Vector Bin Packing with Multiple-Choice*
(Extended Abstract)

Boaz Patt-Shamir** and Dror Rawitz

School of Electrical Engineering,
Tel-Aviv University, Tel-Aviv 69978, Israel
{boaz,rawitz}@eng.tau.ac.il

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., [1]). In the *multidimensional* flavor of bin packing, each item has sizes in several dimensions, and the bins have limited size in each dimension [2]. 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.

* Research supported in part by the Next Generation Video Consortium, Israel.
** Supported in part by the Israel Science Foundation, Grant 664/05.

H. Kaplan (Ed.): SWAT 2010, LNCS 6139, pp. 248–259, 2010.

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 [3].

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, consider the case where $n/2$ 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.

Our results. 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 $\ln 2D + 3$, 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 $T = O(1)$ as well. 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 multi-dimensional *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.

Related work. Classical bin packing (BP) (single dimension, single choice) admits an asymptotic PTAS [4] and an asymptotic fully polynomial-time approximation scheme (asymptotic FPTAS) [5]. Friesen and Langston [6] presented constant factor approximation algorithms for a more general version of 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. For more details about this version of BP see [7] and references therein. Correa and Epstein [8] 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 [8] presented an asymptotic PTAS 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 a factor of $O(D)$ [9,4]. More recently, Chekuri and Khanna [10] 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, Caprara and Sviridenko [11], 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 [12,13]. Frieze and Clarke [14] presented a PTAS for the (single-choice) multidimensional variant of knapsack, but obtaining an FPTAS for multidimensional knapsack is NP-hard [15]. Shachnai and Tamir [16] use the approach of [14] 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., [17,18,19,20]. Branch and bound techniques for MMK are studied in [21,22]. From the algorithmic viewpoint, the first relevant result is by Chandra et al. [23], who present a PTAS for single-dimension, multiple-choice knapsack. The reader is referred to [24] for a comprehensive treatment of knapsack problems.

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.

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 (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 (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

It is possible to obtain approximation ratio of $O(\ln D)$ for MVBP by extending the technique of [10], under the assumption that bin types are of unit weight and that both D and T are constants. 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 [11].

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:

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

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 is the same bin. In the instance of SC, we let $\mathcal{C}$ be the collection of all compatible item sets. Note that a solution to set cover does not immediately solve MVBP, because selecting incarnations and bin-types is an NP-hard problem in its own right. To deal with this issue we have 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):

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

Next, for an instance $\mathcal{C}$ of set cover 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*

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

Let us show that the First-Fit (FF) heuristic for BP is dual oblivious (we use this property later). In FF, 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.

Observation 1. *First-Fit is a 2-dual oblivious algorithm for bin packing.*

Proof. In any solution produced by FF, all non-empty bins except perhaps one are more than half-full. Furthermore, this property holds throughout the execution of FF, 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 $\text{FF}(S) \leq \max\{2\sum_{i \in S} y_i, 1\} \leq 2\sum_{i \in S} y_i + 1$, and hence FF 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 promised by dual obliviousness. Under these assumptions, Algorithm 1 below solves any instance $\mathcal{C}$ of Π.

Algorithm 1

1: Find an optimal solution x^* to (P). Let OPT^* denote its value.
2: Let $\mathcal{C}^+ = \{C \ : \ x^*_C > 0\}$. Let $\mathcal{G} \leftarrow \emptyset, S \leftarrow I$.
3: **while** $\sum_{C \in \mathcal{G}} w_c < \ln \rho \cdot \text{OPT}^*$ **do**
4: Find $C \in \mathcal{C}^+$ for which $\frac{1}{w_C} \sum_{i \in S \setminus C} y_i$ 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 bound the weight of the solution $\mathcal{G} \cup \mathcal{A}$ that is computed by Algorithm 1.

Theorem 1. *Let Π be a (P)-problem. Then for any instance of Π with optimal fractional solution OPT^*, Algorithm 1 outputs $\mathcal{G} \cup \mathcal{A}$ satisfying*

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

where $w_{\max} = \max_t w_t$.

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

$$w(\mathcal{A}) \;\leq\; \rho \textstyle\sum_{i \in S'} y_i + \delta \;\leq\; \rho e^{-\ln \rho} \sum_{i=1}^{n} y_i + \delta \;\leq\; \text{OPT}^* + \delta \;,$$

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,

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

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

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

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

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

It follows that

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

completing the inductive argument. The theorem follows, since by (1) we have

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

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. Also observe that Theorem 1 holds even if x^* is not an optimal solution of (IP), but rather a $(1 + \varepsilon)$-approximation. We use this fact later.

In this section we defined the notion of *dual obliviousness* of an algorithm. We note that Bansal et al. [11] 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 [4] 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 (D) has a PTAS, and hence there exists a PTAS for the LP-relaxation of (P) [25,26].

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 $t(i)$ denote the bin type that can contain the most (fractional) copies of some incarnation of item i, where $j(i)$ and $d(i)$ are the incarnation and dimension that determine this bound. Formally:

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

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 FF, where the size of each item i is $a_{ij(i)d(i)}$.

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 FF 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 $S \subseteq I$, $\text{APPR}(\mathcal{C}|_S) \leq 2D \cdot \sum_{i \in S} y_i + \sum_{t=1}^{T} w_t$. 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 $d'(i) = \text{argmax}_d \left\{ a_{ij'(i)d} / b_{td} \right\}$, i.e., $d'(i)$ is a dimension of bin type t that receives maximum load from (incarnation $j'(i)$ of) item i. Then

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

where the last inequality follows from the compatibility of C.

Now, since FF computes bin assignments that occupy at most twice the sum of bin sizes, we have that $\text{FF}(I_t) \leq w_t \cdot \max\{2 \sum_{i \in I_t} \bar{\ell}_i / w_t, 1\} \leq 2 \sum_{i \in I_t} \bar{\ell}_i + w_t$. Hence, for every instance $\mathcal{I}$ of MVBP we have

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

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

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

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*

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

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 1. *If $D = O(1)$, then there exists a polynomial time algorithm for unweighted* MVBP *that computes a solution whose size is at most* $(\ln 2D + 1)\text{OPT}^* + T + 1$. *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 2. *If $D = O(1)$ and $T = O(\log n)$, then there exists a polynomial time* $(\ln 2D + 3)$*-approximation algorithm for* MVBP.

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

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

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 $(\ln 2D + 3)$-approximate solution for the right guess, the best solution is also a $(\ln 2D + 3)$-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 [14] and of Shachnai and Tamir [16].

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

$$\begin{aligned} \max\ & \textstyle\sum_{i=1}^{n}\sum_{j=1}^{m} w_{ij}x_{ij} & & \\ \text{s.t.}\ & \textstyle\sum_{i=1}^{n}\sum_{j=1}^{m} a_{ijd}x_{ij} \leq 1 & & \forall d \in [D] \\ & \textstyle\sum_{j=1}^{m} x_{ij} \leq 1 & & \forall i \in [n] \\ & x_{ij} \geq 0 & & \forall i \in [n], j \in [m] \end{aligned} \qquad \text{(MMK)}$$

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:

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

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 $j' \neq j$; 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

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

The proof of the next theorem is omitted for lack of space.

Theorem 3. *If $D = O(1)$, then Algorithm 2 is a PTAS for* MMK.

References

1. Papadimitriou, C.H., Steiglitz, K.: Combinatorial optimization: algorithms and complexity. Prentice-Hall, Englewood Cliffs (1981)
2. Kellerer, H., Pferschy, U., Pisinger, D.: Knapsack Problems. Springer, Berlin (2004)
3. Warner, D., Prawda, J.: A mathematical programming model for scheduling nursing personnel in a hospital. Manage. Sci. (Application Series Part 1) 19, 411–422 (1972)
4. Fernandez de la Vega, W., Lueker, G.S.: Bin packing can be solved within 1+epsilon in linear time. Combinatorica 1(4), 349–355 (1981)
5. Karmarkar, N., Karp, R.M.: An efficient approximation scheme for the one-dimensional bin-packing problem. In: 23rd IEEE Annual Symposium on Foundations of Computer Science, pp. 312–320 (1982)

6. Friesen, D.K., Langston, M.A.: Variable sized bin packing. SIAM J. Comput. 15(1), 222–230 (1986)
7. Seiden, S.S., van Stee, R., Epstein, L.: New bounds for variable-sized online bin packing. SIAM J. Comput. 32(2), 455–469 (2003)
8. Correa, J.R., Epstein, L.: Bin packing with controllable item sizes. Information and Computation 206(18), 1003–1016 (2008)
9. Garey, M.R., Graham, R.L., Johnson, D.S., Yao, A.C.: Resource constrained scheduling as generalized bin packing. J. Comb. Theory, Ser. A 21(3), 257–298 (1976)
10. Chekuri, C., Khanna, S.: On multidimensional packing problems. SIAM J. Comput. 33(4), 837–851 (2004)
11. Bansal, N., Caprara, A., Sviridenko, M.: Improved approximation algorithms for multidimensional bin packing problems. In: 47th IEEE Annual Symposium on Foundations of Computer Science, pp. 697–708 (2006)
12. Sahni, S.: Approximate algorithms for the 0/1 knapsack problem. J. ACM 22(1), 115–124 (1975)
13. Ibarra, O.H., Kim, C.E.: Fast approximation algorithms for the knapsack and sum of subset problems. J. ACM 22(4), 463–468 (1975)
14. Frieze, A.M., Clarke, M.R.B.: Approximation algorithms for the m-dimensional $0-1$ knapsack problem: worst-case and probabilistic analyses. Eur. J. Oper. Res. 15, 100–109 (1984)
15. Magazine, M.J., Chern, M.S.: A note on approximation schemes for multidimensional knapsack problems. Math. Oper. Res. 9(12), 244–247 (1984)
16. Shachnai, H., Tamir, T.: Approximation schemes for generalized 2-dimensional vector packing with application to data placement. In: Arora, S., Jansen, K., Rolim, J.D.P., Sahai, A. (eds.) RANDOM 2003 and APPROX 2003. LNCS, vol. 2764, pp. 165–177. Springer, Heidelberg (2003)
17. Akbar, M. M., Manning, E.G., Shoja, G.C., Khan, S.: Heuristic solutions for the multiple-choice multi-dimension knapsack problem. In: International Conference on Computational Science-Part II, pp. 659–668 (2001)
18. Hifi, M., Michrafy, M., Sbihi, A.: Heuristic algorithms for the multiple-choice multidimensional knapsack problem. J. Oper. Res. Soc. 55, 1323–1332 (2004)
19. Parra-Hernández, R., Dimopoulos, N.J.: A new heuristic for solving the multichoice multidimensional knapsack problem. IEEE Trans. on Systems, Man, and Cybernetics—Part A: Systems and Humans 35(5), 708–717 (2005)
20. Akbara, M.M., Rahman, M.S., Kaykobad, M., Manning, E.G., Shoja, G.C.: Solving the multidimensional multiple-choice knapsack problem by constructing convex hulls. Computers & Operations Research 33, 1259–1273 (2006)
21. Khan, M.S.: Quality Adaptation in a Multisession Multimedia System: Model, Algorithms and Architecture. PhD thesis, Dept. of Electrical and Computer Engineering (1998)
22. Sbihi, A.: A best first search exact algorithm for the multiple-choice multidimensional knapsack problem. J. Comb. Optim. 13(4), 337–351 (2007)
23. Chandra, A.K., Hirschberg, D.S., Wong, C.K.: Approximate algorithms for some generalized knapsack problems. Theoretical Computer Science 3(3), 293–304 (1976)
24. Kellerer, H., Pferschy, U., Pisinger, D.: Knapsack Problems. Springer, Heidelberg (2004)
25. Plotkin, S.A., Shmoys, D.B., Tardos, É.: Fast approximation algorithms for fractional packing and covering problems. Math. Oper. Res. 20, 257–301 (1995)
26. Grötschel, M., Lovasz, L., Schrijver, A.: Geometric Algorithms and Combinatorial Optimization. Springer, Heidelberg (1988)