Approximation Schemes for Generalized 2-dimensional Vector Packing with Application to Data Placement

Hadas Shachnai[1,⋆] and Tami Tamir[2]

[1] Bell Laboratories, Lucent Technologies, 600 Mountain Ave. Murray Hill, NJ 07974
[2] Department of Computer Science & Engineering, Univ. of Washington, Box 352350, Seattle, WA 98195. E-mail: tami@cs.washington.edu

Abstract. Suppose that we have a set of items and a set of devices, each possessing two limited resources. Each item requires a given amount 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. We need to allocate the resources to the most profitable (feasible) subset of items. In alternative formulation, we need to *pack* the most profitable subset of items in a set of 2-dimensional bins (knapsacks), in which the capacity in one dimension is *sharable*. Indeed, the special case where we have a single item in each color is the well-known 2*-dimensional vector packing (*2*DVP)* problem. Thus, the problem that we study is strongly NP-hard for a single bin, and 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 our 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

Consider the following optimization problem. Suppose that we have 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 allocated to that item, and

⋆ On leave from the Department of Computer Science, Technion, Haifa 32000, Israel. E-mail: hadas@cs.technion.ac.il

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 we have a single item in each color is the well-known *2-dimensional vector packing problem (2DVP)*. Thus, our problem is strongly NP-hard for a single bin [10] and MAX-SNP hard for multiple bins, already in the case where the bins are identical, and the items have unit profits, i.e., $p_i = 1 \ \forall i, 1 \leq i \leq n$ [16]. We call this problem *vector packing with a shareable dimension* (VPSD).

An important application of VPSD is *data placement* on disks in media-on-demand systems [6, 12, 9]. In such systems (see, e.g., [17, 7]), 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 the VPSD problem, 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, we may have different c_j values. On the other hand, all the broadcast streams require the same (non-shareable) bandwidth; thus, we have $\forall i \ s_i = 1$.

Other applications of VPSD are production planning and scheduling parallel tasks (see in [13]). 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 (i.e., item profits) are proportional to the amounts of resources consumed (i.e., the items sizes).

1.1 Our Results

In the following we summarize our main results.

Single knapsack We show (in Section 2) that VPSD can be optimally solved in pseudo-polynomial time. We then develop an LP based PTAS for general instances. For the subclass of uniform profit/size ratio instances, we develop (in Section 3) a simpler approximation scheme, that is based on extension of a PTAS proposed in [11] for the classical knapsack problem. By this, we obtain the first *combinatorial* PTAS for a non-trivial subclass of instances for 2DVP. In Section 4, we develop *fully polynomial time approximation schemes (FPTAS)* for the subclasses of (i) data placement instances, and (ii) instances with constant number of compartment requirements.

Multiple knapsacks We show (in Section 4) 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. A PTAS is developed (in Section 5) for data placement instances in which the disks are identical (but may have arbitrary storage and load capacities), and the number of distinct file sizes is fixed. Finally, for instances in which M, the distinct number of colors, is fixed, we show (in section 6) that VPSD admits a PTAS, even if the items have arbitrary sizes and profits, and the bins are arbitrary.

In our PTAS for a single knapsack (in Section 2), we combine the guessing technique of [2] with a novel application of the approximation scheme of [3] to the *multidimensional multiple choice knapsack* problem. In our algorithms for uniform ratio instances (in Section 3), we show that a simple greedy algorithm and an approximation scheme proposed for the 0/1 knapsack problem can be extended to VPSD. The idea is to partially reduce VPSD to the knapsack problem, by first considering all the items of each color as a *single* item. Later, we map the *grouped* items back to the original items. While these extensions do not apply for general instances, it may be possible to apply similar ideas for other subclasses of VPSD and 2DVP.

Due to space constraints we state some of the results without proofs.[3]

1.2 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 [8, 5]. In contrast, the *multiple knapsack (MK)* problem is known to be strongly NP-hard [4]. Chekuri and Khanna developed in [2] a PTAS for MK and showed that with slight generalizations this problem becomes APX-hard.

[3] The detailed proofs are given in [14].

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 [3], for the case where d is a fixed constant. As opposed to the combinatorial schemes for the single dimension case, the PTAS in [3] 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 $d = 2$. For the case of $N > 1$ bins, Woeginger showed in [16] that 2-dimensional vector packing is MAX-SNP hard (see also in [1]). Chekuri and Khanna presented in [1] 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 [1] 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 [13] is a special case of VPSD, where $c_k = 1$ for all $1 \leq k \leq M$. The paper [13] 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 [12]. The paper presents an algorithm for the case where all the files are of the *same* (unit) size, and for all $1 \leq j \leq N$, we have the same ratio V_j/C_j for disk j (*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 [6] 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 [6] also presents a PTAS for the data placement problem with unit sized files and uniform ratio disks. Recently, Kashyap and Khuller [9] studied the problem with files of Δ distinct sizes, where Δ 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 $\{1, \ldots, \Delta\}$, 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.

2 Approximation Scheme for a Single Bin

In this section, we discuss the single knapsack version of VPSD. Assume that the knapsack has the volume V and C compartments. We first note that by using a two-level dynamic programming algorithm, VPSD with a single knapsack can be solved optimally in pseudo-polynomial time. (The details are given in [14].) Let $\hat{P}$ be an upper bound on the total profit (indeed, $\hat{P} \leq \sum_i p_i$).

Theorem 1. *VPSD can be solved optimally in* $O(n\hat{P} + M\hat{P}^2C)$ *steps.*

We now describe a PTAS for a single knapsack. Note that, by the result of [10], this is the best we can expect, since our problem is strongly NP-hard.

Assume that we know the optimal profit, P, for our instance. We reduce our problem to the *binary 2-dimensional multiple choice knapsack* (B2D-MCK) problem. That is, for given values of P and ε, 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 the B2D-MCK Problem 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. We need 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 ε and a VPSD instance with n items of M distinct colors, we construct a B2D-MCK instance which consists 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 next integral multiple of $\varepsilon P/M$. In particular, the jth 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 [8]).

Lemma 1. *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 2D-MCK instance.*

Approximating the Optimal Solution for B2D-MCK Given an instance of B2D-MCK, we '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 $p_{min}(S) = \min_{(k,j)\in S} p(k, j)$.

We pack all the items $(k, j) \in S$ and eliminate from the instance all the items $(k, j) \in E(S)$, and the sets S_k from which an item has been selected. In the next step we find an optimal *basic solution* for the following linear program, $LP(S)$

$$
\begin{aligned}
\max \quad & \sum_{k=1}^{M} \sum_{j=1}^{R} p(k, j) x_{kj} \\
s.t. \quad & \sum_{j=1}^{R} x_{kj} \leq 1 \text{ for } k = 1, \ldots, M
\end{aligned}
$$

$$\sum_{k=1}^{M}\sum_{j=1}^{R} x_{kj}s_{kj} \le V$$
$$\sum_{k=1}^{M} c_k \sum_{j=1}^{R} x_{kj} \le C$$
$$x_{kj} = 1 \text{ for } (k,j) \in S, \text{ and } x_{kj} = 0 \text{ for } (k,j) \in E(S)$$
$$x_{kj} \in \{0,1\} \text{ for } k = 1,\ldots,M; \quad j = 1,\ldots,R; \quad (k,j) \notin S \cup E(S)$$

In the linear programming relaxation we allow $0 \le x_{kj} \le 1$. Given an optimal fractional solution, we get an integral solution 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. *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, $S^* = \{(k,j) |\ x^*_{kj} = 1\}$. If $|S^*| < h$ then we are done (the scheme outputs a $(1-\varepsilon)$-approximation to the optimal profit: this is due to the initial guess of P); otherwise, let $S^* = \{(k_1,j_1),\ldots,(k_r,j_r)\}$, such that $p(k_1,j_1) \ge \cdots \ge p(k_r,j_r)$. Let $S^*_h = \{(k_1,j_1),\ldots,(k_h,j_h)\}$, and $\sigma = \sum_{\ell=1}^{h} p(k_\ell,j_\ell)$. Then, for any item $(k,j) \notin (S^*_h \cup E(S^*_h))$, we have $p(k,j) \le \sigma/h$. Let $z^*, \hat{z}$ denote the optimal solution and the solution output by the scheme, respectively. We 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. Now, we have that

$$z^* \le \sum_{k=1}^{M}\sum_{j=1}^{R} p(k,j)x^B_{kj}(S^*_h) \le \sum_{k=1}^{M}\sum_{j=1}^{R} p(k,j)x^I_{kj}(S^*_h) + \delta,$$

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, $F = \{(k,j) |\ x^B_{kj}(S^*_h) > x^I_{kj}(S^*_h)\}$.

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 $0 \le L \le M+2$. Then in the basic solution $\mathbf{x}^B(S^*_h)$, at most L variables can be strictly positive. Thus, at least $L-4$ variables get an integral value (i.e. '1'), and $|F| \le 4$. Note that $\delta < \sigma/h$, since $F \cap (S^*_h \cup E(S^*_h)) = \emptyset$. Hence, we get that $z^* \le \hat{z} + \frac{4\sigma}{h} \le \hat{z} + \frac{4\hat{z}}{h} \le \frac{\hat{z}}{1-\varepsilon}$.

3 Instances with Uniform Profit/Size Ratio

In this section we present algorithms for instances with uniform profit/size ratio, that is, for some $\alpha > 0$, $\forall i, p_i = \alpha s_i$. Our goal is to pack in a single bin a subset of the items whose total size is as large as possible. W.l.o.g. we assume that $\forall k$, $c_k \le C$, $\forall i$, $s_i \le V$.

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 $S_1/c_1 \geq S_2/c_2 \geq \ldots \geq S_M/c_M$.
2. Find in the sorted list the first j colors satisfying $\sum_{k=1}^{j} c_k \leq C$, 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 in A in the knapsack from largest to smallest, ignoring colors, biggest to smallest, 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. *$\mathcal{A}_G$ 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 we sort the colors by profit/compartment ratio, $OPT < S_1 + \ldots + S_{j+1}$. If in step 3 we pack all the items of A, then $a_1 = S_1 + \ldots + S_j$. $Alg = max(a_1, a_2) = max(S_1 + ... + S_j, S_{k^*}) \geq \frac{1}{2}(S_1 + \ldots + S_j + S_{k^*}) \geq \frac{1}{2}(S_1 + \ldots + S_{j+1}) \geq \frac{1}{2}OPT$. If in step 3 we pack only part of the items, then since we pack from largest to smallest we fill at least half of the volume, which is at least $\frac{1}{2}OPT$.

3.2 Approximation Scheme

We now describe a PTAS for the uniform ratio case. Our scheme extends the PTAS of Sahni [11] 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, we 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 $S_1/c_1 \geq S_2/c_2 \geq \ldots$
   (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 4. *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 $O(n^{k_1+1} \cdot M^{k_2})$.*

By selecting $k_1 = k_2 = 1/\varepsilon$ we obtain a PTAS. 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 1. *Algorithm $\mathcal{A}$ is a PTAS for uniform profit/size ratio instances of 2DVP.*

4 Better Algorithms for Special Instances

We now show that better approximations or more efficient algorithms can be obtained for several subclasses of instances.

Theorem 5. *If the compartment requirement of any color class can take one of the values $\eta_1, \ldots, \eta_w$, where w is fixed, then an optimal solution can be computed in $O(M^{w+1}\hat{P})$ steps.*

By scaling the item profits, using the upper bound $\hat{P}$ on the total profit, we may lose only a factor of ε in the approximation ratio.

Theorem 6. *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 $\eta_1, \ldots, \eta_w$, and w is fixed. The running time of the scheme is $O(n \log n + M^{w+2}\frac{n}{\varepsilon})$.*

For instances where all items have the same (unit) size and profit, and each color can have an arbitrary compartment requirement, we get an exact polynomial time algorithm.

Theorem 7. *If $\forall\ 1 \leq i \leq n\ s_i = p_i = 1$, an optimal solution can be computed in $O(Mn^2)$ steps.*

Recall that a data placement instance is given as M files, each having a specified size and a broadcast requirement, which takes a value in $(0, V]$. Thus, for such instances the optimal algorithm runs in $O(MV^2)$ steps. By scaling and rounding the load capacity of the disk, as well as the broadcast requirements of the files,[4] we obtain an FPTAS.

4.1 Packing in Multiple Bins: A Greedy Algorithm

Given an instance of VPSD with N bins, consider an algorithm, $\mathcal{A}_G$, which packs the bins sequentially. In step j, $1 \leq j \leq N$, we use an (approximation or exact) polynomial time algorithm for packing a 'good' subset of the remaining items in bin j. By the analysis of this iterative packing algorithm, as given in [13], and by the above results for a single knapsack, we get

[4] Similar to the proof of Theorem 6. We omit the details.

Theorem 8. *$\mathcal{A}_G$ 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 [2] to get better approximation ratios.

Theorem 9. *$\mathcal{A}_G$ achieves a ratio of $e/(e-1)+\varepsilon$ for VPSD with identical bins, and the ratio $e/(e-1)$ for instances with unit sizes and profits, where e is the base of the natural logarithm.*

5 Approximation Scheme for Data Placement

In this section, we develop a PTAS for data placement instances in which the disks are identical, but may have arbitrary storage and load capacities, 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, $\eta_1, \ldots, \eta_w$, where $w > 1$ is a fixed constant. There are n_k items of color k, $1 \leq k \leq M$. We need to pack a subset of the items in N identical bins, where each bin has volume V and C compartments.

Given a parameter $\varepsilon > 0$, the scheme proceeds as follows. (i) Guess the optimal profit from the packing, P, to within factor $1+\varepsilon$. (Recall that $1 \leq P \leq n$.) (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).

5.1 Guessing the Packed Items

Given a correct guess of P, we omit from the input the items in any color k such that $n_k \leq \varepsilon P/M$. By that we lose at most a factor of ε in the approximation ratio. 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 $(1+\varepsilon)$, we get an instance in which there are $h = O(\ln(M/\varepsilon))$ distinct n_k values.

Now, we partition the item sets to w groups, $S_1, \ldots, S_w$, by their compartment requirements. The item sets having the compartment requirement η_ℓ form the ℓ-th *compartment category*, S_ℓ. We find the subset of packed items by guessing the contribution of each compartment category to the overall profit. Let $P(S_\ell)$ denote the contribution of S_ℓ. We may assume that $P(S_\ell) \geq \varepsilon P/w$. By that, we lose at most a factor of ε from the overall profit. Then, we look for a vector $\boldsymbol{k}$, of integers, $w/\varepsilon \leq k_\ell \leq w/\varepsilon^2$; k_ℓ reflects the contribution of S_ℓ to the overall profit in some optimal packing, in multiples of $\frac{\varepsilon^2 P}{w}$, i.e., $k_\ell \frac{\varepsilon P}{w} \leq P(S_\ell) \leq (k_\ell + 1)\frac{\varepsilon^2 P}{w}$. We seek a vector $\boldsymbol{k} = (k_1, \ldots, k_w)$ satisfying $\sum_{\ell=1}^{w} k_\ell \leq \frac{w}{\varepsilon^2}$. The number of such vectors is at most $\binom{w+w/\varepsilon^2-1}{w-1}$, which is a constant. Given the contribution of S_ℓ, we select from S_ℓ the minimum number of items that provide this profit. That is, we order the sets of items in S_ℓ in non-increasing order by sizes, and select sets of items, starting from the largest set, until we get the desired profit. Note that (at most) one set in S_ℓ may be 'partially selected', i.e., we pack in the bins only some of the items in this set.

5.2 Packing the Items

In packing the selected items, we choose 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 εV; otherwise, it is *small*. Also, the compartment requirement of color k is *large* if $c_k \geq \varepsilon C$; otherwise it is *small*.

Items with Large Compartment Requirements We first pack blocks of items with large compartment requirements. Note that we can pack at most $1/\varepsilon$ such blocks in each bin. We increase the volume of the bins to $V(1+\varepsilon)$, and pack in each bin blocks in the sizes $\varepsilon^2 V, \ldots, V$. Thus, we round up the sizes of small blocks to $\varepsilon^2 V$. After packing the items, we eliminate extra volume, until we get that the sum of packed items in each bin is at most V. By that we lose at most a factor of ε in the approximation ratio. To obtain a constant number of possible block sizes, we modify the input, so that no set of items is too large. This can harm the approximation ratio at most by factor of ε, as formalized in the next lemma.[5]

Lemma 2. *The input I can be transformed to I', which satisfies (i) the size of any set of items is at most V/ε; (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.*

We now describe how we pack the set of items with large compartment requirements. Partition the item sets to the groups $I^1, \ldots, I^h$; all of the item sets in I^r, $1 \leq r \leq h$, are of the same size. We call I^r the r-th *profit category*. Initially, we 'guess' the partition of item sets to blocks. Each partition gives the number of blocks in the sizes $\varepsilon^2 V, ..., V$, taken from the profit categories $\varepsilon^2, ..., V/\varepsilon$. Assuming that block sizes are in multiples of $\varepsilon^2 V$, and the sizes of the item sets are given as integral powers of $(1+\varepsilon)$, we get that the number of coordinates in each partition vector is $O(\lg(1/\varepsilon^3)/\varepsilon^2)$, which is a constant.

Note that in the above partition we define only the sizes of the blocks taken from each profit category I^r, $1 \leq r \leq h$; however, as the item sets in I^r may be of different compartment requirements and different colors, for each of the blocks we need to decide also on the item set in I^r from which it is extracted. This can be done greedily, without harming the approximation ratio of the scheme. Specifically, given the block configuration for I^r, we sort the item sets in I^r in non-decreasing order by their compartment requirements. The blocks are sorted in non-decreasing order by sizes. We now extract the blocks in the list from the sets, starting from the first item set. Once all of the items in the first set are allocated to blocks, we proceed to allocate blocks from the second set, and so on.

---

[5] See also in [6]. We omit the proof.

In this process we allocate 'small' blocks from sets that have 'small' compartment requirement. Clearly, this decreases the potential number of compartments required for packing the item blocks. Now, we have a set of item blocks of given colors that need to be packed in the bins.

Given a correct guess of the partition of the item sets to blocks, we may now assume that each block is packed in a distinct set of compartments in some bin. Hence, we now have an instance of the 2DVP with fixed number of distinct item sizes in each dimension. By defining a *bin configuration* to be the number of items in each size (in both dimensions) in a bin, and guessing the number of bins having each configuration, we can find the optimal packing of the blocks in the bins.

Large Blocks with Small Compartment Requirements In this step we pack large blocks of items whose compartment requirement is at most εC. Since at most $1/\varepsilon$ large blocks can be packed in a bin, we look at items in the profit categories $\varepsilon V, \ldots, V/\varepsilon$. We guess as before the blocks extracted from each of the item sets in the small compartment categories. Then, we can find in polynomial time an optimal packing of these blocks in the bins, using bin configurations.

Packing the Remaining Items Finally, we pack small blocks of items with small compartment requirements, by using linear programming 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 GAP and using a technique of [15] for solving the GAP problem (see the details in [14]).

We summarize our discussion in the following result.

Theorem 10. *The above scheme yields a $(1+\varepsilon)$-approximation for VPSD instances with unit sizes and profits, and w compartment categories. The running time of the scheme is $O(N^{(2e)^{w/\varepsilon^2}} \cdot \frac{1}{\varepsilon^{2w}})$.*

Thus,, we have a PTAS for the data placement problem, with N identical disks and w distinct file sizes, with running time as given in Theorem 10.

Finally, our scheme can be extended to apply to VPSD instances in which the item sizes take at most t distinct sizes, where t is fixed.

6 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 [13] for CCMK. However, since the scheme in [13] uses heavily 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. Our technique relies on a partition of the bins to $O(\log(N/\varepsilon))$ 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.

Theorem 11. *There is a PTAS for VPSD instances in which M is a fixed constant.*

Acknowledgments. We thank Chandra Chekuri for valuable discussions and suggestions.

References

1. C. Chekuri and S. Khanna, On Multi-dimensional Packing Problems. In *Proc. of SODA*, 185–194, 1999.
2. C. Chekuri and S. Khanna. A PTAS for the multiple knapsack problem. In *Proc. of SODA*, 213–222, 2000.
3. 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.
4. M.R. Garey and D.S. Johnson. Strong NP-completeness results: Motivations, examples, and implications. *J. of the ACM*, 25:499–508, 1978.
5. 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.
6. 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.
7. S. Ghandeharizadeh and R.R. Muntz. Design and implementation of scalable continuous media servers. *Parallel Computing J.*, 24(1):91–122, 1998.
8. 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.
9. S. Kashyap and S. Khuller, Algorithms for Non-Uniform Size Data Placement on Parallel Disks. Submitted 2003
10. 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
11. S. Sahni. Approximate Algorithms for the 0/1 knapsack problem, *J. of the ACM*, 22, 115–124, 1975.
12. H. Shachnai and T. Tamir, On Two Class-Constrained Versions of the Multiple Knapsack Problem. *Algorithmica*, vol. 29, 442–467, 2001.
13. H. Shachnai and T. Tamir, Polynomial Time Approximation Schemes for Class-Constrained Packing Problems. *J. of Scheduling*, vol. 4(6), pp. 313–338, 2001.
14. H. Shachnai and T. Tamir, Approximation Schemes for Generalized 2-dimensional Vector Packing with Application to Data Placement. `http://www.cs.technion.ac.il/~hadas/PUB/vpsd.ps`.
15. D. S. Shmoys and E. Tardos, Scheduling unrelated machines with Costs. In *Proc. of SODA*, 1993.
16. G.J. Woeginger. There is no asymptotic PTAS for two-dimensional vector packing. *Information Processing Letters* 64(6): 293-297, 1997.
17. 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.