Discrete Applied Mathematics 160 (2012) 1591–1600

Contents lists available at SciVerse ScienceDirect

Discrete Applied Mathematics

journal homepage: www.elsevier.com/locate/dam

Vector bin packing with multiple-choice☆

Boaz Patt-Shamir*, Dror Rawitz

School of Electrical Engineering, Tel Aviv University, Tel Aviv 69978, Israel

ARTICLE INFO

Article history:
Received 16 March 2011
Received in revised form 7 December 2011
Accepted 14 February 2012
Available online 14 March 2012

Keywords:
Approximation algorithms
Multiple-choice vector bin packing
Multiple-choice multidimensional knapsack

ABSTRACT

We consider a variant of *bin packing* called *multiple-choice vector bin packing*. In this problem, we are given a set of n items, where each item can be selected in one of several D-dimensional *incarnations*. We are also given T bin types, each with its own cost and D-dimensional size. Our goal is to pack the items in a set of bins of minimum overall cost. The problem is motivated by scheduling in networks with guaranteed quality of service (QoS), but due to its general formulation it has many other applications as well. We present an approximation algorithm that is guaranteed to produce a solution whose cost is about $\ln D$ times the optimum. For the running time to be polynomial we require $D = O(1)$ and $T = O(\log n)$. This extends previous results for *vector bin packing*, in which each item has a single incarnation and there is only one bin type. To obtain our result we also present a PTAS for the multiple-choice version of *multidimensional knapsack*, where we are given only one bin and the goal is to pack a maximum weight set of (incarnations of) items in that bin.

1. Introduction

Bin packing, where one needs to pack a given set of items using the least number of limited-space containers (called bins), is one of the fundamental problems of combinatorial optimization (see, e.g., [22]). In the *multidimensional* flavor of bin packing, each item has sizes in several dimensions, and the bins have limited size in each dimension [18]. In this paper, we consider a natural generalization of multidimensional bin packing that occurs frequently in practice, namely *multiple-choice* multidimensional bin packing. In this variant, items and space are multidimensional, and in addition, each item may be selected in one of a few *incarnations*, each with possibly different sizes in the different dimensions. Similarly, bins can be selected from a set of types, each bin type with its own size cap in each dimension, and possibly different cost. The problem is to select incarnations of the items and to assign them to bins so that the overall cost of bins is minimized.

Multidimensionality models, the case where the objects to pack have costs in several incomparable budgets. For example, consider a distribution network (e.g., a cable-TV operator), which needs to decide which data streams to provide. Streams typically have prescribed bandwidth requirements, monetary costs, processing requirements etc., while the system typically has limited available bandwidth, a bound on the amount of funds dedicated to buying content, bounded processing power etc. The multiple-choice variant models, for example, the case where digital objects (such as video streams) may be taken in one of a variety of formats with different characteristics (e.g., bandwidth and processing requirements), and similarly, digital bins (e.g., server racks) may be configured in more than one way. The multiple-choice multidimensional variant is useful in many scheduling applications such as communication under Quality of Service (QoS) constraints, and including work-plans for nursing personnel in hospitals [29].

☆ A preliminary version of this work appears in Proc. 12th Scandinavian Symposium and Workshops on Algorithm Theory (SWAT), 2010. Research supported in part by the Next Generation Video (NeGeV) Consortium, Israel.

* Corresponding author. Fax: +972 3 640 7595.
E-mail addresses: boaz@eng.tau.ac.il (B. Patt-Shamir), rawitz@eng.tau.ac.il (D. Rawitz).

0166-218X/$ – see front matter © 2012 Elsevier B.V. All rights reserved.
doi:10.1016/j.dam.2012.02.020

Specifically, in this paper we consider the problem of *multiple-choice vector bin packing* (abbreviated MVBP, see Section 2 for a formal definition). The input to the problem is a set of *n items* and a set of *T bin types*. Each item is represented by at most *m incarnations*, where each incarnation is characterized by a D-dimensional vector representing the size of the incarnation in each dimension. Each bin type is also characterized by a D-dimensional vector representing the capacity of that bin type in each dimension. We are required to pack all items in the minimal possible number of bins, i.e., we need to select an incarnation for each item, select a number of required bins from each type, and give an assignment of item incarnations to bins so that no bin exceeds its capacity in any dimension. In the weighted version of this problem each bin type has an associated cost, and the goal is to pack item incarnations into a set of bins of minimum cost.

Before stating our results, we note that naïve reductions to the single-choice model do not work. For example, assume we are given n 2-dimensional single incarnation items, $n/2$ items of size $(2/n, 1)$ and $n/2$ items of size $(1, 2/n)$, and two 2-dimensional bin types whose sizes are $(1, n/2)$ and $(n/2, 1)$. In this case the first $n/2$ items can be packed together in a single type-1 bin but require $n/2$ type-2 bins, while the other $n/2$ items fit together in a single type-2 bin but require $n/2$ type-1 bins. If one uses only one bin type, the cost is dramatically larger than the optimum—even with one incarnation per item. Regarding the choice of item incarnation, one may try to use only a cost-effective incarnation for each item (using some natural definition). However, it is not difficult to see that this approach results in approximation ratio $\Omega(D)$ even when there is only one bin type.

1.1. Our results

In this paper we present a polynomial-time approximation algorithm for the multiple-choice vector bin packing problem in the case where D (the number of dimensions) is a constant. The approximation ratio for the general weighted version is $\ln 2D + 3 + \varepsilon$, for any constant $\varepsilon > 0$, assuming that T (the number of bin types) satisfies $T = O(\log n)$. For the unweighted case, the approximation ratio can be improved to $\ln 2D + 1 + \varepsilon$, for any constant $\varepsilon > 0$, if $T = O(1)$. Without any assumption on T, we can guarantee, in the unweighted case, cost of $(\ln(2D) + 1)\text{OPT} + T + 1$, where OPT denotes the optimal cost. To the best of our knowledge, this is the first approximation algorithm for the problem with multiple choice, and it is as good as the best solution for single-choice vector bin packing (see below).

As an aside, to facilitate our algorithm we consider the multiple-choice multidimensional *knapsack* problem (abbreviated MMK), where we are given a single bin and the goal is to load it with the maximum weight set of (incarnations of) items. We present a polynomial-time approximation scheme (PTAS) for MMK for the case where the dimension D is constant. The PTAS for MMK is used as a subroutine in our algorithm for MVBP.

1.2. Related work

Classical bin packing (BP) (single dimension, single choice) admits an asymptotic PTAS (APTAS) [9] and an asymptotic fully polynomial-time approximation scheme (AFPTAS) [17]. Friesen and Langston [10] presented constant factor approximation algorithms for *variable-sized* BP in which a fixed collection of bin sizes is allowed, and the cost of a solution is the sum of sizes of used bins. Murgolo [21] gave an AFPTAS for this version of BP. Epstein and Levin [8] obtained an APTAS for generalized cost variable-sized BP. Online variable-sized BP was considered in [27]. Correa and Epstein [7] considered BP with controllable item sizes. In this version of BP each item has a list of pairs associated with it. Each pair consists of an allowed size for this item, and a nonnegative penalty. The goal is to select a pair for each item so that the number of bins needed to pack the sizes plus the sum of penalties is minimized. Correa and Epstein [7] presented an APTAS that uses bins of size slightly larger than 1.

Regarding multidimensionality, it has been long known that vector bin packing (VBP, for short) can be approximated to within $D + \frac{1}{3}$ [12] and $D + \varepsilon$ [9]. More recently, Chekuri and Khanna [5] presented an $O(\log D)$-approximation algorithm for VBP, for the case where D is constant. They also showed that approximating VBP for arbitrary dimension is as hard as graph coloring, implying that it is unlikely that VBP admits approximation factor smaller than $\sqrt{D}$. The best known approximation ratio for VBP is due to Bansal et al. [3], who gave a polynomial-time approximation algorithm for constant dimension D with approximation ratio arbitrarily close to $\ln D + 1$. Our algorithm for MVBP is based on their ideas.

Knapsack admits an FPTAS [25,15]. Frieze and Clarke [11] presented a PTAS for the (single-choice) multidimensional variant of knapsack, but obtaining an FPTAS for multidimensional knapsack is NP-hard [20]. Since MMK generalizes KNAPSACK, it follows that our PTAS for MMK is the best possible result, unless $P = \text{NP}$. Shachnai and Tamir [28] use the approach of [11] to obtain a PTAS for a special case of 2-dimensional multiple-choice knapsack. Our algorithm for MMK extends their technique to the general case. MMK was studied extensively by practitioners. There are many specialized heuristics for MMK, see, e.g., [1,14,23,2]. Branch and bound techniques for MMK are studied in [19,26]. From the algorithmic viewpoint, the first relevant result is by Chandra et al. [4], who present a PTAS for single-dimension, multiple-choice knapsack. The reader is referred to [18] for a comprehensive treatment of knapsack problems.

1.3. Paper organization

The remainder of this paper is organized as follows. In Section 2 we formalize the problems. Our approximation algorithm for the MVBP problem is given in Section 3. This algorithm uses the PTAS for the MMK problem that is presented in Section 4. An alternative algorithm for unweighted MVBP is given in the Appendix.

2. Problem statements

We now formalize the optimization problems we deal with. For a natural number n, let $[n] \stackrel{\text{def}}{=} \{1, 2, \ldots, n\}$ (we use this notation throughout the paper).

Multiple-choice multidimensional knapsack problem (MMK).

Instance: A set of *n items*, where each item is a set of m or fewer D-dimensional *incarnations*. Incarnation j of item i has size $a_{ij} \in (\mathbb{R}^+)^D$, in which the dth dimension is a real number $a_{ijd} \geq 0$.
In the *weighted* version, each incarnation j of item i has weight $w_{ij} \geq 0$.
Solution: A set of incarnations, at most one of each item, such that the total size of the incarnations in each dimension d is at most 1.
Goal: Maximize the number (*weighted version:* total weight) of incarnations in a solution.

When $D = m = 1$, this is the classical Knapsack problem (KNAPSACK).

Multiple-choice vector bin packing (MVBP).

Instance: Same as for unweighted MMK, with the addition of T *bin types*, where each bin type t is characterized by a vector $b_t \in (\mathbb{R}^+)^D$. The dth coordinate of b_t is called the *capacity* of type t in dimension d, and denoted by b_{td}.
In the *weighted* version, each bin type t has a weight $w_t \geq 0$.
Solution: A set of bins, each assigned a bin type and a set of item incarnations, such that exactly one incarnation of each item is assigned to any bin, and such that the total size of incarnations assigned to a bin does not exceed its capacity in any dimension.
Goal: Minimize number (*weighted version:* total weight) of assigned bins.

When $m = 1$ we get VBP, and the special case where $D = m = 1$ is the classical bin packing problem (BP).

3. Multiple-choice vector bin packing

As shown in the Appendix it is possible to obtain approximation ratio of $O(\ln D)$ for MVBP by extending the technique of [5], under the assumption that bin types are of unit weight and that both D and T are constants. However, in this section we present a stronger result, namely an $O(\log D)$-approximation algorithm for MVBP, assuming that $D = O(1)$ and $T = O(\log n)$. Our algorithm is based on and extends the work of [3].

The general idea is as follows. We first encode MVBP using a covering linear program with exponentially many variables, but polynomially many constraints. We find a near optimal fractional solution to this (implicit) program using a separation oracle of the dual program. (The oracle is implemented by the MMK algorithm from Section 4.) We assign some incarnations to bins using a greedy rule based on some "well behaved" dual solution (the number of greedy assignments depends on the value of the solution to the primal program). Then we are left with a set of unassigned items, but due to our greedy rule we can assign these remaining items to a relatively small number of bins.

3.1. Covering formulation

We start with the transformation of MVBP to weighted Set Cover (SC). An instance of SC is a family of sets $\mathcal{C} = \{C_1, C_2, \ldots\}$ and a cost $w_C \geq 0$ for each $C \in \mathcal{C}$. We call $\bigcup_{C \in \mathcal{C}} C$ the *ground set* of the instance, and usually denote it by I. The goal in SC is to choose sets from $\mathcal{C}$ whose union is I and whose overall cost is minimal. Clearly, SC is equivalent to the following integer program:

$$\begin{aligned} \min \quad & \sum_{C \in \mathcal{C}} w_C \cdot x_C \\ \text{s.t.} \quad & \sum_{C \ni i} x_C \geq 1 \quad \forall i \in I \\ & x_C \in \{0, 1\} \quad \forall C \in \mathcal{C} \end{aligned} \tag{P}$$

where x_C indicates whether the set C is in the cover. A linear program relaxation is obtained by replacing the integrality constraints of (P) by positivity constraints $x_C \geq 0$ for every $C \in \mathcal{C}$. The above formulation is very general. We shall henceforth call problems whose instances can be formulated as in (P) for some $\mathcal{C}$ and w_C values, (P)*-problems*.

In particular, MVBP is a (P)-problem, as the following reduction shows. Let $\mathcal{I}$ be an instance of MVBP. Construct an instance $\mathcal{C}$ of SC as follows. The ground set of $\mathcal{C}$ is the set of items in $\mathcal{I}$, and sets in $\mathcal{C}$ are the subsets of items that can be assigned to some bin. Formally, a set C of items is called *compatible* if and only if there exists a bin type t and an incarnation mapping $f : C \to [m]$ such that $\sum_{i \in C} a_{if(i)d} \leq b_{td}$ for every dimension d, i.e., if there is a way to accommodate all members of C in the same bin. Using this definition, it seems that we should let $\mathcal{C}$ be the collection of all compatible item sets in the SC instance. However, note that in this case a solution to the SC instance does not immediately solves MVBP, because selecting

incarnations and a bin-type is an NP-hard problem in its own right.[1] To deal with this issue we introduce one variable for each possible *assignment* of incarnations and bin type. Namely, we may have more than one variable for a compatible item subset.

3.2. Dual oblivious algorithms

We shall be concerned with approximation algorithms for (P)-problems which have a special property with respect to the dual program. First, we define the dual to the LP-relaxation of (P):

$$\begin{aligned} \max \quad & \sum_{i\in I} y_i \\ \text{s.t.} \quad & \sum_{i\in C} y_i \le w_C \quad \forall C \in \mathcal{C} \\ & y_i \ge 0 \quad \forall i \in I. \end{aligned} \tag{D}$$

Next, for an instance $\mathcal{C}$ of sc and a set S, we define the *restriction of* $\mathcal{C}$ *to* S by

$$\mathcal{C}|_S \stackrel{\text{def}}{=} \{C \cap S \mid C \in \mathcal{C}\},$$

namely we project out all elements not in S. Note that for any S, a solution to $\mathcal{C}$ is also a solution to $\mathcal{C}|_S$: we may only discard some of the constraints in (P). We now arrive at our central concept.

Definition 1 (*Dual Obliviousness*)**.** Let Π be a (P)-problem. An algorithm A for Π is called ρ-*dual oblivious* if there exists a constant δ such that for every instance $\mathcal{C} \in \Pi$ there exists a dual solution $y \in \mathbb{R}^n$ to (D) satisfying, for all $S \subseteq I$, that

$$A(\mathcal{C}|_S) \le \rho \cdot \sum_{i\in S} y_i + \delta.$$

Let us show that the FIRST-FIT heuristic for BP is dual oblivious (we use this property later). In FIRST-FIT, the algorithm scans the items in arbitrary order and places each item in the left most bin which has enough space to accommodate it, possibly opening a new bin if necessary. A newly open bin is placed to the right of rightmost open bin. The following observation is based on Johnson's analysis of NEXT-FIT [16].

Observation 1. FIRST-FIT *is a* 2*-dual oblivious algorithm for bin packing.*

Proof. In any solution produced by FIRST-FIT, all non-empty bins except perhaps one are more than half-full. Furthermore, this property holds throughout the execution of FIRST-FIT, and regardless of the order in which items are scanned. It follows that if we let $y_i = a_i$, where a_i is the size of the ith item, then for every $S \subseteq I$ we have

$$\text{FIRST-FIT}\,(S) \le \max\left\{2\sum_{i\in S} y_i,\, 1\right\} \le 2\sum_{i\in S} y_i + 1,$$

and hence FIRST-FIT is dual oblivious for BP with $\rho = 2$ and $\delta = 1$. □

The usefulness of dual obliviousness is expressed in the following result. Let Π be a (P)-problem, and suppose that APPR is a ρ-dual oblivious algorithm for Π. Suppose further that we can efficiently find the dual solution y to (D) promised by dual obliviousness (Definition 1). Under these assumptions, Algorithm 1 solves any instance $\mathcal{C}$ of Π using a near-optimal fractional solution x^* of (P) that can be computed in polynomial time.

Algorithm 1

1: Compute a fractional solution x^* to (P). Let W^* denote its value.
2: Let $\mathcal{C}^+ = \{C : x^*_C > 0\}$. Let $\mathcal{G} \leftarrow \emptyset$, $S \leftarrow I$.
3: **while** $S \neq \emptyset$ and $\sum_{C\in\mathcal{G}} w_C < \ln\rho \cdot W^*$ **do**
4: Find $C \in \mathcal{C}^+$ for which $\frac{1}{w_C}\sum_{i\in S\cap C} y_i$ is maximized;
5: $\mathcal{G} \leftarrow \mathcal{G} \cup \{C\}$, $S \leftarrow S \setminus C$.
6: **end while**
7: Apply APPR to the residual instance S, obtaining solution $\mathcal{A}$.
8: **return** $\mathcal{G} \cup \mathcal{A}$.

We now bound the weight of the solution $\mathcal{G} \cup \mathcal{A}$ that is computed by Algorithm 1.

[1] An algorithm that selects item incarnations and a bin type can be used to solve PARTITION. Given a PARTITION instance $(a_1, \ldots, a_n)$, we construct the following instance: there are n items, where item i has two incarnations, $(a_i, 0)$ and $(0, a_i)$, and one bin type whose size is (A, A), where $A = \frac{1}{2}\sum_i a_i$.

Theorem 1. *Let Π be a (P)-problem and suppose that* APPR *is a ρ-dual oblivious algorithm (with an additive factor δ) for Π. Then for any instance of Π with a fractional solution x^*, Algorithm 1 outputs $\mathcal{G} \cup \mathcal{A}$ satisfying*

$$w(\mathcal{G} \cup \mathcal{A}) \leq (\ln \rho + 1)W^* + \delta + w_{\max},$$

where $W^* = \sum_C w_C \cdot x^*_C$ and $w_{\max} = \max_t w_t$.

Proof. Clearly, $w(\mathcal{G}) < \ln \rho \cdot W^* + w_{\max}$. It remains to bound the weight of $\mathcal{A}$. Let S' be the set of items not covered by $\mathcal{G}$. If $S' = \emptyset$, then we are done. Otherwise we prove that $\sum_{i \in S'} y_i \leq \frac{1}{\rho} \sum_{i \in I} y_i$, which implies

$$w(\mathcal{A}) \leq \rho \sum_{i \in S'} y_i + \delta \leq \rho \cdot \frac{1}{\rho} \cdot \sum_{i=1}^{n} y_i + \delta \leq W^* + \delta,$$

proving the theorem.

Let $C_k \in \mathcal{C}^+$ denote the kth subset added to $\mathcal{G}$ during the greedy phase, and let $S_k \subseteq I$ be the set of items not covered after the kth subset was chosen. Define $S_0 = I$. We prove, by induction on $|\mathcal{G}|$, that for every k,

$$\sum_{i \in S_k} y_i \leq \prod_{q=1}^{k} \left(1 - \frac{w_{C_q}}{W^*}\right) \cdot \sum_{i \in I} y_i. \tag{1}$$

For the base case we have trivially $\sum_{i \in S_0} y_i \leq \sum_{i \in I} y_i$. For the inductive step, assume that

$$\sum_{i \in S_{k-1}} y_i \leq \prod_{q=1}^{k-1} \left(1 - \frac{w_{C_q}}{W^*}\right) \cdot \sum_{i \in I} y_i.$$

By the greedy rule and the pigeonhole principle, we have that

$$\frac{1}{w_{C_k}} \sum_{i \in S_{k-1} \cap C_k} y_i \geq \frac{1}{W^*} \sum_{i \in S_{k-1}} y_i.$$

It follows that

$$\sum_{i \in S_k} y_i = \sum_{i \in S_{k-1}} y_i - \sum_{i \in S_{k-1} \cap C_k} y_i \leq \left(1 - \frac{w_{C_k}}{W^*}\right) \sum_{i \in S_{k-1}} y_i \leq \prod_{q=1}^{k} \left(1 - \frac{w_{C_q}}{W^*}\right) \cdot \sum_{i \in I} y_i,$$

completing the inductive argument.

Now let k be the number of iterations of the while loop. By (1) we have

$$\sum_{i \in S'} y_i \leq \prod_{q=1}^{k} \left(1 - \frac{w_{C_q}}{W^*}\right) \cdot \sum_{i \in I} y_i \leq \prod_{q=1}^{k} \left(1 - \frac{\frac{1}{k} \sum_q w_{C_q}}{W^*}\right) \cdot \sum_{i \in I} y_i \leq \left(1 - \frac{\ln \rho}{k}\right)^k \cdot \sum_{i \in I} y_i,$$

where the third inequality follows from $\sum_{q=1}^{k} w_{C_q} \geq \ln \rho \cdot W^*$. The theorem follows, since by (1) we have

$$\sum_{i \in S'} y_i \leq (1 - \ln \rho / k)^k \cdot \sum_{i \in I} y_i \leq e^{-\ln \rho} \sum_{i \in I} y_i,$$

as required. □

Note that if x^* can be found in polynomial time, and if APPR is a polynomial-time algorithm, then Algorithm 1 runs in polynomial time. In what follows we will use Theorem 1 with a $(1 + \varepsilon)$-approximate solution to the LP-relaxation of (P).

In this section we defined the notion of *dual obliviousness* of an algorithm. We note that Bansal et al. [3] defined a more general property of algorithms called *subset obliviousness*. (For example, a subset oblivious algorithm is associated with several dual solutions.) Furthermore, Bansal et al. showed that the asymptotic PTAS for BP from [9] with minor modifications is subset oblivious and used it to obtain a subset oblivious $(D + \varepsilon)$-approximation algorithm for MVBP. This paved the way to an algorithm for VBP, whose approximation guarantee is arbitrarily close to $\ln D + 1$. However, in the case of MVBP, using the above APTAS for BP (at least in a straightforward manner) would lead to a subset oblivious algorithm whose approximation guarantee is $(DT + \varepsilon)$. In the next section we present a $2D$-dual oblivious algorithm for weighted MVBP that is based on First-Fit.

3.3. Algorithm for multiple-choice vector bin packing

We now apply the framework of Theorem 1 to derive an approximation algorithm for MVBP. There are several gaps we need to fill.

First, we need to solve (P) for MVBP, which consists of a polynomial number of constraints (one for each item), but an exponential number of variables. We circumvent this difficulty as follows. Consider the dual of (P). The *separation problem* of the dual program in our case is to find (if it exists) a subset C with $\sum_{i\in C} y_i > w_C$ for given item profits $y_1, \ldots, y_n$. The separation problem can therefore be solved by testing, for each bin type, whether the optimum is greater than w_t, which in turn is simply an MMK instance, for which we present a PTAS in Section 4. In other words, the separation problem of the dual program (D) has a PTAS, and hence there exists a PTAS for the LP-relaxation of (P) [24,13].

Second, we need to construct a dual oblivious algorithm for MVBP. To do that, we introduce the following notation. For every item $i \in I$, incarnation j, dimension d, and bin type t we define the *load* of incarnation j of i on the dth dimension of bins of type by $\ell_{ijtd} = a_{ijd}/b_{td}$. For every item $i \in I$ we define the *effective load* of i as

$$\bar{\ell}_i = \min_{1\le j\le m, 1\le t\le T} \left\{ w_t \cdot \max_d \ell_{ijtd} \right\}.$$

Also, let $j(i)$ and $t(i)$ be the incarnation and bin type that allow the cheapest packing of item i, and let $d(i)$ be the worst dimension with respect to this packing. Formally:

$$j(i) = \operatorname*{argmin}_j \min_t \{w_t \cdot \max_d \ell_{ijtd}\}$$
$$t(i) = \operatorname*{argmin}_t \{w_t \cdot \max_d \ell_{ij(i)td}\}$$
$$d(i) = \operatorname*{argmax}_d \ell_{ij(i)t(i)d}.$$

Namely, we assume that $j(i)$, $t(i)$ and $d(i)$ are the choices of j, t and d that are taken in the definition of $\bar{\ell}_i$.

Our dual oblivious algorithm APPR for MVBP is as follows:

1. Divide the item set I into T subsets by letting $I_t \stackrel{\text{def}}{=} \{i : t(i) = t\}$.
2. Pack each subset I_t in bins of type t using FIRST-FIT, where the size of each item i is $a_{ij(i)d(i)}$.

(The second step of the algorithm was used in [9].)

Observe that the size of incarnation $j(i)$ of item i in dimension $d(i)$ is the largest among all other sizes of this incarnation. Hence, the solution computed by FIRST-FIT is feasible for I_t.

We now show that this algorithm is $2D$-dual oblivious.

Lemma 2. *Algorithm* APPR *above is a polynomial time* $2D$*-dual oblivious algorithm for* MVBP.

Proof. Consider an instance of MVBP with item set I, and let the corresponding set cover problem instance be $\mathcal{C}$. We show that there exists a dual solution $y \in \mathbb{R}^n$ such that for any $S \subseteq I$,

$$\text{APPR}\,(\mathcal{C}|_S) \le 2D \cdot \sum_{i\in S} y_i + \sum_{t=1}^{T} w_t.$$

Define $y_i = \bar{\ell}_i/D$ for every i. We claim that y is a feasible solution to (D). Let $C \in \mathcal{C}$ be a compatible item set. C induces some bin type t, and an incarnation $j'(i)$ for each $i \in C$. Let $d'(i) = \operatorname{argmax}_d \left\{a_{ij'(i)d}/b_{td}\right\}$, i.e., $d'(i)$ is a dimension of bin type t that receives maximum load from (incarnation $j'(i)$ of) item i. Then

$$\begin{aligned}
\sum_{i\in C} y_i &= \sum_{d=1}^{D} \sum_{\substack{i\in C \\ i:d'(i)=d}} \frac{\bar{\ell}_i}{D} \\
&\le \frac{1}{D} \sum_{d=1}^{D} \sum_{\substack{i\in C \\ i:d'(i)=d}} w_t \cdot \ell_{ij'(i)td} \\
&= \frac{w_t}{D} \sum_{d=1}^{D} \sum_{\substack{i\in C \\ i:d'(i)=d}} \frac{a_{ij'(i)d}}{b_{td}} \\
&\le \frac{w_t}{D} \sum_{d=1}^{D} \frac{1}{b_{td}} \cdot b_{td} \\
&= w_t,
\end{aligned}$$

where the last inequality follows from the compatibility of C.

Now, since FIRST-FIT computes bin assignments that occupy at most twice the sum of bin sizes, we have that

$$\text{FIRST-FIT}\,(I_t) \le w_t \cdot \max\left\{2\sum_{i\in I_t}\bar{\ell}_i/w_t,\, 1\right\} \le 2\sum_{i\in I_t}\bar{\ell}_i + w_t.$$

Hence, for every instance $\mathcal{I}$ of MVBP we have

$$\begin{aligned}
\text{APPR}\,(\mathcal{I}) &= \sum_{t=1}^{T}\text{FIRST-FIT}\,(I_t)\\
&\le \sum_{t=1}^{T}\left(2\sum_{i\in I_t}\bar{\ell}_i + w_t\right)\\
&= 2\sum_{i\in I}\bar{\ell}_i + \sum_{t=1}^{T} w_t\\
&= 2D\sum_{i\in I} y_i + \sum_{t=1}^{T} w_t\\
&\le 2D\cdot \text{OPT}^* + \sum_{t=1}^{T} w_t.
\end{aligned}$$

Furthermore, for every $S \subseteq I$ we have

$$\text{APPR}\,(\mathcal{C}|_S) = \sum_{t=1}^{T}\text{FIRST-FIT}\,(S\cap I_t) \le 2\sum_{i\in S}\bar{\ell}_i + \sum_{t=1}^{T} w_t = 2D\sum_{i\in S} y_i + \sum_{t=1}^{T} w_t,$$

and we are done. □

Based on Theorem 1 and Lemma 2 we obtain our main result.

Theorem 2. *If $D = O(1)$, then there exists a polynomial time algorithm for* MVBP *with T bin types that computes a solution whose size is at most*

$$(\ln 2D + 1 + \varepsilon)\text{OPT}^* + \sum_{t=1}^{T} w_t + w_{\max}.$$

Note that while the approximation ratio is logarithmic in D, the running time of the algorithm is exponential in D. Theorem 2 implies the following result for unweighted MVBP:

Corollary 3. *If $D = O(1)$, then there exists a polynomial time algorithm for unweighted* MVBP *that computes a solution whose size is at most*

$$(\ln 2D + 1 + \varepsilon)\text{OPT}^* + T + 1.$$

Furthermore, if $T = O(1)$, then there exists a polynomial time $(\ln 2D + 1 + \varepsilon)$-approximation algorithm for unweighted MVBP, *for every $\varepsilon > 0$.*

We also have the following for weighted MVBP.

Corollary 4. *If $D = O(1)$ and $T = O(\log n)$, then there exists a polynomial time $(\ln 2D + 3 + \varepsilon)$-approximation algorithm for* MVBP.

Proof. The result follows from the fact that as we show, we may assume that $\sum_t w_t \le \text{OPT}$. In this case, due to Theorem 2 we have that the cost of the computed solution is at most

$$(\ln 2D + 1 + \varepsilon)\text{OPT}^* + \sum_{t=1}^{T} w_t + w_{\max} \le (\ln 2D + 3 + \varepsilon)\text{OPT}.$$

The above assumption is fulfilled by the following wrapper for our algorithm: Guess which bin types are used in some optimal solution. Iterate through all $2^T - 1$ guesses, and for each guess, compute a solution for the instance that contains only the bin types in the guess. Output the best solution. Since our algorithm computes a $(\ln 2D + 3 + \varepsilon)$-approximate solution for the right guess, the best solution is also a $(\ln 2D + 3 + \varepsilon)$-approximation. □

4. Multiple-choice multidimensional knapsack

In this section we present a PTAS for weighted MMK for the case where D is a constant. Our construction extends the algorithms of Frieze and Clarke [11] and of Shachnai and Tamir [28].

We first present a linear program of MMK, where the variables x_{ij} indicate whether the jth incarnation of the ith item is selected.

$$\begin{aligned} \max \quad & \sum_{i=1}^{n}\sum_{j=1}^{m} w_{ij}x_{ij} \\ \text{s.t.} \quad & \sum_{i=1}^{n}\sum_{j=1}^{m} a_{ijd}x_{ij} \le 1 \quad \forall d \in [D] \\ & \sum_{j=1}^{m} x_{ij} \le 1 \quad \forall i \in [n] \\ & x_{ij} \ge 0 \quad \forall i \in [n],\ j \in [m]. \end{aligned} \tag{MMK}$$

The first type of constraints make sure that the load on the knapsack in each dimension is bounded by 1; the second type of constraints ensures that at most one copy of each element is taken into the solution. Constraints of the third type indicate the relaxation: the integer program for MMK requires that $x_{ij} \in \{0, 1\}$.

Our PTAS for MMK is based on the linear program (MMK). Let $\varepsilon > 0$. Suppose we somehow guess the heaviest q incarnations that are packed in the knapsack by some optimal solution, for $q = \min\{n, \lceil D/\varepsilon \rceil\}$. Formally, assume we are given a set $G \subseteq [n]$ of at most q items and a function $g : G \to [m]$ that selects incarnations of items in G. In this case we can assign values to some variables of (MMK) as follows:

$$x_{ij} = \begin{cases} 1, & \text{if } i \in G \text{ and } j = g(i) \\ 0, & \text{if } i \in G \text{ and } j \ne g(i) \\ 0, & \text{if } i \notin G \text{ and } w_{ij} > \min\{w_{\ell g(\ell)} \mid \ell \in G\}. \end{cases}$$

That is, if we guess that incarnation j of item i is in the optimal solution, then $x_{ij} = 1$ and $x_{ij'} = 0$ for $j' \ne j$; also, if the jth incarnation of item i weighs more than some incarnation in our guess, then $x_{ij} = 0$. Denote the resulting linear program MMK (G, g).

Let $x^*(G, g)$ be an optimal (fractional) solution of MMK (G, g). The idea of Algorithm 2 below is to simply round down the values of $x^*(G, g)$. We show that if G and g are indeed the heaviest incarnations in the knapsack, then the rounded-down solution is very close to the optimum. Therefore, in the algorithm we loop over all possible assignments of G and g and output the best solution.

Algorithm 2

1: **for all** $G \subseteq [n]$ such that $|G| \le q$ and $g : G \to [m]$ **do**
2: $\quad b_d(G, g) \leftarrow 1 - \sum_{i \in G} a_{ig(i)d}$ for every $d \in [D]$
3: $\quad$ **if** $b_d(G, g) \ge 0$ for every d **then**
4: $\quad\quad$ Compute an optimal basic solution $x^*(G, g)$ of MMK(G, g)
5: $\quad\quad$ $x_{ij}(G, g) \leftarrow \lfloor x^*_{ij}(G, g) \rfloor$ for every i and j
6: $\quad$ **end if**
7: $\quad$ $x \leftarrow \operatorname{argmax}_{x(G,g)} w \cdot x(G, g)$
8: **end for**
9: **return** x

Theorem 5. *If $D = O(1)$, then Algorithm 2 is a PTAS for* MMK.

Proof. Regarding running time, note that there are $O(n^q)$ choices of G, and $O(m^q)$ choices of g for each choice of G, and hence, the algorithm runs for $O((nm)^q) = O((nm)^{\lceil D/\varepsilon \rceil})$ iterations, i.e., time polynomial in the input length, for constant D and ε.

Regarding approximation, fix an optimal integral solution x^I to (MMK). If x^I assigns at most q incarnations of items to the knapsack, then we are done. Otherwise, let G^I be the set of items that correspond to the q heaviest incarnations selected to the knapsack by x^I. For $i \in G^I$, let $g^I(i)$ denote the incarnation of i that was put in the knapsack by x^I. Consider the iteration of Algorithm 2 in which $G = G^I$ and $g = g^I$. Clearly, $w \cdot x^I \le w \cdot x^*(G^I, g^I)$. Let n' denote the number of items that are not contained in G^I and were not eliminated by g^I because all their incarnations weigh too much. Let k be the number of variables of the form x_{ij} in MMK (G^I, g^I). When using slack form, we add $D + n'$ slack variables: each constraint of the first type is written as $\sum_{i=1}^{n}\sum_{j=1}^{m} a_{ijd}x_{ij} + s_d = 1$, for $1 \le d \le D$, and each constraint of the second type is written as $\sum_{j=1}^{m} x_{ij} + s'_i = 1$, for $1 \le i \le n'$. Thus, the total number of variables in the program to $k + D + n'$, and since MMK (G^I, g^I) has $D + n'$ constraints (excluding positivity constraints $x_{ij} \ge 0$), it follows any basic solution of MMK (G^I, g^I) has at most $D + n'$ positive variables. Since by constraints of the second type in MMK (G^I, g^I) there is at least one positive variable for each item, it follows that

$x^*(G^l, g^l)$ has at most D non-integral entries, and therefore, by rounding down $x^*(G^l, g^l)$ we lose at most D incarnations of items. Let $W^l = \sum_{i \in G^l} w_{ig^l(i)}$. Then each incarnation lost due to rounding weighs at most W^l/q (because it is not one of the q heaviest). We conclude that

$$w \cdot x(G^l, g^l) \geq w \cdot x^*(G^l, g^l) - D \cdot \frac{W^l}{q} \geq w \cdot x^*(G^l, g^l)\left(1 - \frac{D}{q}\right) \geq w \cdot x^l\left(1 - \frac{D}{q}\right) = \frac{\text{OPT}}{1+\varepsilon},$$

and we are done. □

Acknowledgments

The work of Boaz Patt-Shamir was supported in part by the Israel Science Foundation (grant 1372/09) and by Israel Ministry of Science and Technology.

Appendix. Alternative algorithm for unweighted multiple-choice vector bin-packing

In this section we present an $O(\log D)$-approximation algorithm for MVBP that is based on the algorithm for VBP by Chekuri and Khanna [5]. This algorithm is simpler than the one presented in Section 3, but it only applies unweighted MVBP, and its running time is exponential in both D and T.

Consider an assignment of n items to bins. Clearly, there is no need to use more than n bins. Hence, if a feasible solution uses α_t bins of type t, for any t, we have $\sum_t \alpha_t \leq n$. A vector $(\alpha_1, \ldots, \alpha_T) \in \mathbb{N}^T$, where $\sum_t \alpha_t \leq n$ is called a *bin configuration*. Any feasible solution of MVBP corresponds to some bin configuration.

The first stage of our algorithm is to find the bin configuration $(\alpha_1, \ldots, \alpha_T)$ with least number of bins $\alpha = \sum_t \alpha_t$ that can contain fractional assignments of the items. That is, we find a bin configuration $(\alpha_1, \ldots, \alpha_T)$ with least number of bins α for which the following linear program has a fractional feasible solution:

$$\begin{aligned}
&\sum_{j=1}^{m}\sum_{k=1}^{\alpha} x_{ijk} = 1 \quad \forall i \in [n] \\
&\sum_{i=1}^{n}\sum_{j=1}^{m} a_{ijd}x_{ijk} \leq b_{\text{type}(k)d} \quad \forall k \in [\alpha],\ d \in [D] \\
&x_{ijk} \geq 0 \quad \forall i \in [n],\ j \in [m],\ k \in [\alpha]
\end{aligned} \tag{2}$$

where type(k) is the bin type of bin k. The variable x_{ijk} indicates whether the jth incarnation of the ith item is assigned to the kth bin. The first set of constraints ensures that one incarnation of each item is assigned to some bin. The second set of constraints makes sure that the load in any bin in any dimension is bounded by the appropriate capacity. Notice that since $\alpha \leq n$, there are $\binom{n+T-1}{n}$ interesting bin configurations. Since $\binom{n+T-1}{n} = O(n^{T-1})$ if T is a constant, we are able to find the required bin configuration in polynomial time.

Observation 3. $\alpha \leq \text{OPT}$.

Next, we show that a basic feasible solution of (2) is close to integral, namely that most of its entries are integral.

Lemma 4. *Let x be a basic feasible solution of (2) that corresponds to the bin configuration $(\alpha_1, \ldots, \alpha_T)$ with minimum number of bins $\alpha = \sum_t \alpha_t$. Then, x induces at most MD fractional assignments of items.*

Proof. The number of constraints in (2), excluding constraints of the form $x_{ijk} \geq 0$, is $n + \alpha D$, and the number of variables in is $nm\alpha$. When using slack form, we add αD slack variables, and this brings us to $nm\alpha + \alpha D$ variables. It follows that a basic solution x has at most $n + \alpha D$ positive entries.

Now, since $\sum_{j=1}^{m}\sum_{k=1}^{\alpha} x_{ijk} = 1$ for every $i \in [n]$, at least one entry that corresponds to the ith item is positive, for every i. That is, there is at least one non-zero variable among the set of variables $\{x_{ijk} : j \in [m], k \in [\alpha]\}$ for every i. Since there are at most $n + \alpha D$ positive entries of x, only up to αD items correspond to two or more positive entries. If follows that at most αD items have fractional assignments and the rest have integral assignments. □

Let $\varepsilon > 0$ be a constant that will be determined later. Our algorithm has three phases:

1. We start by finding a basic solution x of (2) that correspond to the optimal α. From Lemma 4 it follows that x contains at most αD fractional bin assignments. Namely, x determines the incarnation and bin of every item, apart from a set S of at most αD items.
2. Next, we place the items in S in new bins as follows. We repeatedly find a set $T \subseteq S$ of size $\ell = \lceil 1/\varepsilon \rceil$ items that can be placed together in one of the bins using exhaustive search. Observe that there are at most $\binom{n}{\ell} = O(n^\ell)$ such sets, and for each such set there are m^ℓ possible combinations of incarnations. When we can no longer pack any ℓ items in one bin, this process terminates. Since S contains at most αD items, we used up to $\frac{\alpha D}{\ell}$ bins during this phase.

3. Consider the remaining items. Since no ℓ of them can be placed in any bin, we are left with a set cover instance in which sets are smaller than ℓ. Each set corresponds to a subset of the remaining items that have incarnations that can be placed together in some bin. There are at most $\sum_{i=1}^{\ell-1}\binom{n}{i} = O(n^{\ell-1})$ possible item sets and each set has $m^{\ell-1}$ incarnations that may be placed in T possible bins. The next phase is simply to use the greedy algorithm for set cover whose approximation ratio is $H_{\ell-1}$ [6].

To summarize, we used at most $\alpha \leq \text{OPT}$ bins for the items whose bin and incarnation were determined by the basic solution x. We used $\frac{\alpha D}{\ell} \leq \varepsilon D \cdot \text{OPT}$ bins to pack items in sets of size ℓ and $H_{\ell-1} \cdot \text{OPT} = O(\ln \frac{1}{\varepsilon}) \cdot \text{OPT}$ bins to pack the rest of the items. This brings us to the following result:

Theorem 6. *Let D and T be constants. There exists a polynomial-time $(1+\varepsilon D+O(\ln \frac{1}{\varepsilon}))$-approximation algorithm for unweighted* MVBP*, for any constant $\varepsilon > 0$.*

By setting $\varepsilon = \frac{1}{D}$ we obtain the following:

Corollary 7. *Let D and T be constants. There exists a polynomial time $O(\log D)$-approximation algorithm for unweighted* MVBP.

References

[1] M.M. Akbar, E.G. Manning, G.C. Shoja, S. Khan, Heuristic solutions for the multiple-choice multi-dimension knapsack problem, in: International Conference on Computational Science—Part II, 2001, pp. 659–668.
[2] M.M. Akbar, M.S. Rahman, M. Kaykobad, E.G. Manning, G.C. Shoja, Solving the multidimensional multiple-choice knapsack problem by constructing convex hulls, Comput. Oper. Res. 33 (2006) 1259–1273.
[3] N. Bansal, A. Caprara, M. Sviridenko, Improved approximation algorithms for multidimensional bin packing problems, in: 47th IEEE Annual Symposium on Foundations of Computer Science, 2006, pp. 697–708.
[4] A.K. Chandra, D.S. Hirschberg, C.K. Wong, Approximate algorithms for some generalized knapsack problems, Theoret. Comput. Sci. 3 (3) (1976) 293–304.
[5] C. Chekuri, S. Khanna, On multidimensional packing problems, SIAM J. Comput. 33 (4) (2004) 837–851.
[6] V. Chvátal, A greedy heuristic for the set-covering problem, Math. Oper. Res. 4 (3) (1979) 233–235.
[7] J.R. Correa, L. Epstein, Bin packing with controllable item sizes, Inform. Comput. 206 (8) (2008) 1003–1016.
[8] L. Epstein, A. Levin, An aptas for generalized cost variable-sized bin packing, SIAM J. Comput. 38 (1) (2008) 411–428.
[9] W. Fernandez de la Vega, G.S. Lueker, Bin packing can be solved within 1+epsilon in linear time, Combinatorica 1 (4) (1981) 349–355.
[10] D.K. Friesen, M.A. Langston, Variable sized bin packing, SIAM J. Comput. 15 (1) (1986) 222–230.
[11] A.M. Frieze, M.R.B. Clarke, Approximation algorithms for the m-dimensional $0-1$ knapsack problem: worst-case and probabilistic analyses, European J. Oper. Res. 15 (1984) 100–109.
[12] M.R. Garey, R.L. Graham, D.S. Johnson, A.C. Yao, Resource constrained scheduling as generalized bin packing, J. Combin. Theory Ser. A 21 (3) (1976) 257–298.
[13] M. Grötschel, L. Lovasz, A. Schrijver, Geometric Algorithms and Combinatorial Optimization, Springer-Verlag, 1988.
[14] M. Hifi, M. Michrafy, A. Sbihi, Heuristic algorithms for the multiple-choice multidimensional knapsack problem, J. Oper. Res. Soc. 55 (2004) 1323–1332.
[15] O.H. Ibarra, C.E. Kim, Fast approximation algorithms for the knapsack and sum of subset problems, J. ACM 22 (4) (1975) 463–468.
[16] D.S. Johnson, Near-optimal bin packing algorithms, Ph.D. Thesis, MIT, Cambridge, MA, 1973.
[17] N. Karmarkar, R.M. Karp, An efficient approximation scheme for the one-dimensional bin-packing problem, in: 23rd IEEE Annual Symposium on Foundations of Computer Science, 1982, pp. 312–320.
[18] H. Kellerer, U. Pferschy, D. Pisinger, Knapsack Problems, Springer-Verlag, 2004.
[19] M.S. Khan, Quality adaptation in a multisession multimedia system: model, algorithms and architecture, Ph.D. Thesis, Dept. of Electrical and Computer Engineering, 1998.
[20] M.J. Magazine, M.-S. Chern, A note on approximation schemes for multidimensional knapsack problems, Math. Oper. Res. 9 (2) (1984) 244–247.
[21] F.D. Murgolo, An efficient approximation scheme for variable-sized bin packing, SIAM J. Comput. 16 (1) (1987) 149–161.
[22] C.H. Papadimitriou, K. Steiglitz, Combinatorial Optimization: Algorithms and Complexity, Prentice-Hall, 1981.
[23] R. Parra-Hernández, N.J. Dimopoulos, A new heuristic for solving the multichoice multidimensional knapsack problem, IEEE Trans. Syst. Man Cybern. A 35 (5) (2005) 708–717.
[24] S.A. Plotkin, D.B. Shmoys, É Tardos, Fast approximation algorithms for fractional packing and covering problems, Math. Oper. Res. 20 (1995) 257–301.
[25] S. Sahni, Approximate algorithms for the 0/1 knapsack problem, J. ACM 22 (1) (1975) 115–124.
[26] A. Sbihi, A best first search exact algorithm for the multiple-choice multidimensional knapsack problem, J. Comb. Optim. 13 (4) (2007) 337–351.
[27] S.S. Seiden, R. van Stee, L. Epstein, New bounds for variable-sized online bin packing, SIAM J. Comput. 32 (2) (2003) 455–469.
[28] H. Shachnai, T. Tamir, Approximation schemes for generalized 2-dimensional vector packing with application to data placement, in: 6th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems, in: LNCS, vol. 2764, 2003, pp. 129–148.
[29] D. Warner, J. Prawda, A mathematical programming model for scheduling nursing personnel in a hospital, Manage. Sci. 19 (1972) 411–422 (Application Series Part 1).