

Vector Bin Packing with Multiple-Choice*

Extended Abstract
(Track A Submission)

Boaz Patt-Shamir^{1**} and Dror Rawitz¹

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

Abstract. We consider the following variant of bin packing. We need to pack a given set of items in the least possible number of bins. Each item can be selected in one of several *incarnations*, where each incarnation has sizes in D dimensions. Bins can also be selected from several bin types, each with its D -dimensional size. The problem has many applications, for example in the context of QoS scheduling. We present an approximation algorithm that is guaranteed to produce a solution whose cost is about $\ln D$ times the optimum, assuming D is constant. This extends previous results, in which each item has a single incarnation and there was only one bin type (*vector bin packing*). 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 the largest number of (incarnations of) items in that bin.

Keywords: Approximation Algorithms, Multiple-Choice Vector Bin Packing, Multiple-Choice Multidimensional Knapsack.

1 Introduction

Fitting items of the maximum possible value in a given space, and arranging a given set of items in the minimal possible space, are two of the fundamental problems of combinatorial optimization, known as packing and covering, respectively, with have many important variants [1]. In the *multidimensional* flavor of packing, each item has sizes in several dimensions, and the space has limited size in each dimension [2]. In this paper we consider a generalization of multidimensional packing, namely *multiple-choice* multidimensional packing. In this variant, each item has a few *incarnations*, each with possibly different sizes in the different dimensions, and similarly, there are a few types of the space containers called bins, in which the (incarnations of) items should be placed, each bin type with its own size cap in each dimension.

* Research supported in part by the Next Generation Video Consortium, Israel.

** Supported in part by the Israel Science Foundation, Grant 664/05.

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 is motivated, for example, by the fact that 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 (that may include processing, storage, and communication hardware) may be configured in more than one way.

More specifically, in this paper we consider the problem of *multiple-choice vector bin packing* (see Section 2 for formal definition). The input to the problem is a set of *items* and a set of *bin types*. Each item is a set of *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 requested 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.

The multiple-choice multidimensional variant is useful in many applications, ranging from scheduling connections with different Quality of Service (QoS) constraints, to planning schedules of nursing personnel in hospitals [3].

Our results. In this paper we show, perhaps surprisingly, that adding the multiple-choice twist to vector bin packing does not affect its complexity dramatically. Specifically, we give a polynomial-time approximation algorithm for the multiple-choice vector bin packing problem (abbreviated MVBP): given any instance and constant $\varepsilon > 0$, the algorithm produces a solution with $(\ln(2D) + 1 + \varepsilon)\text{OPT}$ bins, where OPT denotes the minimal possible number of bins required to pack that instance. To the best of our knowledge, this is the first (asymptotic) 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). Our result extends to the weighted case (where each bin has a weight), with approximation ratio $O(\ln D + T)$, where T is the number of bin types.

As an aside, to facilitate our algorithm we also improve on the best results for multiple-choice multidimensional *knapsack* problem (abbreviated MMK), where we are given a single “bin” and the goal is to load it with the maximal possible number of (incarnations of) items. Specifically, we present a polynomial-time approximation scheme (PTAS) for MMK. The algorithm for MMK is used as a subroutine in our algorithm for MVBP.

Related work. Classical bin packing (BP) admits an asymptotic PTAS [4, 5]. Regarding multidimensionality, it has been long known that vector bin packing (VBP, for short) can be approximated to within a factor of $O(D)$ [6]. More recently, Chekuri and Khanna [8] presented an $O(\log D)$ -approximation algorithm

for VBP, for the case where D is constant. [8] also shows 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 [9], 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.

For the knapsack problem, Frieze and Clarke [16] presented a PTAS for the (single-choice) multidimensional variant, but obtaining an FPTAS for multidimensional knapsack is NP-hard [7]. Shachnai and Tamir [17] use the approach of [16] to obtain a PTAS for a special case of 2-dimensional multiple-choice knapsack. Our algorithm for MMK extends their technique to the general MMK case. Multiple-choice knapsack (MMK) was studied extensively by practitioners. Heuristics for MMK abound, see, e.g., [10–14]. From the algorithmic viewpoint, the first relevant result is by Chandra et al. [15], who present a PTAS for single-dimension, multiple-choice knapsack.

Paper organization. In Section 2 we formalize the problems. In Section 3 we present our solution to the MMK problem, which is used in our solution to the MVBP problem, presented in Section 4.

2 Problem Statements

We now formalize the optimization problems we deal with. For a natural number n , let $[n] \stackrel{\text{def}}{=} \{1, 2, \dots, n\}$ (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 less D -dimensional incarnations. Incarnation j of item i has size $a_{ij} \in (\mathbb{R}^+)^D$, in which the d th 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 d th 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 of (*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 Multidimensional Knapsack

In this section we present a PTAS for weighted MMK. Our construction extends the algorithms of Frieze and Clarke [16] and of Shachnai and Tamir [17].

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

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

In the program, there is a constraint that makes 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 $g : G \rightarrow [m]$ 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 j th incarnation of item i weighs more than some incarnation in our guess, then $x_{ij} = 0$. Denote the resulting linear program $\text{MMK}(G, g)$.

Let $x^*(G, g)$ be an optimal solution to $\text{MMK}(G, g)$. The idea of Algorithm 1 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.

Theorem 1. *Algorithm 1 is a PTAS for MMK.*

Algorithm 1

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

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 trivially 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 1 in which $G = G^I$ and $g = g^I$. Clearly, $w \cdot x^I \leq w \cdot x^*(G^I, g^I)$. Let n' denote the number of items that were not chosen by (G^I, g^I) or were eliminated because their incarnations weigh too much. Let k be the number of variables of the form x_{ij} in $\text{MMK}(G^I, g^I)$. 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 $1 \leq d \leq D$, and each constraint of the second type is written as $\sum_{j=1}^m x_{ij} + s'_i = 1$ for $1 \leq i \leq n'$. Thus, the total number of variables in the program is $k + D + n'$, and since $\text{MMK}(G^I, g^I)$ has $D + n'$ constraints (excluding positivity constraints $x_{ij} \geq 0$), it follows any basic solution of $\text{MMK}(G^I, g^I)$ has at most $D + n'$ positive variables. Since by constraints of the second type in $\text{MMK}(G^I, g^I)$ there is at least one positive variable for each item, it follows that $x^*(G^I, g^I)$ has at most D non-integral entries, and therefore, by rounding down $x^*(G^I, g^I)$ we lose at most D incarnations of items. Let $W^I = \sum_{i \in G^I} w_{ig^I(i)}$. Then each incarnation lost due to rounding weighs at most W^I/q (because it is not one of the q heaviest). We can therefore conclude that

$$\begin{aligned} w \cdot x(G^I, g^I) &\geq w \cdot x^*(G^I, g^I) - D \cdot W^I/q \\ &\geq w \cdot x^*(G^I, g^I)(1 - D/q) \\ &\geq w \cdot x^I(1 - D/q) \\ &= \frac{\text{OPT}}{1 + \varepsilon}, \end{aligned}$$

and we are done. □

4 Multiple-Choice Vector Bin Packing

In this section we present our main result, namely, an $O(\log D)$ -approximation algorithm for MVBP, assuming that D and T (number of dimensions and bin types, respectively) are constants. Our algorithm extends ideas the work of [9].

The general idea is as follows. We first encode MVBP using a covering linear programming formulation 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 3.) 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.

4.1 Covering Formulation

We start with the transformation of MVBP to Set Cover (SC). An instance of SC is a family of sets $\mathcal{C} = \{C_1, C_2, \dots\}$. 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 the fewest possible number of the sets from \mathcal{C} whose union is I . 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. Restricting the set of allowable instances results in special cases we shall call (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 $f : C \rightarrow [m]$ 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. 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).

4.2 Dual Oblivious Algorithms

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

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

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 (family of instances of (P)). 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 any bin which has enough space to accommodate it, possibly opening a new bin if necessary.

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 i th 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 $\delta = 1$. \square

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. Then the following algorithm solves any instance \mathcal{C} of Π :

1. (*linear programming*) Find an optimal solution x^* to (P). Let OPT^* denote its value.
2. (*greedy phase*) Let $\mathcal{C}^+ = \{C : x_C^* > 0\}$. Let $\mathcal{G} \leftarrow \emptyset, S \leftarrow I$.

Repeat while $\sum_{C \in \mathcal{G}} w_C < \ln \rho \cdot \text{OPT}^*$:

Find $C \in \mathcal{C}^+$ for which $\frac{1}{w_C} \sum_{i \in S \setminus C} y_i$ is minimized;

Let $\mathcal{G} \leftarrow \mathcal{G} \cup \{C\}, S \leftarrow S \setminus C$.

3. (*residual solution*) Apply APPR to the residual instance S , obtaining solution \mathcal{A} .
4. Output $\mathcal{G} \cup \mathcal{A}$.

Theorem 2. *Let Π be a (P)-problem. Then for any instance of Π with optimal fractional solution OPT^* , the algorithm above 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

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

proving the theorem.

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

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

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

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

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

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

and we are done. \square

Observe that Theorem 2 holds even if x^* is not an optimal solution of (P), but rather a $(1 + \varepsilon)$ -approximation. We use this fact later.

4.3 Algorithm for Multiple-Choice Vector Bin Packing

We now apply the framework of Theorem 2 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 exists) a subset C with $\sum_{i \in C} y_i > w_C$ for given item profits y_1, \dots, 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 have presented a PTAS in Section 3. 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) [18, 19].

Second, we need to come up with 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 d th dimension of bins of type t by $\ell_{ijtd} = a_{ijd}/b_{td}$. For every item $i \in I$ we define the *effective load* of i as

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

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:

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

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 $d'(i) = \operatorname{argmax}_d \{a_{ij'(i)d}/b_{td}\}$, 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. \square

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

Theorem 3. *There exists a polynomial time algorithm for MVBP with T bin types that computes a solution whose size is at most*

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

This implies the following result for unweighted MVBP:

Corollary 1. *There exists a polynomial time algorithm for unweighted MVBP with T bin types that computes a solution whose size is at most*

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

That is, 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. *There exists a polynomial time $O(T + \ln 2D)$ -approximation algorithm for MVBP.*

Proof. The result follows from the fact that we may assume that $\max_t w_t \leq \text{OPT}$. This assumption is fulfilled by the following wrapper for our algorithm. Guess which is the most expensive bin type used in an optimal solution, and remove bin types that are more expensive. Compute a solution for all possible guesses, and output the best solution. Since our algorithm computes an r -approximate solution for the right guess, then best solution is also r -approximation.

References

1. Papadimitriou, C.H., Steiglitz, K.: Combinatorial optimization : algorithms and complexity. Prentice-Hall (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** (1972) 411–422
4. Fernandez de la Vega, W., Lueker, G.S.: Bin packing can be solved within $1+\varepsilon$ in linear time. Combinatorica **1**(4) (1981) 349–355

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. (1982) 312–320
6. 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) (1976) 257–298
7. Magazine, M.J., Chern, M.S.: A note on approximation schemes for multidimensional knapsack problems. *Mathematics of Operations Research* **9**(2) (1984) 244–247
8. Chekuri, C., Khanna, S.: On multidimensional packing problems. *SIAM Journal on Computing* **33**(4) (2004) 837–851
9. Bansal, N., Caprara, A., Sviridenko, M.: Improved approximation algorithms for multidimensional bin packing problems. In: 47th IEEE Annual Symposium on Foundations of Computer Science. (2006) 697–708
10. Khan, M.S.: Quality Adaptation in a Multisession Multimedia System: Model, Algorithms and Architecture. PhD thesis, Dept. of Electrical and Computer Engineering (1998)
11. Hifi, M., Michrafy, M., Sbihi, A.: Heuristic algorithms for the multiple-choice multidimensional knapsack problem. *Journal of the Operational Research Society* **55** (2004) 1323–1332
12. Parra-Hernández, R., Dimopoulos, N.J.: A new heuristic for solving the multiple-choice multidimensional knapsack problem. *IEEE Trans. on Systems, Man, and Cybernetics—Part A: Systems and Humans* **35**(5) (2005) 708–717
13. Akbara, M.M., Rahmanb, M.S., Kaykobadb, M., Manninga, E., Shojaa, G.: Solving the multidimensional multiple-choice knapsack problem by constructing convex hulls. *Computers & Operations Research* **33** (2006) 1259–1273
14. Sbihi, A.: A best first search exact algorithm for the multiple-choice multidimensional knapsack problem. *Journal of Combinatorial Optimization* **13**(4) (2007) 337–351
15. Chandra, A.K., Hirschberg, D.S., Wong, C.K.: Approximate algorithms for some generalized knapsack problems. *Theoretical Computer Science* **3**(3) (1976) 293–304
16. Frieze, A.M., Clarke, M.R.B.: Approximation algorithms for the m -dimensional 0 – 1 knapsack problem: worst-case and probabilistic analyses. *European Journal of Operational Research* **15** (1984) 100–109
17. Shachnai, H., Tamir, T.: Approximation schemes for generalized 2-dimensional vector packing with application to data placement. In: 6th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems. Number 2764 in LNCS (2003) 129–148
18. Plotkin, S.A., Shmoys, D.B., Tardos, É.: Fast approximation algorithms for fractional packing and covering problems. *Mathematics of Operations Research* **20** (1995) 257–301
19. Grötschel, M., Lovasz, L., Schrijver, A.: *Geometric Algorithms and Combinatorial Optimization*. Springer-Verlag (1988)