed in polynomial time. Given the contribution of each profit set I^j in an optimal packing, we select for the set U the items of I^j with smallest *sizes* whose total profit accumulates to the required contribution. These small items can clearly replace any set of items with the same contribution from $\mathcal{D}^j$. Thus, U is feasible.

3.2 Our PTAS for CCMK - An Overview

In the following we adopt the guessing approach in [5], for obtaining a PTAS for CCMK. For some $t \geq 1$, assume that there are t types of bins, such that bins of type j have the same capacity, v_j, and the same number of compartments, c_j. Let $c = \max_j c_j$ be the maximal number of compartments in any type of bins. We denote by B_j the set of bins of type j; let $m_j = |B_j|$ be the number of bins in B_j. Our schema proceeds in three steps:

1. Guess a subset $U \subseteq I$ of items that will be packed in the bins, such that $P(U) \geq (1-\varepsilon)OPT(I)$, and U has a feasible class-constrained packing in the knapsacks.

2. Guess a partition of the bins to at most

$$r = \sum_{\ell=1}^{c} \binom{M}{\ell} = O(M^c) \tag{1}$$

color sets, i.e., associate with each subset of bins the set of colors that can be packed in these bins. We call a subset of bins of the same type and color set a *block*; thus, the resulting number of blocks is $T \leq t \cdot r$.

3. Find a feasible packing of $U' \subseteq U$ in the bin blocks, such that $P(U') \geq (1-\varepsilon)P(U)$.

As in [5], the overall scheme is more involved since we iterate between the guessing and packing steps. This is due to the fact that $OPT(I)$ is not known in advance. However, the maximal number of guessing-packing iterations is bounded by the maximal number of guesses for a value P, such that

$$(1-\varepsilon)OPT(I) \leq P \leq OPT(I). \tag{2}$$

To bound this number, we show a γ-approximation for the problem for some constant $0 < \gamma \leq 1$.

Claim 3.1 *Let $c_m = \min_j c_j$ be the minimal number of compartments in any type of bins, then the CCMK problem has a polynomial time $\frac{c_m}{2M}$-approximation algorithm.*

Proof: The following algorithm, $\mathcal{A}$, is a $\frac{c_m}{2M}$-approximation algorithm for CCMK:

1. Ignore the colors and find a $\frac{1}{2}$-approximate solution for MK for the input I (such an algorithm is given is [32]).

2. For each knapsack, K_j, find the c_j most valuable colors in the resulting packing and remove all the items which are not in these colors.

Let z_1 be the optimal profit for the non class-constrained problem, and let z_2 be the profit obtained in step 1 of $\mathcal{A}$. Let z^* be the optimal profit for the class constrained problem, and let z be the profit obtained by $\mathcal{A}$.

Note that since we pack in each knapsack items of at most c_j colors, $\mathcal{A}$ produces a valid packing. In addition,

$$z \geq \frac{c_m}{M} z_2 \geq \frac{c_m}{2M} z_1 \geq \frac{c_m}{2M} z^*.$$

Thus, $\mathcal{A}$ is a $\frac{c_m}{2M}$-approximation algorithm for the CCMK problem. ∎

Having a $\frac{c_m}{2M}$-approximation for $OPT(I)$, we can find a value of P satisfying (2) using a binary search over the resulting range.

Corollary 3.2 *The number of guessing-packing iterations for finding P which satisfies (2) is at most* $\lceil \log(\frac{2M}{\varepsilon}) \rceil$.

In Sections 3.3, 3.5, and 3.6 we describe in detail the implementation of each of the steps in our PTAS. The implementation of the second and the third steps varies, depending on t, the number of bin types. As a result, the time complexity of our PTAS depends also on that parameter.

3.3 Guessing Items

We now show that the first step, of guessing items, can be generalized to the class-constrained version. This generalization is straightforward: after reducing (to h) the distinct number of profits in I (as detailed in Section 3.1), we further partition each profit set I^j into M color sets $I^j_1, \ldots, I^j_M$, with the items in each set having the same profit and the same color. For a subset of items, $\mathcal{D}$, whose packing gives the profit P, denote by $\mathcal{D}^j_i$ the subset of items in $\mathcal{D}$ in the jth profit class and in color i. We now guess an $h \times M$-matrix, $K = \{k_{i,j}\}$, of integers, $1 \leq k_{i,j} \leq hM/\varepsilon^2$; $k_{i,j}$ reflects the contribution of $\mathcal{D}^j_i$ to the overall profit in some optimal packing, in multiples of $\frac{\varepsilon^2 P}{Mh}$, i.e.,

$$k_{i,j}\frac{\varepsilon^2 P}{Mh} \leq P(\mathcal{D}^j_i) \leq (k_{i,j}+1)\frac{\varepsilon^2 P}{Mh}. \tag{3}$$

We seek a matrix $K = \{k_{i,j}\}$ satisfying $\sum_{j=1}^{h}\sum_{i=1}^{M} k_{i,j} \leq \frac{Mh}{\varepsilon^2}$. Given the contribution of each profit-color set I^j_i in an optimal packing, we select for U the items of I^j_i with smallest *sizes* whose total profit accumulates to the required contribution.

In order to show that the above algorithm finds in polynomial time a set $U \subseteq I$ as required in the guessing step, we need to show the following claims:

Claim 3.3 *Solving the problem for the rounded profit instance I' rather than I, can decrease the optimal profit by a factor of at most* $(1-\varepsilon)$.

Proof: The proof follows from the fact that for any subset U of I, U is transformed into a set $U' \subseteq I'$ whose profit satisfies

$$(1-\varepsilon)P(U) \leq \frac{P(U)}{1+\varepsilon} \leq \frac{P(U')\varepsilon P}{n} \leq P(U).$$

In particular, this holds for an optimal subset of packed items. ∎

Claim 3.4 *There exists an $h \times M$-matrix, K, of integers, $1 \leq k_{i,j} \leq hM/\varepsilon^2$, such that the set U induced by K has a feasible class-constrained packing and $P(U) \geq (1-\varepsilon^2)P$.*

Proof: Let $\mathcal{D}$ be the subset of items as defined above. Recall that U is the subset of items induced by the matrix K; then, from the right inequality in (3) we get that

$$P = P(\mathcal{D}) = \sum_{j=1}^{h}\sum_{i=1}^{M} P(\mathcal{D}_i^j) \leq \sum_{j=1}^{h}\sum_{i=1}^{M} k_{i,j}\frac{\varepsilon^2 P}{Mh} + \varepsilon^2 P = P(U) + \varepsilon^2 P.$$

This yields the claim. ∎

Claim 3.5 *The number of possible guesses for the matrix K is $n^{O(\frac{M}{\varepsilon}\ln(1/\varepsilon))}$.*

Proof: Note that the total number matrices K is the number of hM-tuples whose coordinates sum to hM/ε^2, given by $\binom{hM+\frac{hM}{\varepsilon^2}-1}{hM-1}$. Now, for any $n \geq l \geq 0$, $\binom{n}{l} \leq (\frac{en}{l})^l$. Hence, we get that

$$\binom{hM+\frac{hM}{\varepsilon^2}-1}{hM-1} \leq (\frac{3e}{\varepsilon^2})^{hM} = e^{h\cdot M\cdot\ln(\frac{3e}{\varepsilon^2})} = n^{O(\frac{M}{\varepsilon}\ln(1/\varepsilon))}.$$

This completes the proof. ∎

Combining Claims 3.3, 3.4, and 3.5, we have

Corollary 3.6 *For a given P, a set U such that $P(U) \geq (1-\varepsilon)P$, and U has a feasible class-constrained packing in the knapsacks, can be found in $n^{O(\frac{M}{\varepsilon}\ln(1/\varepsilon))}$ guesses.*

3.4 Reducing the Number of Distinct Item Sizes

Using the restricted number of distinct profit values we can also reduce the number of distinct sizes in the set U to $O(\ln n/\varepsilon^2)$. This stage is similar to the corresponding step in [5], and is based on the *shifting* technique introduced in the PTAS for bin-packing in [19]. Let A be a set of a items with identical profits, identical colors and arbitrary sizes. We order the items in A in non-decreasing order by sizes and divide them into $g = (1 + 1/\varepsilon)$ groups $A_1, ..., A_g$ with $A_1, ..., A_{g-1}$ containing $\lfloor a/g \rfloor$ items each and A_g containing the remaining items. We discard the items in A_{g-1} and for each $i < g-1$ we increase the size of every item in A_i to the size of the smallest item in A_{i+1}. Since A is ordered by size, no item in A_i is larger than the smallest item in A_{i+1} for $1 \leq i < g$. Thus, if A has a feasible packing then the modified instance also has one. We discard at most an ε fraction from the profit (of the items in A_{g-1}) and the modified instance has at most $2/\varepsilon$ distinct sizes ($g-2$ sizes of the items in $A_1, ..., A_{g-2}$ and at most g distinct sizes of the items in A_g). Applying this to each profit-color class in U, we can now assume that U consists of $s \leq O(M \ln n/\varepsilon^2)$ classes, i.e., $U_1 \cup \cdots \cup U_s$, and the items in U_k are of the same size, the same profit,and the same color.

3.5 Instances with a Fixed Number of Bin Types

Assume that $t \geq 1$ is a *fixed* constant. In the second step we partition the bins into blocks. We need to determine $m_{j,l}$, the number of bins of type j associated with the lth color set. Thus, the number of guesses is $O(n^T)$ (since we need to determine the size of T blocks, and each block can consist of at most n bins).

The third step of our scheme is implemented as follows. Let $n_k = |U_k|$ be the number of items in the class U_k, $1 \leq k \leq s$. We first partition U_k to $1 \leq T' \leq T$ subsets, where T' is the number of distinct blocks in which the items of U_k are placed. Clearly, the items of U_k can only be placed in blocks that include the color of the items in U_k in their color set. Denote by $n_{k,b}$ the number of items in the class U_k that are placed in the block b. For $i = 1, \ldots, 2T/\varepsilon$, let $\sigma_i^k = \lceil \frac{i\varepsilon n_k}{2T} \rceil$.

Claim 3.7 *By considering only partitions of U_k in which $\forall b, \exists i,\ n_{k,b} = \sigma_i^k$, we can decrease the optimal profit gained by items of U_k at most by factor $(1-\varepsilon)$.*

Proof: First note that if $\frac{\varepsilon n_k}{2T} \leq 1$ then clearly all possible integer values for $n_{k,b}$ are considered. If $\frac{\varepsilon n_k}{2T} > 1$ then $\forall 1 \leq i \leq 2T/\varepsilon$,

$$\begin{aligned} \sigma_i^k - \sigma_{i-1}^k &= \lceil \frac{i\varepsilon n_k}{2T} \rceil - \lceil \frac{(i-1)\varepsilon n_k}{2T} \rceil \leq \frac{i\varepsilon n_k}{2T} + 1 - \frac{(i-1)\varepsilon n_k}{2T} \\ &= 1 + \frac{\varepsilon n_k}{2T} \leq \frac{2\varepsilon n_k}{2T} = \frac{\varepsilon n_k}{T}. \end{aligned} \tag{4}$$

Consider any partition of U_k into the blocks. For any block b, if there is no i such that $n_{k,b} = \sigma_i^k$ then we can remove the less-profitable items until we reach some σ_i^k. By (4) we remove at most $\frac{\varepsilon n_k}{T}$ items. Thus, we remove a total of at most εn_k items - with an overall loss of at most a factor of ε from the total profit of U_k. ∎

Therefore, we can assume that $n_{k,b}$ is one of $2T/\varepsilon$ values. We can now describe the partition of U_k as a list of pairs (b, i), where $1 \leq b \leq T$ and $1 \leq i \leq 2T/\varepsilon$. The number of such subsets of pairs for U_k is at most $2^{O(T^2/\varepsilon)}$, and as we need to consider all the classes, $U_1, \ldots, U_s$, the overall number of guesses is $2^{s \cdot T^2/\varepsilon} = 2^{O((M \ln n/\varepsilon^2) \cdot (T^2/\varepsilon))} = n^{O(MT^2/\varepsilon^3)}$.

By the *guessing items* step of our scheme, the set U has a feasible packing. In particular, there exists a partition of the U_k's among the T blocks which corresponds to a feasible packing. We show that we can pack $U' \subseteq U$ such that $P(U') \geq (1-\varepsilon)P(U)$. In fact, for each block, b, we have guessed a set of items that has a feasible packing in b. Since the bins in each block are identical, we can use, for each block separately, any known PTAS for bin packing (see, e.g., [22] whose time complexity is $O(n/\varepsilon^2)$) and take for each block the $m_{j,l}$ most profitable bins. Since such a PTAS packs *all* the guessed items in at most $(1+\varepsilon)m_{j,l}$ bins, we discard at most ε of the profit. If $m_{j,l} \leq \lceil \frac{1}{\varepsilon} \rceil$ then we

can use a PTAS for a constant number of bins (given in [10]). Thus, for a given partition of the U_k's to the blocks, we can find a packing with profit at least $(1-\varepsilon)P(U)$ in $O(Tn/\varepsilon^2)$ steps.

We summarize in the next result

Theorem 3.8 *The CCMK with a fixed number of distinct colors, $M \geq 1$, and a fixed number of bin types, $t \geq 1$, admits a PTAS, whose running time is $O(\frac{T}{\varepsilon^2} n^{O(\frac{M}{\varepsilon}\ln(1/\varepsilon)+\frac{MT^2}{\varepsilon^3})})$.*

Proof: The proof follows from a combination of (i) number of guesses of the value of P; (ii) the number of guesses of the matrix K; (iii) the number of guesses of the partition of U to bin blocks, and (iv) the complexity of the PTAS for bin packing, for each guess.

Thus, we get that the overall running time is at most

$$\lceil \log(\frac{2M}{\varepsilon}) \rceil \cdot O(n^{O(\frac{M}{\varepsilon}\ln(1/\varepsilon))} \cdot n^{O(MT^2/\varepsilon^3)} \cdot \frac{Tn}{\varepsilon^2})$$

which yields the statement of the theorem. ∎

3.6 CCMK with Arbitrary Number of Bin Types

Suppose that the number of bin types is $t = O(\lg n)$. Recall that $m_j = |B_j|$ is the number of bins of the jth type. We need to determine for all $1 \leq j \leq t$, $1 \leq l \leq r$ the number $m_{j,l}$ of bins of type j associated with the lth color set. A naive implementation of this step of our scheme results in $O(n^{O(lgn)})$ guesses. Thus, we limit the possible partitions of the bins to blocks as follows. For $i = 0, \ldots, 2r/\varepsilon$, let $\beta_i^j = \lceil \frac{i\varepsilon m_j}{2r} \rceil$.

Claim 3.9 *By considering only partitions of m_j in which $\forall l, \exists i$, $m_{j,l} = \beta_i^j$, we can decrease the optimal profit gained by items packed in the bins of B_j by a factor of at most $(1-\varepsilon)$.*

Proof: First note that if $\frac{\varepsilon m_j}{2r} \leq 1$ then clearly all possible integer values for $m_{j,l}$ are considered. If $\frac{\varepsilon m_j}{2r} > 1$ then, as in the proof of Claim 3.7, $\forall 1 \leq i \leq 2r/\varepsilon$,

$$\beta_i^j - \beta_{i-1}^j \leq \frac{\varepsilon m_j}{r}. \tag{5}$$

Consider any partition of m_j among the r color sets. For any color set l, if there is no i such that $m_{j,l} = \beta_i^j$ then we can add bins of type j until we reach some β_k^j. From (5) we add at most $\frac{\varepsilon m_j}{r}$ bins. Thus, summing over all color sets, we add a total of at most εm_j bins. Once the placement is completed we can pick for each j the m_j most profitable bins with an overall loss of at most a factor of ε from the total profit. ∎

Hence, we can consider only partitions in which the number of bins allocated to the lth color set is a multiple of $\varepsilon m_j/2r$, and the total number of bins of type j is at most $(1+\varepsilon)m_j$. Since $m_{j,l}$

is one of $2r/\varepsilon$ values, we can describe the partition of m_j as a list of pairs (l,i), where $1 \leq l \leq r$ and $1 \leq i \leq \frac{2r}{\varepsilon}$. We get that the number of guesses is $2^{O(r^2/\varepsilon)}$. The overall number of guesses, when taking all bin types, is $2^{O(\frac{r^2}{\varepsilon}\log n)} = n^{O(r^2/\varepsilon)}$.

In the third step we adapt the packing steps in [5], with the bins partitioned to blocks, as defined above.

Indeed, in the general case, t can be as large as N. The packing steps in [5] assume that there are $O(\lg(n/\varepsilon))$ subsets of bins such that the capacities of bins from the same subset are in the range $[v, v(1+\varepsilon))$ for some v. In the following we show that the set of bins can be partitioned in a way that fits this assumption.

Lemma 3.10 *Any set of bins with t bin types can be transformed into a set with $O(\lg(n/\varepsilon))$ subsets of bin types, such that the bins in each subset have the same number of compartments and their capacities are in the range $[v, v(1+\varepsilon))$ for some v, and for any $U \subseteq I$ that has a feasible packing in the original set, there exists $U'' \subseteq U$ that has a feasible packing in the transformed set, and $P(U'') \geq (1-\varepsilon)P(U)$.*

Proof: Let v_{max} denote the maximal capacity of a bin. Consider the set of bins whose capacities are in the range $[\varepsilon v_{max}/n^{\frac{3M}{\varepsilon}}, v_{max}]$. We partition this set of bins into subsets, $A^0, A^1, \ldots$, such that A^j consists of the bins whose capacities are in the range

$$[\ \frac{\varepsilon v_{max}}{n^{\frac{3M}{\varepsilon}}}(1+\varepsilon)^j,\ \frac{\varepsilon v_{max}}{n^{\frac{3M}{\varepsilon}}}(1+\varepsilon)^{j+1}).$$

Distinguishing between bins by number of compartments, we partition each A^j into at most c subsets, $A^j_1, \ldots, A^j_c$. We now have at most $c\lg(n^{\frac{3M}{\varepsilon}}/\varepsilon)$ different subsets if bins.

Now, for any $U \subseteq I$ that is packed in the original set of bins, we can add $w \leq M$ bins, one for each color $1 \leq i \leq M$, and pack, for each color, i, all the items in color i, previously packed in bins of sizes smaller than $\varepsilon v_{max}/n^{\frac{3M}{\varepsilon}}$, in a *single* bin with a single compartment. The total capacity of the w added bins is at most $\varepsilon v_{max}/n^{(\frac{3M}{\varepsilon}-1)} \leq \varepsilon v_{max}/n^{\frac{2M}{\varepsilon}}$. Therefore, we can transform the given instance to an instance, where all the bins, except for $w \leq M$, are of sizes in the range $[\varepsilon v_{max}/n^{\frac{3M}{\varepsilon}}, v_{max}]$, arranged in the sets $\{A^j_i\}$ defined above.

Since the set U is unknown in advance, we have to guess $w \leq M$ and the partition of the extra capacity of the small bins (which is at most $\varepsilon v_{max}/n^{\frac{2M}{\varepsilon}}$) among the w new bins. The capacity of each of the new bins will be a multiple $1 \leq k \leq n$ of the minimal size $\varepsilon v_{max}/n^{\frac{3M}{\varepsilon}}$. Thus, the overall number of guesses is $n^w \leq n^{O(M)}$.

Finally, after the items of U are packed in the transformed set of bins, we need to eliminate the extra w bins. Clearly, we can remove any set of w bins with larger capacities and at least

one compartment. Consider the set of bins of $T'' = \lg(n^{\frac{2M}{\varepsilon}}/\varepsilon)$ types with largest capacities. The number of bins in the set is at least T'', and

$$\begin{aligned} T'' + w &\leq \frac{2M}{\varepsilon} \lg n + \lg \frac{1}{\varepsilon} + M \\ &\leq (1+\varepsilon)\frac{2M}{\varepsilon} \lg n + (1+\varepsilon) \lg \frac{1}{\varepsilon} \\ &\leq (1+\varepsilon)T''. \end{aligned}$$

That is, we increase the number of bins in this set at most by factor $(1+\varepsilon)$. Since the smallest bin size in this set is $\varepsilon v_{max}/n^{\frac{2M}{\varepsilon}}$, we can select the T'' most profitable bins. The set of items, U'', packed in the bins that were not omitted from the instance has profit at least $(1-\varepsilon)P(U)$. ∎

We can now proceed to apply the above PTAS to the resulting set of bins. Thus,

Theorem 3.11 *For any instance I of the CCMK, in which $M \geq 1$ is fixed, there is a PTAS which obtains a total profit of $(1-\varepsilon)OPT(I)$ in time $O(n^{O(\frac{M}{\varepsilon}\ln(1/\varepsilon)+r^2/\varepsilon+\ln(1/\varepsilon)/\varepsilon^8)})$, where r is given in (1).*

Proof: We get the overall complexity of our PTAS by combining the following steps: (i) guessing a value of P; (ii) guessing the matrix K; (iii) guessing the partition of U to bin blocks; (iv) guessing the partition of the extra capacity among the new bins, and (v) packing the items [5]. Thus, we have that the overall running time is at most

$$\lceil \log(\frac{2M}{\varepsilon}) \rceil \cdot O(n^{O(\frac{M}{\varepsilon}\ln(1/\varepsilon)+r^2/\varepsilon+M+\ln(1/\varepsilon)/\varepsilon^8)})$$

which yields the desired result. ∎

4 The Class-Constrained 0-1 Knapsack Problem

In this section we consider the *class-constrained 0-1 knapsack problem (CCKP)*, in which we need to place a subset U of I in a single knapsack of size $v \geq \max\{s(u)|u \in I\}$. The objective is to pack items of maximal value from I in the knapsack, such that the sum of the sizes of the packed items does not exceed v, and the number of different colors in the knapsack does not exceed c.

Throughout this section we assume that the numbers c and M are given as part of the input (Otherwise, using an FPTAS for the classic 0-1 knapsack problem, we can examine all the $\binom{M}{c}$ possible subsets of c colors). We handle separately instances from three categories:

1. Instances with *uniform* value, in which all the items have the same value (regardless of their size or color). In the context of production planning (see in Section 1.2), such instances reflect systems in which the revenue is proportional to the *number* of items produced (regardless of their types and demands for resources)

2. Instances with *color-based* values, in which the items in each color have the same value (and arbitrary sizes), i.e., for $1 \leq i \leq M$, the value of any item in color i is p_i. This class reflects systems in which the revenue depends on the product type, regardless of its demand for resources

3. Instances with *arbitrary* values, independent of the item sizes and colors.

For uniform-value instances we present a polynomial time optimal algorithm. Note that the problem becomes NP-hard when we have to pack the items into two (or more) knapsacks. Formally,

Theorem 4.1 *The CCMK problem is NP-hard for uniform-value instances.*

Proof: We show a reduction from the *Partition problem*, which is known to be NP-hard [12]. The partition problem consists of a finite set A of $2k$ items with integral sizes $a_1, a_2, \ldots, a_{2k}$. The problem is to determine, if there exists a subset A' of A such that $|A'| = k$ and $\sum_{j \in A'} a_j = \sum_{j \in A \setminus A'} a_j$.

Given an instance for partition, $A = \{a_1, a_2, \ldots, a_{2k}\}$, let $\sum_{j=1}^{2k} a_j = 2S$ (if the sum is odd then clearly no partition exists). We construct an input for the CCKP with two knapsacks with $v = S$ and $c = k$. The items to be packed are of $M = 2k$ colors. There are $2k$ items, one in each color: the ith item has size a_i and value $p_i = 1$, $1 \leq i \leq M$. It is easy to verify that there is a packing with value $2k$ if and only if there exists a desired partition. ∎

When dealing with uniform-value instances, packing a subset of items of *maximal value* is equivalent to packing the *maximal number* of items. Intuitively, we would like to pack small items first: indeed, when $c \geq M$ the color constraints can be ignored, and the resulting problem can be solved optimally, by a greedy algorithm which adds to the knapsack the smallest available item, until no item can be added. However, when $c < M$, the greedy approach ceases to be the optimal. In fact, as we show in Appendix B a simple greedy algorithms can perform poorly.

For non-uniform values, we present an FPTAS for the CCKP. Our FPTAS has different time complexity when applied to instances with color-based values, and to instances with arbitrary values. Note that for instances with non-uniform values the problem is NP-hard. Indeed, when $c = M = n$, i.e., there is one item in each color and no color-constraints, we get an instance of the classic 0-1 knapsack problem. Note that the above also implies that the problem is NP-hard for instances with color-based values.

As in the FPTASs for the non class-constrained problem [19], and for the cardinality-constrained problem [2], we combine scaling of the profits with dynamic programming (DP). However, in the class-constrained problem we need two levels of DP recursions, as described below.

4.1 An Optimal Solution Using Dynamic Programming

Assume that we have an upper bound, $\hat{P}$, on the optimal value of the solution to our problem. Given such an upper bound, we can formulate an algorithm, based on dynamic programming, to compute the optimal solution. The time complexity of the algorithm is polynomial in $\hat{P}, n$ and M.

For each color $i \in \{1, \ldots, M\}$, let n_i denote the number of items of color i in the instance I. Thus, $n = \sum_{i=1}^{M} n_i$. Let $f_i(a, \ell)$ denote the smallest size sum of items with total value a of ℓ distinct colors out of the colors $1, \ldots, i$, where $i = 1, \ldots, M$; $a = 0, \ldots, \hat{P}$; $\ell = 1, \ldots, c$.

4.1.1 Dynamic Programming for Instances with Color-Based values

When packing instances with color-based values, in any optimal solution, we can replace the packed items of color i by the smallest items of color i, for all $1 \leq i \leq M$. Thus, we consider below only solutions in which we pack the smallest items of each color. For each $1 \leq j \leq n_i$, let $s_i(j)$ denote the size sum of the j smallest items of color i. Recall that p_i is the value of the items of color i.

The dynamic programming recursion is defined as follows. We initialize

$$f_0(a, \ell) = \begin{cases} 0 & a = 0 \text{ and } \ell = 0 \\ +\infty & \text{otherwise} \end{cases}$$

and $f_i(a, 0) = +\infty \ \ \forall 1 \leq i \leq M, \ \forall a > 0$. Then, for $i = 1, \ldots, M$, the entries of f_i can be computed from those of f_{i-1} by using for $\ell \in \{1, ...c\}$, $a \in \{0, ..., \hat{P}\}$ the formula

$$f_i(a, \ell) = \min \begin{cases} f_{i-1}(a, \ell) \\ \min\limits_{1 \leq j \leq n_i, \ a \leq jp_i} (f_{i-1}(a - jp_i, \ell - 1) + s_i(j)) \end{cases}$$

The first line considers the case in which color i is not included in the packing. The second line considers the cases in which color i is included and j items of color i are packed.

The value of an optimal solution for the problem is given by

$$\max_{a=0,\ldots,\hat{P};\ \ell=0,\ldots,c} \{a : f_M(a, \ell) \leq v\}.$$

The time complexity of the algorithm is $O(\sum_{i=1}^{M} n_i \hat{P} c) = O(n\hat{P}c)$.

For the special case of uniform-value instances, we can assume w.l.o.g., that $\forall u \in I, p(u) = 1$. Since we pack at most n items, we can bound the maximal profit by $\hat{P} = n$, and we get an optimal $O(n^2 c)$ algorithm.

4.1.2 Dynamic Programming for Arbitrary Values

When the items have arbitrary values, the algorithm consists of two stages. In the first stage we calculate for each color i the value $h_{i,k}(a)$, which denotes the smallest size sum of items with total value a, out of the first k items of color i (the items in each color are given in arbitrary order). $h_{i,k}(a)$ is calculated for $k = 1, \ldots, n_i$, and $a = 0, \ldots, \hat{P}$.

For each color $i = 1, \ldots, M$, the DP algorithm calculates h_i as follows. We initialize $h_{i,0}(0) = 0$ and $h_{i,0}(a) = +\infty$ for $a > 0$. Denote by p_k^i, s_k^i the value and size of the kth item of color i, respectively.

For $k = 1, \ldots, n_i$, the entries of $h_{i,k}$ can be computed from those of $h_{i,k-1}$ by using the formula

$$h_{i,k}(a) = \begin{cases} h_{i,k-1}(a) & \text{if } a < p_k^i \\ \min(h_{i,k-1}(a), h_{i,k-1}(a - p_k^i) + s_k^i) & \text{otherwise} \end{cases}$$

The time complexity of computing the table h_i is $O(\hat{P}n_i)$, which gives a total of $O(\hat{P}n)$ for the calculation of all the tables h_i, $\forall 1 \leq i \leq M$.

In the second stage of the algorithm, we calculate the tables f_i, $\forall 1 \leq i \leq M$. The table f_i can be calculated using the tables h_i, $\forall 1 \leq i \leq M$. We initialize

$$f_0(a, \ell) = \begin{cases} 0 & a = 0 \text{ and } \ell = 0 \\ +\infty & \text{otherwise} \end{cases}$$

and $f_i(a, 0) = +\infty$ $\forall 1 \leq i \leq M$, $\forall a > 0$. Then, for $i = 1, \ldots, M$, the entries of f_i can be computed from those of f_{i-1} by using for $\ell \in \{1, ...c\}$, $a \in \{0, ..., \hat{P}\}$ the formula

$$f_i(a, \ell) = \min \begin{cases} f_{i-1}(a, \ell) \\ \min\limits_{1 \leq a' \leq a} (f_{i-1}(a - a', \ell - 1) + h_{i,n_i}(a')) \end{cases}$$

The first line considers the case in which color i is not included in the packing. The second line considers the cases in which color i is included in the packing and contributes a' to the total value.

The value of an optimal solution for the problem is given by

$$\max_{a=0,\ldots,\hat{P};\ \ell=0,\ldots,c} \{a : f_M(a, \ell) \leq v\}.$$

The time complexity of the recursion is $O(Mc\hat{P}^2)$. Adding the time needed to construct the tables h_i, which is $O(\hat{P}n)$, we have a total of $O(Mc\hat{P}^2 + n\hat{P})$.

4.2 FPTAS for Non-Uniform Values 0-1 Knapsack Problem

By using the pseudo-polynomial algorithm above we can devise an FPTAS for instances with non-uniform values. First, we need an upper bound on the value, z^*, of an optimal solution. Such an upper bound can be obtained from the $\frac{c}{2M}$-approximation algorithm, $\mathcal{A}$ (given in Section 3.2). Let z be the profit obtained by $\mathcal{A}$. Then, clearly, $\frac{2M}{c}z$ is an upper bound on the value of z^*.

We scale the item values by replacing each value p_i by $q_i = \lceil \frac{p_i n}{\varepsilon z} \rceil$, where $1-\varepsilon$ is the required approximation ratio. Now, we need to scale accordingly the upper bound, $\hat{P}$, on the profit.

Lemma 4.2 *$\hat{P} = \frac{2M}{c}\lceil \frac{n}{\varepsilon} \rceil + n$ is an upper bound for the scaled problem.*

Proof: Recall that $\frac{2M}{c}z$ is an upper bound for the non-scaled instance. Since we pack at most n items, the rounding of the q_i's can add at most n to the total scaled value. Thus, if we can achieve profit larger than $\hat{P}$ in the scaled problem, the same items achieve profit larger than $\frac{2M}{c}\lceil \frac{n}{\varepsilon} \rceil \frac{z\varepsilon}{n} \geq \frac{2M}{c}z$ in the non-scaled instance. ∎

Finally, we apply the DP scheme and return the optimal 'scaled' solution as the approximated solution for the non-scaled instance. (In color-based instances, the scaled values are used in the calculation of f; in arbitrary-value instances, the scaled values are used in the calculation of the tables h_i). As in [19] and [2] this scaling technique yields an FPTAS for CCKP.

Theorem 4.3 *The above algorithm is an FPTAS for CCKP, whose running time is $O(Mn^2/\varepsilon)$ for instances with color-based values, and $O(\frac{1}{\varepsilon}M^2n^2(1+\frac{M}{c\varepsilon}))$ for instances with arbitrary values.*

Proof: We first show that the returned solution value is at least $(1-\varepsilon)z^*$. Let R^* and R^*_s be an optimal item set for the original and for the scaled problem, respectively. The sizes of the items and the knapsack are not influenced by the scaling, thus, R^*_s is a feasible solution for the original problem. For a set U, let $q(U) = \sum_{j\in U} q(j)$ be the total 'scaled' values of the items in U. R^*_s is optimal for the scaled problem, thus,

$$q(R^*_s) \geq q(R^*). \tag{6}$$

By definition, for any value p_i, we have,

$$\frac{z\varepsilon}{n}(q_i - 1) < p_i \leq \frac{z\varepsilon}{n}q_i. \tag{7}$$

Using Equations 6 and 7 we get that

$$p(R^*) - p(R^*_s) \leq \frac{z\varepsilon}{n}(q(R^*) - (q(R^*_s) - |R^*_s|)) \leq \frac{z\varepsilon}{n}|R^*_s| \leq z\varepsilon \leq z^*\varepsilon.$$

The time complexity is obtained by setting $\hat{P} = \frac{2M}{c}\lceil \frac{n}{\varepsilon} \rceil + n$ in the time complexities of the pseudo-polynomial algorithms. For instances with color-based values, the time complexity of the

algorithm is $O(nc\hat{P}) = O(Mn^2/\varepsilon)$. For arbitrary values we get $O(Mc\hat{P}^2+n\hat{P}) = O(\frac{1}{\varepsilon}M^2n^2(1+\frac{M}{c\varepsilon}))$. ■

For the CCMK problem it is now natural to analyze the greedy algorithm, which packs the knapsacks sequentially by applying the above FPTAS for a single knapsack to the remaining items. Recently, it was shown in [5] that this algorithm is a $(2+\varepsilon)$-approximation for the MK problem. The main idea in the proof is that for any knapsack, K_j, the set of items that are packed in K_j in some optimal placement and are not packed at all by Greedy, is available for Greedy when it fills K_j. The same argument can be used when analyzing this algorithm for the class-constrained case. Thus, the result in [5] carries over to the CCMK problem.

Theorem 4.4 *Greedy(ε) yields a $(2+\varepsilon)$-approximation for CCMK.*

Since MK is a special case of CCMK, it follows from [5] that the bound is tight; also, the performance of the algorithm cannot be improved by ordering the knapsacks in non-increasing order by their capacities.

5 Generalized Class-Constrained Packing

Recall, that in GCCP, each item $u \in I$ is associated with a set $C(u)$ of colors. Denote by $C(K_j)$ the set of colors to which the knapsack K_j allocates compartments ($|C(K_j)| \leq c_j$). Then, u can be packed in K_j iff $C(u) \cap C(K_j) \neq \emptyset$. We show that the GCCP problem is APX-hard, that is, there exists $\varepsilon_1 > 0$ such that it is NP-hard to decide whether an instance has a maximal profit P, or if every legal packing has profit at most $(1-\varepsilon_1)P$. This hardness result holds even for a single knapsack, and for instances, I, in which all the items have the same size and the same value. Moreover, for each item $u \in I$, the cardinality of $C(u)$ is a constant (at most 4).

We use an approximation-preserving reduction from the maximum 3-bounded 3-dimensional matching problem (3DM-3), defined as follows.

Input: A set of triplets $T \subseteq X \times Y \times Z$, where $|X| = |Y| = |Z| = n$; the number of occurrences of any item of $X \cup Y \cup Z$ in T is at most 3. The number of triplets is $|T| \geq n$.

Output: A 3-dimensional matching in T of maximal cardinality, i.e., a subset $T' \subseteq T$, such that any item in X, Y, Z appears at most once in T', and $|T'|$ is maximal.

Kann showed in [21] that 3DM-3 is APX-hard, that is, there exists $\varepsilon_0 > 0$ such that it is NP-hard to decide whether an instance has a matching of size n, or if every matching has size at most $(1-\varepsilon_0)n$.

Theorem 5.1 *The GCCP problem is APX-hard, even for a single knapsack and instances with uniform value and size.*

Proof: Given an instance of 3DM-3, we construct the following instance I for the generalized class-constrained 0-1 knapsack problem. The knapsack has capacity $3n$, and n compartments. There are $3n$ items: for each item i of X we have an item x_i, and similarly y_j for each $j \in Y$ and z_k for $k \in Z$. All the items $u \in I$ have the same size $s(u) = 1$, and the same value $p(u) = 1$. Let t be the cardinality of T, and denote by $e_1, \ldots, e_t$ the triplets in T. The color sets of the items are as follows.

$$C(x_i) = \{x_i\} \cup \{e_\ell | i \in e_\ell\}$$

$$C(y_j) = \{y_j\} \cup \{e_\ell | j \in e_\ell\}$$

$$C(z_k) = \{z_k\} \cup \{e_\ell | k \in e_\ell\}$$

Note that the reduction is polynomial, and since the 3DM instance is 3-bounded, $|C(u)| \leq 4$ for each $u \in I$.

Clearly, colors of type x_i, y_j, z_k can contribute at most one to the total profit, while colors of type e_ℓ can contribute at most three. For a given packing, each color of type e_ℓ which contributes three to the profit, induces a triplet in T, and since each item can be packed only once, the set of these triplets form a matching in the 3DM-3 instance.

Let t^* be the size of an optimal matching, and let p^* be the maximal profit from packing I.

Claim 5.2 *For any $\varepsilon_1 > 0$, if $p^* = 3n$ then $t^* = n$, and if $p^* \leq (1 - \varepsilon_1)3n$ then $t^* \leq n(1 - \frac{3}{2}\varepsilon_1)$.*

Proof: First, if $p^* = 3n$, then by the above discussion, each of the n compartments must be allocated to a color of type e_ℓ and a matching of size n is induced. Thus, $t^* = n$. Assume that $t^* > n - x$, for some $0 \leq x \leq n$. Then we can pack three items of each of the $n - x$ colors of type e_ℓ participating in the matching, and at least x additional items of other colors - one of each color. Thus, $p^* > 3(n - x) + x = 3n - 2x$. In particular, for $x = \frac{3}{2}n\varepsilon_1$, we get that if $p^* \leq (1 - \varepsilon_1)3n$ then $t^* \leq n(1 - \frac{3}{2}\varepsilon_1)$. ∎

Thus, if for any $\varepsilon_1 > 0$ we could decide whether a given instance, I, has maximal profit P, or if every packing has profit at most $(1 - \varepsilon_1)P$, then, taking $\varepsilon_1 = \frac{2}{3}\varepsilon_0$ we could contradict the APX-hardness of 3DM-3. ∎

Remark 5.1 *Note that the GCCP instance constructed in our reduction has $M = 3n + t$ distinct colors. The question whether GCCP is APX-hard when M is a constant remains open.*

5.1 Class-Constrained Packing with Knapsack-Dependent Colors

In this section we consider another generalization of class-constrained packing, in which the color of each item depends on the knapsack in which it is placed. That is, each item $u \in I$ has a size

$s(u)$, a profit $p(u)$ and an N-vector, $\vec{c}_u$, where $\vec{c}_u(\ell)$ is the color of u when it is placed in the knapsack K_ℓ, $1 \leq \ell \leq N$. This generalization is motivated from production systems in which the configuration required for processing a task may be different on different machines.

Theorem 5.3 *The class constrained packing problem with knapsack-dependent colors is APX-hard, even for instances with uniform value and size, and identical knapsacks.*

Proof: Given an instance of 3DM-3, we construct the following instance I for the class-constrained problem with knapsack dependent colors. Let t be the cardinality of T. There are $N = t$ knapsacks with $v_\ell = 3$, $c_\ell = 1, \forall 1 \leq \ell \leq t$. For each item i of X we have an item x_i, and similarly y_j for each $j \in Y$ and z_k for $k \in Z$. There are $2t$ *additional* items, arranged in pairs w_h^1, w_h^2, $\forall 1 \leq h \leq t$.

All the items $u \in I$ have the same size $s(u) = 1$, and the same value $p(u) = 1$.

Denote by $e_1, \ldots, e_t$ the triplets in T. The color vectors of the items are as follows.

$$\vec{c}_{x_i}(\ell) = \begin{cases} e_\ell & \text{if } i \in e_\ell \\ x_i & \text{otherwise} \end{cases} \qquad \forall 1 \leq \ell \leq N,\ 1 \leq i \leq n$$

$$\vec{c}_{y_j}(\ell) = \begin{cases} e_\ell & \text{if } j \in e_\ell \\ y_j & \text{otherwise} \end{cases} \qquad \forall 1 \leq \ell \leq N,\ 1 \leq j \leq n$$

$$\vec{c}_{z_k}(\ell) = \begin{cases} e_\ell & \text{if } k \in e_\ell \\ z_k & \text{otherwise} \end{cases} \qquad \forall 1 \leq \ell \leq N,\ 1 \leq k \leq n$$

$$\vec{c}_{w_h^1}(\ell) = \vec{c}_{w_h^2}(\ell) = w_h \qquad \forall 1 \leq \ell \leq N,\ 1 \leq h \leq t$$

That is, the ℓth triplet of T has a unique color, e_ℓ, and an item in this triplet is colored with e_ℓ if and only if it is placed in K_ℓ; otherwise, the item is colored with a unique color. In addition, each pair of the additional items has a unique color (independent of the knapsack in which it is placed).

Note that the reduction is polynomial, and since the 3DM instance is 3-bounded, each item has at most 4 distinct colors.

Let t^* be the size of an optimal matching, and let p^* be the maximal profit from packing I.

Claim 5.4 *For any $0 \leq x \leq n$, $p^* = 2t + n - x$ if and only if $t^* = n - x$.*

Proof: Since each knapsack has a single compartment, it can accommodate three items if and only if they compose a triplet in T. Also, since we can always place in a knapsack one pair of the "additional" items, in any optimal packing each knapsack includes at least two items. Thus,

there is a maximal matching of size $n - x$ iff in an optimal packing there are $n - x$ knapsacks with three items in each and $t - (n - x)$ knapsacks with two items in each. That is, $t^* = n - x$ iff $p^* = 3(n - x) + 2(t - (n - x)) = 2t + n - x$. ∎

In particular, for $x = \varepsilon_0 n$, we get that

$$p^* \in [2t + n - \varepsilon_0 n, 2t + n] \text{ if and only if } \quad t^* \in [(1 - \varepsilon_0)n, n]. \tag{8}$$

Let $\varepsilon_1 = \frac{\varepsilon_0 n}{2t+n}$. Note that $t \leq 3n$, since the number of occurrences of any item of $X \cup Y \cup Z$ in T is at most 3. Thus, ε_1 is a fixed constant.

Assume that for any $\varepsilon > 0$ we could decide whether a given instance, I, has maximal profit $2t + n$, or if every packing has profit at most $(1 - \varepsilon)(2t + n)$. In particular, this holds for ε_1. However, since $(1 - \varepsilon_1)(2t + n) = (1 - \frac{\varepsilon_0 n}{2t+n})(2t + n) = 2t + n - \varepsilon_0 n$, using (8) we can contradict the APX-hardness of 3DM-3. ∎

Acknowledgment

We would like to thank the anonymous referee, whose comments helped to improve the presentation of our results.

References

[1] N. Alon, Y. Azar, G. J. Woeginger and T. Yadid Approximation Schemes for Scheduling. In *Proc. of the 10th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 493–500, 1997.

[2] A. Caprara, H. Kellerer, U. Pferschy, and D. Pisinger. Approximation algorithms for knapsack problems with cardinality constraints. *European Journal of Operations Research*, 123:333–345, 2000.

[3] A.K. Chandra, D.S. Hirschberg, and C.K. Wong. Approximate algorithms for some generalized knapsack problems. *Theoretical Computer Science*, 3:293–304, 1976.

[4] C. Chekuri and S. Khanna. On multi-dimensional packing problems. In *Proc. of the 10th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 185–194, 1999.

[5] C. Chekuri and S. Khanna. A PTAS for the multiple knapsack problem. In *Proc. of the 11th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 213–222, 2000.

[6] E.G. Coffman, M.R. Garey, and D.S. Johnson. Approximation Algorithms for Bin Packing: A Survey. in *Approximation Algorithms for NP-hard Problems*. D.S. Hochbaum (Ed.). PWS Publishing Company, 1995.

[7] M. Dawande, J. Kalagnanam, and J. Sethuraman. Variable sized bin packing with color-constraints. Technical report, IBM, T.J.Watson Research Center, Yorktown Heights, NY 10598, 1998.

[8] L. Epstein and J. Sgall. Approximation schemes for scheduling on uniformly related and identical parallel machines. In *Proc. of the 7th European Symposium on Algorithms*, volume 1643 of *Lecture Notes in Computer Science*, pages 151–162. Springer-Verlag, 1999.

[9] C.E. Ferreira, A. Martin and, R. Weismantel. Solving multiple knapsack problems by cutting planes. *SIAM J. on Opt.*, 6:858 –877, 1996.

[10] A.M. Frieze and M.R.B. Clarke. Approximation algorithms for the m-dimensional $0-1$ knapsack problem: Worst-case and probabilistic analyses. *European Journal of Operational Research*, 15:100–109, 1984.

[11] M.R. Garey and D.S. Johnson. Strong NP-completeness results: Motivations, examples, and implications. *Journal of the ACM*, 25:499–508, 1978.

[12] M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman, 1979.

[13] G.V. Gens and E.V. Levner. Computational complexity of approximation algorithms for combinatorial problems. In *Proc. of the 8th International Symp. on Mathematical Foundations of Computer Science*, volume 74 of *Lecture Notes in Computer Science*, pages 292–300. Springer-Verlag, 1979.

[14] S. Ghandeharizadeh and R.R. Muntz. Design and implementation of scalable continuous media servers. *Parallel Computing Journal*, 24(1):91-122, 1998.

[15] L. Golubchik, S. Khanna, S. Khuller, R. Thurimella, and A. Zhu. Approximation algorithms for data placement on parallel disks. In *Proc. of the 11th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 223–232, 2000.

[16] D.S. Hochbaum and D.B. Shmoys. Using dual approximation algorithms for scheduling problems: Practical and theoretical results. *Journal of the ACM*, 34(1):144–162, 1987.

[17] M.S. Hung and J.C. Fisk. A heuristic routine for solving large loading problems. *Naval Research Logistical Quarterly*, 26(4):643–50, 1979.

[18] T. Ibaraki and N. Katoh. *Resource Allocation Problems - Algorithmic Approaches.* The MIT Press, 1988.

[19] O.H. Ibarra and C.E. Kim. Fast approximation for the knapsack and the sum of subset problems. *Journal of the ACM*, 22(4):463–468, 1975.

[20] G. Ingargiola and J. F. Korsh. An algorithm for the solution of 0-1 loading problems. *Operations Research*, 23(6):110–119, 1975.

[21] V. Kann. Maximum bounded 3-dimensional matching is max SNP-complete. *Information Processing Letters*, 37:27–35, 1991.

[22] N. Karmakar and R.M. Karp. An efficient approximation scheme for the one-dimensional bin packing problem. In *Proc. of the 23rd Symp. on Foundations of Computer Science*, pages 312–320, 1982.

[23] H. Kellerer. A polynomial time approximation scheme for the multiple knapsack problem. In *Proc. of APPROX'99*, Lecture Notes in Computer Science, pages 51–62. Springer-Verlag, 1999.

[24] H. Kellerer and U. Pferschy. Cardinality constrained bin-packing problems. *Annals of Operations Research* 92:335-348, 1999.

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

[26] E. Y-H Lin. A bibliographical survey on some well-known non-standard knapsack problems. *Information Systems and Oper. Research*, 36:4, pp. 274–317, 1998.

[27] S. Martello and P. Toth. Algorithms for knapsack problems. *Annals of Discrete Math.*, 31:213–258, 1987.

[28] R. Motwani. Lecture notes on approximation algorithms. Technical report, Dept. of Computer Science, Stanford Univ., CA, 1992.

[29] H. Shachnai and T. Tamir. On two class-constrained versions of the multiple knapsack problem. *Algorithmica.* To appear.

[30] H. Shachnai and T. Tamir. Multiprocessor scheduling with machine allotment and parallelism constraints. Manuscript, 2000.

[31] H. Shachnai and J. Turek. Multiresource malleable task scheduling. *Information Processing Letters*, 70:47–52, 1999.

[32] D.B. Shmoys and E. Tardos. An approximation algorithm for the generalized assignment problem. *Mathematical Programming A*, 62:461–474, 1993.

[33] J. Turek, J. Wolf, and P. Yu. Approximate algorithms for scheduling parallelizable tasks. In *Proceedings of the 4th Annual Symposium on Parallel Algorithms and Architectures*, pages 323–332, San Diego, CA, June 1992.

[34] W.F. Vega and G.S. Leuker. Bin packing can be solved within $1+\varepsilon$ in linear time. *Combinatorica*, 1:349–355, 1981.

[35] J.L. Wolf, P.S. Yu, and H. Shachnai. Disk load balancing for video-on-demand systems. *ACM Multimedia Systems Journal*, 5:358–370, 1997.

A Approximating CCBP by Packing Large Items First

For any $0 < \varepsilon < \frac{1}{2}$, we show the existence of instances of CCBP, for which an optimal packing of the large items (i.e., $u \in I$, such that $s(u) \geq \varepsilon$) cannot be extended to a $(1+\varepsilon)$-approximation to the optimal. We assume that $\frac{1}{\varepsilon}$ is an integer.

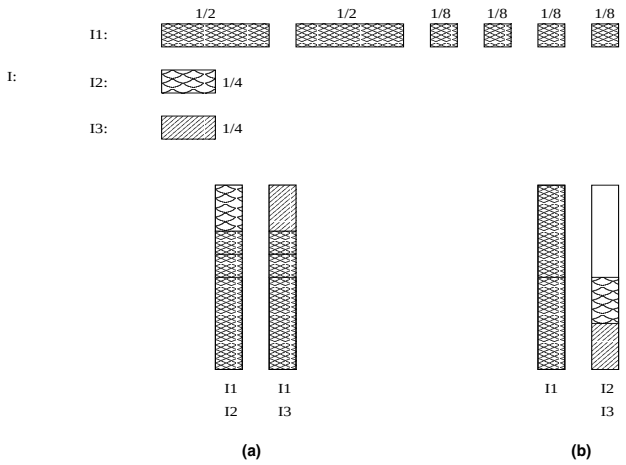

Figure 1: Packing large items first

Consider the following instance of the CCBP problem (demonstrated in Figure 1, for $\varepsilon = \frac{1}{4}$).

- $I = I_1 \cup I_2 \cup I_3$ (that is, $M = 3$). I_1 consists of two items of size $\frac{1}{2}$ and $\frac{2}{\varepsilon} - 4$ items of size $\frac{\varepsilon}{2}$; I_2 consists of one item of size ε, and I_3 consists of one item of size ε.
- The bins have capacity 1 and two compartments, that is, $c = 2$.

An optimal packing of I uses two bins (see Figure 1(a)): in each bin we pack one item of size $\frac{1}{2}$ and $\frac{1}{\varepsilon} - 2$ small items of I_1, and the single item of I_2 or I_3. The size sum of the items in each bin is therefore $\frac{1}{2} + (\frac{1}{\varepsilon} - 2)\frac{\varepsilon}{2} + \varepsilon = 1$. A possible optimal packing of the large items of I (of size $\geq \varepsilon$) is presented in Figure 1(b). The first bin is filled with the two large items of I_1 and the second bin includes the two items of I_2 and I_3. Since there is no available compartment for I_1 in the second bin, and no additional place for all the small items of I_1 in the first bin (even if we allow an overflow of ε), we need to 'open' an additional bin for the small items, resulting in a $\frac{3}{2}$ ratio.

B Performance of Greedy for the CCKP

We show that a natural greedy algorithm can perform poorly when solving the class-constrained 0-1 knapsack problem, even for instances with uniform value. Consider the following algorithm, $\mathcal{A}_1$.

1. Sort the items by sizes (small items first).
2. Repeat as long as possible: if the knapsack contains items of less than c colors, add to the knapsack the smallest available item, otherwise, add the smallest available item from one of the c colors that are included in the knapsack.

Claim B.1 *$\mathcal{A}_1$ has an unbounded approximation ratio.*

Proof: For some $\alpha > 1$, consider an instance consisting of items of $M = 2c$ different colors: c colors with one item of size $\varepsilon < \frac{v}{\alpha c}$ and additional item of size v; and c colors with α items of size $\frac{v}{\alpha c}$ each. The optimum packs $c\alpha$ items of the last c colors, while the algorithm $\mathcal{A}_1$ packs the c smallest items of the first c colors, resulting in approximation ratio of α. ∎