

# Approximation Schemes for Generalized Two Dimensional Vector Packing with Application to Data Placement\*

Hadas Shachnai<sup>†</sup>

Tami Tamir<sup>‡</sup>

## Abstract

Given is a set of items and a set of devices, each possessing two limited resources. Each item requires given amounts of the resources. Further, each item is associated with a profit and a color, and items of the same color can *share* the use of one resource. The goal is to allocate the resources to the most profitable (feasible) subset of items. In alternative formulation, the goal is to *pack* the most profitable subset of items in a set of two-dimensional bins (knapsacks), in which the capacity in one dimension is *sharable*. Indeed, the special case where there is a single item in each color is the well-known *two-dimensional vector packing (2DVP)* problem. Thus, unless  $P=NP$ , the problem that we study does not admit a *fully polynomial time approximation scheme (FPTAS)* for a single bin, and is MAX-SNP hard for multiple bins. Our problem has several important applications, including *data placement* on disks in media-on-demand systems.

We present approximation algorithms as well as optimal solutions for some instances. In some cases, our results are similar to the best known results for 2DVP. Specifically, for a single knapsack, we show that the problem is solvable in pseudo-polynomial time and develop a *polynomial time approximation scheme (PTAS)* for general instances. For a natural subclass of instances we obtain a simpler scheme. This yields the first *combinatorial PTAS* for a non-trivial subclass of instances for 2DVP. For multiple knapsacks, we develop a PTAS for a subclass of instances arising in the data placement problem. Finally, we show that when the number of distinct colors in the instance is fixed, our problem admits a PTAS, even if the items have arbitrary sizes and profits, and the bins are arbitrary.

## 1 Introduction

### 1.1 Problem Statement

Consider the following optimization problem. Given is a set of  $n$  items and a set of  $N$  devices, each possessing a limited supply of two resources. Each item requires given amounts of the resources. Further, an item is associated with a profit, that is obtained if the resources are

---

\*A preliminary version of this paper appeared in the Proceedings of the 6th International Workshop on Approximation Algorithms (APPROX), Princeton, August 2003.

<sup>†</sup>Department of Computer Science, The Technion, Haifa 32000, Israel. E-mail: [hadas@cs.technion.ac.il](mailto:hadas@cs.technion.ac.il). Part of this work was done while the author was on a leave in Bell Laboratories, Lucent Technologies, 600 Mountain Ave., Murray Hill, NJ 07974.

<sup>‡</sup>School of Computer Science, The Interdisciplinary Center, Herzliya, Israel. E-mail: [tami@idc.ac.il](mailto:tami@idc.ac.il)

allocated to that item, and a color; items of the same color can *share* the use of one of the resources. The goal is to allocate the resources to a subset of the items, subject to availability constraints, such that the overall profit is maximized.

Formally, suppose that the  $j$ -th device,  $1 \leq j \leq N$ , has  $V_j$  and  $C_j$  units from the first and second resource, respectively. Each item  $i$ ,  $1 \leq i \leq n$ , is associated with a profit,  $p_i$ . Also, item  $i$  requires  $s_i$  units from the first resource and  $c_i$  units from the second resource. We assume that the second resource can be *shared* by some items. Specifically, the instance  $I$  is partitioned into  $M$  sets, by colors; all items of the same color  $k$ ,  $1 \leq k \leq M$ , require the same amount,  $c_k$ , from the second resource and can share its use. The goal is to select a feasible most-profitable subset of items. A subset is feasible if the total allocation from the first (second) resource on the  $j$ -th device does not exceed  $V_j$  ( $C_j$ ), for  $1 \leq j \leq N$ .

In alternative formulation, the above set of items needs to be packed into  $N$  bins (knapsacks); the  $j$ -th bin has capacity  $V_j$  and  $C_j$  compartments. Each item can be packed in any of the bins. When the first item of color  $k$  is packed in some bin,  $c_k$  compartments are allocated to this *color*; additional items of color  $k$  will be accommodated in the same set of compartments. A packing is feasible if the total size of the packed items in any bin,  $j$ , is at most  $V_j$ , and the total number of compartments allocated in bin  $j$  is at most  $C_j$ ,  $1 \leq j \leq N$ . The goal is to pack a subset of the items of maximum total profit.

Indeed, the special case where there is a single item from each color is the well-known *two-dimensional vector packing problem (2DVP)*. Thus, our problem does not admit an FPTAS for a single bin [14] and is MAX-SNP hard for multiple bins, already in the case where the bins are identical, and the items have unit profits, i.e.,  $p_i = 1 \forall i, 1 \leq i \leq n$  [20]. We call this problem *vector packing with a sharable dimension (VPSD)*.

## 1.2 Applications

A natural application of VPSD is *data placement* on disks in media-on-demand systems [8, 11]. In such systems (see, e.g., [22, 9]), a large database of  $M$  video program files is stored on a centralized server. Each program file,  $k$ ,  $1 \leq k \leq M$ , is associated with a number of desired broadcasts of this file,  $n_k$ , and a size (storage requirement),  $c_k$ . The files are stored on  $N$  shared disks. Each disk,  $j$ , is characterized by (i) its storage capacity,  $C_j$ , that is, the total size of the files that can reside on it, and (ii) its load capacity,  $V_j$ , which is the number of data streams that can be read simultaneously from that disk. The files need to be placed on the disks so as to maximize the total number of requests for broadcasts that can be satisfied simultaneously. In the resulting instance of VPSD, the bins represent disks, and the items are broadcast requests. To satisfy a request, some disk has to broadcast a data stream to the client. This disk must hold a copy of the requested file. Note that storage is a shared resource – that can be used by all the streams broadcasting the same data from the same disk. Different files may have different sizes, thus, they may have different  $c_k$  values. On the other hand, all the broadcast streams require the same (non-sharable) bandwidth; thus,  $\forall k \ s_k = 1$ .

Another application is *distributed caching* [4], in which the items to be packed are requests

to access stored data. Requests of the same type (color) need to access the same data and can therefore share the storage allocated to this data. On the other hand, each request requires non-sharable bandwidth.

Other applications of VPSD are production planning and scheduling parallel tasks (see in [18]). Of particular interest in our study is the subclass of *uniform profit/size ratio* instances of VPSD, in which for some  $\alpha > 0$ ,  $\forall i p_i = \alpha s_i$ . Such instances naturally arise in real systems, where client payments for service (item profits) are proportional to the amounts of resources consumed (item sizes).

### 1.3 Our Results

In the following we summarize our main results.

- For a single knapsack, we give
  - An optimal pseudo-polynomial time algorithm.
  - An LP-based PTAS for general instances (see Section 2.2).
  - For the subclass of uniform profit/size ratio instances, a simpler approximation scheme, that is based on extension of a PTAS proposed in [16] for the classical knapsack problem (see Section 3). By this, we obtain the first *combinatorial* PTAS for a non-trivial subclass of instances for 2DVP.
  - An FPTAS for the subclasses of (i) instances with constant number of compartment requirements, and (ii) data placement instances (see Section 4).
- For multiple knapsacks,
  - We show (in Section 5) that an iterative greedy algorithm achieves the ratio of  $(2+\varepsilon)$  for instances with arbitrary bin sizes, and  $(\frac{e}{e-1} + \varepsilon)$  when the bins are identical.
  - We develop (in Section 6) a PTAS for data placement instances in which the disks are identical, and the number of distinct file sizes is fixed.
  - For instances in which  $M$ , the distinct number of colors, is fixed, we show (in Section 7) that VPSD admits a PTAS, even if the items have arbitrary sizes and profits, and the bins are arbitrary.

### 1.4 Related Work

Packing problems in single dimension have been extensively studied. Since these problems are NP-hard, most of the research work in this area focused on finding approximation algorithms. The classic *0-1 knapsack problem* admits an FPTAS; that is, for any  $\varepsilon > 0$ , a  $(1 - \varepsilon)$ -approximation for the optimal solution can be found in  $O(n/\varepsilon^2)$  steps [10, 7]. In contrast, the *multiple knapsack (MK)* problem is known to be strongly NP-hard [6]. Chekuri and Khanna developed in [3] a PTAS for MK and showed that with slight generalizations this problem becomes APX-hard.

Packing problems in higher dimensions (also known as *d-dimensional vector packing*) are known to be substantially harder to solve, exactly or approximately. The best known result for a single knapsack is a PTAS due to Frieze and Clarke [5], for the case where  $d$  is a fixed constant. As opposed to the combinatorial schemes for the single dimension case, the PTAS in [5] uses as a procedure a *linear program*. To the best of our knowledge, none of the later published work on the  $d$ -dimensional knapsack problem gives a *combinatorial* scheme, even for the case where  $d = 2$ . Recently, Fleischer et al. [4] designed a PTAS for VPSD in a single knapsack. The scheme of [4] simplifies the PTAS in Section 2.2 and handles also the fractional version of the problem.

For the case of  $N > 1$  bins, Woeginger showed in [20] that two-dimensional vector packing is MAX-SNP hard (see also in [2]). Chekuri and Khanna presented in [2] a PTAS for the *vector scheduling* problem, in which our goal is to schedule a set of jobs, given by  $d$ -dimensional vectors, on a set of machines, so as to minimize the maximum completion time (or makespan) over all dimensions. The scheme in [2] yields a *dual* PTAS for  $d$ -dimensional vector packing in  $N \geq 1$  bins, where the bins have  $d$  equal-sized dimensions, and  $d$  is a fixed constant. The *class constrained multiple knapsack (CCMK)* problem introduced in [18] is a special case of VPSD, where  $c_k = 1$  for all  $1 \leq k \leq M$ . The paper [18] presents a PTAS for any instance of CCMK in which  $M$ , the number of distinct colors of items, is fixed.

The data placement problem was initially studied in [17]. The paper presents an algorithm for the case where all the files are of the *same* (unit) size, and for all  $1 \leq j \leq N$ , the ratio  $V_j/C_j$  is equal (*uniform ratio disks*). The paper shows that the algorithm achieves a ratio of  $1 - 1/(1 + C_{min})$  to the optimal, where  $C_{min} = \min_j C_j$ . Golubchik et al. gave in [8] a tighter analysis of this algorithm and showed that it achieves the ratio  $1 - 1/(1 + \sqrt{C_{min}})^2$ , and that this ratio is optimal for *any* algorithm for this problem. The paper [8] also presents a PTAS for the data placement problem with unit sized files and uniform ratio disks. Kashyap and Khuller [11] studied the problem with files of  $\Delta$  distinct sizes, where  $\Delta$  is fixed. They presented an algorithm that achieves a ratio of  $\frac{C-\Delta}{C+\Delta} \left( 1 - 1/(1 + \sqrt{\frac{C}{2\Delta}})^2 \right)$ , where file sizes are in  $\{1, \dots, \Delta\}$ , and  $C$  is the storage capacity of the disks. They also showed that this algorithm can be combined with an algorithm that runs in polynomial time when  $C$  is fixed, to get a PTAS for the data placement problem with constant number of file sizes.

Other related works deal with media-on-demand systems. Dynamic algorithms for the data placement problem were suggested in [22]. In [15] the load on the disks is balanced dynamically by replications and deletions of media-files. Better performance of the MOD system can be achieved also by data sharing techniques. i.e., data retrieved for one request can be used to service additional requests. This can be done by batching (see e.g., in [21]) or buffering [13, 1]. Each of these techniques can be applied independently of the data placement scheme, to improve the overall performance of the system.

## 2 Algorithms for a Single Bin

### 2.1 A Pseudo-polynomial Time Algorithm

In this section, we discuss the single knapsack version of VPSD. We first describe a pseudo-polynomial time algorithm,  $\mathcal{A}$ , which solves optimally VPSD for a single knapsack. Suppose that the knapsack has capacity  $V$  and  $C$  compartments. The algorithm is based on a two-stage dynamic programming procedure. Let  $n_k$  denote the number of items of color  $k$ , and assume that  $p_{max} = \max_{1 \leq i \leq n} p_i$  is the maximum profit of any item, and  $\hat{P} \leq \sum_i p_i$  is an upper bound on the optimal profit.

- In the first stage, the values  $h_{k,r}(a)$  are computed recursively;  $h_{k,r}(a)$  denotes the minimum total size of a subset of items, out of the first  $r$  in color  $k$ , such that the total profit of the subset is  $a$ , where  $1 \leq k \leq M$ ,  $1 \leq r \leq n_k$ , and  $0 \leq a \leq \hat{P}$ .
- In the second stage, the values of  $h_{k,r}(a)$  are used to compute recursively  $f_k(a, \ell)$ , the minimum total size of a subset of items whose colors are among the first  $k$ ,  $1 \leq k \leq M$ , such that the items use  $\ell$  compartments,  $1 \leq \ell \leq C$ , and the total profit of the items is  $a$ ,  $1 \leq a \leq \hat{P}$ .

Formally, the calculation of the values  $h_{k,r}(a)$  is done by initializing

$$h_{k,r}(a) = \begin{cases} 0 & a = 0 \\ +\infty & -p_{max} \leq a < 0 \\ +\infty & r = 0, 1 \leq a \leq \hat{P} \end{cases}$$

Then, for  $k = 1, \dots, M$ , the entries of  $h_{k,r}$  can be computed from those of  $h_{k,r-1}$  by using for  $r \in \{1, \dots, n_k\}$ ,  $a \in \{0, \dots, \hat{P}\}$  the formula

$$h_{k,r}(a) = \min \begin{cases} h_{k,r-1}(a) \\ s_r^k + h_{k,r-1}(a - p_r^k) \end{cases}$$

where  $s_r^k, p_r^k$  are the size and profit of the  $r$ -th item of color  $k$ .

For calculation of the values  $f_k(a, \ell)$ , initially set

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

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

$$f_k(a, \ell) = \min \begin{cases} f_{k-1}(a, \ell) \\ \min_{1 \leq a' \leq a} (f_{k-1}(a - a', \ell - c_k) + h_{k,n_k}(a')) \end{cases}$$

The first line covers the case where no item of color  $k$  is added to the solution; in the second line, we add items of color  $k$  and find the minimum total size of the packed items, considering all possible contributions  $1 \leq a' \leq a$  of items in color  $k$  to the overall profit.

The optimal solution is given by

$$\max_{a=0,\dots,P; \ell=1,\dots,C} \{a : f_M(a, \ell) \leq V\}.$$

Hence, we have

**Theorem 2.1** *VPSD can be solved optimally in  $O(n\hat{P} + M\hat{P}^2C)$  steps.*

Note that if  $C$ , the number of compartments in the knapsack, is polynomial in the input size, then the above yields an FPTAS for the problem.

## 2.2 Approximation Scheme for a Single Bin

In this section, we describe a PTAS for a single knapsack. Note that, by the result of [14], this is the best we can expect since the problem does not admit an FPTAS. Assume that the optimal profit,  $P$ , for our instance is known. We reduce our problem to the *binary 2-dimensional multiple choice knapsack* (B2D-MCK) problem. That is, for given values of  $P$  and  $\varepsilon$ , we define an instance for B2D-MCK, whose optimal solution induces a solution for VPSD with profit at least  $(1 - \varepsilon)P$ . We then develop a PTAS for the B2D-MCK problem. By combining the reduction and the PTAS for B2D-MCK, we get a PTAS for VPSD. Note that  $P$  can be ‘guessed’ in polynomial time within factor  $(1 + \varepsilon)$ , using binary search over the range  $(\max_i p_i, \sum_i p_i)$ .

### Reduction to Binary 2-dimensional Multiple Choice Knapsack

Recall that an instance of B2D-MCK consists of a single 2-dimensional knapsack and  $M$  sets of items. Each item has a 2-dimensional size and is associated with a profit. The goal is to pack a subset of items of maximal total profit. A packing is feasible if it does not exceed the volume in any dimension, and at most one item is packed from each set.

Given the value of  $P$ , the parameter  $\varepsilon$  and a VPSD instance with  $n$  items of  $M$  distinct colors, construct a B2D-MCK instance consisting of a single 2-dimensional knapsack with capacities  $b_1 = V$  and  $b_2 = C$ , and  $M$  sets of items; each set  $S_k$  has  $R = M/\varepsilon$  items,  $1 \leq k \leq M$ . Each of the items in  $S_k$  represents a subset of the items in the VPSD instance which are of color  $k$ , and whose total profit is rounded down to the nearest integral multiple of  $\varepsilon P/M$ . In particular, the  $j$ th item in  $S_k$ , denoted as  $(k, j)$ , is given by the triple  $(s_{kj}, c_k, p(k, j))$ :  $s_{kj}$  is the minimal total size of a subset of items in color  $k$  whose total profit is  $p(k, j) = (j\varepsilon P)/M$ . This total size can be computed using dynamic programming for the items of  $S_k$  with the rounded profits (as in the FPTAS for the classic knapsack problem [10]).

**Lemma 2.2** *If there exists a solution with profit  $P$  for the VPSD instance, then there exists a solution with profit at least  $(1 - \varepsilon)P$  for the binary B2D-MCK instance.*

**Proof:** Given an optimal solution for VPSD with profit  $P$ , assume that the contribution of color  $k$  is in the range  $[\frac{j\varepsilon P}{M}, \frac{(j+1)\varepsilon P}{M})$ ; then, in the input for B2D-MCK, the profit of the corresponding item in  $S_k$  is rounded down to  $j\varepsilon P/M$ . Due to similar rounding of the profits contributed by other colors, the total loss in profit is at most  $\varepsilon P$ .  $\square$

## Approximating the Optimal Solution for B2D-MCK

Given an instance of B2D-MCK, ‘guess’ the set  $S$  of most profitable items in the optimal solution, where  $|S| = h = \min(M, \lfloor \frac{4(1-\varepsilon)}{\varepsilon} \rfloor)$ . Let  $E(S)$  be the subset of items with profits that are larger than the minimal profit of any item in  $S$ , that is,  $E(S) = \{(k, j) \notin S \mid p(k, j) > p_{\min}(S)\}$ , where  $p_{\min}(S) = \min_{(k, j) \in S} p(k, j)$ .

All the items  $(k, j) \in S$  are packed, and all the items  $(k, j) \in E(S)$  as well as the sets  $S_k$  from which an item has been selected, are eliminated from the instance. In the next step we find an optimal *basic solution* for the following linear program.

$$\begin{aligned}
 (LP(S)) \quad \max \quad & \sum_{k=1}^M \sum_{j=1}^R p(k, j) x_{kj} \\
 \text{s.t.} \quad & \sum_{j=1}^R x_{kj} \leq 1 \text{ for } k = 1, \dots, M \\
 & \sum_{k=1}^M \sum_{j=1}^R x_{kj} s_{kj} \leq V \\
 & \sum_{k=1}^M c_k \sum_{j=1}^R x_{kj} \leq C \\
 & x_{kj} = 1 \text{ for } (k, j) \in S, \text{ and } x_{kj} = 0 \text{ for } (k, j) \in E(S) \\
 & 0 \leq x_{kj} \leq 1 \text{ for } k = 1, \dots, M; \quad j = 1, \dots, R; \quad (k, j) \notin S \cup E(S)
 \end{aligned}$$

Given an optimal fractional solution, an integral solution is produced by rounding down to 0 the fractional variables in the solution. The output for B2D-MCK consists of the items in  $S$  and the items  $(k, j)$  for which  $x_{kj} = 1$ .

**Theorem 2.3** *The above scheme achieves a ratio of  $(1 - \varepsilon)$  to the optimal B2D-MCK profit.*

**Proof:** Let  $\mathbf{x}^*$  be an optimal solution for the linear program  $LP(S)$ , and let  $S^*$  be the corresponding subset of items, that is,  $S^* = \{(k, j) \mid x_{kj}^* = 1\}$ . If  $|S^*| < h$  then we are done (the scheme outputs the optimal solution for the B2D-MCK instance, since we guess  $S^*$ ); otherwise, let  $S^* = \{(k_1, j_1), \dots, (k_r, j_r)\}$ , such that  $p(k_1, j_1) \geq \dots \geq p(k_r, j_r)$ . Let  $S_h^* = \{(k_1, j_1), \dots, (k_h, j_h)\}$ , and  $\sigma = \sum_{\ell=1}^h p(k_\ell, j_\ell)$ . Then, for any item  $(k, j) \notin (S_h^* \cup E(S_h^*))$ , it holds that  $p(k, j) \leq \sigma/h$ . Let  $z^*, \hat{z}$  denote the optimal solution and the solution output by the scheme, respectively. Denote by  $\mathbf{x}^B(S_h^*), \mathbf{x}^I(S_h^*)$  the basic and integral solutions of  $LP(S)$  as computed by the scheme for the initial guess  $S_h^*$ . It holds that

$$z^* \leq \sum_{k=1}^M \sum_{j=1}^R p(k, j) x_{kj}^B(S_h^*) \leq \sum_{k=1}^M \sum_{j=1}^R p(k, j) x_{kj}^I(S_h^*) + \delta, \quad (1)$$

where  $\delta = \sum_{(k, j) \in F} p(k, j)$ , and  $F$  is the set of items for which the basic variable was a fraction, that is,  $F = \{(k, j) \mid x_{kj}^B(S_h^*) > x_{kj}^I(S_h^*)\}$ .

Recall that in any *basic* solution for a linear program, the number of non-zero variables is bounded by the number of tight constraints in some optimal solution (since non-tight constraints can be omitted). Assume that in the optimal (fractional) solution of  $LP(S_h^*)$  there are  $L$  tight constraints, where  $0 \leq L \leq M + 2$ . Then in the basic solution  $\mathbf{x}^B(S_h^*)$ , at most  $L$  variables can be strictly positive. On the other hand, if there are  $L$  tight constraints, then at least for  $L - 2$  sets  $S_k$ , the sum of variables with positive values is *exactly* 1. Thus,  $F$  gets the maximum size when  $L - 4$  variables are assigned the value ‘1’, and 4 variables (associated with two sets) are assigned some fractional values in  $(0, 1)$ , i.e.,  $|F| \leq 4$ . Note that  $\delta < 4\sigma/h$ , since  $F \cap (S_h^* \cup E(S_h^*)) = \emptyset$ . Hence, from (1), it follows that  $z^* \leq \hat{z} + \frac{4\sigma}{h} \leq \hat{z} + \frac{4\hat{z}}{h} \leq \frac{\hat{z}}{1-\varepsilon}$ .  $\square$

We summarize in the next result.

**Theorem 2.4** *There is a PTAS for VPSD with a single bin.*

**Proof:** We first show that the above scheme yields a  $(1 + \varepsilon)$ -approximation to the optimum. Suppose that the optimal value is  $P$ , then  $P$  can be guessed to within factor  $(1 - \varepsilon)$ . Let  $P'$  be the guessed value, then, by Lemma 2.2, there exists an optimal value for the B2D-MCK constructed by the scheme whose value satisfies  $OPT_{B2D-MCK} \geq (1 - \varepsilon)P$ . By Theorem 2.3, our scheme outputs for the B2D-MCK instance a solution whose value is at least

$$(1 - \varepsilon)OPT_{B2D-MCK} \geq (1 - \varepsilon)^3 P.$$

Hence, by taking in the scheme  $\varepsilon' = \varepsilon/3$  we get a  $(1 + \varepsilon)$ -approximation.

To analyze the running time of the scheme, note first that the linear program can be solved in time that is polynomial in  $n$ . In addition, the subset  $S_h^*$  can be guessed in  $O(n^{O(1/\varepsilon)})$  steps, therefore we get a PTAS.  $\square$

### 3 Single Bin and Items with Uniform Profit/Size Ratio

In this section we present algorithms for instances with uniform profit/size ratio, that is, for some  $\alpha > 0$ ,  $\forall i, p_i = \alpha s_i$ . The goal is to pack in a single bin a subset of the items whose total size is as large as possible. We show that for this case the simple greedy algorithm and PTAS for classic 0/1 knapsack can be extended to apply for VPSD. We note that these or similar extensions do not fit for arbitrary instances. In both algorithms, the idea is to partly-reduce VPSD to knapsack, by first considering all items of each color as a single item; later on, these items are separated back to the original items.

W.l.o.g. we assume that  $\forall k, c_k \leq C$ ,  $\forall i, s_i \leq V$  (otherwise, omit from the instance items that cannot be packed). Also, we assume that  $M, C$  are given as part of the input. If any of the parameters  $V, C$  or  $M$  is fixed, then the problem reduces to the classic knapsack problem: If  $M$  is fixed, the subset of colors accommodated in the knapsack can be found by exhaustive enumeration of all possible subsets of colors whose total compartment requirement is at most  $C$ . This is done in  $O(2^M)$  steps, which is a constant. If  $C$  is fixed then the set of at most  $C$  distinct colors in an optimal solution can be found in  $O(CM^C)$  steps, which is polynomial. Then solve the Knapsack problem where the input is the subset of items of the selected colors.

If  $V$  is a constant then only a constant number of items can be packed in the knapsack and the problem is solvable in  $O(n^V)$  steps.

### 3.1 A Greedy 2-approximation Algorithm

Let  $S_k$  be the total size of items with color  $k$ ,  $1 \leq k \leq M$ . Consider the following greedy algorithm  $\mathcal{A}_G$ .

1. Sort the colors such that  $S_1/c_1 \geq S_2/c_2 \geq \dots \geq S_M/c_M$ .
2. Determine the set of colors which are allocated compartments: these are the first  $j$  colors in the sorted list such that  $\sum_{k=1}^j c_k \leq C$ , and  $\sum_{k=1}^{j+1} c_k > C$ . Let  $A$  be the set of all items in the selected colors.
3. Pack the items of  $A$  in the knapsack from largest to smallest, ignoring colors, while there is enough space. Let  $a_1$  denote the total size of items packed this way.
4. Let  $k^*$  be the color with maximal total size. Let  $a_2$  be the total size of items that are packed from color  $k^*$  when adding items greedily, from largest to smallest, as long as there is enough space.
5. Select (and pack accordingly) the maximum between  $a_1$  and  $a_2$ .

**Theorem 3.1**  $\mathcal{A}_G$  yields a 2-approximation for uniform-ratio instances.

**Proof:** If the total size of items in color  $k^*$  is more than  $V$ , then  $a_2 > V/2$ , otherwise,  $a_2 = S_{k^*}$ . If  $a_2 > V/2$ , we are done (since  $OPT \leq V$ ). Consider the case that  $a_2 = S_{k^*}$ . Since the colors are sorted by profit/compartments ratio,  $OPT < S_1 + \dots + S_{j+1}$ . If in step 3 all the items of  $A$  are packed, then  $a_1 = S_1 + \dots + S_j$ . The total size packed by  $\mathcal{A}_G$  is  $S(\mathcal{A}_G) = \max(a_1, a_2) = \max(S_1 + \dots + S_j, S_{k^*}) \geq \frac{1}{2}(S_1 + \dots + S_j + S_{k^*}) \geq \frac{1}{2}(S_1 + \dots + S_{j+1}) \geq \frac{1}{2}OPT$ . If in step 3 only part of the items are packed, then since they are packed from largest to smallest at least half of the bin is filled, which is at least  $\frac{1}{2}OPT$ .  $\square$

### 3.2 Approximation Scheme

We now describe a PTAS for the uniform ratio case. Our scheme extends the PTAS of Sahni [16] for the classical knapsack problem. Let  $k_1, k_2$  be constants (to be determined). Algorithm  $\mathcal{A}$  proceeds as follows. For any possible selection of at most  $k_1$  items from  $I$ , and for any possible selection of at most  $k_2$  colors among those that do not appear in the  $k_1$  items, do the following.

1. Let  $V'$  be the remaining volume ( $V'$  equals  $V$  minus the total size of the  $k_1$  items). Let  $C'$  be the remaining number of compartments ( $C'$  equals  $C$  minus the total compartment demand of the  $k_2$  colors and the  $k_1$  items).
2. If this selection of items and colors is infeasible (that is,  $V' < 0$  or  $C' < 0$ ) stop; otherwise,

3. Let  $T$  be the set of the  $k_2$  selected colors and the colors of the  $k_1$  items.
  - (a) Pack the  $k_1$  items.
  - (b) Add the other items of the  $T$  colors in arbitrary order as long as there is enough space.
  - (c) If there is no space while adding these items, terminate with the packed items; otherwise,
  - (d) Sort the colors that do not belong to  $T$  such that  $S_1/c_1 \geq S_2/c_2 \geq \dots$
  - (e) Add items of color  $c_1$  in arbitrary order, then items of color  $c_2$  and so on, as long as there are enough space and enough compartments.

**Theorem 3.2** For all  $k_1, k_2$ ,  $\mathcal{A}$  has approximation ratio  $R_{\mathcal{A}} \leq 1 + \frac{1}{\min(k_1, k_2)}$  and running time  $O(n^{k_1+1} \cdot M^{k_2})$ .

**Proof:** The number of possible selections of  $k_1$  items and  $k_2$  colors is  $O(n^{k_1} \cdot M^{k_2})$ . For each such subset, the amount of work done by  $\mathcal{A}$  is  $O(n)$ , assuming there is a single preprocessing of sorting the color-sets. We now turn to analyze the approximation ratio of  $\mathcal{A}$ .

Let  $OPT$  be any optimal solution. If  $|OPT| \leq k_1$  we are done, since the solution of  $OPT$  will be considered at some iteration. Let  $H = \{a_1, a_2, \dots, a_{k_1}\}$  be the set of  $k_1$  most profitable items in  $OPT$ . Let  $W$  be the set of at most  $k_2$  most profitable colors in  $OPT$ , among those that are not presented in  $H$ . There exists an iteration of  $\mathcal{A}$  in which  $H$  and  $W$  are considered. We show that the profit gained by  $\mathcal{A}$  in this iteration yields the statement of the theorem. Consider the list  $L_1 = OPT \setminus H = \{a_{k_1+1}, \dots, a_x\}$  of the remaining items of  $OPT$ , in the order they are considered by  $\mathcal{A}$ . By the algorithm, the items of  $OPT$  whose colors are in  $T$  appear first in  $L_1$ , then the items of  $OPT$  whose color has the maximal  $S_j/c_j$  ratio and so on. The internal order of the items whose colors are in  $T$  and in each color-set is arbitrary.

Recall that at some point  $\mathcal{A}$  will try  $H$  as the initial set of  $k_1$  packed items. The algorithm will then add greedily the remaining items, as long as the volume and compartment constraints allow. If all the items are packed,  $\mathcal{A}$  is clearly optimal. Otherwise, at some point there is not enough space for the next item, or there are not enough compartments for the next color-set. We distinguish between these two cases.

**Claim 3.3** If  $\mathcal{A}$  stops due to space constraint, then  $R_{\mathcal{A}} \leq 1 + 1/k_1$ .

**Proof:** Let  $m$  be the index of the first item in  $L_1$  which is not placed into the knapsack by  $\mathcal{A}$ , i.e. items  $a_{k_1+1}, \dots, a_{m-1}$  are packed. We assume that  $a_m$  is not placed in the knapsack because  $V_f$ , the remaining empty space at that point, is smaller than  $s_m$ . At this time, when  $a_m$  is rejected, the knapsack contains the items from  $H$ , the items  $a_{k_1+1}, \dots, a_{m-1}$  and some items which are not in  $OPT$ .

Let  $G$  denote the set of items that are placed in the knapsack so far by the greedy stage of  $\mathcal{A}$  (Steps (b)-(e)). These are all the items added to the knapsack up to this point that are not in  $H$ . The items in  $G \setminus OPT$  are of total size

$$\Delta = V - (V_f + \sum_{i=1}^{m-1} s_i). \quad (2)$$

Thus, the total size of the items in  $G$  is  $S_G \geq \sum_{i=k_1+1}^{m-1} s_i + \Delta$ . We conclude that the total size of the items in  $OPT$  is

$$\begin{aligned} S(OPT) &= \sum_{i=1}^x s_i = \sum_{i=1}^{k_1} s_i + \sum_{i=k_1+1}^{m-1} s_i + \sum_{i=m}^x s_i \\ &\leq S_H + (S_G - \Delta) + (V - \sum_{i=1}^{m-1} s_i) \\ &= S_H + S_G + V_f < S_H + S_G + s_m \end{aligned}$$

The last equality follows from (2). Since  $\mathcal{A}$  packs at least  $H \cup G$ , we get that  $S(\mathcal{A}) \geq S_H + S_G$  which implies  $S(OPT) - S(\mathcal{A}) < s_m$ . Given that there are at least  $k_1$  items with a profit at least as large as  $a_m$  (those selected for  $H$ ), we get that  $s_m \leq S(OPT)/(k_1 + 1)$ . This gives the approximation ratio.  $\square$

**Claim 3.4** *If  $\mathcal{A}$  stops due to compartment constraint, then  $R_{\mathcal{A}} \leq 1 + 1/k_2$ .*

**Proof:** Let  $C(OPT)$  denote the set of colors participating in  $OPT$ . Recall that  $W = \{b_1, b_2, \dots, b_{k_2}\}$  is the set of  $k_2$  most profitable colors in  $C(OPT)$ , and that  $T = W \cup \{b_{k_2+1}, \dots, b_t\}$  is the set of colors of  $W$  and the colors presented in the  $k_1$  most profitable items. We know that the total compartment demand of the items of colors in  $T$  is at most  $C$  (otherwise,  $\mathcal{A}$  would have been terminated in Step 2). Also, since  $\mathcal{A}$  does not terminate due to volume constraint (this is handled in Claim 3.3), it must be that  $\mathcal{A}$  packs all the items of colors in  $T$ . By our selection of  $H$  and  $W$  we know that  $T \subseteq C(OPT)$ .

Consider the list  $L_2 = \{b_{t+1}, \dots, b_z\}$  of the remaining colors in  $C(OPT)$  in the order they are considered by  $\mathcal{A}$  in Step (d). That is,  $b_{t+1}$  is the color from  $C(OPT) \setminus T$  having the maximal  $S_j/c_j$  ratio, and so on. Let  $j$  be the index of the first color in  $L_2$  for which there are not enough compartments in the knapsack, i.e. all the items of colors in  $b_{t+1}, \dots, b_{j-1}$  are packed. That is, the first item of color  $b_j$  is rejected because  $C_f$ , the number of remaining available compartments at that point, is smaller than  $c_j$ . At this time of the rejection, the knapsack contains all the items of colors in  $T$ , all the items of colors  $b_{t+1}, \dots, b_{j-1}$  and all the items of additional colors which are not in  $C(OPT)$ .

Let  $C(G)$  denote the colors that are placed in the knapsack so far by the greedy stage of  $\mathcal{A}$  (Steps (d)-(e)). These are all the colors added to the knapsack up to this point that are not in  $T$ . The colors in  $C(G) \setminus C(OPT)$  have total compartment request

$$\Gamma = C - (C_f + \sum_{i=1}^{j-1} c_i). \quad (3)$$

Moreover, each of these colors has profit per compartment ratio at least  $S_j/c_j$  since it was considered earlier by the greedy stage of  $\mathcal{A}$ . It follows that the total size of packed items of the colors in  $C(G)$  is

$$S_{C(G)} \geq \sum_{i=t+1}^{j-1} S_i + \Gamma \frac{S_j}{c_j}. \quad (4)$$

Consider now the colors  $b_j, \dots, b_z$  that are in  $C(OPT)$  but are not allocated compartments by  $\mathcal{A}$ . For each of these colors  $i$ ,  $S_i/c_i \leq S_j/c_j$  (by the sorting in Step (d)). Thus, the total size of items in these colors is

$$\sum_{i=j}^z S_i = \sum_{i=j}^z \frac{S_i c_i}{c_i} \leq \frac{S_j}{c_j} \sum_{i=j}^z c_i.$$

Note that, since  $OPT$  is feasible,  $\sum_{i=1}^z c_i \leq C$ . Using (3), it follows that

$$\sum_{i=j}^z c_i \leq C - \sum_{i=1}^{j-1} c_i = \Gamma + C_f. \quad (5)$$

We conclude that the total size of the items in colors of  $C(OPT)$  is

$$\begin{aligned} S(C(OPT)) \leq \sum_{i=1}^z S_i &= \sum_{i=1}^t S_i + \sum_{i=t+1}^{j-1} S_i + \sum_{i=j}^z S_i \\ &\leq S_T + (S_{C(G)} - \Gamma \frac{S_j}{c_j}) + \frac{S_j}{c_j} (\Gamma + C_f) \\ &= S_T + S_{C(G)} + C_f \frac{S_j}{c_j} \\ &< S_T + S_{C(G)} + S_j \end{aligned}$$

The second inequality follows from (4) and (5), and the last inequality follows from the fact that  $c_j > C_f$ . Clearly,  $S(OPT) \leq S(C(OPT))$  since in  $OPT$  only a subset of the items from each color in  $C(OPT)$  may be packed. Now, since  $\mathcal{A}$  packs at least all the items in colors of  $T \cup C(G)$ , it holds that  $S(\mathcal{A}) \geq S_T + S_{C(G)}$  which implies  $S(OPT) - S(\mathcal{A}) < S_j$ . Since  $T$  contains at least the  $k_2$  most profitable colors in  $OPT$ , the total contribution to  $OPT$  of the items with color  $j$  is at most  $OPT/(k_2 + 1)$ . This gives the approximation ratio.  $\square$

Combining Claims 3.3 and 3.4 we get the statement of the theorem.  $\square$

A PTAS is obtained by selecting  $k_1 = k_2 = 1/\varepsilon$ .

Consider the subclass of 2DVP instances in which the size of any item  $i$  in each dimension is arbitrary, and the profit  $p_i$  is proportional to the size in one dimension. For such instances, we have a combinatorial approximation scheme, as formalized in the next result.

**Corollary 3.5** *Algorithm  $\mathcal{A}$  is a PTAS for uniform profit/size ratio instances of 2DVP.*

## 4 Better Algorithms for Special Instances

This section shows that better approximations or more efficient algorithms can be obtained for several subclasses of instances. In Section 4.1, we discuss inputs in which the compartment requirement of any color can take one of  $w$  values,  $\eta_1, \dots, \eta_w$ , where  $w$  is a fixed constant. In Section 4.2, we discuss inputs for the data placement problem. For both classes of inputs we give a pseudo-polynomial time algorithms, which are then transformed into FPTASs.

### 4.1 Constant Number of Compartment Requirements

**Theorem 4.1** *If the compartment requirement of any color class can take one of the values  $\eta_1, \dots, \eta_w$ , for some fixed  $w \geq 1$ , then an optimal solution can be computed in  $O(n\hat{P} + M^w \hat{P}^2)$ ,*

where  $\hat{P}$  is an upper bound on the total profit.

**Proof:** For finding an optimal packing of the items, the following two-stage dynamic programming procedure is used.

- In the first stage, compute recursively (as in Section 2.2) the values of  $h_{k,r}(a)$ , for  $1 \leq k \leq M$ ,  $1 \leq r \leq n_k$ , and  $1 \leq a \leq \hat{P}$ . Recall that  $h_{k,r}(a)$  denotes the minimum total size of a subset of items, out of the first  $r$  in color  $k$ , such that the total profit of the subset is  $a$ .
- In the second stage, use the values of  $h_{k,r}(a)$  to compute recursively  $f_k(a, d_1, \dots, d_w)$ , the minimum size sum of a subset of items whose colors are among the first  $k$ ,  $1 \leq k \leq M$ , such that  $d_\ell$  colors from compartment category  $\ell$ ,  $1 \leq \ell \leq w$ , are used, and the total profit of the items is  $a$ ,  $1 \leq a \leq \hat{P}$ . Initialize

$$f_0(a, d_1, \dots, d_w) = \begin{cases} 0 & a = 0 \text{ and } d_1 = \dots = d_w = 0 \\ +\infty & \text{otherwise} \end{cases}$$

Assume that the  $k$ -th color has compartment requirement  $\eta_r$ , for some  $1 \leq r \leq w$ , then use the following formula.

$$f_k(a, d_1, \dots, d_w) = \min \begin{cases} f_{k-1}(a, d_1, \dots, d_w) \\ \min_{1 \leq a' \leq a} (f_{k-1}(a - a', d_1, \dots, d_r - \eta_r, \dots, d_w) + h_{k,n_k}(a')) \end{cases}$$

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

$$\max_{a=0, \dots, \hat{P}; 0 \leq d_1, \dots, d_w \leq M} \{a : f_M(a, d_1, \dots, d_w) \leq V \text{ and } \sum_{\ell=1}^d d_\ell \eta_\ell \leq C\}.$$

Computing the  $h_{k,r}(a)$  values requires  $O(n \cdot \hat{P})$  steps, and the recursive computation of  $f_k(a, d_1, \dots, d_w)$  is done in  $O(M^w \hat{P}^2)$  steps, which gives the statement of the theorem.  $\square$

By scaling the item profits, using the upper bound  $\hat{P}$  on the total profit, only a factor of  $\varepsilon$  from the optimal solution might be lost, as formalized in the next result.

**Theorem 4.2** *There is an FPTAS for VPSD instances with a single knapsack, in which the compartment requirement of any color class can take one of the values  $\eta_1, \dots, \eta_w$ , and  $w$  is fixed. The running time of the scheme is  $O(n \log n + M^{w+2}(\frac{n}{\varepsilon})^2)$ .*

**Proof:** The instance is scaled in two steps.

- First, find a  $2M$ -approximation to the optimal profit. This is done by taking all the colors  $k$  for which  $c_k \leq C$ , and generating an input consisting of these items, such that all items are of the same color. The well-known greedy algorithm yields a 2-approximation for the knapsack problems (with no colors). Next, assign to the packed items their original colors, and select the color that achieving the maximum profit.
- Let  $z$  be the solution value output by the  $2M$ -approximation algorithm. Then set for each item,  $i$ ,  $p'_i = \lceil \frac{p_i n}{\varepsilon z} \rceil$ , and set  $\hat{P} = 2M \lceil n/\varepsilon \rceil + n$  as an upper bound for the optimal profit.

As in the classic FPTASs for the Knapsack problem (based on a scheme of Ibarra and Kim [10], see also in [12]), the above scaling yields a  $(1 + \varepsilon)$ -approximation to the optimal profit. Hence, using the greedy  $2M$ -approximation and applying dynamic programming as described in the proof of Theorem 4.1, we get that the overall complexity is

$$O(n \log n + n\hat{P} + M^w \hat{P}^2) = O(n \log n + M^{w+2} \cdot \left(\frac{n}{\varepsilon}\right)^2).$$

□

## 4.2 Data Placement on a Single Disk

In the following we show that the data placement problem on a single disk admits an FPTAS. We first consider the subclass of instances of VPSD in which all items have the same (unit) size and the same (unit) profit, and each color set may have arbitrary compartment requirement. We show that for such instances VPSD is solvable in polynomial time.

**Theorem 4.3** *If  $\forall 1 \leq i \leq n$ ,  $s_i = p_i = 1$ , an optimal solution can be computed in  $O(M \cdot \min(V, n))$  steps.*

**Proof:** First, we use dynamic programming to compute recursively  $g_k(a)$ , the minimum number of compartments needed for packing a subset of items whose total profit is  $a$  (in other words, for packing  $a$  items), from the first  $k$  colors. Formally, for all  $0 \leq k \leq M$ , initialize  $g_k(a) = 0$  for  $a \leq 0$ . Also,  $g_0(a) = +\infty$ , for  $a > 0$ ; then we calculate  $g_k(a)$  for  $a \in \{1, \dots, \hat{P}\}$ , where  $\hat{P} = \min(n, V)$ . This is done using the formula

$$g_k(a) = \min \begin{cases} g_{k-1}(a) \\ g_{k-1}(a - n_k) + c_k \end{cases}$$

In calculating  $g_k(a)$ , we consider two possibilities. First, no item of color  $k$  is packed, then  $g_k(a) = g_{k-1}(a)$ ; and second, some items of color  $k$  are packed, in which case we allocate to color  $k$   $c_k$  compartments. If  $a > n_k$ , then out of  $a$  packed items, we pack  $n_k$  in color  $k$ , and the remaining  $a - n_k$  items are packed in additional  $g_{k-1}(a - n_k)$  compartments. Given the  $g$  values, an optimal solution is found by taking  $\max_{0 \leq a \leq \hat{P}} \{a : g_M(a) \leq C\}$ . □

Recall that an instance of the data placement problem consists of  $M$  files: the  $k$ -th file has a specified size,  $c_k$ , and a broadcast requirement,  $n_k$ , which takes a value in  $(0, V]$ . We can reduce a data placement instance to an instance of VPSD, where all items have unit sizes and unit profits, and the number of items of color  $k$  is  $n_k$ . Using Theorem 4.3, we can find an exact solution for this VPSD instance in  $O(M \cdot \min(n, V))$  steps, where  $n = \sum_{k=1}^M n_k$ . Since the input for data placement consists of the value of  $V$  and the sets of values  $\{c_k\}$  and  $\{n_k\}$ , for  $1 \leq k \leq M$ , this exact algorithm becomes pseudo-polynomial.

We now show how to obtain an FPTAS. We first note that any data placement instance has uniform profit/size ratio with  $\alpha = 1$ . Thus, we can obtain a 2-approximate solution,  $z$ , using algorithm  $\mathcal{A}_G$  (see in Section 3.1). We take  $\hat{V} = 2 \cdot \lceil \frac{M}{\varepsilon} \rceil + M$ , and for any  $1 \leq k \leq M$ , we scale the broadcast requirement of the  $k$ -th file by taking  $\hat{n}_k = \frac{n_k \cdot M}{\varepsilon \cdot z}$ . Given the scaled instance, we can now get a  $(1 + \varepsilon)$ -approximation for data placement by using

dynamic programming, as described in the proof of Theorem 4.3. The running time of  $\mathcal{A}_G$  is  $O(M \log M)$ . Adding the time required for computing the  $g$  values, we get an overall running time of  $O(M \log M + M\hat{V}) = O(M^2/\varepsilon)$ .

**Theorem 4.4** *There is an FPTAS for data placement on a single disk.*

## 5 Packing in Multiple Bins: A Greedy Algorithm

Given an instance of VPSD with  $N$  bins, consider an algorithm,  $\mathcal{A}_{MG}$ , which packs the bins sequentially. In step  $j$ ,  $1 \leq j \leq N$ , use an (approximation or exact) polynomial time algorithm for packing a ‘good’ subset of the remaining items in bin  $j$ . The analysis of this iterative packing algorithm is identical to the analysis given in [18] for the classic multiple knapsack problem. and by the above results for a single knapsack, we get

**Theorem 5.1**  *$\mathcal{A}_{MG}$  is a  $(2 + \varepsilon)$ -approximation algorithm for VPSD, and 2-approximation for instances with unit sizes and profits.*

In the special case where all the bins are identical, we can use a result for the *generalized assignment problem (GAP)* in [3] to get better approximation ratios.

**Theorem 5.2**  *$\mathcal{A}_{MG}$  achieves ratio  $e/(e - 1) + \varepsilon$  for VPSD with identical bins, and ratio  $e/(e - 1)$  for instances with unit sizes and profits, where  $e$  is the base of the natural logarithm.*

## 6 An Approximation Scheme for Data Placement

In this section, we develop a PTAS for data placement instances in which the disks are identical and the number of distinct file sizes is fixed.

In terms of the VPSD problem, we consider instances  $I$  consisting of  $n$  items, of  $M$  distinct colors; for any item  $i$ ,  $1 \leq i \leq n$ ,  $p_i = s_i = 1$ . The compartment requirement of color  $k$  can take one of  $w$  possible values,  $\eta_1, \dots, \eta_w$ , where  $w > 1$  is a fixed constant. There are  $n_k$  items of color  $k$ ,  $1 \leq k \leq M$ . Since all items have unit sizes and profits, we may assume that there is a *single* set of items in each color. We need to pack a subset of the items in  $N$  identical bins, where each bin has volume  $V$  and  $C$  compartments.

Throughout the execution of the scheme, it is assumed that no set of items is too large, namely, for all  $1 \leq k \leq M$ ,  $n_k \leq V/\varepsilon$ . This can harm the approximation ratio at most by factor of  $\varepsilon$ , as shown in the next lemma, due to Golubchik et al. [8].

**Lemma 6.1** *The input  $I$  can be transformed to  $I'$ , which satisfies (i) the size of any set of items is at most  $V/\varepsilon$ ; (ii) any packing of items in  $I'$  can be mapped to a packing of items in  $I$  of the same profit. (iii)  $OPT(I') \geq (1 - \varepsilon)OPT(I)$ , where  $OPT(I)$  is the optimal profit from packing  $I$ .*

Given a parameter  $\varepsilon > 0$ , the scheme proceeds as follows. (i) Guess the optimal profit from the packing,  $1 \leq P \leq n$ . (ii) Guess the subset of items that are packed in the bins. (iii) Pack the selected items, distinguishing between items with ‘large’ and ‘small’ compartment

requirements. In the latter case, we further distinguish between packings of ‘large’ and ‘small’ blocks (we define a *block* below).

Given a correct guess of  $P$ , omit from the input the items in any color  $k$  such that  $n_k \leq \varepsilon P/M$ . By that, at most a factor of  $\varepsilon$  in the approximation ratio is lost. Dividing the value of  $n_k$ , for each of the remaining colors, by  $\varepsilon P/M$ , and rounding down to the nearest integral power of  $(1 + \varepsilon)$ , results in an instance in which there are  $h = O(\ln M/\varepsilon)$  distinct  $n_k$  values.

Now, partition the item sets to  $w$  groups,  $S_1, \dots, S_w$ , by their compartment requirements. The item sets having the compartment requirement  $\eta_\ell$  form the  $\ell$ -th *compartment category*,  $S_\ell$ .

In packing items in the bins, we select in each step a subset of the items (or, *block*) in some color. We distinguish between the sizes of the packed blocks and the compartment requirements of the corresponding items. We say that a block is *large* if its size is at least  $\varepsilon V$ ; otherwise, it is *small*. Also, the compartment requirement of color  $k$  is *large* if  $c_k \geq \varepsilon C$ ; otherwise it is *small*.

## 6.1 Packing Items with Large Compartment Requirements

First, we pack blocks of items with large compartment requirements. Note that at most  $1/\varepsilon$  such blocks can be packed in each bin. For any  $1 \leq \ell \leq w$ , such that  $\eta_\ell \geq \varepsilon C$ , we call  $S_\ell$  a *large compartment category*. To pack items of the large compartment categories, we consider blocks of sizes which are integral multiples of  $\varepsilon^2 V$ . Also, in determining the numbers of blocks of each size in a bin, we take bins of size  $V(1 + \varepsilon)$ . Thus, if in some optimal solution a block of size  $\alpha \varepsilon^2 V$  is packed in a bin, where  $0 < \alpha \leq 1/\varepsilon^2$ , then we round  $\alpha$  to the next integral value. After packing the items, we omit in each bin at most  $\varepsilon V$  items, to satisfy the volume constraint of the input. This may result in a loss of at most factor of  $\varepsilon$  in the overall profit.

Initially, we guess the number of blocks in sizes  $\varepsilon^2 V, \dots, V$  contributed to the solution by each of the large compartment categories. Taking the block sizes to be integral multiples of  $\varepsilon^2 V$ , each compartment category may contribute blocks of at most  $1/\varepsilon^2$  distinct sizes. We define a *block configuration* to be the number of blocks of each size contributed to the packing by a subset of items of a given color. Since  $n_k \leq V/\varepsilon$  for all  $1 \leq k \leq M$ , overall, each item set can contribute at most  $1/\varepsilon^3$  blocks. Thus, the total number of block configurations is  $L = O((1/\varepsilon^3)^{1/\varepsilon^2})$ . Now, we guess the number of item sets in each of the large compartment categories having each configuration. This requires  $O(M^{wL})$  steps. To select the packed items, we allocate each block configuration to an item set having a minimum number of items that is at least as large as the total number of items in this block configuration. Thus, we have determined the blocks of items of large compartment categories that need to be packed in the bins.

Given a correct guess of the blocks contributed by the item sets, we may assume that each block is packed in a *distinct* set of compartments in some bin. (Otherwise, we glue together several blocks that *share* the same compartment set into one large block.) Hence, the result is an instance of the 2DVP with fixed number of distinct item sizes in each dimension.

We now define a *bin configuration* to be the number of items in each size (in both dimensions) in a bin. Specifically, let  $1 \leq c \leq 1/\varepsilon$ ,  $1 \leq b \leq 1/\varepsilon^2$  be the distinct number of item sizes in the first (=compartment) and the second (=block size) dimension, respectively; then, each configuration is a vector  $(n_{11}, \dots, n_{1b}, \dots, n_{c1}, \dots, n_{cb})$ , where  $0 \leq n_{ij} \leq 1/\varepsilon$ ,  $1 \leq i \leq c$ ,  $1 \leq j \leq b$ . The total number of bin configurations is  $R = O((1/\varepsilon)^{c \cdot b}) = O((1/\varepsilon)^{\varepsilon^{-3}})$ , which is a constant, and the number of bins in each configuration can be guessed in  $O(N^R)$  steps. This gives an optimal packing of the blocks in the bins.

## 6.2 Packing Large Blocks with Small Compartment Requirements

In this step, we pack large blocks of items whose compartment requirement is smaller than  $\varepsilon C$ . We now guess the number of blocks of sizes  $\varepsilon V, \dots, V$ , as integral multiples of  $\varepsilon^2 V$ , extracted from small compartment categories. Since at most  $1/\varepsilon$  large blocks can be packed in a bin, it is possible to find (as before) in polynomial time an optimal packing of the blocks in the bins.

## 6.3 Packing the Remaining Items

Finally, we pack small blocks of items with small compartment requirements, by solving a linear programming relaxation and rounding the (fractional) solution. The program takes as input the set of small blocks generated from the remaining sets of items; we need to allocate for each block of items of color  $k$  a set of  $c_k$  compartments in some bin. This is done by reducing our problem to the *general assignment problem (GAP)* and by using a technique of [19] for solving GAP.

In partitioning the remaining items to blocks, assume that blocks generated from the same set of items are packed in different bins; thus, two blocks of the same color do not share the same set of compartments. We now reduce any instance of our problem to a GAP instance, where the cost of packing item  $k$  in bin  $j$  is  $c_{kj}$  (In our case,  $c_{kj} = c_k$  for all  $1 \leq j \leq N$ ).

As a first step, we ‘guess’ the small blocks contributed by the remaining item sets. Note that a naive search over all possible partitions to blocks may result in exponential running time. We argue, however, that it suffices to consider only partitions in which item sets whose sizes are smaller than  $\varepsilon V/w$  are packed as one block in some bin, while item sets with larger sizes are partitioned to blocks whose sizes are integral multiples of  $\varepsilon^2 V/w$ . For packing the remaining item sets, we allow to use bins of size  $V(1 + \varepsilon)$ . Suppose that, in some optimal solution, an item set contributes  $\frac{\alpha \varepsilon^2 V}{w}$  to the profit in bin  $j$ , then we round this profit (=block size) to the next multiple of  $\frac{\varepsilon^2 V}{w}$ . Since the total number of such blocks in the bin is at most  $w/\varepsilon$ , the total packed volume in this bin may increase to at most  $V(1 + \varepsilon)$ . We then omit from the bin the extra volume, losing at most factor of  $\varepsilon$  in the total profit. It is possible to guess the small blocks while guessing the large blocks of items with small compartment requirements, i.e, we include in the block configurations of the item sets blocks of sizes  $\varepsilon V/w, \dots, \varepsilon V, \dots, V$ ; then, the length of each block configuration vector is  $O(w + 1/\varepsilon^2)$ .

Now, given a *good* guess of the blocks contributed to the solution by the remaining items, we use a linear program for packing these items in the bins. For convenience, we map the blocks

to *items* in an instance of GAP; that is, any block that belonged to an item from compartment category  $\ell$  becomes in the GAP instance an item of the same size and the packing cost  $\eta_\ell$ , in any of the bins. The size of the resulting GAP instance is equal to the number of blocks generated in the above partition. Denote this number by  $m$ . Note that we can now ignore the colors of the blocks.

Let  $V_j, C_j$  be the remaining volume and available number of compartments in bin  $j$ , after packing the blocks in the previous steps. Denote by  $x_{ij} \in \{0, 1\}$  the indicator variable for the assignment of an item  $i$  (in the GAP instance) to bin  $j$ . We formulate the packing problem as the following integer program.

$$\begin{aligned} & \text{maximize} && \sum_{i=1}^m \sum_{j=1}^N x_{ij} s_i \\ & \text{subject to :} && \sum_{i=1}^m c_i x_{ij} \leq C_j \text{ for } j = 1, \dots, N \\ & && \sum_{i=1}^m s_i x_{ij} \leq V_j \text{ for } j = 1, \dots, N \\ & && \sum_{j=1}^N x_{ij} = 1 \text{ for } i = 1, \dots, m \end{aligned}$$

We solve the linear programming relaxation of this program, in which  $x_{ij} \in [0, 1]$ . Now, we round the optimal fractional solution, by defining a bipartite graph and by solving the maximum weighted matching problem on this graph. We define a separate bipartite graph for each compartment category. As shown in [19], the solution for the matching problem yields an integral packing of the items in the bins, such that there is at most one extra item from each compartment category in each bin. By taking in our scheme  $\varepsilon' = \varepsilon/w$ , we get that the total extra volume in any bin is at most  $\varepsilon V$ . By omitting the extra items, we lose at most a factor of  $\varepsilon$  from the profit.

We summarize our discussion in the following result.

**Theorem 6.2** *The above is a PTAS for VPSD instances with unit sizes and profits, and  $w$  compartment categories, where  $w$  is some fixed constant. The running time of the scheme is polynomial in  $N$  and  $M$  and exponential in  $w$  and  $1/\varepsilon$ .*

Note that our scheme yields a PTAS with the same running time for instances of the data placement problem.

**Corollary 6.3** *There is a PTAS for the data placement problem with  $N$  identical disks and fixed number of distinct file sizes.*

## 7 Approximation Scheme for Fixed Number of Colors

For instances in which the distinct number of colors is fixed, we show that VPSD admits a PTAS, even if the items have arbitrary sizes and profits, and the bins are arbitrary. Our scheme builds on an approximation technique presented in [18] for class constrained multiple knapsack. However, since the scheme in [18] crucially relies on the fact that the number of compartments in each bin can be bounded by some constant  $\leq M$  (since the compartment requirement of any color  $k$  is  $c_k = 1$ ), we need to use a different approach. The heart of the

scheme is a partition of the bins into  $O(\log(n/\varepsilon))$  types, by rounding up the volumes, and by eliminating compartments, such that in the resulting instance all the bins of the same type have (almost) the same volume and the same number of compartments. This is done without harming the approximation ratio of the scheme.

Given a parameter  $\varepsilon > 0$ , the scheme proceeds in the following steps. (i) Guess the optimal profit,  $OPT(I)$ , within factor  $1 - \varepsilon$ . (ii) Guess the subset  $U \subseteq I$  of items that are packed in the bins, such that  $P(U) \geq (1 - \varepsilon)OPT(I)$  and there exists a feasible packing of  $U$ . (iii) Partition the bins into  $O(\log(n/\varepsilon))$  types. (iv) Determine how many bins of each type are assigned to each subset of colors. (v) Find a feasible packing of  $U' \subseteq U$  in the bins, such that  $P(U') \geq (1 - \varepsilon)P(U)$ .

We describe in detail below the steps of the scheme.

**Guessing the Packed Items** Having guessed  $P \geq OPT(I)(1 - \varepsilon)$ , we guess  $U \subseteq I$ , a subset of items whose total profit satisfies  $P(U) \geq (1 - \varepsilon)P$ , which has a feasible packing in the bins. This can be done in polynomial time, as shown in the next result.

**Lemma 7.1** [18] *For a given  $P$ , a set  $U$  such that  $P(U) \geq (1 - \varepsilon)P$ , and  $U$  has a feasible packing (by compartments) in the bins, can be guessed in  $n^{O(\frac{M}{\varepsilon} \ln(1/\varepsilon))}$  steps.*

**Defining Bin Types** Let  $r = 2^M$  be the number of possible color sets (i.e., subsets of colors), to be packed in the bins. We number the subsets of colors in non-decreasing order by the sum of compartment requirements; let  $c_{s\ell}$  denote the sum of requirements of the  $\ell$ -th subset. For any bin  $B_j$  having  $c_j$  compartments,  $B_j$  belongs to  $A_\ell$  if the maximum sum of compartment requirements that can be assigned to  $B_j$  belongs to the  $\ell$ -th subset (ties are broken arbitrarily).

**Lemma 7.2** *For any  $1 \leq \ell \leq r$ , the set  $A_\ell$  can be transformed into a set with  $O(\log(n/\varepsilon))$  bin types, such that (i) the bins in each subset have the same number of compartments and their capacities are in the range  $[v, v(1 + \varepsilon))$  for some  $v$ ; (ii) for any  $U_\ell \subseteq I$  that has a feasible packing in  $A_\ell$ , there exists  $U'_\ell \subseteq U_\ell$  that has a feasible packing in the transformed set, and  $P(U'_\ell) \geq (1 - \varepsilon)P(U_\ell)$ .*

**Proof:** For a given  $1 \leq \ell \leq r$ , let  $v_{max} = v_{max}(\ell)$  denote the maximal capacity of a bin in  $A_\ell$ . Let  $g$  be the minimum integer such that the number of bins of sizes in  $[\varepsilon v_{max}/n^{\frac{gM}{\varepsilon}}, v_{max}]$  is at least

$$T'' = \frac{M}{\varepsilon}. \quad (6)$$

Note that there exists such  $g \geq 0$ , since w.l.o.g. we may assume that the number of bins in  $A_\ell$  is  $\Omega(\log(n/\varepsilon))$ . Consider the set of bins in  $A_\ell$  whose capacities are in the range  $[\varepsilon v_{max}/n^{\frac{(g+1)M}{\varepsilon}}, v_{max}]$ . We partition this set of bins into subsets,  $A_\ell^0, A_\ell^1, \dots$ , such that  $A_\ell^j$  consists of the bins whose capacities are in the range

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

We now have  $O(\log(n^{\frac{(g+1)M}{\varepsilon}}/\varepsilon))$  different subsets of bins in  $A_\ell$ .

Suppose that  $U_\ell \subseteq I$  is packed in the original set of bins in  $A_\ell$ , then we add  $w \leq M$  bins, at most one for each color  $1 \leq k \leq M$ , and pack all the items in color  $k$ , previously packed in bins of sizes smaller than  $\varepsilon v_{max}/n^{\frac{(g+1)M}{\varepsilon}}$ , in a *single* bin having  $c_{sl}$  compartments. The total capacity of the  $w$  added bins is at most  $\varepsilon v_{max}/n^{(\frac{(g+1)M}{\varepsilon} - 1)} \leq \varepsilon v_{max}/n^{\frac{gM}{\varepsilon}}$ . Therefore, we can transform the set  $A_\ell$  to a set of bins, where all the bins, except for  $w \leq M$ , are of sizes in the range  $[\varepsilon v_{max}/n^{\frac{(g+1)M}{\varepsilon}}, v_{max}]$ , arranged in the sets  $\{A_\ell^j\}$  defined above.

Since the set  $U_\ell$  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  $\varepsilon v_{max}/n^{\frac{gM}{\varepsilon}}$ ) among the  $w$  new bins in  $A_\ell$ . The capacity of each of the new bins will be a multiple  $1 \leq d \leq n$  of  $\varepsilon v_{max}/n^{\frac{(g+1)M}{\varepsilon}}$ . Thus, the overall number of guesses is  $n^w \leq n^{O(M)}$ .

Finally, after the items of  $U_\ell$  are packed in the transformed set of bins  $A_\ell$ , we need to eliminate the extra  $w$  bins. Clearly, we can remove from  $A_\ell$  any set of  $w$  bins with the largest capacities.

Consider the set of  $T'' + w$  bins in  $A_\ell$  having the largest capacities, where  $T''$  is defined in (6). Note that  $T'' + w \leq T'' + M \leq T''(1 + \varepsilon)$ .

Since this set of bins contains at least  $T''$  bins of size  $\varepsilon v_{max}/n^{\frac{gM}{\varepsilon}}$  or larger, which belonged originally to  $A_\ell$ , we can select the  $T''$  most profitable bins, thus guaranteeing that each of the  $w$  eliminated bins are at least of the size of a new bin, and the total profit of  $U'_\ell$ , the remaining set of packed items, is at least  $(1 - \varepsilon)P(U_\ell)$ . Repeating this for any  $1 \leq \ell \leq r$ , we get the statement of the theorem.  $\square$

**Assigning Color Sets to Bins** Now, we determine how many bins of each type  $A_\ell^j$ , for all  $\ell, j$ , are assigned a given subset of colors (out of the  $r = 2^M$  possible subsets). This gives a partition of the bins into *blocks*, such that all the bins in the same block have (almost) the same volume and the same number of compartments, and are assigned the same subset of colors.

Note that since the number of bin types is  $t = r \cdot O(\log(n/\varepsilon))$ , we can use a technique for guessing the assignments of color sets to the bins, as in the CCMK problem. The following was shown in [18].

**Lemma 7.3** *For  $t = O(\log n)$  bin types, the assignment of color sets to the bin types can be done in  $2^{\frac{r^2}{\varepsilon} \log n} = n^{O(r^2/\varepsilon)}$  steps.*

**Packing the Items** Having assigned to each bin a subset of colors, we can apply the packing step for CCMK to obtain a feasible packing of  $U' \subseteq U$  in the bin blocks, such that  $P(U') \geq (1 - \varepsilon)P(U)$ . As shown in [18], this can be done in polynomial time.

Putting together the guessing and packing steps, we get the following.

**Theorem 7.4** *There is a PTAS for VPSD instances in which  $M$  is a fixed constant.*

**Acknowledgments.** We thank Chandra Chekuri for valuable discussions and suggestions. Thanks also to Samir Khuller for helpful comments on the paper.

## References

- [1] A. Bar-Noy, R. E. Ladner, and T. Tamir. Scheduling Techniques for Media-on-Demand. In *Proc. of SODA*, 791–800, 2003.
- [2] C. Chekuri and S. Khanna, On Multi-dimensional Packing Problems. *SIAM J. on Computing*, 33(4):837–851, 2004.
- [3] C. Chekuri and S. Khanna. A PTAS for the multiple knapsack problem. *SIAM J. on Computing*, 35(3):713–728, 2006.
- [4] L. Fleischer, M.X. Goemans, V.S. Mirrokn and M. Sviridenko. Tight approximation algorithms for maximum general assignment problems. *Proc of SODA 2006*, 611–620.
- [5] A. M. Frieze and M.R.B. Clarke, Approximation Algorithms for the m-dimensional 0-1 knapsack problem: worst-case and probabilistic analyses. In *European J. of Operational Research*, 15(1):100–109, 1984.
- [6] M.R. Garey and D.S. Johnson. Strong NP-completeness results: Motivations, examples, and implications. *J. of the ACM*, 25:499–508, 1978.
- [7] G.V. Gens and E.V. Levner. Computational complexity of approximation algorithms for combinatorial problems. In *Proc. of the 8th Int. Symp. on Mathematical Foundations of Computer Science*, 292–300, 1979.
- [8] L. Golubchik, S. Khanna, S. Khuller, R. Thurimella, and A. Zhu. Approximation algorithms for data placement on parallel disks. In *Proc. of SODA*, 223–232, 2000.
- [9] S. Ghandeharizadeh and R.R. Muntz. Design and implementation of scalable continuous media servers. *Parallel Computing J.*, 24(1):91–122, 1998.
- [10] O.H. Ibarra and C.E. Kim, Fast Approximation for the Knapsack and the Sum of Subset Problems, *J. of the ACM*, 22(4), pages 463–468, 1975.
- [11] S. Kashyap and S. Khuller, Algorithms for Non-Uniform Size Data Placement on Parallel Disks. *FST & TCS*, 2003.
- [12] H. Kellerer, U. Pferschy and D. Pisinger. *Knapsack Problems*. Springer, 2004.
- [13] M. Kamath, K. Ramamritham, and D. Towsley. Continuous media sharing in multimedia database systems. In *Proc. of the 4th Intl. Conf. on Database Systems for Advanced Applications*, 1995.
- [14] B. Korte, R. Schrader. On the existence of fast approximation schemes. In *O. Magasarian, R. Meyer, S. Robinson (eds.): Nonlinear Programming 4*, 415–437, Academic Press, 1981
- [15] P.W.K. Lie, J.C.S. Lui, and L. Golubchik. Threshold-based dynamic replication in large-scale video-on-demand systems. In *Proc. of the Eighth International Workshop on Research Issues in Database Engineering (RIDE)*, 52–59, 1998.

- [16] S. Sahni. Approximate Algorithms for the 0/1 knapsack problem, *J. of the ACM*, 22, 115–124, 1975.
- [17] H. Shachnai and T. Tamir, On Two Class-Constrained Versions of the Multiple Knapsack Problem. *Algorithmica*, vol. 29, 442–467, 2001.
- [18] H. Shachnai and T. Tamir, Polynomial Time Approximation Schemes for Class-Constrained Packing Problems. *J. of Scheduling*, vol. 4(6), pp. 313–338, 2001.
- [19] D. S. Shmoys and E. Tardos, Scheduling unrelated machines with Costs. In *Proc. of SODA*, 1993.
- [20] G.J. Woeginger. There is no asymptotic PTAS for two-dimensional vector packing. *Information Processing Letters* 64(6): 293-297, 1997.
- [21] J.L. Wolf, P.S. Yu, and H. Shachnai. Scheduling issues in video-on-demand systems. *Multimedia Information Storage and Management*, 183–207, 1996.
- [22] J.L. Wolf, P.S. Yu, and H. Shachnai. Disk load balancing for video-on-demand systems. *ACM Multimedia Systems J.*, 5:358–370, 1997.