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

Hadas Shachnai [*†]   Tami Tamir [‡]

Abstract

Suppose that we have a set of devices, each possessing two limited resources, and a set of n items. 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.

[*] Computer Science Department, The Technion, Haifa 32000, Israel. E-mail: hadas@cs.technion.ac.il.

[†] Currently on a leave in Bell Laboratories, Lucent Technologies, 600 Mountain Ave., Murray Hill, NJ 07974.

[‡] Department of Computer Science & Engineering, Univ. of Washington, Box 352350, Seattle, WA 98195. E-mail: tami@cs.washington.edu.

1 Introduction

Consider the following optimization problem. Suppose that we have a set of N devices, each possessing a limited supply of two resources, and a set of n items. 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 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 so as to maximize the 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 [11] 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, 13, 10]. 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 media. Note that storage is a shared resource $-$ that can be used by all the streams broadcasting the same media from the same disk. Different media 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 [14]). 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 [12] 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 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.

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.

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 [14] is a special case of VPSD, where $c_k = 1$ for all $1 \leq k \leq M$. The paper [14] 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 [13]. 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 [10] 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.

Due to space constraints some of the proofs are omitted. Other proofs are given in the Appendix.

2 Algorithms for a Single Bin and Arbitrary Instances

In this section, we discuss the single knapsack version of VPSD. Assume that the knapsack has the volume V and C compartments. We first show that the problem is solvable in pseudo-polynomial time. In the sequel, we develop a PTAS. Note that, by the result of [11], this is the best we can expect since our problem is strongly NP-hard.

2.1 An Optimal Pseudo-polynomial Time Algorithm

In the Appendix we show that VPSD with a single knapsack can be optimally solved in pseudo-polynomial time, using a two-level dynamic programming algorithm. Let $\hat{P}$ be an upper bound on the total profit (indeed, $\hat{P} \leq \sum_i p_i$).

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

2.2 Approximation Scheme

We now describe a PTAS for a single knapsack. 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. The goal is to pack items of maximal total profit in the knapsack. 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 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 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.

$$
\begin{array}{lll}
(LP(S)) & \text{maximize} & \displaystyle\sum_{k=1}^{M}\sum_{j=1}^{R} p(k,j)x_{kj} \\
& \textit{subject to}: & \displaystyle\sum_{j=1}^{R} x_{kj} \leq 1 \text{ for } k = 1, \ldots, M \\
& & \displaystyle\sum_{k=1}^{M}\sum_{j=1}^{R} x_{kj}s_{kj} \leq V \\
& & \displaystyle\sum_{k=1}^{M} c_k \sum_{j=1}^{R} x_{kj} \leq C \\
& & x_{kj} = 1 \text{ for } (k,j) \in S, \text{ and } x_{kj} = 0 \text{ for } (k,j) \in E(S) \\
& & x_{kj} \in \{0,1\} \text{ for } k = 1, \ldots, M; \quad j = 1, \ldots, R; \quad (k,j) \notin S \cup E(S)
\end{array}
$$

In the linear programming relaxation we allow $0 \leq x_{kj} \leq 1$. 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 2.3 *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, $S^* = \{(k, j) | \; x^*_{kj} = 1\}$. 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 $p(k_1, j_1) \geq \cdots \geq p(k_r, j_r)$. 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 $p(k, j) \leq \sigma/h$. 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

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

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,

$$F = \{(k, j) |\ x_{kj}^B(S_h^*) > x_{kj}^I(S_h^*)\}.$$

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. Thus, at least $L-4$ variables get an integral value (i.e. '1'), and $|F| \leq 4$. Note that $\delta < \sigma/h$, since $F \cap (S_h^* \cup E(S_h^*)) = \emptyset$. Hence, we get that $z^* \leq \hat{z} + \frac{4\sigma}{h} \leq \hat{z} + \frac{4\hat{z}}{h} \leq \frac{\hat{z}}{1-\varepsilon}$. ∎

3 Algorithms for 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. We assume that for all the colors $c_k \leq C$ and for all the items $s_i \leq V$ (otherwise, we can omit from the instance the non-packable items). Also, we assume that M, C are given as part of the input. Indeed, if any of the parameters V, C or M is a constant then the problem reduces to the classic knapsack problem: in this case, we can check all the possibilities to select the r colors that will be packed, where $1 \leq r \leq C$. This requires solving the knapsack problem at most $\sum_{r=1}^{C} \binom{M}{C} = O(CM^C)$ times.

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.

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 3.1 *The greedy algorithm 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 [12] 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 presented in 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 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 $O(n)$ (We assume 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 $H = \{a_1, a_2, \ldots, a_{k_1}\}$ 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 $L_1 = OPT \setminus H = \{a_{k_1+1}, \ldots, a_x\}$ 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, the statement of the theorem follows from the next claims (The proofs are given in the Appendix).

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

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

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

4 Better Algorithms for Special Instances

In this section, we show that better approximations or more efficient algorithms can be obtained for several subclasses of instances. In particular, we prove (in the Appendix) the following results.

Theorem 4.1 *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 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* $\eta_1, \ldots, \eta_w$, *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 4.3 *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,[1] 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 [14], and by the above results for a single knapsack, we get

Theorem 4.4 $\mathcal{A}_G$ *is a* $2{+}\varepsilon$*-approximation algorithm for VPSD, and* 2*-approximation for instances with unit sizes and profits.*

[1] Similar to the proof of Theorem 4.2. We omit the details.

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 4.5 *$\mathcal{A}_G$ 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.

We formulate the scheme in terms of VPSD. That is, we consider instances I consisting of n items, of M distinct colors; for any item $i, 1 \le i \le n$, $p_i = s_i = 1$. 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, $1 \le k \le M$. We need to pack a subset of the items in N identical bins, where each bin has volume V and C compartments.

Overview of the scheme: Given a parameter $\varepsilon > 0$, the scheme proceeds in the following steps. (i) Guess the optimal profit from the packing, P, to within factor $1+\varepsilon$ (Since the profit is equal to the total number of packed items, $1 \le P \le VN$). (ii) Round the numbers n_k of items having color k. This will guarantee that the running time of the scheme is polynomial in M, N and n. (iii) Guess the subset of items that are packed in the bins. (iv) 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). Thus, we do the packing in three phases: (a) pack items with large compartment requirements; (b) pack large blocks of items with small compartment requirements, and (c) pack the remaining small blocks.

5.1 Preprocessing the Input

We first omit from the input items of 'non-popular' colors. Formally, for any k, if $n_k \le \varepsilon P/M$ we remove all the items from color k. By that we lose at most a factor of ε in the approximation ratio. We then divide by $\varepsilon P/M$ the value of n_k for each of the remaining colors, and round down the values of n_k to the nearest integral power of $(1+\varepsilon)$. Thus, we get an instance in which there are $h = O(\ln(M/\varepsilon))$ distinct n_k values. In other words, we partition the item sets into h groups $I^1, \ldots, I^h$, such that all of the item sets in I^r, $1 \le r \le h$, are of the same size. Since the (potential) profit from each item set is equal to its size, we call I^r the r-th *profit category*.

5.2 Guessing the Packed Items

We now guess the subset of items that are packed in the bins. 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 $P(S_\ell) \ge \varepsilon P/w$. By that, we lose at most a factor of ε from the overall profit. Then, we look for a vector $\vec{k}$, of integers, $w/\varepsilon \le k_\ell \le w/\varepsilon^2$; 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., $k_\ell \frac{\varepsilon P}{w} \le P(S_\ell) \le (k_\ell+1)\frac{\varepsilon^2 P}{w}$. We seek a vector $\vec{k} = (k_1, \ldots, k_w)$ satisfying $\sum_{\ell=1}^{w} k_\ell \le \frac{w}{\varepsilon^2}$. 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 the 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.

5.3 Packing the Items

In packing the selected items, we choose in each step a subset of the items in some color; we call these subsets *blocks*. We pack the guessed items in three steps, distinguishing 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, we say that the compartment requirement of color k is *large* if $c_k \geq \varepsilon C$; otherwise it is *small*.

5.3.1 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.

Lemma 5.1 *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.*

The proof is based on replacing any large set of items, of size n_k, by $\lceil n_k \cdot \varepsilon / V \rceil$ sets of items, of distinct colors, and showing that any packing of the resulting instance I' can be transformed to a packing of I, with a loss of at most factor of ε in the overall profit.[2] We now describe how we pack the set of items with large compartment requirements.

Guessing a partition of item sets to blocks: Initially, we 'guess' the partition of item sets to blocks. This is done by exhaustive search over all possible partitions. 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. In this process we allocate 'small' blocks from sets that have 'small'

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

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.

Packing the blocks: 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 (otherwise, we can glue together several blocks that share the same compartment set into one large block). 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.

5.3.2 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 in Section 5.3.1) the blocks extracted from each of the item sets in the small compartment categories. Then, we can find an optimal packing of these blocks in the bins, using bin configurations. This can be done in polynomial time, since the number of block sizes and the distinct number of compartment requirements are fixed.

5.3.3 Packing the Remaining Items

Finally, we turn to pack small blocks of items with small compartment requirements. This is done 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 in the Appendix).

We summarize our discussion in the following result.

Theorem 5.2 *The above scheme yields a $(1+\varepsilon)$-approximation for VPSD instances with unit sizes and profits, and d compartment categories. The running time of the scheme is $O(n^{(2e)^{d/\varepsilon^2}} \cdot \frac{1}{\varepsilon^{2d}})$.*

It can be shown (details omitted) 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.

Finally, for disk placement instances we get the following.

Theorem 5.3 *The disk placement problem with identical disks and fixed number of distinct file sizes admits a PTAS whose running time is $O(N^{(2e)^{d/\varepsilon^2}} \cdot \frac{1}{\varepsilon^{2d}})$.*

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 [14] for CCMK. However, since the scheme in [14] 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. We prove the following in the Appendix.

Theorem 6.1 *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] 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.

[10] S. Kashyap and S. Khuller, Algorithms for Non-Uniform Size Data Placement on Parallel Disks. Submitted 2003.

[11] 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

[12] S. Sahni. Approximate Algorithms for the 0/1 knapsack problem, *J. of the ACM*, 22, 115–124, 1975.

[13] H. Shachnai and T. Tamir, On Two Class-Constrained Versions of the Multiple Knapsack Problem. *Algorithmica*, vol. 29, 442–467, 2001.

[14] H. Shachnai and T. Tamir, Polynomial Time Approximation Schemes for Class-Constrained Packing Problems. *J. of Scheduling*, vol. 4(6), pp. 313–338, 2001.

[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.

A Some Proofs

Proof of Theorem 2.1: In the following we 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 $\hat{P}$ is an upper bound on the total profit.

- In the first stage, we compute recursively $h_{k,r}(a)$, the minimum size sum 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 $1 \leq a \leq \hat{P}$.
- In the second stage, we use the values of $h_{k,r}(a)$ to compute recursively $f_k(a, \ell)$, the minimum size sum of a subset of items whose colors are among the first k, $1 \leq k \leq M$, such that the items use ℓ compartments, $1 \leq \ell \leq C$, and the total profit of the items is a, $1 \leq a \leq \hat{P}$.

This is done by setting

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

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

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

The optimal solution is given by

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

∎

Proof of Claim 3.3: 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 $a_{k_1+1}, \ldots, a_{m-1}$ 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 $a_{k_1+1}, \ldots, a_{m-1}$ and some items which are not in OPT.

Let G denote the items that are placed in the knapsack so far by the greedy stage of $\mathcal{A}$ (Steps (c)-(f)). 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 $\Delta = V - (V_f + \sum_{i=1}^{m-1} s_i)$. 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

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

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 to H), we get that $s_m \leq S(OPT)/(k_1 + 1)$. This implies the approximation ratio. ∎

Proof of Claim 3.4: Let $C(OPT)$ denote the set of colors participating in OPT. Recall that $W = \{b_1, b_2, \ldots, b_{k_2}\}$ is the set of k_2 most profitable colors in $C(OPT)$, and that $T = W \cup \{b_{k_2+1}, \ldots, b_t\}$ 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 constraints (this is handled in 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 $L_2 = \{b_{t+1}, \ldots, b_z\}$ of the remaining colors in $C(OPT)$ in the order they are considered by $\mathcal{A}$ in Step (f). 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 $b_{t+1}, \ldots, b_{j-1}$ are packed. That is, that 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 the items of colors $b_{t+1}, \ldots, b_{j-1}$ 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 $\mathcal{A}$ (Step (f)). 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 $\Gamma = C - (C_f + \sum_{i=1}^{j-1} c_i)$. 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 $S_{C(G)} \geq \sum_{i=t+1}^{j-1} S_i + \Gamma \frac{S_j}{c_j}$.

Consider now the colors $b_j, \ldots, b_z$ that are in $C(OPT)$ but are not allocated compartments by $\mathcal{A}$. For each of these colors $S_i/c_i \leq S_j/c_j$ (according to the sorting in Step (e)). 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 $\sum_{i=1}^{z} c_i \leq C$, thus, $\sum_{i=j}^{z} c_i \leq C - \sum_{i=1}^{j-1} c_i = \Gamma + C_f$.

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

$$\begin{aligned} S(C(OPT)) \leq \sum_{i=1}^{z} S_i &= \sum_{i=1}^{t} S_i + \sum_{i=t+1}^{j-1} S_i + \sum_{i=j}^{z} S_i \\ &\leq S_T + (S_{C(G)} - \Gamma \frac{S_j}{c_j}) + \frac{S_j}{c_j}(\Gamma + C_f) \\ &= S_T + S_{C(G)} + C' \frac{S_j}{c_j} < S_T + S_{C(G)} + S_j \end{aligned}$$

Clearly, $S(OPT) \leq S(C(OPT))$ since in OPT we might pack only a subset of the items from each color in $C(OPT)$. Now, since $\mathcal{A}$ packs at least all the items in colors of $T \cup C(G)$, we have that $S(\mathcal{A}) \geq S_T + S_{C(G)}$ which implies $S(OPT) - S(\mathcal{A}) < S_j$. Since T includes 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 implies the competitive ratio. ∎

Proof of Theorem 4.1: For finding an optimal packing of the items, we can use the following two-stage dynamic programming procedure. As before, we assume that $\hat{P}$ is an upper bound on the total profit.

- In the first stage, we compute recursively (as in Section 2.1) the values of $h_{k,r}(a)$, for $1 \leq k \leq M$, $1 \leq r \leq n_k$, and $1 \leq a \leq \hat{P}$.
- In the second stage, we use the values of $h_{k,r}(a)$ to compute recursively $f_k(a, w_1, \ldots, w_d)$, the minimum size sum of a subset of items whose colors are among the first k, $1 \leq k \leq M$, such that we use w_ℓ colors from compartment category ℓ, $1 \leq \ell \leq d$, and the total profit of the items is a, $1 \leq a \leq \hat{P}$. We initialize

$$f_0(a, w_1, \ldots, w_d) = \begin{cases} 0 & a = 0 \text{ and } w_1 = \cdots = w_d = 0 \\ +\infty & \text{otherwise} \end{cases}$$

  Assume that the k-th color has compartment requirement $\eta_{\ell'}$, then we use the following formula.

$$f_k(a, w_1, \ldots, w_d) = \min \begin{cases} f_{k-1}(a, w_1, \ldots, w_d) \\ \min\limits_{1 \leq a' \leq a} (f_{k-1}(a - a', w_1, \ldots, w_{\ell'-1}, \ldots, w_d) + h_{k,n_k}(a')) \end{cases}$$

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

$$\max_{a=0,\ldots,\hat{P};\ 0 \leq w_1,\ldots,w_d \leq M} \{a : f_M(a, w_1, \ldots, w_d) \leq V \text{ and } \sum_{\ell=1}^{d} w_\ell \eta_\ell \leq C\}.$$

∎

Proof of Theorem 4.2: We scale the instance in two steps.

(i) First, we find an r-approximation to the optimal profit. We can take, for example, all the colors k for which $c_k \leq C$, and generate an input that consists 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). The algorithm completes by assigning to the packed items their original colors, and selecting the color that achieved the maximum profit. This yields a $2M$-approximation for VPSD.

(ii) Let z be the solution value output by our r-approximation algorithm. Then we take for each item, i, $p'_i = \lceil \frac{p_i n}{\varepsilon z} \rceil$, and set $\hat{P} = r\lceil n/\varepsilon \rceil + n$ as an upper bound for the optimal profit.

As shown in [13], the above scaling yields a $(1+\varepsilon)$-approximation to the optimal profit (by using the configuration graph we get an optimal solution for the scaled instance). Hence, using the greedy $2M$-approximation, we get that the overall complexity is

$$O(n \log n + M^{w+1}\hat{P}) = O(n \log n + M^{w+1}(2M \cdot \lceil \frac{n}{\varepsilon} \rceil + n)),$$

which yields the statement of the theorem. ∎

Proof of Theorem 4.3: 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, from the first k colors. The values of $g_k(a)$ is computed for $1 \leq a \leq n$, and $1 \leq k \leq M$. Given the table g, we reconstruct the set of items for which a is maximal and $g_n(a) \leq C$. By definition of g, this set is an optimal solution. ∎

Proof sketch of Theorem 6.1: We outline below the technique that we use in the initial steps, which define the partition of the bins to *blocks*. The steps in which we pack the items are similar to the packing steps in the scheme of [14], and will be given in detail in the full version of the paper. Our scheme for VPSD instances with fixed number of colors proceeds as follows.

1. Initially, we partition the bins to $O(\lg(n/\varepsilon))$ *types*, such that the bins in any type j have volume in the range $[v, v(1+\varepsilon)]$, for some v, and the same number of compartments. More specifically, let v_{max} denote the maximal capacity of a bin. Consider the set of bins whose capacities are in the range $[\varepsilon v_{max}/n^{\frac{3M}{\varepsilon}}, v_{max}]$. We partition this set of bins into subsets, $A^0, A^1, \ldots$, such that A^j consists of the bins whose capacities are in the range
$$[\frac{\varepsilon v_{max}}{n^{\frac{3M}{\varepsilon}}}(1+\varepsilon)^j,\ \frac{\varepsilon v_{max}}{n^{\frac{3M}{\varepsilon}}}(1+\varepsilon)^{j+1}).$$
We further distinguish between the bins by the number of compartments. Let $r = 2^M$ be the number of subsets of colors that can be accommodated in any bin, and let $c_{s\ell}$ be the sum of the compartment requirements of the ℓ-th color set, $1 \leq \ell \leq r$. A bin b belongs to A^j_ℓ if the color set with *largest* sum of compartment requirements which can be accommodated in b is ℓ (ties are broken arbitrarily). Thus, we round down the number of compartments in all the bins in A^j_ℓ to $c_{s\ell}$. Hence, we have at most $r \cdot \log(n^{\frac{3M}{\varepsilon}}/\varepsilon) = O(\log(n/\varepsilon))$ distinct bin types.
Now, for any $U \subseteq I$ that is packed in the original set of bins, we can add $w \leq M$ bins, one for each color $1 \leq k \leq M$, and pack, for each color k, all the items in color k previously packed in bins of sizes smaller than $\varepsilon v_{max}/n^{\frac{3M}{\varepsilon}}$, in a *single* bin with c_k compartments. The total capacity of the w added bins is at most $\varepsilon v_{max}/n^{(\frac{3M}{\varepsilon}-1)} \leq \varepsilon v_{max}/n^{\frac{2M}{\varepsilon}}$. Therefore, we can transform the given instance to an instance where all the bins, except for $w \leq M$, are of sizes in the range $[\varepsilon v_{max}/n^{\frac{3M}{\varepsilon}}, v_{max}]$, arranged in the sets $\{A^j_\ell\}$ defined above.
Since the set U is unknown in advance, we have to guess $w \leq M$ and the partition of the extra capacity of the small bins (which is at most $\varepsilon v_{max}/n^{\frac{2M}{\varepsilon}}$) among the w new bins. The capacity of each of the new bins is a multiple $1 \leq z \leq n$ of the minimal size $\varepsilon v_{max}/n^{\frac{3M}{\varepsilon}}$. Thus, the overall number of guesses is $n^w \leq n^{O(M)}$.
After packing the items, it can be shown that by removing the w least profitable bins in each type, we lose at most a factor of ε in the approximation ratio.
2. In the second step, we guess (similar to [14]) the subset $U \subseteq I$ of items that are packed in the bins, such that $P(U) \geq (1-\varepsilon)OPT(I)$, and the packing is feasible.
3. Now, we determine how many bins of each type are assigned a given subset of colors (out of the $r = 2^M$ possible subsets). This gives a partition of the bins to *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.

4. Finally, we find (similar to [14]) a feasible packing of $U' \subseteq U$ in the bin blocks, such that $P(U') \geq (1-\varepsilon)P(U)$.

∎

B Approximation Scheme for Data Placement

B.1 Packing Small Blocks of Items with Small Compartment Requirements

In partitioning the remaining items to blocks, we assume that each block is packed in a different bin, thus, two blocks of the same color do not share the same set of compartments, and we can reduce our problem to GAP, where the cost of packing item k in bin j is c_{kj}. (In our case, $c_{kj} = c_k$ for all $1 \leq j \leq N$.) We can now use the technique of [15] for solving the GAP problem.

As a first step, we need to 'guess' the partition of the remaining item sets into small blocks. We note that a naive search over all possible partitions may result in exponential running time. We argue, however, that we can look only at partitions in which items whose size is smaller than $\varepsilon V/d$ are packed as one block in some bin, while items with larger sizes are partitioned to blocks whose sizes are integral multiples of $\varepsilon V/d$. This can be justified as follows. Consider the total contribution (i.e., profit) of each compartment category to bin j in some optimal solution. If any category contributes less than $\varepsilon V/d$, we can round up this contribution to $\varepsilon V/d$. After the items are packed, we can eliminate the extra volume (of at most εV) from bin j. Thus, we lose at most a factor of ε in the approximation ratio. In addition, when we pack in bin j items from some compartment category ℓ whose total profit is $r\varepsilon V/d$, we can only benefit from using the smallest number of blocks that achieve this profit. Thus, we can lose only a factor of ε in the approximation ratio by looking at partitions of the item sets to blocks of sizes that are integral multiples of $\varepsilon V/d$. The number of such partitions is $M^{O(d/\varepsilon)}$.

Now, given a *good* partition of the remaining items to small blocks, we use a linear program for packing the items in the bins. For convenience, we map the blocks to *items* in an instance of the GAP problem; that is, any block that belonged to an item from compartment category ℓ, becomes in the GAP instance an item of the same size and the packing cost η_ℓ, 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 may 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 to an item i (in the GAP instance) to bin j. We can formulate the packing problem as the following integer program.

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

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 solving the maximum weighted matching problem on that graph. We define a separate bipartite graph for each compartment category. As shown in [15], 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 $\varepsilon' = \varepsilon/d$, we get that the total extra volume in any bin is at most εV. We can omit the extra items and lose at most a factor of ε in the approximation ratio.