European Journal of Operational Research 233 (2014) 43–63

Contents lists available at ScienceDirect

European Journal of Operational Research

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

Discrete Optimization

Multidimensional dual-feasible functions and fast lower bounds for the vector packing problem

Cláudio Alves [a,*], José Valério de Carvalho [a], François Clautiaux [b], Jürgen Rietz [a]

[a] Departamento de Produção e Sistemas, Escola de Engenharia, Universidade do Minho, 4710-057 Braga, Portugal
[b] Université des Sciences et Technologies de Lille, LIFL UMR CNRS 8022, 59655 Villeneuve d'Ascq, France

ARTICLE INFO

Article history:
Received 27 January 2012
Accepted 7 August 2013
Available online 23 August 2013

Keywords:
Dual-feasible functions
Vector packing problem
Fast lower bounds

ABSTRACT

In this paper, we address the 2-dimensional vector packing problem where an optimal layout for a set of items with two independent dimensions has to be found within the boundaries of a rectangle. Many practical applications in areas such as the telecommunications, transportation and production planning lead to this combinatorial problem. Here, we focus on the computation of fast lower bounds using original approaches based on the concept of dual-feasible functions.

Until now, all the dual-feasible functions proposed in the literature were 1-dimensional functions. In this paper, we extend the principles of dual-feasible functions to the m-dimensional case by introducing the concept of vector packing dual-feasible function, and we propose and analyze different new families of functions. All the proposed approaches were tested extensively using benchmark instances described in the literature. Our computational results show that these functions can approximate very efficiently the best known lower bounds for this problem and improve significantly the convergence of branch-and-bound algorithms.

1. Introduction

The m-dimensional vector packing problem (mD-VPP, with $m \in \mathbb{N} \setminus \{0\}$) is a generalization of the well known 1-dimensional bin packing problem (1D-BPP). In the latter, items with different lengths have to be packed into a minimum number of larger objects, or bins. In the mD-VPP, there are m feasibility (capacity) constraints, one for each dimension of the problem, i.e. the sum of the lengths of all packed items must not exceed the bin size in any of the m directions. Unlike in m-dimensional bin packing problems, the dimensions of the mD-VPP are independent. As an example, for $m = 2$, these dimensions may represent the volume and the weight of an object. When $m = 1$, the problem reduces to the 1D-BPP. Hence, the mD-VPP is $\mathcal{NP}$-hard in the strong sense (Garey and Johnson, 1978). In this paper, we focus on the 2D-VPP. However, whenever it is possible, we will generalize our results to the m-dimensional case.

The mD-VPP was described first in Garey et al. (1976) to address a special multiprocessor scheduling problem with resource constraints. In that paper, the authors analyzed the worst-case behavior of a generalization of the well-known first fit heuristic used for the 1D-BPP, and they explored two variants based on a preordering of the items (the first fit decreasing and the level heuristic).

Other approximation schemes and worst-case performance results concerning the mD-VPP were reported in Yao (1980), de la Vega and Lueker (1981), Woeginger (1997), Chekuri and Khanna (1999), and Kellerer and Kotov (2003). In Yao (1980), Yao showed that for any $o(n \log n)$-time algorithm A, there is an instance where A generates a solution that uses at least m times the number of bins of the optimal solution. de la Vega and Lueker (1981) showed that a solution with $m + \varepsilon$ times the optimal number of bins can be found in linear time for the mD-VPP by applying an extension of their algorithm for the 1D-BPP. In Chekuri and Khanna (1999), described an approximation algorithm for the mD-VPP with a better worst-case performance ratio of $1 + \varepsilon m + O(\ln \varepsilon^{-1})$.

For the 2D-VPP, Woeginger (1997) proved that there is no polynomial time approximation scheme for this special case unless $\mathcal{P} = \mathcal{NP}$. Recently, Kellerer and Kotov (2003) proposed an $O(n \log n)$-time algorithm with an asymptotic worst-case performance ratio of 2 for the 2D-VPP. In Chang et al. (2005), described an $O(n^2)$-time algorithm that extends the algorithm of Kellerer and Kotov (2003), and that leads to solutions with no more than $1/(1 - \varrho)$ times the number of bins of the optimal solution plus one, with ϱ representing the largest fraction between the sizes of the items and the sizes of the bins. In Bansal et al. (2006), described polynomial time randomized algorithms for the 2D-VPP with a worst-case ratio of nearly 1.525.

* Corresponding author. Tel.: +351 253604765.
E-mail addresses: claudio@dps.uminho.pt (C. Alves), vc@dps.uminho.pt (J.V. de Carvalho), francois.clautiaux@univ-lille1.fr (F. Clautiaux), juergen_rietz@gmx.de (J. Rietz).

0377-2217/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.ejor.2013.08.011

Spieksma was the first to propose an exact approach for the 2D-VPP in Spieksma (1994). The author described different lower bounding procedures, and developed a branch-and-bound algorithm that relies on these bounds. In this paper, the author described also a heuristic based on the first fit decreasing rule.

The approaches of Spieksma (1994) were later improved by Caprara and Toth in Caprara and Toth (2001). The authors discussed different lower bounding procedures based on bounds for the 1D-BPP and on the linear relaxation of a column generation model. They described algorithms for computing the bounds, they explored the worst-case performance behavior of these bounds, and they proposed heuristics and exact algorithms for the problem. Extensive computational results were reported on instances with up to 200 different items. These results show an improvement compared to previous approaches.

The bounds obtained by column generation are of excellent quality. However, they are obtained with a high computational cost. For the 1-dimensional cutting-stock problem, fast lower bounds can be obtained using the so-called dual-feasible functions (DFFs) (Lueker, 1983). These functions are related to dual solutions of the column generation model, and therefore there is always a function leading to the optimal value yielded by this model.

One of the contributions of this paper is the extension of the concept of DFFs to the m-dimensional case. Several difficulties arise when one wants to generalize the results obtained for 1-dimensional DFFs. They are induced by the fact that the comparison between item sizes have to be done componentwise, and therefore, *monotonicity* and *superadditivity* take a different sense. In this paper, we propose several new families of functions, and we analyze those properties which are relevant to evaluate the quality of the lower bounds that these functions can generate. We describe different lower bounding procedures based on these functions, and we report on the complexity of each one. Our approaches were tested on benchmark instances of the literature. Our computational experiments show that high quality bounds can be obtained for this problem using the vector packing dual-feasible functions proposed in this paper. Furthermore, we tested our lower bounding procedures within a branch-and-bound algorithm to evaluate their potential impact on the convergence of this type of algorithms. We analyzed the behavior of the algorithm with and without our lower bounding schemes. The results obtained show that significant improvements can be achieved with the former approach.

The paper is organized as follows. In Section 2, we introduce the notation that will be used throughout the paper, and we describe two integer programming formulations for the mD-VPP. In Section 3, we briefly recall the basics of dual-feasible functions. In Section 4, we define formally the concept of vector packing dual-feasible functions, and we explore some general properties of these functions. The quality of the lower bounds obtained with vector packing dual-feasible functions is discussed in Section 5. In Section 6, we propose and analyze new classes of vector packing dual-feasible functions. The lower bounding procedures that can be derived from these functions and their corresponding complexity are described in Section 7. Extensive computational experiments are reported and discussed in Section 8. Some final conclusions are drawn in Section 9.

2. The vector packing problem

2.1. Definition and notation

An instance $E := (n; \mathbf{L}; \mathbf{b})$ of the mD-VPP consists in a set $\{1, 2, \dots, n\}$ of n items, whose sizes are given in the matrix $\mathbf{L} = (l_{11}, l_{12}, \dots, l_{1m}; \dots; l_{n1}, l_{n2}, \dots, l_{nm}) \in [0,1]^{n\times m}$ (with $\mathbf{l}_i$ being the ith row-vector of $\mathbf{L}$) and with order demands $\mathbf{b} = (b_1, \dots, b_n)^\top \in (\mathbb{N} \setminus \{0\})^n$. In order to simplify the presentation, we will assume that all the sizes are normalized, such that the bins become m-dimensional unit cubes.

The m D-VPP consists in finding a partition of the set of items into a minimum number of subsets such that the items in each subset fit into a bin, i.e. the sum of the sizes in each dimension does not exceed 1 for any subset. Hence, a pattern $\mathbf{a} \in \mathbb{N}^n$ is feasible, if the following capacity constraints on all the m dimensions hold:

$$\sum_{i=1}^{n} a_i \times \ell_{id} \leqslant 1, \quad d = 1, \dots, m. \tag{1}$$

Let $\mathbf{w} := (1, 1, \dots, 1)^\top \in \mathbb{R}^m$. The capacity constraints (1) can be stated as follows: $\mathbf{L}^\top \mathbf{a} \leqslant \mathbf{w}$.

For given vectors $\mathbf{s}, \mathbf{t} \in \mathbb{R}^m$ the relation signs $\leqslant$, $\geqslant$, $<$, and $>$ will be used if the relation is componentwise true. For example, $\mathbf{s} \leqslant \mathbf{t}$ will stand for $s_i \leqslant t_i$, $i = 1, \dots, m$. Furthermore, an interval $[\mathbf{s}, \mathbf{t}]$ will consist of all $\mathbf{z} \in \mathbb{R}^m$ with $\mathbf{s} \leqslant \mathbf{z} \leqslant \mathbf{t}$.

From this point forward, the null vector will be denoted by $\mathbf{o} := (0, 0, \dots, 0)^\top \in \mathbb{R}^m$.

2.2. Integer programming models

2.2.1. A compact formulation

Let K be an upper bound for the number of bins that are necessary to pack all the items. The mD-VPP can be formulated using an assignment model similar to the Kantorovich model proposed in Kantorovich (1960):

$$\min \sum_{j=1}^{K} y_j \tag{2}$$

$$\text{s.t.} \sum_{j=1}^{K} x_{ij} \geqslant b_i, \ i = 1, \dots, n, \tag{3}$$

$$\sum_{i=1}^{n} l_{id} x_{ij} \leqslant y_j, \ d = 1, \dots, m, \ j = 1, \dots, K, \tag{4}$$

$$y_j \in \{0, 1\}, \ j = 1, \dots, K, \tag{5}$$

$$x_{ij} \in \{0, 1\}, \ i = 1, \dots, n, \ j = 1, \dots, K. \tag{6}$$

The binary variables y_j, $j = 1, \dots, K$, indicate if the bin j is used ($y_j = 1$) or not ($y_j = 0$), while the x_{ij} variables, $i = 1, \dots, n$, $j = 1, \dots, K$, determine the inclusion of an item i in the bin j. The objective function (2) consists in minimizing the number of bins required to pack the items. Constraints (3) ensure that the demands of the items are fulfilled. The capacity constraints are represented by (4). There is a capacity constraint for each bin and dimension of the problem.

This model was presented and discussed also in Caprara and Toth (2001) for $m = 2$. It suffers from the same typical weaknesses of the assignment models for cutting and packing problems: a weak continuous lower bound and symmetry. As shown in Caprara (1998), the lower bound provided by the linear programming (LP) relaxation of (2)–(6), rounded up to the next integer, equals the bound L_C given by Spieksma in Spieksma (1994). For the general m-dimensional problem, this bound is equal to:

$$\max_{d=1,\dots,m} \left\{ \left\lceil \sum_{i=1}^{n} l_{id} \right\rceil \right\}. \tag{7}$$

2.2.2. A column generation model

In Caprara and Toth (2001), the authors described a column generation model for the mD-VPP which is obtained from (2)–(6) by dualizing the demand constraints (3). The resulting model has an exponential number of columns that represent the feasible

patterns satisfying the capacity constraints (1). Let P denote this set of patterns. The model states as follows.

$$\min \sum_{p\in P}\lambda_p \tag{8}$$

$$\text{s.t.} \sum_{p\in P}a_{ip}\lambda_p \geqslant b_i,\ i=1,\dots,n, \tag{9}$$

$$\lambda_p \geqslant 0,\ \text{and integer},\ p\in P. \tag{10}$$

The coefficients a_{ip} represent the number of times an item i is used in pattern p. The integer variables λ_p represent the number of times a pattern p is used. The objective function (8) consists in minimizing the total number of patterns (and hence, bins) used to fulfill the item demands (9).

This reformulation leads to a pricing subproblem that is a multidimensional knapsack problem. Since this problem does not have the integrality property, the lower bound given by the LP relaxation of (8)–(10) will be at least equal to the bound provided by the LP relaxation of (2)–(6). In practice, this bound is quite strong, but its computation is time consuming for any large scale instances as shown in Caprara and Toth (2001).

3. Dual-feasible functions

Dual-feasible functions were first proposed by Johnson in Johnson (1973). These functions can be used to compute lower bounds for different combinatorial optimization problems, and also to derive valid inequalities for integer programming problems (Clautiaux et al., 2010; Fekete and Schepers, 2001). A function $f:[0,1]\to[0,1]$ is a *dual-feasible function (DFF)*, if for any finite set $\{x_i\in\mathbb{R}_+ : i\in J\}$ of nonnegative real numbers, the following holds:

$$\sum_{i\in J}x_i\leqslant 1 \Rightarrow \sum_{i\in J}f(x_i)\leqslant 1.$$

A DFF f is a *maximal dual-feasible function (MDFF)*, if there is no other DFF $g:[0,1]\to[0,1]$ such that $g(x)\geqslant f(x)$, for all $x\in[0,1]$. For $f:[0,1]\to[0,1]$ to be a MDFF, it is necessary and sufficient (Clautiaux et al., 2010; Rietz et al., 2010) that f is symmetric, i.e.

$$f(x)+f(1-x)=1 \quad \text{for all} \quad x\in[0,1/2],$$

$f(0)=0$, and f verifies the following superadditivity condition:

$$f(x_1+x_2)\geqslant f(x_1)+f(x_2) \text{ for all } x_1,x_2 \text{ with } 0<x_1\leqslant x_2<1/2 \text{ and } x_1+x_2\leqslant 2/3.$$

All the DFFs described in the literature are 1-dimensional functions. One of the contributions of this paper is to extend for the first time the concept of DFF to the m-dimensional case. A comprehensive survey and comparative study of the 1-dimensional DFFs proposed in the literature is provided in Clautiaux et al. (2010). This study allowed to identify the following DFFs as some of the best functions proposed so far. From this point forward, we will use the abbreviation $\mathrm{frac}(x):=x-\lfloor x\rfloor$, for any $x\in\mathbb{R}$, to denote the non-integer part of the real expression x.

- $f_{FS,1}$, proposed in Fekete and Schepers (2001), with $k\in\mathbb{N}\setminus\{0\}$:

$$f_{FS,1}(x;k)=\begin{cases} x, & \text{if } (k+1)\times x\in\mathbb{N},\\ \lfloor (k+1)x\rfloor/k, & \text{otherwise,}\end{cases}$$

- $f_{CCM,1}$, proposed in Carlier et al. (2007), for any $C\in\mathbb{R}$ and $C\geqslant 1$:

$$f_{CCM,1}(x;C)=\begin{cases} \lfloor Cx\rfloor/\lfloor C\rfloor, & \text{if } 0\leqslant x<1/2,\\ \frac{1}{2}, & \text{if } x=\frac{1}{2},\\ 1-f_{CCM,1}(1-x), & \text{if } \frac{1}{2}<x\leqslant 1,\end{cases}$$

- $f_{BJ,1}$ proposed in Burdett and Johnson (1977), with $C\in\mathbb{R}$ and $C\geqslant 1$:

$$f_{BJ,1}(x;C)=\frac{1}{\lfloor C\rfloor}\times\left(\lfloor Cx\rfloor+\max\left\{0,\frac{\mathrm{frac}(Cx)-\mathrm{frac}(C)}{1-\mathrm{frac}(C)}\right\}\right).$$

4. Vector packing dual-feasible functions

Dual-feasible functions (DFFs) have been used essentially to compute bounds and inequalities for cutting and packing, routing and scheduling problems (Clautiaux et al., 2010). The functions described in the literature are defined exclusively for 1-dimensional domains. Computing lower bounds for the mD-VPP by separating the problem into m instances of the 1D-BPP, and by applying these functions independently to each one of these instances may lead to arbitrarily bad results. In this paper, we introduce the concept of vector packing dual-feasible functions, where the 1-dimensional domain is now replaced by an m-dimensional one. These functions can be applied directly to mD-VPP instances without separating the m dimensions, and hence, they may lead to much stronger lower bounds for the mD-VPP. Our objective is to approximate the lower bounds provided by the LP relaxation of (8)–(10) through efficient (polynomial-time) lower bounding procedures that rely on these m-dimensional dual-feasible functions.

In this section, we introduce formally the definitions of vector packing dual-feasible function and maximal vector packing dual-feasible function. We show that some general properties for the 1-dimensional case can be generalized to the multidimensional case, and we give a complete characterization of *maximal* functions for the m-dimensional case. Finally, we show how to build such *maximal* functions from non-*maximal* ones by forcing symmetry.

4.1. Definitions

A formal definition of a vector packing dual-feasible function is given next.

Definition 1. A function $f:[0,1]^m\to[0,1]$ is a *vector packing dual-feasible function* (VP-DFF), if for all instances of the mD-VPP and all feasible patterns $\mathbf{a}\in\mathbb{N}^n$ satisfying (1), the following inequality holds:

$$\sum_{i=1}^{n}a_i\times f\left(\mathbf{I}_i^{\top}\right)\leqslant 1.$$

A VP-DFF is said to be *maximal* if there is no other VP-DFF that dominates it in the sense stated in the following definition.

Definition 2. A VP-DFF f is *maximal* (VP-MDFF), if there is no other VP-DFF g with $g(\mathbf{x})\geqslant f(\mathbf{x})$ for all $\mathbf{x}\in[0,1]^m$.

Some simple VP-DFFs are, for example, the projections to the jth coordinate of the argument-vector ($j=1,\dots,m$), i.e. $f_j(\mathbf{x})=x_j$. These functions lead to lower bounds for the mD-VPP which are already known from the 1D-BPP.

4.2. General properties of VP-MDFFs

The necessary conditions from the 1-dimensional case stated in Carlier and Néron (2007) for a function to be maximal are still valid for the higher-dimensional case. However, it remains to be checked how the higher-dimensional case can be described and if stronger sufficient conditions are needed. These ideas are explored in the following theorems.

Theorem 1. *Any VP-MDFF $f:[0,1]^m\to[0,1]$ has necessarily the following properties:*

1. *f is superadditive, i.e. for all $\mathbf{x},\mathbf{y}\in[0,1]^m$ with $\mathbf{x}+\mathbf{y}\leqslant\mathbf{w}$, it holds that*

$$f(\mathbf{x}+\mathbf{y}) \geqslant f(\mathbf{x})+f(\mathbf{y}); \tag{11}$$

2. *f is non-decreasing:*

$$f(\mathbf{x}) \leqslant f(\mathbf{y}), \text{ if } \mathbf{o} \leqslant \mathbf{x} \leqslant \mathbf{y} \leqslant \mathbf{w}; \tag{12}$$

3. *f is symmetric, i.e. for all* $\mathbf{x} \in [0,1]^m$, *it holds that*

$$f(\mathbf{x})+f(\mathbf{w}-\mathbf{x}) = 1, \tag{13}$$

and especially $f(\mathbf{w}) = 1$ *and* $f(\frac{1}{2}\mathbf{w}) = 1/2$.

Proof. To prove (11), we define first the function $g : [0,1]^m \to \mathbb{R}$ as $g(\mathbf{x}) := \sup_{\mathbf{y}\in[\mathbf{o},\mathbf{x}]} \{f(\mathbf{y})+f(\mathbf{x}-\mathbf{y})\}$.

We will show that g is a VP-DFF dominating f, and hence $g \equiv f$. Since

$$g(\mathbf{x}) \geqslant f(\mathbf{o})+f(\mathbf{x}) \text{ and } f(\mathbf{o}) \geqslant 0,$$

one has $g(\mathbf{x}) \geqslant f(\mathbf{x})$ for all $\mathbf{x} \in [0,1]^m$. Let J be any finite index set, and $\mathbf{x}_i \in [0,1]^m$ such that $\sum_{i\in J}\mathbf{x}_i \leqslant \mathbf{w}$. Due to the construction of g, for all $\varepsilon > 0$, there are $\mathbf{y}_i \in [\mathbf{o},\mathbf{x}_i]$ with $i \in J$, such that

$$\sum_{i\in J} g(\mathbf{x}_i) \leqslant \varepsilon + \sum_{i\in J}(f(\mathbf{y}_i)+f(\mathbf{x}_i-\mathbf{y}_i)) \leqslant \varepsilon + 1,$$

because f is a VP-DFF. Since the inequality $\sum_{i\in J} g(\mathbf{x}_i) \leqslant \varepsilon + 1$ holds for all $\varepsilon > 0$, it follows that $\sum_{i\in J} g(\mathbf{x}_i) \leqslant 1$, and hence g is a VP-DFF too. That implies $g \equiv f$, because f is a VP-MDFF. Since $f(\mathbf{x}) \geqslant f(\mathbf{y})+f(\mathbf{x}-\mathbf{y})$ for all $\mathbf{x}, \mathbf{y} \in [0,1]^m$ with $\mathbf{x} \geqslant \mathbf{y}$, the assertion (11) follows by replacing $\mathbf{x}$ by $\mathbf{x}+\mathbf{y}$.

The property (12) follows, because $\mathbf{y} \geqslant \mathbf{x}$ implies $\mathbf{y}-\mathbf{x} \in [0,1]^m$, and since $f(\mathbf{y}) \geqslant f(\mathbf{x})+f(\mathbf{y}-\mathbf{x})$ and $f(\mathbf{y}-\mathbf{x}) \geqslant 0$, we have that $f(\mathbf{y}) \geqslant f(\mathbf{x})$.

Now, we prove the validity of the property (13) by addressing first the case where $\mathbf{x} \in [0,1]^m \setminus \{\frac{1}{2}\mathbf{w}\}$. Let $r \in \{1,\dots,m\}$, with $x_r \neq 1/2$, and let $h:[0,1]^m \to [0,1]$ be the function defined as follows:

$$h(\mathbf{x}) := \begin{cases} f(\mathbf{x}), & \text{if } x_r \leqslant 1/2, \\ 1-f(\mathbf{w}-\mathbf{x}), & \text{otherwise.} \end{cases}$$

We want to show that $h \equiv f$. Clearly, we have $h(\mathbf{x}) \geqslant f(\mathbf{x})$ for all $\mathbf{x} \in [0,1]^m$, because $f(\mathbf{x}) \leqslant 1-f(\mathbf{w}-\mathbf{x})$. Otherwise, f would not be a VP-DFF. To see that h is a VP-DFF too, we choose again a finite index set J and $\mathbf{x}_i \in [0,1]^m$ with $\sum_{i\in J}\mathbf{x}_i \leqslant \mathbf{w}$. Suppose that $1 \in J$ and $x_{1r} > 1/2$. Then, $x_{ir} < 1/2$ for all $i \in J\setminus\{1\}$, and hence

$$\begin{aligned}\sum_{i\in J} h(\mathbf{x}_i) &= 1-f(\mathbf{w}-\mathbf{x}_1)+\sum_{i\in J\setminus\{1\}} f(\mathbf{x}_i) \\ &\leqslant 1-f(\mathbf{w}-\mathbf{x}_1) \\ &\quad +f\left(\sum_{i\in J\setminus\{1\}}\mathbf{x}_i\right), \text{ due to (11) and because } \sum_{i\in J\setminus\{1\}}\mathbf{x}_i \\ &\leqslant \mathbf{w}-\mathbf{x}_1, \leqslant 1, \text{ by the monotonicity (12).}\end{aligned}$$

Since f is maximal, we have $f \equiv h$. If $x_r > 1/2$, then $f(\mathbf{x}) = h(\mathbf{x}) = 1-f(\mathbf{w}-\mathbf{x})$ due to the definition of h. If $x_r < 1/2$, then $f(\mathbf{w}-\mathbf{x}) = h(\mathbf{w}-\mathbf{x}) = 1-f(\mathbf{w}-(\mathbf{w}-\mathbf{x})) = 1-f(\mathbf{x})$. Therefore, Property (13) follows except for $\mathbf{x} = \frac{1}{2}\mathbf{w}$.

Since f is a VP-DFF, it follows that $f(\mathbf{o}) = 0$ and $f(\frac{1}{2}\mathbf{w}) \leqslant 1/2$. Analogously to the previous paragraph, we define the function $h':[0,1]^m \to [0,1]$ as follows:

$$h'(\mathbf{x}) := f(\mathbf{x}), \text{ for all } \mathbf{x} \in [0,1]^m \setminus \left\{\frac{1}{2}\mathbf{w}\right\}, \text{ and } h'\left(\frac{1}{2}\mathbf{w}\right) := 1/2.$$

Similar arguments (with $\mathbf{x}_1 := \frac{1}{2}\mathbf{w}$) show that h' is a VP-DFF, and hence, since f is maximal, it follows that $f \equiv h'$. □

The properties (11)–(13) of Theorem 1 are also sufficient conditions for a function $f:[0,1]^m \to [0,1]$ to be a VP-MDFF. However, it is possible to derive sufficient conditions that are more restricted. These sufficient conditions are stated in Theorem 2. This theorem will help to simplify the proofs of maximality that will be described in the next sections. Before introducing these new sufficient conditions, we describe first in Lemma 1 an additional assertion that will be needed to prove the maximality of a VP-DFF.

Lemma 1. *If a VP-DFF* $f:[0,1]^m \to [0,1]$ *satisfies the symmetry condition (13), then f is a VP-MDFF.*

Proof. Suppose that there is a VP-DFF $g:[0,1]^m \to [0,1]$, with $g(\mathbf{x}) > f(\mathbf{x})$ for a given $\mathbf{x} \in [0,1]^m$. The symmetry of f implies

$$f(\mathbf{w}-\mathbf{x}) = 1-f(\mathbf{x}) > 1-g(\mathbf{x}) \geqslant g(\mathbf{w}-\mathbf{x}),$$

otherwise the contradiction $g(\mathbf{x})+g(\mathbf{w}-\mathbf{x}) > 1$ would arise. Since $f(\mathbf{w}-\mathbf{x}) > g(\mathbf{w}-\mathbf{x})$, the VP-DFF f is not dominated by any another one, and hence it is a VP-MDFF. □

Theorem 2. *Given two constants* $r, s \in \{1,\dots,m\}$ *and a function* $f:[0,1]^m \to [0,1]$, *the following conditions are sufficient for f to be a VP-MDFF:*

1. *Eq. (13) is true for all* $\mathbf{x} \in [0,1]^m$ *with* $x_r \leqslant 1/2$;
2. *Inequality (11) holds for all* $\mathbf{x}, \mathbf{y} \in [0,1]^m$ *with* $\mathbf{x}+\mathbf{y} \leqslant \mathbf{w}$, *and* $x_s \leqslant y_s \leqslant 1/2$ *and* $x_s + y_s \leqslant 2/3$.

Proof. First, we prove that the conditions 1 and 2 imply respectively that (13) and (11) hold for all $\mathbf{x}, \mathbf{y} \in [0,1]^m$ with $\mathbf{x}+\mathbf{y} \leqslant \mathbf{w}$. Then, we will show that f is a VP-DFF and that it is maximal by resorting to Lemma 1.

Suppose that $\mathbf{y} \in [0,1]^m$ with $y_r > 1/2$. Let $\mathbf{x} := \mathbf{w}-\mathbf{y}$. That yields $\mathbf{x} \in [0,1]^m$ with $x_r < 1/2$ such that the condition 1. implies that (13) holds for this $\mathbf{x}$. That gives $1 = f(\mathbf{x})+f(\mathbf{w}-\mathbf{x}) = f(\mathbf{w}-\mathbf{y})+f(\mathbf{y})$, and hence (13) holds also for $\mathbf{y}$ and therefore for all $\mathbf{x} \in [0,1]^m$.

Let $\mathbf{x},\mathbf{y} \in [0,1]^m$ be given, with $\mathbf{x}+\mathbf{y} \leqslant \mathbf{w}$. Now, we show that (11) holds:

- if $x_s > y_s$, then we can exchange $\mathbf{x}$ and $\mathbf{y}$. That does not change the validity of (11), i.e. the inequality (11) holds now if and only if it was true before. Therefore, we may assume $x_s \leqslant y_s$ in the following, and replace the inequality chain $x_s \leqslant y_s \leqslant 1/2$ in condition 2 by $x_s, y_s \leqslant 1/2$ without any influence on the validity of the theorem;
- if $y_s > 1/2$, then (13) implies $f(\mathbf{x}+\mathbf{y}) = 1-f(\mathbf{w}-\mathbf{y}-\mathbf{x})$. Since $x_s + (1-x_s-y_s) < 1/2$ and $0 \leqslant x_s < 1/2$, the condition 2. implies (11) for the arguments $\mathbf{x}$ and $\mathbf{w}-\mathbf{x}-\mathbf{y}$, because if $x_s > 1-x_s-y_s$ then the same exchange can be done as in the previous paragraph. That yields $f(\mathbf{w}-\mathbf{y}-\mathbf{x})+f(\mathbf{x}) \leqslant f(\mathbf{w}-\mathbf{y})$. Hence, we have

  $$f(\mathbf{x}+\mathbf{y}) \geqslant 1-(f(\mathbf{w}-\mathbf{y})-f(\mathbf{x})) = 1-1+f(\mathbf{y})+f(\mathbf{x}),$$

  by the symmetry condition (13). Since for $y_s > 1/2$, the superadditivity condition (11) holds, we may assume that $y_s \leqslant 1/2$ in the following;
- if $x_s \leqslant y_s \leqslant 1/2$ and $x_s + y_s > 2/3$, then $1-x_s-y_s < 1/3$ and $1-y_s < 2/3$ (because of $x_s \leqslant y_s$ and $x_s+y_s > 2/3$). Hence, the condition 2 allows to apply the inequality (11) with the arguments $\mathbf{x}$ and $\mathbf{w}-\mathbf{x}-\mathbf{y}$, leading to $f(\mathbf{w}-\mathbf{y}-\mathbf{x})+f(\mathbf{x}) \leqslant f(\mathbf{w}-\mathbf{y})$, and, by symmetry, we have

  $$\begin{aligned} f(\mathbf{x}+\mathbf{y}) &= 1-f(\mathbf{w}-\mathbf{y}-\mathbf{x}) \geqslant 1-(f(\mathbf{w}-\mathbf{y})-f(\mathbf{x})) \\ &= f(\mathbf{x})+f(\mathbf{y}). \end{aligned}$$

Hence, the conditions 1 and 2 imply the superadditivity (11) for all $\mathbf{x}, \mathbf{y} \in [0,1]^m$ with $\mathbf{x}+\mathbf{y} \leqslant \mathbf{w}$.

The superadditivity and the range imply the monotonicity (12). For any $\mathbf{x}, \mathbf{y} \in [0,1]^m$ with $\mathbf{x} \leqslant \mathbf{y}$, one has $f(\mathbf{y}) \geqslant f(\mathbf{x}) + f(\mathbf{y}-\mathbf{x}) \geqslant f(\mathbf{x})$. Additionally, one gets $f(2\mathbf{o}) \geqslant 2f(\mathbf{o})$. Hence $f(\mathbf{o}) \leqslant 0$ and therefore $f(\mathbf{o}) = 0$ and $f(\mathbf{w}) = 1$ due to the symmetry.

These results are used to show that f is a VP-DFF. For any finite index set J and vectors $\mathbf{x}_i \in [0,1]^m$ with $\sum_{i\in J}\mathbf{x}_i \leqslant \mathbf{w}$, it follows from the superadditivity condition (11) by induction that $\sum_{i\in J} f(\mathbf{x}_i) \leqslant f\left(\sum_{i\in J}\mathbf{x}_i\right) \leqslant f(\mathbf{w}) = 1$. Hence, f is a VP-DFF. Furthermore, the function f is maximal because of the symmetry condition (13) and Lemma 1. □

In the following propositions, we show that the functions resulting from the convex combination of VP-MDFFs or from the composition of a VP-MDFF with a MDFF remain maximal.

Proposition 1. *Any convex combination of VP-MDFFs is a VP-MDFF.*

Proof. Let be given n' VP-MDFFs f_i, $i = 1, \ldots, n'$, and a set of n' positive real numbers λ_i, $i = 1, \ldots, n'$, such that $\sum_{i=1}^{n'} \lambda_i = 1$. We will denote by f the function obtained through the convex combination of the VP-MDFFs f_i with coefficients λ_i, i.e. $f := \sum_{i=1}^{n'} \lambda_i \times f_i$. For any finite set J and vectors $\mathbf{x}_i \in [0,1]^m$ with $\sum_{i\in J}\mathbf{x}_i \leqslant \mathbf{w}$, we have

$$\begin{aligned}\sum_{i\in J} f(\mathbf{x}_i) &= \sum_{i\in J}\sum_{j=1}^{n'} \lambda_j \times f_j(\mathbf{x}_i)\\ &= \sum_{j=1}^{n'} \lambda_j \times \sum_{i\in J} f_j(\mathbf{x}_i)\\ &\leqslant \sum_{j=1}^{n'} \lambda_j \times 1 = 1.\end{aligned}$$

Hence, f is a VP-DFF according to Definition 1.

Now, we will prove that f is symmetric in the sense of (13). That will imply that f is maximal due to Lemma 1. By definition, all the functions f_i, $i = 1, \ldots, n'$ are maximal, and hence they must be symmetric according to Theorem 1. As a consequence, one gets for any $\mathbf{x} \in [0,1]^m$ that

$$\begin{aligned}f(\mathbf{x}) + f(\mathbf{w}-\mathbf{x}) &= \sum_{i=1}^{n'} \lambda_i \times f_i(\mathbf{x}) + \sum_{i=1}^{n'} \lambda_i \times f_i(\mathbf{w}-\mathbf{x})\\ &= \sum_{i=1}^{n'} \lambda_i \times (f_i(\mathbf{x}) + f_i(\mathbf{w}-\mathbf{x}))\\ &= \sum_{i=1}^{n'} \lambda_i = 1,\end{aligned}$$

and hence the symmetry condition holds also for the function f. Therefore, f is maximal. □

Proposition 2. *The composition of a VP-MDFF f with a MDFF g, i.e. $g(f(\cdot))$, is a VP-MDFF.*

Proof. Let g: $[0,1] \to [0,1]$ be a maximal DFF, and let f: $[0,1]^m \to [0,1]$ be a VP-MDFF. For any finite set J and vectors $\mathbf{x}_i \in [0,1]^m$ such that $\sum_{i\in J}\mathbf{x}_i \leqslant \mathbf{w}$, we have

$$\sum_{i\in J} g(f(\mathbf{x}_i)) \leqslant 1,$$

since $\sum_{i\in J} f(\mathbf{x}_i) \leqslant 1$. Hence, $g(f(\cdot))$ is a VP-DFF according to Definition 1.

By definition, the functions f and g are maximal, and hence they are both symmetric according to Theorem 1. For every $\mathbf{x} \in [0,1]^m$, it follows that

$$g(f(\mathbf{x})) + g(f(\mathbf{w}-\mathbf{x})) = g(f(\mathbf{x})) + g(1-f(\mathbf{x})) = 1,$$

such that $g(f(\cdot))$ is symmetric too and therefore maximal according to Lemma 1. □

In Nemhauser and Wolsey (1998), Nemhauser and Wolsey described valid inequalities for independence systems obtained from superadditive functions. A set $S \in \mathbb{Z}_+^n$ is an *independence system*, if $(0,\ldots,0)^\top \in S$, and $\mathbf{y} \leqslant \mathbf{x} \Rightarrow \mathbf{y} \in S$ holds, for $\mathbf{x} \in S$ and $\mathbf{y} \in \mathbb{Z}_+^n$. An additional requirement on $S := \{\mathbf{x} \in \mathbb{Z}_+^n : \mathbf{A}\mathbf{x} \leqslant \mathbf{b}\}$ is that all the coefficients should be nonnegative integers, and $b_i \geqslant \max_j a_{ij}$, for all i and $a_{ij} \in \mathbf{A}$. Hence, one obtains an equivalent inequality by division by b_i, such that a system of the kind $\mathbf{L}^\top\mathbf{x} \leqslant \mathbf{w}$, with $\mathbf{x} \in \mathbb{Z}_+^n$ and $\mathbf{L} \in ([0,1] \cap \mathbb{Q})^{n\times m}$, arises like in our vector packing problem. On the contrary, we did not require integrality constraints or that the data should be rational.

Analogously to Proposition 2 of Clautiaux et al. (2010), the definition of VP-DFFs immediately yields valid inequalties for the system $\mathbf{L}^\top\mathbf{x} \leqslant \mathbf{w}$, with $0 \leqslant \ell_{ij} \leqslant 1$ and $x_j \in \mathbb{Z}_+^n$, for $i = 1, \ldots, m$ and $j = 1, \ldots, n$, namely $\sum_{j=1}^n f(\mathbf{l}^j)x_j \leqslant 1$, where $\mathbf{l}^j$ is the jth column of the matrix $\mathbf{L}$. The proof is straightforward, because $\sum_{j=1}^n f(\mathbf{l}^j)x_j = \sum_{j=1}^n \sum_{k=1}^{x_j} f(\mathbf{l}^j) \leqslant 1$, since $\sum_{j=1}^n \sum_{k=1}^{x_j} \mathbf{l}^j \leqslant \mathbf{w}$.

The similarities between Nemhauser and Wolsey and our superadditive, non-decreasing functions are the following. The domain $D(\mathbf{b}) := \{\mathbf{d} \in \mathbb{Z}_+^m : \mathbf{d} \leqslant \mathbf{b}\}$ is replaced by $[0,1]^m$, and the function F is normalized to $F(\mathbf{w}) = 1$ and the range $[0,1]$. Then F has to be non-decreasing and superadditive. F is maximal if and only if it is symmetric in the sense (13).

4.3. Creating VP-MDFFs by forcing symmetry

In this section, we show how a VP-MDFF can be built from a superadditive m-dimensional superadditive vector function by forcing symmetry. This result is a generalization of a scheme developed for the 1-dimensional case and described in Clautiaux et al. (2010).

Proposition 3. *Let f:$[0,1]^m \to [0,1]$ be a superadditive function, and M be any subset of $[0,1]^m \setminus \{\frac{1}{2}\mathbf{w}\}$ such that:*

1. *for all $\mathbf{x} \in [0,1]^m \setminus \{\frac{1}{2}\mathbf{w}\}$, the following equivalence holds:*
$$\mathbf{x} \in M \iff \mathbf{w}-\mathbf{x} \notin M; \tag{14}$$
2. *for any $\mathbf{x}, \mathbf{y} \in M$, it holds that*
$$\mathbf{x}+\mathbf{y} \not\leqslant \mathbf{w}. \tag{15}$$

The following function g:$[0,1]^m \to [0,1]$ which is built from f is a VP-MDFF:

$$g(\mathbf{x}) := \begin{cases} 1/2, & \text{if } 2\mathbf{x} = \mathbf{w},\\ 1-f(\mathbf{w}-\mathbf{x}), & \text{if } \mathbf{x} \in M,\\ f(\mathbf{x}), & \text{otherwise.}\end{cases}$$

Proof. For this proof, we resort to Theorem 2. That requires to verify that g is both symmetric and superadditive.

First, we show that the range of g belongs to $[0,1]$. Because of the superadditivity and the range of the function f, this function f is also non-decreasing, and hence, we have that

$$f(\mathbf{w}) \leqslant 1 \text{ and } f\left(\frac{1}{2}\mathbf{w}\right) \leqslant \frac{1}{2}.$$

Therefore, the range of g belongs to $[0,1]$ too.

Let $A := [0,1]^m \setminus (M \cup \{\frac{1}{2}\mathbf{w}\})$. Because of (14), one gets for any $\mathbf{x} \in M$ that

$$g(\mathbf{x}) + g(\mathbf{w}-\mathbf{x}) = 1 - f(\mathbf{w}-\mathbf{x}) + f(\mathbf{w}-\mathbf{x}) = 1,$$

and, analogously, for any $\mathbf{x} \in A$ that

$$g(\mathbf{x}) + g(\mathbf{w}-\mathbf{x}) = f(\mathbf{x}) + 1 - f(\mathbf{w}-(\mathbf{w}-\mathbf{x})) = 1.$$

Furthermore, we have also $g(\frac{1}{2}\mathbf{w}) = 1/2$, and hence, the function g is symmetric.

It remains to be proved that g is also superadditive. For that purpose, let $\mathbf{x}, \mathbf{y} \in [0,1]^m$, with $\mathbf{x}+\mathbf{y} \leqslant \mathbf{w}$. Because of (15), one has $\mathbf{o} \in A$, and hence $\mathbf{w} \in M$. The superadditivity of f implies $f(2\mathbf{o}) \geqslant 2f(\mathbf{o})$ and $f(\mathbf{o}) \leqslant 0$, and hence we have that $f(\mathbf{o}) = 0$ due to the range of f.

To show that g is superadditive, we have to resort to the following case distinction that covers all the possibilities concerning which case of the definition of g has to be used, i.e. to which of the sets A, $\{\frac{1}{2}\mathbf{w}\}$ or M each of the arguments $\mathbf{x}$, $\mathbf{y}$ and $\mathbf{x}+\mathbf{y}$ belongs.

1. If $\mathbf{x}$, $\mathbf{y}$, $\mathbf{x}+\mathbf{y} \in A$, then $g(\mathbf{x}+\mathbf{y}) - g(\mathbf{x}) - g(\mathbf{y}) = f(\mathbf{x}+\mathbf{y}) - f(\mathbf{x}) - f(\mathbf{y}) \geqslant 0$ due to the superadditivity of f.
2. If $\mathbf{x}, \mathbf{y} \in A$ and $\mathbf{x}+\mathbf{y} = \frac{1}{2}\mathbf{w}$, then $g(\mathbf{x}) + g(\mathbf{y}) = f(\mathbf{x}) + f(\mathbf{y}) \leqslant f(\mathbf{x}+\mathbf{y}) \leqslant \frac{1}{2} = g(\mathbf{x}+\mathbf{y})$.
3. If $\mathbf{x}, \mathbf{y} \in A$ and $\mathbf{x}+\mathbf{y} \in M$, then $g(\mathbf{x}+\mathbf{y}) - g(\mathbf{x}) - g(\mathbf{y}) = 1 - f(\mathbf{w}-\mathbf{x}-\mathbf{y}) - f(\mathbf{x}) - f(\mathbf{y}) \geqslant 1 - f(\mathbf{w}) \geqslant 0$.
4. It is impossible that $\mathbf{y} = \frac{1}{2}\mathbf{w}$ and $\mathbf{x}+\mathbf{y} \in A$, because in that case, we would have $\mathbf{w}-\mathbf{x}-\mathbf{y} \in M$ and $\mathbf{w}-\mathbf{x}-\mathbf{y} \leqslant \frac{1}{2}\mathbf{w}$ in contradiction to (15).
5. If $\mathbf{y} = \mathbf{x}+\mathbf{y}$, then $\mathbf{x} = \mathbf{o}$, implying $g(\mathbf{x}) = 0$, because $f(\mathbf{o}) = 0$ and $\mathbf{o} \in A$. Therefore, the superadditivity of g becomes obvious.
6. If $\mathbf{x} \in A$, $\mathbf{y} = \frac{1}{2}\mathbf{w}$ and $\mathbf{x}+\mathbf{y} \in M$, then $g(\mathbf{x}+\mathbf{y}) - g(\mathbf{x}) - g(\mathbf{y}) = \frac{1}{2} - f(\mathbf{w}-\mathbf{x}-\mathbf{y}) - f(\mathbf{x}) \geqslant \frac{1}{2} - f(\mathbf{w}-\mathbf{y}) \geqslant 0$.
7. It cannot occur that $\mathbf{y} \in M$ and $\mathbf{x}+\mathbf{y} \in A$, because $\mathbf{w}-\mathbf{x}-\mathbf{y} \in M$ and $\mathbf{w}-\mathbf{x}-\mathbf{y}+\mathbf{y} = \mathbf{w}-\mathbf{x}$ contradicts (15).
8. It can also not happen that $\mathbf{y} \in M$ and $\mathbf{x}+\mathbf{y} = \frac{1}{2}\mathbf{w}$, because of the contradiction $2\mathbf{y} \leqslant \mathbf{w}$.
9. If $\mathbf{x} \in A$ and $\mathbf{y}, \mathbf{x}+\mathbf{y} \in M$, then $g(\mathbf{x}+\mathbf{y}) - g(\mathbf{x}) - g(\mathbf{y}) = 1 - f(\mathbf{w}-\mathbf{x}-\mathbf{y}) - f(\mathbf{x}) - 1 + f(\mathbf{w}-\mathbf{y}) = f(\mathbf{w}-\mathbf{y}) - f(\mathbf{w}-\mathbf{y}-\mathbf{x}) - f(\mathbf{x}) \geqslant 0$.
10. If $\mathbf{x} = \mathbf{y} = \frac{1}{2}\mathbf{w}$, then $g(\mathbf{x}+\mathbf{y}) = g(\mathbf{w}) = 1 - f(\mathbf{o}) = 1 = g(\mathbf{x}) + g(\mathbf{y})$, because $\mathbf{w} \in M$ and $f(\mathbf{o}) = 0$.
11. It cannot happen that $\mathbf{x} = \frac{1}{2}\mathbf{w}$ and $\mathbf{y} \in M$, because that would imply $\mathbf{y} = \mathbf{x}+\mathbf{y}-\mathbf{x} \leqslant \frac{1}{2}\mathbf{w}$ in contradiction to (15).
12. Because of (15), it is also impossible that $\mathbf{x}, \mathbf{y} \in M$.

All the remaining cases are obtained by exchanging $\mathbf{x}$ and $\mathbf{y}$, and hence, they are similar to the previous cases. Hence, all conditions of Theorem 2 are satisfied, such that g is a VP-MDFF. □

The set M in Proposition 3 can be chosen in various ways. For instance, we may have

$$M := [0,1] \times [0,1] \times \ldots \times [0,1] \times \left(\frac{1}{2}, 1\right] \cup [0,1] \times \ldots \times [0,1] \times \left(\frac{1}{2}, 1\right] \times \left\{\frac{1}{2}\right\} \cup \ldots \cup \left(\frac{1}{2}, 1\right] \times \left\{\frac{1}{2}\right\} \times \ldots \times \left\{\frac{1}{2}\right\}$$

(as the union of m parts). For $m = 2$, this set becomes $[0,1] \times (\frac{1}{2}, 1] \cup (\frac{1}{2}, 1] \times \{\frac{1}{2}\}$, i.e. the upper half of the unit square, where only a part of the border belongs to M. Additionally, M could be chosen for example as follows

$$M := \{\mathbf{x} \in [0,1]^2 : x_1 + x_2 > 1\} \cup \left\{\mathbf{x} \in \left(\frac{1}{2}, 1\right] \times [0,1] : x_1 + x_2 = 1\right\}.$$

5. Lower bounds based on VP-DFFs

Let $E := (n; \mathbf{l}; \mathbf{b})$ be an instance of the mD-VPP, and f a VP-DFF. A valid lower bound for the number of bins that are necessary to pack the items of E can be computed from f as follows:

$$z[f] := \sum_{i=1}^{n} b_i \times f(\mathbf{l}_i). \tag{16}$$

Let z_{CG} denote the optimal value of the LP relaxation of the column generation model (8)–(10). The following proposition shows the relation between z_{CG} and the value given by any VP-DFF.

Proposition 4. *For any VP-DFF f: $[0,1]^m \to [0,1]$, the bound $z[f]$ is never above z_{CG}. Moreover, there is at least one VP-DFF $\hat{f}$ with $z[\hat{f}] = z_{CG}$.*

Proof. Even if the number of feasible patterns can be huge, it remains finite. Therefore, the theory about linear optimization problems, especially the duality theory, can be applied. Without loss of generality assume that

$$\mathbf{l}_i \neq \mathbf{l}_j, \quad \text{for all } i,j \in \{1,\ldots,n\} \text{ with } i \neq j. \tag{17}$$

The dual of the LP relaxation of (8)–(10) states as follows:

$$\max\ \mathbf{b}^\top \pi \tag{18}$$

$$\text{s.t.} \sum_{i=1}^{n} a_{ip}\pi_i \leqslant 1,\ p \in P, \tag{19}$$

$$\pi_i \geqslant 0,\ i = 1,\ldots,n. \tag{20}$$

Because $\mathbf{l}_i \in [0,1]^m$, every item fits alone into one bin. Therefore, there is a feasible solution to (8)–(10). Because of the non-negativity constraints on the λ_p variables of (8)–(10), the objective function (8) is bounded in the direction of the optimization by e.g. zero. Hence, optimal solutions of the LP relaxation of (8)–(10) and the dual problem (18)–(20) exist, and the optimal objective function values of both problems are the same (by the strong duality theorem).

For every feasible pattern $\mathbf{a}$, it holds that $\sum_{i=1}^{n} a_i * f(\mathbf{l}_i) \leqslant 1$ by definition of VP-DFFs. Therefore, $\pi_i := f(\mathbf{l}_i)$ for $i := 1,\ldots,n$, fulfills the demand (19). Condition (20) is clearly met due to the range of f. Hence, the chosen π is feasible for the problem (18)–(20), such that $z[f] \leqslant z_{CG}$ follows.

Let $\hat{\boldsymbol{\pi}}$ be an optimal solution of the problem (18)–(20). A VP-DFF $\hat{f} : [0,1]^m \to [0,1]$ can be defined by

$$\hat{f}(\mathbf{x}) := \begin{cases} \hat{\pi}_i, & \text{if } \mathbf{x} = \mathbf{l}_i, i \in \{1,\ldots,n\}, \\ 0, & \text{if } \mathbf{x} \neq \mathbf{l}_i, \text{for all } i \in \{1,\ldots,n\} \end{cases}$$

because of the constraints (19) and (20), even if $\hat{f}$ is in general not maximal. This function $\hat{f}$ is well defined due to the assumption (17) and yields $z[\hat{f}] = z_{CG}$ according to the strong duality theorem. □

6. New classes of VP-MDFFs

In this section, we propose new classes of VP-MDFFs. When general schemes for generating VP-MDFFs are proposed, we describe and analyze some specific functions that can be obtained from these schemes. To simplify the notation, the parameters of the functions will be omitted whenever it is possible.

6.1. Class I: functions based on projections into 1-dimensional domains

Our first set of VP-MDFFs is based on the projection of the m-dimensional data into 1-dimensional domains. The following proposition gives a formal definition of these VP-MDFFs.

Proposition 5. *Let g:$[0,1] \to [0,1]$ be a MDFF and $\mathbf{u} \in \mathbb{R}_+^m$ with $\mathbf{u}^\top\mathbf{w} = 1$. The function f:$[0,1]^m \to [0,1]$ with*

$$f(\mathbf{x}) := g(\mathbf{u}^\top\mathbf{x})$$

is a VP-MDFF.

Proof. Each of the projections $g_j:[0,1]^m \to [0,1]$ with $g_j(\mathbf{x}) = x_j$ is a VP-MDFF. To prove that, we resort to Theorem 2. First, the domain and range of these projections are in accordance with Theorem 2. The symmetry is given due to

$$g_j(\mathbf{x}) + g_j(\mathbf{w}-\mathbf{x}) = x_j + (1-x_j) = 1,$$

for any $\mathbf{x} \in [0,1]^m$. The superadditivity also holds since

$$g_j(\mathbf{x}+\mathbf{y}) = x_j + y_j = g_j(\mathbf{x}) + g_j(\mathbf{y}),$$

for all $\mathbf{x}, \mathbf{y} \in [0,1]^m$ with $\mathbf{x}+\mathbf{y} \leqslant \mathbf{w}$.

According to Proposition 1, any convex combination of VP-MDFFs is a VP-MDFF, and therefore $\mathbf{x} \mapsto \mathbf{u}^\top \mathbf{x}$ is also a VP-MDFF. Furthermore, as shown in Proposition 2, the composition of a VP-MDFF and a MDFF yields again a VP-MDFF, and hence f is a VP-MDFF. □

The function described in Corollary 1 is obtained from Proposition 5 using the MDFF $f_{FS,1}$ proposed in Fekete and Schepers (2001) as the function g.

Corollary 1. *Let $\mathbf{v} \in \mathbb{R}^m_+$ such that $\mathbf{v}^\top \mathbf{w} \in \mathbb{N} \setminus \{0,1\}$. The following function $f : [0,1]^m \to \mathbb{R}_+$ is a VP-MDFF:*

$$f(\mathbf{x}) := \begin{cases} \frac{\mathbf{v}^\top \mathbf{x}}{\mathbf{v}^\top \mathbf{w}}, & \text{if } \mathbf{v}^\top \mathbf{x} \in \mathbb{N}, \\ \frac{\lfloor \mathbf{v}^\top \mathbf{x} \rfloor}{\mathbf{v}^\top \mathbf{w}-1}, & \text{otherwise.} \end{cases}$$

Proof. Let $\mathbf{u} := \frac{1}{\mathbf{v}^\top \mathbf{w}} \times \mathbf{v}$ (and hence, $\mathbf{u} \in \mathbb{R}^m_+$ with $\mathbf{u}^\top \mathbf{w} = 1$), and let $k := \mathbf{v}^\top \mathbf{w} - 1$. If we set $g := f_{FS,1}$ in Proposition 5 and apply the definition of the function $f_{FS,1}$, we get

$$\begin{aligned} f_{FS,1}(\mathbf{u}^\top \mathbf{x}; k) &= \begin{cases} \mathbf{u}^\top \mathbf{x}, & \text{if} (k+1) \times \mathbf{u}^\top \mathbf{x} \in \mathbb{N}, \\ \lfloor (k+1) \times \mathbf{u}^\top \mathbf{x} \rfloor / k, & \text{otherwise,} \end{cases} \\ &= \begin{cases} \frac{\mathbf{v}^\top \mathbf{x}}{\mathbf{v}^\top \mathbf{w}}, & \text{if } \mathbf{v}^\top \mathbf{w} \times \frac{\mathbf{v}^\top \mathbf{x}}{\mathbf{v}^\top \mathbf{w}} \in \mathbb{N}, \\ \lfloor \mathbf{v}^\top \mathbf{w} \times \frac{\mathbf{v}^\top \mathbf{x}}{\mathbf{v}^\top \mathbf{w}} \rfloor / (\mathbf{v}^\top \mathbf{w} - 1), & \text{otherwise,} \end{cases} \end{aligned}$$

which leads to the definition of the function $f(\mathbf{x})$ of the corollary. □

The next function is obtained from Proposition 5 and the MDFF $f_{BJ,1}$ described in Burdett and Johnson (1977).

Corollary 2. *Let $\mathbf{v} \in \mathbb{R}^m_+$ be any vector with $\mathbf{v}^\top \mathbf{w} \geqslant 1$. Then, the function $f : [0,1]^m \to \mathbb{R}_+$ with*

$$f(\mathbf{x}) := \left(\lfloor \mathbf{v}^\top \mathbf{x} \rfloor + \max\left\{0, \frac{\mathrm{frac}(\mathbf{v}^\top \mathbf{x}) - \mathrm{frac}(\mathbf{v}^\top \mathbf{w})}{1 - \mathrm{frac}(\mathbf{v}^\top \mathbf{w})}\right\} \right) \Big/ \lfloor \mathbf{v}^\top \mathbf{w} \rfloor$$

is a VP-MDFF.

Proof. Let $\mathbf{u} := \frac{1}{\mathbf{v}^\top \mathbf{w}} \times \mathbf{v}$ (and hence, $\mathbf{u} \in \mathbb{R}^m_+$ with $\mathbf{u}^\top \mathbf{w} = 1$), and let $C := \mathbf{v}^\top \mathbf{w}$. If we set $g := f_{BJ,1}$ in Proposition 5 and apply the definition of the function $f_{BJ,1}$, we obtain

$$\begin{aligned} f_{BJ,1}(\mathbf{u}^\top \mathbf{x}; C) &= \left(\lfloor C \times \mathbf{u}^\top \mathbf{x} \rfloor + \max\left\{0, \frac{\mathrm{frac}(C \times \mathbf{u}^\top \mathbf{x}) - \mathrm{frac}(C)}{1 - \mathrm{frac}(C)}\right\} \right) \Big/ \lfloor C \rfloor \\ &= \left(\lfloor \mathbf{v}^\top \mathbf{w} \times \mathbf{v}^\top \mathbf{x} / \mathbf{v}^\top \mathbf{w} \rfloor + \max\left\{0, \frac{\mathrm{frac}(\mathbf{v}^\top \mathbf{w} \times \mathbf{v}^\top \mathbf{x} / \mathbf{v}^\top \mathbf{w}) - \mathrm{frac}(\mathbf{v}^\top \mathbf{w})}{1 - \mathrm{frac}(\mathbf{v}^\top \mathbf{w})}\right\} \right) \Big/ \lfloor \mathbf{v}^\top \mathbf{w} \rfloor. \end{aligned}$$ □

Note that if $\mathbf{v}^\top \mathbf{w} \in \mathbb{N}$, then the function f of Corollary 2 is only a convex combination of the projections $f_1, \ldots, f_m$.

6.2. Class II

Some of the ideas of the 1-dimensional MDFFs discussed in Clautiaux et al. (2010) can be adapted for the m-dimensional VPP. For instance, the function due to Martello and Toth, which maps small items to zero and large ones to 1, while the other items remain unchanged, can be generalized as follows. Note that the difficulty in this generalization is to find a suitable definition of *small* and *large* items when vectors are involved.

Proposition 6. *Let $h : [0,1]^m \to \mathbb{R}$ be non-decreasing with $h(\mathbf{x}) + h(\mathbf{w}-\mathbf{x}) > 0$ for all $\mathbf{x} \in [0,1]^m$, and let $g:[0,1]^m \to [0,1]$ be a VP-MDFF. The following functions $f_1, f_2:[0,1]^m \to [0,1]$ are VP-MDFFs:*

$$f_1(\mathbf{x}) := \begin{cases} 0, & \text{if } h(\mathbf{x}) \leqslant 0 \\ 1, & \text{if } h(\mathbf{w}-\mathbf{x}) \leqslant 0, \\ g(\mathbf{x}), & \text{otherwise} \end{cases} \quad f_2(\mathbf{x}) := \begin{cases} 0, & \text{if } h(\mathbf{x}) < 0 \\ 1, & \text{if } h(\mathbf{w}-\mathbf{x}) < 0. \\ g(\mathbf{x}), & \text{otherwise} \end{cases}$$

Proof. For this proof, we resort to Lemma 1. We will show that f_1 is a VP-DFF, i.e. that for any $\mathbf{x},\mathbf{y} \in [0,1]^m$ with $\mathbf{x}+\mathbf{y} \leqslant \mathbf{w}$, it follows that $f_1(\mathbf{x}+\mathbf{y}) \leqslant 1$. Then, the symmetry will be verified.

To start the proof that f_1 is a VP-DFF, several situations with respect to $\mathbf{x}$, $\mathbf{y} \in [0,1]^m$ have to be analyzed. First, observe that $\mathbf{x}+\mathbf{y} \leqslant \mathbf{w}$ implies $\mathbf{x} \leqslant \mathbf{w}-\mathbf{y}$ and $h(\mathbf{x}) \leqslant h(\mathbf{w}-\mathbf{y})$, due to the monotonicity of h. We distinguish the following cases:

1. If $h(\mathbf{w}-\mathbf{y}) \leqslant 0$, then $f_1(\mathbf{x}) = 0$. Since $f_1(\mathbf{y}) = 1$, one has $f_1(\mathbf{x}) + f_1(\mathbf{y}) = 1$;
2. The case $h(\mathbf{w}-\mathbf{x}) \leqslant 0$ is similar to the previous one;
3. Assume that $h(\mathbf{w}-\mathbf{x}) > 0$ and $h(\mathbf{w}-\mathbf{y}) > 0$. Then, $f_1(\mathbf{x}) + f_1(\mathbf{y}) \leqslant g(\mathbf{x}) + g(\mathbf{y}) \leqslant 1$, because g is a VP-DFF.

In all cases, it follows that $f_1(\mathbf{x}) + f_1(\mathbf{y}) \leqslant 1$. Using induction, that implies $\sum_{i \in J} f_1(\mathbf{x}_i) \leqslant 1$, for any finite index set J of vectors $\mathbf{x}_i \in [0,1]^m$ with $\sum_{i \in J} \mathbf{x}_i \leqslant \mathbf{w}$. Hence, f_1 is a VP-DFF.

To prove that f_1 is maximal, according to Lemma 1, it remains to show that f_1 is symmetric. For this purpose, recall that g is by definition a VP-MDFF. According to Theorem 1, g must be symmetric. To prove that f_1 is symmetric, we distinguish the following cases with regard to $\mathbf{x} \in [0,1]^m$:

1. if $h(\mathbf{x}) > 0$ and $h(\mathbf{w}-\mathbf{x}) > 0$, then $f_1(\mathbf{x}) = g(\mathbf{x})$ and $f_1(\mathbf{w}-\mathbf{x}) = g(\mathbf{w}-\mathbf{x})$. The symmetry of g yields $f_1(\mathbf{x}) + f_1(\mathbf{w}-\mathbf{x}) = g(\mathbf{x}) + g(\mathbf{w}-\mathbf{x}) = 1$;
2. if $h(\mathbf{x}) \leqslant 0$, then $f_1(\mathbf{x}) = 0$ and $f_1(\mathbf{w}-\mathbf{x}) = 1$, and hence $f_1(\mathbf{x}) + f_1(\mathbf{w}-\mathbf{x}) = 1$;
3. if $h(\mathbf{w}-\mathbf{x}) \leqslant 0$, then $f_1(\mathbf{x}) = 1$ and $f_1(\mathbf{w}-\mathbf{x}) = 0$, and hence $f_1(\mathbf{x}) + f_1(\mathbf{w}-\mathbf{x}) = 1$.

Therefore, f_1 is a VP-MDFF. The proof for f_2 is similar. □

Corollary 3. *Let $\mathbf{u} \in [0, \frac{1}{2}]^m$, and let $g:[0,1]^m \to [0,1]$ be a VP-MDFF. The following functions $f_1, f_2:[0,1]^m \to [0,1]$ are also VP-MDFFs:*

$$f_1(\mathbf{x}) := \begin{cases} 0, & \text{if } \mathbf{x} \leqslant \mathbf{u} \text{ and } \mathbf{x} \neq \frac{1}{2}\mathbf{w} \\ 1, & \text{if } \mathbf{x} \geqslant \mathbf{w}-\mathbf{u} \text{ and } \mathbf{x} \neq \frac{1}{2}\mathbf{w}, \\ g(\mathbf{x}), & \text{otherwise} \end{cases} \quad f_2(\mathbf{x}) := \begin{cases} 0, & \text{if } \mathbf{x} < \mathbf{u} \\ 1, & \text{if } \mathbf{x} > \mathbf{w}-\mathbf{u}. \\ g(\mathbf{x}), & \text{otherwise} \end{cases}$$

Proof. In that case, we can use in Proposition 6 for example

$$h(\mathbf{x}) := \begin{cases} 0, & \text{if } \mathbf{x} \leqslant \mathbf{u} \not\ni \mathbf{x} \neq \frac{1}{2}\mathbf{w}, \\ 1, & \text{otherwise,} \end{cases}$$

or

$$h(\mathbf{x}) := \begin{cases} 0, & \text{if } \mathbf{x} < \mathbf{u}, \\ 1, & \text{otherwise,} \end{cases}$$

because $h(\mathbf{x}) = 0$ for an $\mathbf{x} \in [0,1]^m$ implies $\mathbf{x} \neq \frac{1}{2}\mathbf{w}$ and $\mathbf{x} \leqslant \mathbf{u}$, hence $\mathbf{w} - \mathbf{x} \geqslant \mathbf{w} - \mathbf{u}$ and finally $h(\mathbf{w} - \mathbf{x}) = 1$. □

Corollary 4. *Let $\|\cdot\|_p$ be an $\mathcal{L}^p$-norm in $\mathbb{R}^m$ with $1 \leqslant p \leqslant \infty$, i.e.*

$$\|\mathbf{x}\|_\infty = \max_{r=1,\ldots,m} |x_r| \text{ and } \|\mathbf{x}\|_p = \sqrt[p]{\sum_{r=1}^m |x_r|^p} \text{ for } p < \infty.$$

Let $g{:}[0,1] \to [0,1]$ be a VP-MDFF and $\varepsilon \in (0, \|\mathbf{w}\|_p/2)$. The following function $f{:}[0,1]^m \to [0,1]$ is a VP-MDFF:

$$f(\mathbf{x}) := \begin{cases} 0, & \text{if } \|\mathbf{x}\|_p \leqslant \varepsilon, \\ 1, & \text{if } \|\mathbf{w} - \mathbf{x}\|_p \leqslant \varepsilon, \\ g(\mathbf{x}), & \text{otherwise.} \end{cases}$$

Proof. The function $h(\mathbf{x}) := \|\mathbf{x}\|_p - \varepsilon$ is non-decreasing in $[0,1]^m$. Because of

$$h(\mathbf{x}) + h(\mathbf{w} - \mathbf{x}) = \|\mathbf{x}\|_p + \|\mathbf{w} - \mathbf{x}\|_p - 2\varepsilon \geqslant \|\mathbf{w}\|_p - 2\varepsilon > 0 \quad \text{for any } \mathbf{x} \in [0,1]^m,$$

both the prerequisites of h in Proposition 6 are satisfied. Using this function h within f_1 in Proposition 6 leads to the function f. □

Corollary 5. *Let $g' : [0,1]^m \to \mathbb{R}$ be any non-decreasing function, and let $r \in \{1, \ldots, m\}$. The following function $f{:}[0,1]^m \to [0,1]$ is a VP-MDFF:*

$$f(\mathbf{x}) := \begin{cases} 0, & \text{if } 2\mathbf{w}^\top\mathbf{x} < m \text{ and } g'(\mathbf{x}) < 0, \\ 1, & \text{if } 2\mathbf{w}^\top\mathbf{x} > m \text{ and } g'(\mathbf{w} - \mathbf{x}) < 0, \\ x_r, & \text{otherwise.} \end{cases} \tag{21}$$

Proof. Here, the function $f_2(\mathbf{x})$ of Proposition 6 is used with

$$h(\mathbf{x}) := \begin{cases} \max\{g'(\mathbf{x}), -1\}, & \text{if } 2\mathbf{w}^\top\mathbf{x} < m \text{ and } g'(\mathbf{x}) < 0, \\ 2, & \text{otherwise.} \end{cases}$$

This function h is non-decreasing. If one has for an $\mathbf{x} \in [0,1]^m$ that $h(\mathbf{x}) < 2$ then $2\mathbf{w}^\top\mathbf{x} < m$, hence $2\mathbf{w}^\top(\mathbf{w} - \mathbf{x}) > m$, because $\mathbf{w}^\top\mathbf{w} = m$. That implies $h(\mathbf{w} - \mathbf{x}) + h(\mathbf{x}) \geqslant 2 - 1 > 0$. Of course, $g'(\mathbf{x}) < 0 \Leftrightarrow \max\{g'(\mathbf{x}), -1\} < 0$. Furthermore, the projections $\mathbf{x} \mapsto x_r$ are used as the function g in the definition of f_2 in Proposition 6s, with $r = 1, \ldots, m$. These projections are VP-MDFFs as shown in Proposition 5, and hence, our proof is complete. □

6.3. Class III

In the following proposition, we introduce a new VP-MDFF. We describe first the general function for the m-dimensional case. For the sake of clarity, the function for the 2-dimensional case is given next in Corollary 6. The idea consists in assigning the value zero to very small items and the value 1 to large ones, while another VP-MDFF is applied to the remaining items.

Proposition 7. *Let $g{:}[0,1]^m \to [0,1]$ be a VP-MDFF and $\mathbf{u} \in [0,1/2]^m$. The following function $f{:}[0,1]^m \to [0,1]$*

$$f(\mathbf{x};\mathbf{u}) := \begin{cases} 0, & \text{if } \exists i \in \{1,\ldots,m\} \text{ with } x_i < u_i \text{ and } \nexists j \in \{1,\ldots,i\} \text{ with } x_j > 1 - u_j, \\ 1, & \text{if } \exists i \in \{1,\ldots,m\} \text{ with } x_i > 1 - u_i \text{ and } \nexists j \in \{1,\ldots,i\} \text{ with } x_j < u_j, \\ g(\mathbf{x}), & \text{otherwise,} \end{cases}$$

is a VP-MDFF.

Proof. The proof relies on Theorem 2. First, we will show that f is symmetric, and then that it is non-decreasing. The latter will be used to prove the superadditivity of f.

Recall that since g is a VP-MDFF, it is symmetric due to Theorem 1. To show that f is symmetric, we distinguish the following three cases for a given $\mathbf{x} \in [0,1]^m$:

1. if $\mathbf{u} \leqslant \mathbf{x} \leqslant \mathbf{w} - \mathbf{u}$, then $f(\mathbf{x}) = g(\mathbf{x})$ and also $f(\mathbf{w} - \mathbf{x}) = g(\mathbf{w} - \mathbf{x})$, because $\mathbf{w} - \mathbf{u} \geqslant \mathbf{w} - \mathbf{x} \geqslant \mathbf{u}$. Hence, $f(\mathbf{x}) + f(\mathbf{w} - \mathbf{x}) = g(\mathbf{x}) + g(\mathbf{w} - \mathbf{x}) = 1$;
2. if there is an $i \in \{1, \ldots, m\}$ with $x_i < u_i$ but no $j \in \{1, \ldots, i\}$ with $x_j > 1 - u_j$, then $f(\mathbf{x}) = 0$. In that case, one has $f(\mathbf{w} - \mathbf{x}) = 1$ according to the second line in the definition of f. Hence, $f(\mathbf{x}) + f(\mathbf{w} - \mathbf{x}) = 1$;
3. if there is an $i \in \{1, \ldots, m\}$ with $x_i > 1 - u_i$ and no $j \in \{1, \ldots, i\}$ with $x_j < u_j$, then $f(\mathbf{x}) = 1$, and the first line in the definition of f applies to $f(\mathbf{w} - \mathbf{x})$. Hence, $f(\mathbf{x}) + f(\mathbf{w} - \mathbf{x}) = 1$.

To show that f is non-decreasing, let $\mathbf{x}, \mathbf{y} \in [0,1]^m$ with $\mathbf{x} \leqslant \mathbf{y}$. We distinguish the following three cases with respect to $f(\mathbf{x})$:

1. the monotonicity is obvious for $f(\mathbf{x}) = 0$, because $0 \leqslant g(\mathbf{y}) \leqslant 1$ for all $\mathbf{y} \in [0,1]^m$, and hence $f(\mathbf{y}) \geqslant 0$;
2. if there is an $i \in \{1, \ldots, m\}$ with $x_i > 1 - u_i$ but no $j \in \{1, \ldots, i\}$ with $x_j < u_j$, then $f(\mathbf{x}) = 1$. It follows that $y_i \geqslant x_i > 1 - u_i$ and $y_j \geqslant x_j \geqslant u_j$, for all $j \in \{1, \ldots, i\}$, and hence $f(\mathbf{y}) = 1$. As a consequence, it holds that $f(\mathbf{y}) \geqslant f(\mathbf{x})$;
3. if the last line in the definition of f applies, then $f(\mathbf{x}) = g(\mathbf{x})$ and $\mathbf{u} \leqslant \mathbf{x} \leqslant \mathbf{y}$. Hence, $f(\mathbf{y}) \in \{1, g(\mathbf{y})\}$, and $f(\mathbf{y}) \geqslant g(\mathbf{y})$ because $g(\mathbf{y}) \leqslant 1$. According to Theorem 1, it holds that $g(\mathbf{y}) \geqslant g(\mathbf{x})$. It follows that $f(\mathbf{x}) = g(\mathbf{x}) \leqslant g(\mathbf{y}) \leqslant f(\mathbf{y})$.

It remains to show that f is superadditive. For this purpose, let $\mathbf{x}, \mathbf{y} \in [0,1]^m$ with $\mathbf{x} + \mathbf{y} \leqslant \mathbf{w}$. Because of the monotonicity of f, the superadditivity is obvious for the case where $f(\mathbf{x}) = 0$ or $f(\mathbf{y}) = 0$. Therefore, we will assume that $f(\mathbf{x}) > 0$ and $f(\mathbf{y}) > 0$. Suppose that there is an $i \in \{1, \ldots, m\}$ with $x_i > 1 - u_i$ and $x_j \geqslant u_j$, for all $j \in \{1, \ldots, i\}$. Because of $\mathbf{y} \leqslant \mathbf{w} - \mathbf{x}$, it follows that $y_i < u_i$ and $y_j \leqslant 1 - u_j$, for all $j \in \{1, \ldots, i\}$, and hence, $f(\mathbf{y}) = 0$ in contradiction to the assumption $f(\mathbf{y}) > 0$. Exchanging $\mathbf{x}$ and $\mathbf{y}$ would yield a similar result. Therefore, $f(\mathbf{x}) = g(\mathbf{x})$ and $f(\mathbf{y}) = g(\mathbf{y})$. Since f is non-decreasing and $f(\mathbf{x}) > 0$, it follows that $f(\mathbf{x} + \mathbf{y}) > 0$. Hence, $f(\mathbf{x} + \mathbf{y}) \geqslant g(\mathbf{x} + \mathbf{y})$ according to the definition of f, because $g(\mathbf{x} + \mathbf{y}) \leqslant 1$. Since g is a VP-MDFF, it is also superadditive due to Theorem 1. Therefore, $g(\mathbf{x} + \mathbf{y}) \geqslant g(\mathbf{x}) + g(\mathbf{y}) = f(\mathbf{x}) + f(\mathbf{y})$, such that f is superadditive. □

Corollary 6. *The function $f{:}[0,1]^2 \to [0,1]$ with $0 \leqslant u_1, u_2 \leqslant 1/2$ and $r, q \in \{1,2\}$, which is defined as*

$$f(\mathbf{x};u_1,u_2,r,q) := \begin{cases} 1, & \text{if } x_q > 1 - u_1 \text{ or } (x_q \geqslant u_1 \text{ and } x_{3-q} > 1 - u_2), \\ x_r, & \text{if } 1 - u_1 \geqslant x_q \geqslant u_1 \text{ and } 1 - u_2 \geqslant x_{3-q} \geqslant u_2, \\ 0, & \text{otherwise,} \end{cases} \tag{22}$$

is a VP-MDFF.

Proof. For $q = 1$, this function is a special case of the generalization described in Proposition 7 with $m = 2$, $\mathbf{u} = (u_1, u_2)^\top$ and g being the projection $\mathbf{x} \mapsto x_r$. If $q = 2$, then x_1 and x_2 are tested in reverse order, i.e. first x_2 is compared with u_1 and $1 - u_1$, and only if that does not immediately lead to the function value 0 or 1 then x_1 is examined. That yields $f(\mathbf{x};u_1,u_2,r,2) = f(x_2,x_1;u_1,u_2,3-r,1)$. That case leads again to a VP-MDFF. □

6.4. Class IV

In this subsection, we introduce a new VP-MDFF in its general form, and a similar function for the 2-dimensional case. This VP-MDFF depends on a non-decreasing function whose properties are stated below. At the end of the subsection, we describe how this non-decreasing function should be chosen so as to obtain the best lower bounds with the corresponding VP-MDFFs.

Proposition 8. *Let $m \in \mathbb{N} \setminus \{0\}$, $k \in (1/3, 1/2]$, $g:[0,1]^m \to [0,1]$ be a non-decreasing function with*

$$\mathbf{x}, \mathbf{y} \in [0,1]^m, \mathbf{x} + \mathbf{y} \geqslant \mathbf{w} \Rightarrow g(\mathbf{x}) + g(\mathbf{y}) \geqslant 1, \tag{23}$$

and let M be a subset of $[0,1]^m \setminus \{\frac{1}{2}\mathbf{w}\}$ such that $\mathbf{x} \in M \Leftrightarrow \mathbf{w} - \mathbf{x} \notin M$ for all $\mathbf{x} \in [0,1]^m \setminus \{\frac{1}{2}\mathbf{w}\}$.

The function $f:[0,1]^{m+1} \to [0,1]$ defined as

$$f(\mathbf{x}, x_{m+1}; k) := \begin{cases} 0, & \text{if } x_{m+1} < k, \\ 1, & \text{if } x_{m+1} > 1-k, \\ 1/2, & \text{if } x_1 = x_2 = \ldots = x_{m+1} = 1/2, \\ g(\mathbf{x}), & \text{if } 1/2 < x_{m+1} \leqslant 1-k \text{ or } (x_{m+1} = 1/2 \text{ and } \mathbf{x} \in M), \\ 1 - g(\mathbf{w}-\mathbf{x}), & \text{if } k \leqslant x_{m+1} < 1/2 \text{ or } (x_{m+1} = 1/2 \text{ and } \mathbf{w}-\mathbf{x} \in M), \end{cases} \tag{24}$$

is a VP-MDFF.

Proof. The proof relies on Theorem 2. First, we verify that the function f is well defined, i.e. for any argument there is exactly one value assigned. Then, we show that f is symmetric, non-decreasing and finally superadditive.

If $x_{m+1} \neq 1/2$ then depending only on x_{m+1}, exactly one of the first, second, fourth or fifth line of the definition of f is assigned to an argument. The 1st line applies namely for $x_{m+1} < k$, the 5th line for $k \leqslant x_{m+1} < 1/2$, the 4th line for $1/2 < x_{m+1} \leqslant 1-k$ and the 2nd line for $x_{m+1} > 1-k$.

If $x_{m+1} = 1/2$ then the third, fourth or fifth line is selected depending on $\mathbf{x}$, namely the 3rd line for $\mathbf{x} = \frac{1}{2}\mathbf{w}$, the 4th line for $\mathbf{x} \in M$ and the 5th line otherwise. This is so because for any $\mathbf{x} \in [0,1]^m$ it holds either $\mathbf{x} = \frac{1}{2}\mathbf{w}$ or $\mathbf{x} \in M$ or $\mathbf{w} - \mathbf{x} \in M$ due to the definition of M.

To prove the symmetry of f, we distinguish the following cases with regard to the value of x_{m+1}:

1. if $x_1 = \cdots = x_{m+1} = 1/2$, then $f(\mathbf{x}, x_{m+1}) = f(\frac{1}{2}\mathbf{w}, 1/2) = 1/2 = 1 - f(\mathbf{w}-\mathbf{x}, 1-x_{m+1})$;
2. if $x_{m+1} < k$, then $f(\mathbf{x}, x_{m+1}) = 0$ and $1 - x_{m+1} > 1-k$. Hence, $f(\mathbf{w}-\mathbf{x}, 1-x_{m+1}) = 1$;
3. if $x_{m+1} > 1-k$, then $f(\mathbf{x}, x_{m+1}) = 1$ and $1 - x_{m+1} < k$. Hence, $f(\mathbf{w}-\mathbf{x}, 1-x_{m+1}) = 0$;
4. if $1/2 < x_{m+1} \leqslant 1-k$, then $k \leqslant 1 - x_{m+1} < 1/2$. Hence, $f(\mathbf{x}, x_{m+1}) = g(\mathbf{x})$ and $f(\mathbf{w}-\mathbf{x}, 1-x_{m+1}) = 1 - g(\mathbf{x})$;
5. if $k \leqslant x_{m+1} < 1/2$, $1-k \geqslant 1-x_{m+1} > 1/2$. Hence, $f(\mathbf{x}, x_{m+1}) = 1 - g(\mathbf{w}-\mathbf{x})$ and $f(\mathbf{w}-\mathbf{x}, 1-x_{m+1}) = g(\mathbf{w}-\mathbf{x})$;
6. if $x_{m+1} = 1/2$ and $\mathbf{x} \in M$, $f(\mathbf{x}, x_{m+1}) = g(\mathbf{x})$, and hence $f(\mathbf{w}-\mathbf{x}) = 1 - g(\mathbf{w} - (\mathbf{w}-\mathbf{x}))$;
7. if $x_{m+1} = 1/2$ and $\mathbf{w} - \mathbf{x} \in M$, $f(\mathbf{x}, x_{m+1}) = 1 - g(\mathbf{w}-\mathbf{x})$ and $\mathbf{w} - (\mathbf{w}-\mathbf{x}) \in M$. Hence, $f(\mathbf{w}-\mathbf{x}, x_{m+1}) = g(\mathbf{w}-\mathbf{x})$, such that the symmetry is valid in this case too.

To prove that f is non-decreasing, let $\mathbf{x}, \mathbf{y} \in [0,1]^m$ and x_{m+1}, $y_{m+1} \in [0,1]$, with $\mathbf{x} \leqslant \mathbf{y}$ and $x_{m+1} \leqslant y_{m+1}$. We have to show that $f(\mathbf{x}, x_{m+1}) \leqslant f(\mathbf{y}, y_{m+1})$. This is obvious for $x_{m+1} < k$ or $y_{m+1} > 1-k$, since for these cases we have respectively $f(\mathbf{x}, x_{m+1}) = 0$ and $f(\mathbf{y}, y_{m+1}) = 1$. Therefore, we assume that $k \leqslant x_{m+1} \leqslant y_{m+1} \leqslant 1-k$. To complete our proof concerning the monotonicity of f, we distinguish the following cases with regard to $(\mathbf{x}, x_{m+1})$ and $(\mathbf{y}, y_{m+1})$:

1. if for $(\mathbf{x}, x_{m+1})$ and $(\mathbf{y}, y_{m+1})$, the same line in the definition of f has to be applied. Then, $\mathbf{x} \leqslant \mathbf{y}$ and the monotonicity of g imply $g(\mathbf{x}) \leqslant g(\mathbf{y})$ and $g(\mathbf{w}-\mathbf{x}) \geqslant g(\mathbf{w}-\mathbf{y})$, and hence $f(\mathbf{x}, x_{m+1}) \leqslant f(\mathbf{y}, y_{m+1})$;
2. if
$$(k \leqslant x_{m+1} < 1/2) \text{ or } (x_{m+1} = 1/2 \text{ and } \mathbf{w} - \mathbf{x} \in M)$$
and
$$(1/2 < y_{m+1} \leqslant 1-k) \text{ or } (y_{m+1} = 1/2 \text{ and } \mathbf{y} \in M),$$
then $f(\mathbf{x}, x_{m+1}) = 1 - g(\mathbf{w}-\mathbf{x})$, $f(\mathbf{y}, y_{m+1}) = g(\mathbf{y})$ and $\mathbf{x} \leqslant \mathbf{y}$ lead to $f(\mathbf{x}, x_{m+1}) \leqslant f(\mathbf{y}, y_{m+1})$, because $g(\mathbf{w}-\mathbf{x}) + g(\mathbf{y}) \geqslant 1$ due to (23), since $\mathbf{w} - \mathbf{x} + \mathbf{y} \geqslant \mathbf{w}$;
3. if $(k \leqslant x_{m+1} < 1/2)$ or $(x_{m+1} = 1/2$ and $\mathbf{w} - \mathbf{x} \in M)$, and $y_1 = \cdots = y_{m+1} = 1/2$, then $g(\mathbf{w}-\mathbf{x}) \geqslant g(\frac{1}{2}\mathbf{w}) \geqslant 1/2$, since $\mathbf{x} \leqslant \mathbf{y} = \frac{1}{2}\mathbf{w}$, and hence
$$f(\mathbf{x}, x_{m+1}) = 1 - g(\mathbf{w}-\mathbf{x}) \leqslant 1/2 = f(\mathbf{y}, y_{m+1});$$
4. The remaining case
$$x_1 = \cdots = x_{m+1} = 1/2 \text{ and } ((1/2 < y_{m+1} \leqslant 1-k) \text{ or } (y_{m+1} = 1/2 \text{ and } \mathbf{y} \in M))$$
can easily be handled by the symmetry of f, namely $k \leqslant 1 - y_{m+1} < 1/2$ or $(1 - y_{m+1} = 1/2$ and $\mathbf{w} - (\mathbf{w}-\mathbf{y}) \in M)$, yielding $f(\mathbf{w}-\mathbf{y}, 1 - y_{m+1}) \leqslant 1/2$ due to the previous case. That gives $1/2 \geqslant 1 - f(\mathbf{y}, y_{m+1})$ by symmetry, and hence $f(\mathbf{y}, y_{m+1}) \geqslant 1/2 = f(\mathbf{x}, x_{m+1})$.

The superadditivity of f is proved as follows. Let $\mathbf{x}, \mathbf{y} \in [0,1]^m$ and $x_{m+1}, y_{m+1} \in [0,1]$, with $\mathbf{x} + \mathbf{y} \leqslant \mathbf{w}$ and $x_{m+1} + y_{m+1} \leqslant 1$. Assume that $f(\mathbf{x}, x_{m+1}) > 0$ and $f(\mathbf{y}, y_{m+1}) > 0$, otherwise the superadditivity is obvious. Therefore, $x_{m+1} \geqslant k$ and $y_{m+1} \geqslant k$, and hence $x_{m+1} + y_{m+1} \geqslant 2k > 1-k$, implying $f(\mathbf{x}+\mathbf{y}, x_{m+1} + y_{m+1}) = 1$. The monotonicity and symmetry of f yield also

$$f(\mathbf{x}, x_{m+1}) \leqslant f(\mathbf{w}-\mathbf{y}, 1-y_{m+1}) = 1 - f(\mathbf{y}, y_{m+1}),$$

and hence

$$f(\mathbf{x}, x_{m+1}) + f(\mathbf{y}, y_{m+1}) \leqslant f(\mathbf{x}+\mathbf{y}, x_{m+1} + y_{m+1}) = 1. \quad \square$$

Note, here it is not necessary to demand the condition (15) to M. For $m = 1$ and $M = (\frac{1}{2}, 1]$, the function (24) becomes

$$f(x_1, x_2; k) = \begin{cases} 0, & \text{if } x_2 < k, \\ 1, & \text{if } x_2 > 1-k, \\ 1/2, & \text{if } x_1 = x_2 = 1/2, \\ g(x_1), & \text{if } 1/2 < x_2 \leqslant 1-k \text{ or } (x_2 = 1/2 \text{ and } x_1 > 1/2), \\ 1 - g(1-x_1), & \text{if } k \leqslant x_2 < 1/2 \text{ or } (x_2 = 1/2 \text{ and } x_1 < 1/2). \end{cases}$$

The following VP-MDFF is similar, but it differs for $x_1 \in \{0,1\}$, i.e. it may get other function values in these cases.

Proposition 9. *Let g be a non-decreasing function defined from $[0,1]$ to $[0,1]$, such that*

$$g(y) + g(1-y) \geqslant 1, \text{ for all } y \in [0,1]. \tag{25}$$

The function $f:[0,1]^2 \to [0,1]$ with $k \in (1/3, 1/2]$ and defined as

$$f(\mathbf{x}; k) := \begin{cases} 0, & \text{if } x_1 = 0 \text{ or } (x_1 < 1 \text{ and } x_2 < k), \\ 1, & \text{if } x_1 = 1 \text{ or } (x_1 > 0 \text{ and } x_2 > 1-k), \\ 1/2, & \text{if } x_1 = x_2 = 1/2, \\ g(x_1), & \text{if } (1/2 < x_2 \leqslant 1-k \text{ and } 0 < x_1 < 1) \text{ or } (x_2 = 1/2 \text{ and } 1/2 < x_1 < 1), \\ 1 - g(1-x_1), & \text{if } (k \leqslant x_2 < 1/2 \text{ and } 0 < x_1 < 1) \text{ or } (x_2 = 1/2 \text{ and } 0 < x_1 < 1/2), \end{cases}$$

is a VP-MDFF.

Proof. The proof relies on Theorem 2, and it is organized as the one for Proposition 8. First, we verify that f is well defined, i.e. we assign to any argument exactly one value. Then, we show that f is symmetric, non-decreasing and superadditive. The analysis can be restricted to the cases with $x_1 \in \{0,1\}$, because for $0 < x_1 < 1$ it is the same as the above given special case of Proposition 8 for $m = 1$ and $M = (\frac{1}{2}, 1]$.

The function f is defined with the required domain and range. If $x_1 \in \{0,1\}$, then the first or second line in the definition of f applies, while the fourth and fifth line explicitly exclude the case $x_1 \in \{0,1\}$.

The symmetry becomes obvious for $x_1 \in \{0,1\}$. If $x_1 = 0$, then $f(\mathbf{x}) = 0$ and $1 - x_1 = 1$, and hence $f(\mathbf{w} - \mathbf{x}) = 1$. If $x_1 = 1$, then $f(\mathbf{x}) = 1$ and $1 - x_1 = 0$, and hence $f(\mathbf{w} - \mathbf{x}) = 0$.

To show that f is non-decreasing, choose any $\mathbf{x}, \mathbf{y} \in [0,1]^2$ with $\mathbf{x} \leqslant \mathbf{y}$ and ($x_1 \in \{0,1\}$ or $y_1 \in \{0,1\}$).

If $x_1 = 0$, then $f(\mathbf{x}) = 0 \leqslant f(\mathbf{y})$. If $x_1 = 1$, then $y_1 = 1$, and hence $f(\mathbf{y}) = 1 = f(\mathbf{x})$. The proof for $y_1 \in \{0,1\}$ is similar. It remains to show that f is superadditive. Let $\mathbf{x}, \mathbf{y} \in [0,1]^2$ with $\mathbf{x} + \mathbf{y} \leqslant \mathbf{w}$. If $f(\mathbf{x}) = 0$ or $f(\mathbf{y}) = 0$, then the monotonicity of f implies that f is superadditive. Therefore, we assume that $f(\mathbf{x}) > 0$ and $f(\mathbf{y}) > 0$. We have $x_1, y_1 > 0$ and $x_2, y_2 \geqslant k$, which yields $x_2 + y_2 \geqslant 2k > 1 - k$ and $x_1 + y_1 > 0$, and hence $f(\mathbf{x} + \mathbf{y}) = 1$. Since f is non-decreasing and symmetric, it follows that $f(\mathbf{y}) \leqslant f(\mathbf{w} - \mathbf{x}) = 1 - f(\mathbf{x})$. Hence, we have $f(\mathbf{x}) + f(\mathbf{y}) \leqslant f(\mathbf{x} + \mathbf{y})$ such that f is superadditive. □

In the following proposition, we show how the g function in Proposition 9 should be defined so as to get the best lower bound that can be obtained from the corresponding VP-DFF $f(\mathbf{x}; k)$ for the 2D-VPP.

Proposition 10. *Given an instance of the 2D-VPP, the best lower bound based on the function f of Proposition 9 can be found with the following function $g:[0,1] \to \{0, 1/2, 1\}$ that depends on the parameters $s, t \in \mathbb{R}$ with $0 \leqslant s \leqslant 1/2$ and $s \leqslant t \leqslant 1 - s$:*

$$g(x) := \begin{cases} 0, & \text{if } x < s, \\ 1/2, & \text{if } s \leqslant x \leqslant t, \\ 1, & \text{if } x > t. \end{cases}$$

Proof. Assume that we are given a function $g:[0,1] \to [0,1]$ that satisfies the prerequisites of Proposition 9, i.e. g is non-decreasing and obeys the condition (25). First, we analyze the structure of the resulting bound (16), then we simplify the function g without decreasing the obtained bound $z[f]$. We will show that an optimal function g can be found, which has only function values in $\{0, 1/2, 1\}$.

If the function f of Proposition 9 is applied to an item of size (y_1, y_2) and with order demand b, then one of the values 0, b, $b/2$, $b \times g(y_1)$ or $b - b \times g(1 - y_1)$ is added in the bound $z[f]$, because $f(\mathbf{y}) \in \{0, 1/2, 1, g(y_1), 1 - g(1 - y_1)\}$. Therefore, applying the function f of Proposition 9 to an instance with n items and a bin size $(1,1)$ yields the lower bound (16) in the form

$$z[f] = c + \sum_{i=1}^{n'} a_i g(x_i), \tag{26}$$

with $n' \in \mathbb{N},\ n' \leqslant n,\ 2c \in \mathbb{N}$ and $a_i \in \mathbb{Z} \setminus \{0\}$, while either x_i or $1 - x_i$ are item sizes.

Assume without loss of generality that $0 < x_1 < \cdots < x_{n'} < 1$. Now we begin to simplify g without affecting the bound (26). Let $x_{n'+1} := 1$. Especially, $g(x) := g(x_1)$ for all $x \in [0, x_1)$ and $g(x) := g(x_i)$ for $x \in (x_{i-1}, x_i)$, $i = 2, \ldots, n' + 1$, preserves the monotonicity of g and the property (25). In this way, g becomes a staircase function. The x_i and the coefficients c and a_i are given; the $g(x_i)$ have to be selected to maximize the expression (26) under the restrictions of g described in Proposition 9, i.e. $0 \leqslant g(x_1) \leqslant \ldots \leqslant g(x_{n'}) \leqslant 1$ and $x_i + x_j \geqslant 1 \Rightarrow g(x_i) + g(x_j) \geqslant 1$ for all $i, j \in \{1, \ldots, n'\}$. Calculating an optimal function g can be done by considering the following arguments based on dominance:

1. if there is an $i \in \{1, \ldots, n' - 1\}$ with $a_i \geqslant 0$, then set $g(x_i) := g(x_{i+1})$, because this is the largest value that is allowed for $g(x_i)$ due to the monotonicity condition. Smaller values for $g(x_i)$ cannot increase the expression (26). Moreover, if the condition (25) is fulfilled for a given value of $g(x_i)$, then it is clearly obeyed for larger values of $g(x_i)$ too. Because of $g(x_i) = g(x_{i+1})$, the expression (26) remains unchanged, if one sets $a_{i+1} := a_{i+1} + a_i$ and after that $a_i := 0$. Such terms with $a_i = 0$ can be cut out immediately;
2. if $a_{n'} \geqslant 0$, then $g(x_{n'}) := 1$ is analogously an optimal choice because of the restriction $g(x_{n'}) \leqslant 1$. To simplify the expression (26), set $c := c + a_{n'}$ and after that $n' := n' - 1$, such that the last summand disappears without changing the value of the bound (26);
3. after eliminating the nonnegative coefficients a_i according to the former points 1 and 2, only negative coefficients a_i may remain in the bound (26). If some negative coefficients (and no positive ones) remained, then the further simplifications depend on the x_i itself, as stated in the following;
4. if $x_1 + x_{n'} < 1$ ($a_1 \leqslant 0$ and $a_{n'} \leqslant 0$), then $g(x_1) := 0$ is an optimal choice, because it has no influence on condition (25) or on the monotonicity of g;
5. if only one term $a_i g(x_i)$ with $a_i < 0$ and $1/2 \leqslant x_i < 1$ remained in the sum (26), then set $g(x_i) := 1/2$ according to the condition (25);
6. If $n' \geqslant 2$ and $x_1 + x_{n'-1} \geqslant 1$, then set $g(x_{n'}) := g(x_{n'-1})$ or equivalently $a_{n'-1} := a_{n'-1} + a_{n'}$ and then $n' := n' - 1$. The monotonicity of g and the condition (25) are not affected by these changes;
7. If $a_1 > a_{n'}$ and $x_{n'-1} < 1 - x_1 \leqslant x_{n'}$, then set $g(x_1) := 1 - g(x_{n'})$, because a smaller value of $g(x_1)$ is not allowed due to condition (25). Larger values of $g(x_1)$ are neither useful for the bound (26) nor necessary. Since $g(x_1) = 1 - g(x_{n'})$, one gets $a_1 g(x_1) + a_{n'} g(x_{n'}) = a_1 + (a_{n'} - a_1) g(x_{n'})$, and hence this case can be simplified by $c := c + a_1$, $a_{n'} := a_{n'} - a_1$ and then $a_1 := 0$.
8. Otherwise, i.e. $a_1 \leqslant a_{n'} \leqslant 0$ and $x_{n'-1} < 1 - x_1 \leqslant x_{n'}$, an optimal choice is $g(x_1) := 0$, and $g(x_{n'}) := 1$ to maximize the expression (26) under the given conditions.

The simplifications of the bound (26) can be done in a loop, which leads always to $g(x) \in \{0, 1/2, 1\}$ for the considered arguments and an optimal staircase function g. Since that holds for all values k, which have to be explored (and that are not more than $n + 1$ ones), the assertion follows. □

6.5. Class V

Let $\mathbf{s}, \mathbf{t} \in (0,1]^m$, and let u_1, u_2 be feasible (not necessarily optimal) dual values for an instance of the mD-VPP with two items of sizes $\mathbf{s}$ and $\mathbf{t}$, each demanded at least once. The following proposition describes a new family of VP-DFFs. Recall that these functions can be transformed into VP-MDFFs by enforcing symmetry through Proposition 3.

Proposition 11. *The function $f:[0,1]^m \to [0,1]$ is a superadditive VP-DFF:*

$$f(\mathbf{x}) := \max\{a_1 u_1 + a_2 u_2 | a_1, a_2 \in \mathbb{N}, a_1 \mathbf{s} + a_2 \mathbf{t} \leqslant \mathbf{x}\}.$$

Proof. First, we show that the range of f belongs to $[0,1]$. Then, we prove that f is superadditive and finally that f is a VP-DFF.

Since u_1, u_2 are feasible values for the dual problem, it follows that $u_1, u_2 \geqslant 0$. Because of

$$f(\mathbf{x}) \geqslant 0u_1 + 0u_2 = 0 \text{ and } a_1u_1 + a_2u_2 \leqslant 1,$$

for all $a_1, a_2 \in \mathbb{N}$ with $a_1\mathbf{s} + a_2\mathbf{t} \leqslant \mathbf{w}$, the range of f is a subset of $[0,1]$. To show the superadditivity of f, let $\mathbf{x}$, $\mathbf{y} \in [0,1]^m$ with $\mathbf{x} + \mathbf{y} \leqslant \mathbf{w}$. Then, we have

$$\begin{aligned} f(\mathbf{x}+\mathbf{y}) &= \max\{a_1u_1 + a_2u_2 | a_1, a_2 \in \mathbb{N}, a_1\mathbf{s} + a_2\mathbf{t} \leqslant \mathbf{x} + \mathbf{y}\} \\ &\geqslant \max\{a_1^1u_1 + a_2^1u_2 | a_1^1, a_2^1 \in \mathbb{N}, a_1^1\mathbf{s} + a_2^1\mathbf{t} \leqslant \mathbf{x}\} + \\ &\quad \max\{a_1^2u_1 + a_2^2u_2 | a_1^2, a_2^2 \in \mathbb{N}, a_1^2\mathbf{s} + a_2^2\mathbf{t} \leqslant \mathbf{y}\} \\ &= f(\mathbf{x}) + f(\mathbf{y}). \end{aligned}$$

To show that f is a VP-DFF, choose any finite index-set J and vectors $\mathbf{x}^j \in [0,1]^m$ ($j \in J$), with $\sum_{j\in J}\mathbf{x}^j \leqslant \mathbf{w}$. The superadditivity yields $\sum_{j\in J} f(\mathbf{x}^j) \leqslant f(\sum_{j\in J}\mathbf{x}^j) \leqslant 1$ due to the range of f. □

Calculating the function requires to solve an integer optimization problem for every argument $\mathbf{x}$, but the complexity remains low, if the possible values $a_1, a_2 \in \mathbb{N}$ are bounded by a small constant. Without loss of generality, assume that $\max_{i\in\{1,\ldots,m\}} s_i \leqslant \max_{i\in\{1,\ldots,m\}} t_i$. Since there are only two items $\mathbf{s},\mathbf{t} > \mathbf{o}$, the function value can be easily calculated by trying all possible numbers a_2, i.e. $a_2 \in \mathbb{N}$ and $a_2 \leqslant \min\{x_i/t_i : i \in \{1,\ldots,m\}\}$, and setting $a_1 := \lfloor \min_{i\in\{1,\ldots,m\}}(x_i - a_2 * t_i)/s_i \rfloor$. The effort for these calculations is the same as for dynamic optimization with exactly two different items. Therefore, the complexity to calculate a_1 and a_2 is pseudo-polynomial.

7. Implementation details and complexity results

In this section, we discuss the implementation details of the lower bounding procedures related to the VP-MDFFs proposed above, and we report on the complexity of these procedures. We focus on the specific procedures that were implemented and tested, and whose results are reported in Section 8.

First, we recall the complexity of computing lower bounds for the mD-VPP using the column generation approach applied to formulation (8)–(10). The generation of a new column for (8)–(10) requires to solve an m-dimensional knapsack problem, which is well-known to be $\mathcal{NP}$-complete. At each iteration of the column generation algorithm, the corresponding m-dimensional knapsack problem can be solved in pseudo-polynomial time using dynamic programming. Furthermore, the number of columns of (8)–(10) grows exponentially with the number n of items in the problem. Below, we will show that our VP-MDFF based lower bounding procedures for the mD-VPP are much more efficient than the column generation approach. Indeed, all these procedures except the last one are algorithms of polynomial complexity.

Below, we identify the specific VP-MDFFs that were considered. A lower bounding procedure for the mD-VPP is defined from each function. These procedures rely essentially on (16). The details of the implementation of each one of these VP-MDFF based lower bounding procedures and their corresponding complexity are discussed next.

- (a) : VP-MDFF of Proposition 5 with $g = f_{CCM,1}$;
- (b) : VP-MDFF of Corollary 1 (Proposition 5 with $g = f_{FS,1}$);
- (c) : VP-MDFF of Corollary 2 (Proposition 5 with $g = f_{BJ,1}$);
- (d) : VP-MDFF of Corollary 3 (using f_2 and g obtained from Proposition 5 and $f_{CCM,1}$);
- (e) : VP-MDFF of Corollary 4 (with g obtained from Proposition 5 and $f_{CCM,1}$);
- (f) : VP-MDFF of Corollary 5 (with $g(\mathbf{x}) := \|\mathbf{x}\|_p - c$ with constants $p, c \in \mathbb{R}$, $p \geqslant 1$);
- (g) : VP-MDFF of Corollary 6;
- (h) : VP-MDFF of Proposition 9 with the function g described in Proposition 10.
- (i) : VP-DFF of Proposition 11 by forcing symmetry as discussed in Proposition 3.

Cases (a)–(e)

The lower bounding procedures related to (a)–(c) correspond to the lower bounding scheme (16) applied with the VP-MDFFs obtained from Proposition 5 with $g \in \{f_{CCM,1}, f_{FS,1}, f_{BJ,1}\}$. The last two functions lead respectively to the functions of Corollaries 1 and 2. The cases (d) and (e) are associated to VP-MDFFs constructed from Corollaries 3 and 4 with g based on Proposition 5 with $f_{CCM,1}$.

In each one of these cases, the use of the corresponding function causes the complexity $\mathcal{O}(n)$ in the evaluation of the bound (16). These functions are used with pseudo-random parameters. The number of parameter sets that are tried is a constant for each function, i.e. it is independent from the number n of items.

Case (f)

The lower bounding procedure related to (f) relies on (16) and on the VP-MDFF of Corollary 5 with $g'(\mathbf{x}) := \|\mathbf{x}\|_p - c$ and constants $p, c \in \mathbb{R}$, $p \geqslant 1$. Let $I_1 := \{i \in \{1,\ldots,n\} : \mathbf{w}^\top\mathbf{l}_i > m/2 \text{ and } g'(\mathbf{w} - \mathbf{l}_i) < 0\}$ and

$$I_2 := \{i \in \{1,\ldots,n\} \setminus I_1 : \mathbf{w}^\top\mathbf{l}_i \geqslant m/2 \text{ or } g'(\mathbf{l}_i) \geqslant 0\}. \tag{27}$$

Then, the bound (16) can be rewritten as follows

$$z[f] = \sum_{i\in I_1} b_i + \max_{r\in\{1,\ldots,m\}} s_r \text{ , with } \mathbf{s} := \sum_{i\in I_2} b_i \times \mathbf{l}_i. \tag{28}$$

Using (28) simplifies the search for the best parameter values with low complexity. Computing $z[f]$ from (16) requires $\mathcal{O}(n)$ evaluations of the VP-DFF f for each chosen parameter set. In the form (28), the effort $\mathcal{O}(n)$ is needed only once for an initial parameter set. After that, if each change of the parameters modifies the sets I_1 and I_2 by at most one element, recomputing $z[f]$ requires only the complexity $\mathcal{O}(1)$ instead of $\mathcal{O}(n)$. Therefore, if the parameter sets are adapted according to the input data, the total complexity can be reduced by a factor up to n in the evaluation of $z[f]$.

Since choosing p optimally is not obvious, an exponential sequence $p \in \{1, 1.1, 1.1^2, \ldots, 1.1^{29}\}$ is tried. For each of these 30 different values, the remaining parameters c and r in Corollary 5 can be chosen optimally by trying all

$$c \in \{m, \|\mathbf{l}_i\|_p : i \in \{1,\ldots,n\} \wedge \mathbf{w}^\top\mathbf{l}_i < m/2\}.$$

For given parameters p and c, the set I_2 is determined by (27), such that the index r can be found according to (28). The choice $c := m$ is sufficiently large to assign all items the value 0 or 1, except to items $\mathbf{l}_i$ with $\mathbf{w}^\top\mathbf{l}_i = m/2$, because $\|\mathbf{w}\|_p = \sqrt[p]{m} \leqslant m$. To achieve a small complexity, we sort the items first, and then a loop with complexity $\mathcal{O}(mn)$ is executed. Without sorting the items, it would be necessary to try up to $n+1$ values of c, and to evaluate each of the n items according to the function f for the given c with complexity $\mathcal{O}(m)$, such that the total complexity would rise to $\mathcal{O}(mn^2)$. Hence, for each p, the computation of the lower bound can be done with complexity $\mathcal{O}(mn \ln n)$.

The details of this procedure are described in Algorithm 1. Items $\mathbf{l}_i$ with $\mathbf{w}^\top\mathbf{l}_i = m/2$ never get the value 0 or 1 in the first or second line of the definition of the function f in Corollary 5. Therefore, they are separated from the other items and they are always considered in the set I_2 in the bound (28). Other items of sizes $\mathbf{x}$ and $\mathbf{y}$ might be put into one container if $\mathbf{w}^\top\mathbf{x} < m/2 < \mathbf{w}^\top\mathbf{y}$ and $\|\mathbf{x}\|_p = \|\mathbf{w} - \mathbf{y}\|_p$. Therefore, such items are either considered simultaneously in the set I_2 in the bound (28), or none of them. To handle these situations efficiently, including the ordering of the items and the comparisons between them, the following auxiliary function $h : [0,1]^m \to \mathbb{R}_+$ is used:

$$h(\mathbf{x}) := \begin{cases} m, & \text{if } \mathbf{w}^\top \mathbf{x} = m/2, \\ \|\mathbf{x}\|_p, & \text{if } \mathbf{w}^\top \mathbf{x} < m/2, \\ \|\mathbf{w} - \mathbf{x}\|_p, & \text{otherwise.} \end{cases}$$

Since $\|\mathbf{w}\|_p = \sqrt[p]{m}$, it follows that $h(\mathbf{x}) < m$, for all $\mathbf{x}$ for which $\mathbf{w}^\top \mathbf{x} \neq m/2$. Therefore, items $\mathbf{l}_i$ with $\mathbf{w}^\top \mathbf{l}_i = m/2$ will always be placed in the last positions after the sorting that is done at the beginning of the algorithm.

Algorithm 1 (*Lower bounding procedure for the mD-VPP based on the VP-MDFF of case (f)*).

Input:
n items with sizes $\mathbf{l}_i \in [0,1]^m$ and order demands $b_i \in \mathbb{N} \setminus \{0\}$;
p: parameter with $p \geq 1$;
Output:
z: lower bound according to (16) for optimal values $c \in \mathbb{R}$ and $r \in \{1, \ldots, m\}$;
Auxiliary variables:
$i, k \in \mathbb{N}$; $t \in \mathbb{R}$; $\mathbf{s} \in \mathbb{R}^m_+$;

Sort the items in non-decreasing order of $h(\mathbf{l}_i)$;
$\mathbf{s} := \mathbf{o}$; $i := n+1$; $k := 0$; $z := 0$;
repeat
  $i := i - 1$;
  if $(i = 1)$ `or` $(h(\mathbf{l}_i) < m)$ **then** break;
  ;
  $\mathbf{s} := \mathbf{s} + b_i \times \mathbf{l}_i$;
until *false*;
repeat
  $t := \max_{r \in \{1,\ldots,m\}} s_r - k$;
  if $(t > z)$ **then** $z := t$;
  ;
  if $(i = 0)$ **then** break;
  ;
  $t := h(\mathbf{l}_i)$;
  repeat
    $\mathbf{s} := \mathbf{s} + b_i \times \mathbf{l}_i$;
    if $(\mathbf{w}^\top \mathbf{l}_i > m/2)$ **then** $k := k + b_i$;
    ;
    $i := i - 1$;
  until $(i = 0)$ `or` $(h(\mathbf{l}_i) < t)$;
until *false*;
$z := z + k$;

In Algorithm 1, the variable k counts the items which are assigned the value 1 according to Corollary 5, with $g'(\mathbf{x}) = \|\mathbf{x}\|_p - c$ for enough large c, i.e. items $\mathbf{l}_i$ with $\mathbf{w}^\top \mathbf{l}_i > m/2 \wedge c > \|\mathbf{w} - \mathbf{l}_i\|_p$. If $c \leqslant 0$, then $g'(\mathbf{x}) \geqslant 0$ for all $\mathbf{x}$, such that $f(\mathbf{x}) = x_r$ for all $\mathbf{x} \in [0,1]^m$. In that case, only the material bound is calculated, i.e. $\mathbf{s} = \sum_{i=1}^n b_i \times \mathbf{l}_i$ and $z = \max_{r \in \{1,\ldots,m\}} s_r$.

When the statement $t := h(\mathbf{l}_i)$ is reached for a given i, then using $c := t$ implies $h(\mathbf{l}_j) \leqslant c$ for all $j \in \{1, \ldots, i\}$, due to the sorting that is done at the beginning. Items $\mathbf{l}_j$, with $h(\mathbf{l}_j) = c$, will still be counted for $\mathbf{s}$, but all later processed items are assigned values 0 or 1 in the VP-MDFF f of Corollary 5. Therefore, the set I_2 does not contain any of the remaining items for this value c, such that it is only necessary to count the number of remaining items, which will get the value 1 due to the function f of Corollary 5. If in the last `repeat-until`-loop, the counter k is increased, then these items must be put into new containers, but the residual space in these containers can be used for small items. Therefore, it is necessary to subtract k before comparing t with the former bound z.

Since the index i is decremented in every passage of the loops, the complexity without the sorting would be $\mathcal{O}(mn)$. Sorting n items requires $\mathcal{O}(n \ln n)$ comparisons and possible exchanges. Since we have also to consider the dimension m of the items, the complexity becomes $\mathcal{O}(mn \ln n)$.

Case (g)

A lower bounding procedure based on (16) and on the VP-MDFF of Corollary 6 is described in Algorithm 2. Since the functions (21) and (22) share some similarities ($f(\mathbf{x}) = x_r$ for some $\mathbf{x}$ with the parameter $r \in \{1, \ldots, m\}$, and $f(\mathbf{x}) \in \{0,1\}$ otherwise), it is again advantageous to rewrite the bound (16) in the form (28), but with other sets I_1, I_2, namely $I_1 := \{i \in \{1, \ldots, n\} : \ell_{iq} > 1 - u_1 \vee (\ell_{iq} \geqslant u_1 \not\geqslant \ell_{i,3-q} > 1 - u_2)\}$ and $I_2 := \{i \in \{1, \ldots, n\} : u_1 \leqslant \ell_{iq} \leqslant 1 - u_1 \not\geqslant u_2 \leqslant \ell_{i,3-q} \leq 1 - u_2\}$, with $q \in \{1,2\}$. An optimal choice of u_1, u_2 for a given q is possible with $u_1 \in \{1/2, \ell_{iq} : i \in \{1, \ldots, n\} \not\ni \ell_{iq} < 1/2\}$ and $u_2 \in \{1/2, \ell_{i,3-q} : i \in \{1, \ldots, n\} \not\ni \ell_{i,3-q} < 1/2\}$. Since up to $(n+1)^2$ pairs (u_1, u_2) have to be tested, and the computation of the bound (28) for one pair (u_1, u_2) requires a complexity $\mathcal{O}(n)$, a naive implementation would have the total complexity $\mathcal{O}(n^3)$. However, by using a suitable sorting, the complexity decreases to $\mathcal{O}(n^2)$ as it happens in Algorithm 2. The goal of the sorting is that for fixed variables q and u_1, the change of the sets I_1 and I_2 in every step where u_2 is modified applies to at most one element. This objective is achieved by the sorting in Algorithm 2. When $u_2 := \ell_{i,3-q} \leqslant 1/2$ is set for a given $i \in \{1, \ldots, n\}$, it follows for all $j \in \{1, \ldots, i\}$ that either $\ell_{j,3-q} \leqslant u_2$ or $\ell_{j,3-q} > 1 - u_2$. Therefore, except for $\ell_{j,3-q} = u_2$, no more items $j \in \{1, \ldots, i\}$ can belong to I_2. Algorithm 2 is used with $q = 1$ and $q = 2$, and the better of the two bounds is chosen.

Algorithm 2 (*Lower bounding procedure for the mD-VPP based on the VP-MDFF of case (g)*).

Input:
n items with sizes $\mathbf{l}_i \in [0,1]^2$ and order demands $b_i \in \mathbb{N} \setminus \{0\}$;
q: parameter with $q \in \{1,2\}$;
ε: small constant with $\varepsilon > 0$;
Output:
z: lower bound based on (16) and on the VP-MDFF of Corollary 6 for the given q and optimal r, u_1, u_2;
/* u_1 is explicitely set below; u_2 and r are set implicitely. */
Auxiliary variables:
$i, k_1, k_2 \in \mathbb{N}$; $t, u_1, v \in \mathbb{R}$; $\mathbf{s} \in \mathbb{R}^2_+$;

Sort the items in non-decreasing order of $\varepsilon \times \ell_{i,3-q} - |0.5 - \ell_{i,3-q}|$;
$z := -1$; $t := 1/2$;
repeat
  $\mathbf{s} := \mathbf{o}$; $v := 0$; $k_1 := 0$; $k_2 := 0$; $u_1 := t - \varepsilon$;
  /* Try $u_1 \in \{1/2 - \varepsilon, \ell_{iq} - \varepsilon | i \in \{1,\dots,n\} \wedge \ell_{iq} < 1/2\}$. */
  for $i := n$ **downto** 1 **do**
    if $(\ell_{iq} > 1 - u_1)$ **then** $k_1 := k_1 + b_i$;
    ;
    else if $(\ell_{iq} \geq u_1)$ **then**
      $\mathbf{s} := \mathbf{s} + b_i \times \mathbf{l}_i$;
      if $(\ell_{i,3-q} > \varepsilon + 1/2)$ **then** $k_2 := k_2 + b_i$;
      ;
      else
        /* Try $u_2 := \ell_{i,3-q}$ in Corollary 6. Take r according to $t := \max_{r \in \{1,2\}} s_r - k_2$. */
        $t := \max\{s_1, s_2\} - k_2$;
        if $(t > v)$ **then** $v := t$;
        ;
      end
    end
  end
  $t := -1$; $v := v + k_1 + k_2$;
  if $(v > z)$ **then** $z := v$;
  ;
  for $i := n$ **downto** 1 **do**
    if $(\ell_{iq} < u_1 - \varepsilon)$ **and** $(\ell_{iq} > t + \varepsilon)$ **then** $t := \ell_{iq}$;
    ;
  end
until $(t < 0)$;

The worst case is reached when all the values ℓ_{iq} are different and less than 1/2. In this case, the outer loop is passed $n+1$ times with $u_1 \in \{1/2 - \varepsilon, \ell_{iq} - \varepsilon\}$ and $i \in \{1,\dots,n\}$. Therefore, the complexity of the algorithm is $\mathcal{O}(n^2)$.

The variables k_1 and k_2 count the number of items that (may) get the value 1 in the comparison with u_1,u_2 in the VP-MDFF of Corollary 6. If the statement $t := \max\{s_1, s_2\} - k_2$ is reached for a given i, then the sorting at the beginning implies $\ell_{j,3-q} > 1 - \ell_{i,3-q}$ or $\ell_{j,3-q} \leqslant \ell_{i,3-q}$, for all $j \in \{1,\dots,i\}$. Therefore, trying $u_2 := \ell_{i,3-q}$ causes $f(\mathbf{l}_j) \in \{0,1\}$, for all $j \in \{1,\dots,i | \ell_{j,3-q} \neq u_2\}$, i.e. the set I_2 cannot contain later processed items, such that the bound (28) can be computed as in case (*f*).

Case (h)

The lower bounding procedure related to (*h*) relies on the bound (26), which was derived from (16), and on the VP-MDFF of Proposition 9. In our implementation, its complexity is $\mathcal{O}(n^3)$. The details of our implementation are given in Algorithm 3. The function g used in this VP-MDFF is the one described in Proposition 10. After sorting the items, we choose $\mathcal{O}(n)$ times values of the variable k, namely all $k \in \{1/2, \ell_{iq} - \varepsilon : i \in \{1,\dots,n\} \nexists \ 1/3 < \ell_{iq} < 1/2\}$, with $q \in \{1,2\}$ and $\varepsilon > 0$ being an enough small constant. If $q = 2$ then the function f of Proposition 9 is taken literally, while for $q = 1$ the first and second dimension are exchanged. For each value of k, the function g is obtained optimally according to Proposition 10. For this purpose, we use a vector of length $\mathcal{O}(n)$ to represent the function g. This vector is initialized each time with complexity $\mathcal{O}(n^2)$ in the loop `for` $i_2 := n$ `downto 1 do ...` in Algorithm 3. For given values $k \in (1/3, 1/2]$ and $q \in \{1,2\}$, we check first if an item $\mathbf{l}_{i_2}$ is assigned a value 0, 1 or 1/2 in the function f of Proposition 9. If that is not the case, i.e. the function g plays a role in the evaluation of $f(\mathbf{l}_{i_2})$, then the required data with respect to that item in (26) are collected in a vector, which is sorted in non-decreasing order of the arguments of g. The sorting by insertion requires the complexity $\mathcal{O}(n^2)$, and since it is executed up to $n+1$ times, the total complexity of our implementation is $\mathcal{O}(n^3)$. The simplifications according to the case distinction in the proof of Proposition 10 need only the effort $\mathcal{O}(n)$. That is done in Algorithm 3 in the conditional block `if` $(i_3 > 1)$ `then ...` in the same sequence as in the proof of Proposition 10. At the end, only the number c in the expression (26) remains as bound. Hence, the total complexity of computing the bound through this procedure is $\mathcal{O}(n^3)$. Finally, the lower procedure described in Algorithm 3 is applied for $q = 1$ and $q = 2$, and the best bound is chosen.

Algorithm 3 (*Lower bounding procedure for the mD-VPP based on the VP-MDFF of case (h)*).

```
Input:  n items with sizes l_i ∈ [0,1]^2 and order demands b_i ∈ ℕ \ {0};
        q: parameter with q ∈ {1,2}; ε: small constant with ε > 0;
Output: z: lower bound based on (16) and on the VP-MDFF of case (h) for the given q;
Auxiliary variables: s ∈ ℤ; k, c, x1, x2 ∈ ℝ; start ∈ {0,1}; i ∈ ℕ^5; y ∈ ℝ^n; a ∈ ℤ^n;
Sort the items in non-decreasing order of ε × ℓ_{jq} − |0.5 − ℓ_{jq}|;
z := −1; k := 1/2; c := 0; start := 1;
for i1 := n downto 1 do
    if (start = 0) then
        if (ℓ_{i1 q} ≥ 1/2) then goto label1;
        ;
        k := ℓ_{i1 q} − ε;
    end
    i3 := 1; c := 0; start := 0;
    for i2 := n downto 1 do
        if (ℓ_{i2,3−q} > 0) then
            x1 := ℓ_{i2,3−q}; x2 := ℓ_{i2 q}; s := b_{i2};
            if (x1 = 1) or (x2 > 1 − k) then c := c + s;
            ;
            else if (l_{i2} = ½w) then c := c + s/2;
            ;
            else if (x2 ≥ k) then
                i4 := i3;
                if (x2 < 1/2) or (x2 = 1/2 and x1 < 1/2) then {c := c + s; x1 := 1 − x1; s := −s;}
                ;
                repeat i4 := i4 − 1; until (i4 = 0) or (y_{i4} ≤ x1);
                ;
                if (i4 = 0) or (y_{i4} < x1) then
                    i4 := i4 + 1;
                    for i5 := i3 − 1 downto i4 do {y_{i5+1} := y_{i5}; a_{i5+1} := a_{i5};}
                    ;
                    y_{i4} := x1; a_{i4} := 0; i3 := i3 + 1;
                end
                a_{i4} := a_{i4} + s;
            end
        end
    end
    if (i3 > 1) then
        repeat
            i2 := i2 + 1;
            if (i2 = i3) then break;
            ;
            if (a_{i2−1} > 0) then {a_{i2} := a_{i2} + a_{i2−1}; a_{i2−1} := 0;}
            ;
        until false;
        if (a_{i2−1} > 0) then
            i2 := i2 − 1; c := c + a_{i2};
            if (i2 = 1) then goto label1;
            ;
        end
        i3 := 1;
        repeat
            if (y_{i3} + y_{i2−1} < 1) then i3 := i3 + 1;
            ;
            else if (i2 − 1 = i3) then {c := c + a_{i3}/2; break;}
            ;
            else if (y_{i3} + y_{i2−2} ≥ 1) then {i2 := i2 − 1; a_{i2−1} := a_{i2−1} + a_{i2};}
            ;
            else if (a_{i3} > a_{i2−1}) then {c := c + a_{i3}; a_{i2−1} := a_{i2−1} − a_{i3}; i3 := i3 + 1;}
            ;
            else {i2 := i2 − 1; c := c + a_{i2}; i3 := i3 + 1;}
            ;
        until (i3 ≥ i2);
    end
label1: if (c > z) then z := c;
    ;
    if (ℓ_{i1 q} ≤ 1/3) or (ℓ_{i1 q} ≥ 2/3) then break;
    ;
end
```

Case (i)

A lower bounding procedure based on (16) and on the VP-DFF of Proposition 11 with forced symmetry according to Proposition 3 requires first to choose two items as vectors **s** and **t**. Since there are $\mathcal{O}(n^2)$ possibilities, we have to make $\mathcal{O}(n^2)$ calls to a subroutine to obtain the optimal values for a_1, a_2, u_1, u_2 on which this VP-DFF relies. In this subroutine, we compute first the optimal dual values u_1, u_2 for the two items of sizes **s** and **t**. For this step, we need to solve a linear optimization problem. After that, we compute the values a_1 and a_2. The complexity of this embedded subroutine is comparable with the one of dynamic optimization and can therefore be high. Without additional restrictions, this complexity will not be simply a polynomial in n, because a_2 could be as large as $\max_{i\in\{1,\ldots,n\}}\{1/\max_{d\in\{1,\ldots,m\}}\ell_{id}\}$. If all input data are integer with the maximum M, i.e. the container is not normalized to the unit cube, then this ratio can reach M, and hence the total complexity becomes $\mathcal{O}(Mn^2)$, which is pseudo-polynomial. Nevertheless, since we have to consider only two items to evaluate this bound, the complexity may be lower than if n pieces would have been considered.

Table 1
Computational results for the lower bounding procedures: instance set I (part 1).

Inst.	n	z_C	z_{CG}	Average lower bound									Best bound									Equals z_{CG}								
				(a)	(b)	(c)	(d)	(e)	(f)	(g)	(h)	(i)	(a)	(b)	(c)	(d)	(e)	(f)	(g)	(h)	(i)	(a)	(b)	(c)	(d)	(e)	(f)	(g)	(h)	(i)
1	25	6.9	6.9	6.9	6.9	6.9	6.9	6.9	6.9	6.9	6.9	6.9	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
	50	13.5	13.5	13.5	13.5	13.5	13.5	13.5	13.5	13.5	13.5	13.5	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
	100	25.5	25.5	25.5	25.5	25.5	25.5	25.5	25.5	25.5	25.5	25.5	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
	200	50.3	50.3	50.3	50.3	50.3	50.3	50.3	50.3	50.3	50.3	50.3	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
2	25	11.9	14.2	13.8	12.9	13.6	14.0	14.0	13.0	14.0	14.0	14.2	6	3	5	8	8	0	8	8	10	6	3	5	8	8	0	8	8	10
	50	26.9	31.5	31.3	28.1	30.3	31.0	31.5	28.4	31.4	31.4	31.1	8	0	0	6	10	0	9	9	6	8	0	0	6	10	0	9	9	6
	100	51.3	57.4	53.9	52.2	53.2	54.5	54.8	54.2	56.0	55.9	55.5	0	0	0	0	1	0	10	9	5	0	0	0	0	0	0	0	0	0
	200	99.8	113.5	104.5	102.1	104.0	105.2	104.8	105.1	108.7	107.6	108.6	0	0	0	0	0	0	8	5	7	0	0	0	0	0	0	0	0	0
3	25	12.3	14.2	13.8	13.3	13.6	13.8	13.8	12.9	14.0	14.0	14.2	6	3	4	6	6	0	8	8	10	6	3	4	6	6	0	8	8	10
	50	26.5	31.5	31.3	27.9	29.8	30.8	31.3	27.2	31.4	31.4	31.1	8	0	0	3	8	0	9	9	6	8	0	0	3	8	0	9	9	6
	100	50.5	56.9	52.9	51.7	52.1	53.4	53.6	53.2	55.4	55.9	55.3	0	0	0	0	0	0	5	10	4	0	0	0	0	0	0	0	4	0
	200	100.3	113.5	104.4	101.5	103.5	104.7	105.3	104.6	108.2	108.2	108.6	0	0	0	0	0	0	5	5	7	0	0	0	0	0	0	0	0	0
4	25	3.3	3.3	3.3	3.3	3.3	3.3	3.3	3.3	3.3	3.3	3.3	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
	50	7.0	7.0	7.0	7.0	7.0	7.0	7.0	7.0	7.0	7.0	7.0	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
	100	13.0	13.0	13.0	13.0	13.0	13.0	13.0	13.0	13.0	13.0	13.0	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
	200	25.3	25.3	25.3	25.3	25.3	25.3	25.3	25.3	25.3	25.3	25.3	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
5	25	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
	50	4.0	4.0	4.0	4.0	4.0	4.0	4.0	4.0	4.0	4.0	4.0	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
	100	7.0	7.0	7.0	7.0	7.0	7.0	7.0	7.0	7.0	7.0	7.0	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
	200	13.0	13.0	13.0	13.0	13.0	13.0	13.0	13.0	13.0	13.0	13.0	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10

Table 2
Computational results for the lower bounding procedures: instance set I (part 2).

Inst.	n	z_C	z_{CG}	Average lower bound									Best bound									Equals z_{CG}								
				(a)	(b)	(c)	(d)	(e)	(f)	(g)	(h)	(i)	(a)	(b)	(c)	(d)	(e)	(f)	(g)	(h)	(i)	(a)	(b)	(c)	(d)	(e)	(f)	(g)	(h)	(i)
6	25	9.9	10.1	9.7	9.8	9.9	9.8	9.7	9.9	9.9	7.8	9.5	8	9	10	9	8	10	10	0	6	6	7	8	7	6	8	8	0	5
	50	21.3	21.5	21.0	20.9	21.3	20.9	20.9	21.3	21.3	18.5	20.1	7	6	10	6	6	10	10	0	1	5	4	8	4	4	8	8	0	0
	100	40.5	41.0	40.0	40.5	40.5	40.1	40.2	40.5	40.5	33.5	38.0	5	10	10	6	7	10	10	0	0	0	5	5	1	2	5	5	0	0
	200	80.3	81.1	78.8	80.1	80.3	79.6	79.5	80.3	80.3	68.2	75.5	1	8	10	3	3	10	10	0	0	0	1	2	0	0	2	2	0	0
7	25	9.5	9.6	9.5	9.5	9.5	9.4	9.5	9.5	9.5	7.6	9.3	10	10	10	9	10	10	10	0	8	9	9	9	8	9	9	9	0	7
	50	19.6	19.7	19.5	19.6	19.6	19.5	19.5	19.6	19.6	16.6	19.3	9	10	10	9	9	10	10	0	7	8	9	9	8	8	9	9	0	6
	100	39.8	40.2	39.5	39.8	39.8	39.7	39.4	39.8	39.8	33.5	38.7	7	10	10	9	6	10	10	0	1	3	6	6	5	3	6	6	0	0
	200	79.9	80.1	79.2	79.8	79.9	79.3	79.1	79.9	79.9	67.3	78.0	4	9	10	4	4	10	10	0	0	3	7	8	2	3	8	8	0	0
8	25	10.5	13.0	12.8	13.0	13.0	13.0	13.0	10.5	10.5	9.1	9.0	9	10	10	10	10	0	0	0	0	9	10	10	10	10	0	0	0	0
	50	21.1	25.0	25.0	24.8	25.0	24.6	24.5	21.1	21.1	18.7	17.7	10	8	10	9	8	0	0	0	0	10	8	10	9	8	0	0	0	0
	100	39.8	50.0	48.5	50.0	49.9	49.0	50.0	39.8	39.8	33.3	31.3	8	10	9	9	10	0	0	0	0	8	10	9	9	10	0	0	0	0
	200	79.9	100.0	93.3	95.5	99.9	97.8	100.0	79.9	79.9	65.2	63.0	6	4	9	8	10	0	0	0	0	6	4	9	8	10	0	0	0	0
9	25	6.3	7.3	6.3	6.3	6.3	6.3	6.3	6.3	6.3	3.6	5.6	10	10	10	10	10	10	10	0	3	0	0	0	0	0	0	0	0	0
	50	13.5	14.5	13.5	13.5	13.5	13.3	13.4	13.5	13.5	7.7	11.7	10	10	10	8	9	10	10	0	0	0	0	0	0	0	0	0	0	0
	100	25.7	26.7	25.2	25.7	25.7	25.6	25.5	25.7	25.7	15.1	21.7	5	10	10	9	8	10	10	0	0	0	0	0	0	0	0	0	0	0
	200	50.3	50.3	48.6	50.3	50.3	49.1	49.5	50.3	50.3	25.9	41.9	5	10	10	5	5	10	10	0	0	5	10	10	5	5	10	10	0	0
10	24	8.0	8.0	8.0	8.0	8.0	8.0	8.0	8.0	8.0	8.0	8.0	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
	51	17.0	17.0	17.0	17.0	17.0	17.0	17.0	17.0	17.0	17.0	17.0	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
	99	33.0	33.0	33.0	33.0	33.0	33.0	33.0	33.0	33.0	33.0	33.0	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
	201	67.0	67.0	67.0	67.0	67.0	67.0	67.0	67.0	67.0	67.0	67.0	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
Total													302	310	327	306	316	280	342	223	241	260	256	272	259	270	225	259	198	210

Table 3
Computational results for the lower bounding procedures: instance set II (part 1).

Inst.	n	z_C	z_{CG}	Average lower bound									Best bound									Equals z_{CG}								
				(a)	(b)	(c)	(d)	(e)	(f)	(g)	(h)	(i)	(a)	(b)	(c)	(d)	(e)	(f)	(g)	(h)	(i)	(a)	(b)	(c)	(d)	(e)	(f)	(g)	(h)	(i)
1	20	11.9	14.4	13.6	12.6	13.6	13.6	13.4	13.5	14.0	14.0	14.3	4	2	4	4	3	4	7	7	10	3	1	3	3	2	3	6	6	9
	40	23.2	27.1	25.6	24.5	25.4	25.6	25.4	25.9	26.5	26.5	26.9	1	0	0	0	0	4	6	7	10	0	0	0	0	0	3	5	7	8
	60	34.6	40.2	38.6	36.5	38.1	38.8	38.3	38.7	39.2	39.3	40.1	2	0	1	2	1	1	4	4	10	1	0	0	1	1	0	3	4	9
	80	46.9	55.0	52.8	50.8	52.4	53.0	52.3	52.7	53.3	54.2	54.9	1	1	1	2	1	0	2	6	10	1	1	1	2	1	0	1	5	9
	100	56.7	64.3	61.6	59.0	61.2	62.1	61.5	62.2	63.0	63.1	64.1	0	0	0	0	0	1	4	6	10	0	0	0	0	0	1	2	4	8
2	20	9.7	11.5	10.9	10.0	10.8	11.0	11.0	10.6	11.2	10.8	11.3	6	3	6	7	7	5	9	5	10	4	3	5	5	5	3	7	4	8
	40	19.2	21.4	20.5	19.7	20.5	20.5	20.6	20.2	20.7	20.6	21.1	5	2	5	5	6	4	7	5	10	2	0	2	2	3	1	4	3	7
	60	28.5	30.7	30.3	29.6	29.9	30.4	30.2	29.5	30.3	30.1	30.5	8	3	5	9	7	3	8	6	10	6	2	4	7	6	2	7	5	8
	80	38.8	42.4	41.1	40.0	40.7	41.2	41.2	39.9	41.4	41.1	41.5	6	2	3	6	7	0	8	6	9	3	0	0	3	3	0	4	3	4
	100	46.6	49.6	48.4	47.6	47.9	48.5	48.6	47.3	48.6	48.7	49.0	3	0	1	3	4	1	5	5	8	2	0	1	2	2	0	2	3	5
3	20	11.2	13.6	12.4	11.5	12.3	12.3	12.4	12.2	13.2	12.6	13.4	3	1	2	2	3	2	8	5	10	2	1	2	2	2	2	6	4	8
	40	21.5	25.5	23.2	22.5	23.2	23.3	23.1	23.2	24.6	23.7	24.6	1	0	2	1	1	1	9	3	9	0	0	0	0	0	1	4	2	3
	60	32.0	37.3	34.9	33.7	34.7	35.0	35.0	35.0	36.1	35.0	36.6	1	1	1	2	2	3	6	3	9	0	0	0	0	0	1	2	2	4
	80	43.4	51.3	48.1	46.7	48.0	48.3	47.9	47.5	50.0	49.5	50.2	1	1	1	1	1	0	7	7	9	0	0	0	0	0	0	3	2	3
	100	52.3	59.7	55.7	53.9	54.9	55.7	55.3	55.6	58.0	56.5	58.3	0	0	0	0	0	0	6	3	9	0	0	0	0	0	0	0	0	0
4	20	14.8	17.1	16.7	16.4	16.7	16.6	16.7	15.7	17.0	16.7	17.0	7	5	7	7	7	1	9	6	9	7	5	7	7	7	1	9	6	9
	40	29.2	34.6	34.1	33.4	34.0	34.1	33.9	31.9	34.4	34.3	34.4	6	2	5	6	5	1	9	8	9	6	2	5	6	5	1	8	7	8
	60	43.0	52.6	50.7	49.9	50.5	50.6	50.5	46.5	51.9	51.7	51.7	1	1	0	1	1	0	8	7	7	1	1	0	1	1	0	3	4	4
	80	57.4	68.7	67.1	66.4	67.0	67.3	67.0	63.2	68.3	68.3	68.0	2	1	2	2	1	0	9	9	7	1	1	1	2	1	0	7	7	5
	100	72.1	85.1	83.5	82.7	83.0	83.5	83.6	77.5	85.0	85.0	84.4	2	1	1	1	2	0	10	10	5	2	1	1	1	2	0	9	9	5
5	20	14.7	17.9	17.3	14.1	17.3	17.2	17.3	15.9	17.6	17.3	17.9	5	0	5	5	5	0	8	6	10	5	0	5	5	5	0	8	6	10
	40	28.4	34.6	33.8	28.0	33.6	33.7	33.6	31.4	34.5	34.0	34.3	4	0	2	3	2	0	9	6	7	4	0	2	3	2	0	9	6	7
	60	43.6	52.4	51.2	42.3	50.9	51.2	51.2	47.3	52.2	52.0	51.9	3	0	2	3	3	0	8	6	5	3	0	2	3	3	0	8	6	5
	80	57.4	68.2	67.0	56.4	66.5	67.1	67.0	62.9	67.9	67.6	67.8	2	0	2	3	2	0	8	5	8	2	0	2	2	2	0	7	4	7
	100	71.5	85.5	83.9	69.8	82.8	84.1	83.9	76.7	85.1	85.0	84.6	3	0	0	3	3	0	9	8	5	1	0	0	1	1	0	6	5	2

Table 4
Computational results for the lower bounding procedures: instance set II (part 2).

Inst.	n	z_C	z_{CG}	Average lower bound									Best bound									Equals z_{CG}								
				(a)	(b)	(c)	(d)	(e)	(f)	(g)	(h)	(i)	(a)	(b)	(c)	(d)	(e)	(f)	(g)	(h)	(i)	(a)	(b)	(c)	(d)	(e)	(f)	(g)	(h)	(i)
6	20	14.2	18.0	17.5	16.7	17.5	17.6	17.6	17.7	18.0	17.7	18.0	6	2	6	7	7	7	10	7	10	6	2	6	7	7	7	10	7	10
	40	28.2	36.4	35.8	33.8	35.4	35.7	35.5	35.8	36.0	35.4	36.1	6	0	3	5	3	6	8	4	9	5	0	3	4	2	5	7	3	8
	60	42.1	54.9	53.4	51.3	53.3	53.1	53.2	53.4	53.9	53.3	54.5	2	0	1	0	1	3	4	3	10	0	0	0	0	0	2	3	3	7
	80	55.5	71.1	69.8	67.6	69.5	69.9	69.8	70.0	70.1	69.3	70.8	2	1	2	2	2	2	2	2	9	2	1	2	2	2	2	2	2	7
	100	68.4	87.8	85.9	81.8	86.0	86.0	85.5	86.0	86.3	85.9	87.3	1	0	1	1	1	1	3	2	8	1	0	1	1	1	1	3	1	7
7	20	8.7	9.4	8.5	8.6	8.7	8.7	8.7	8.7	9.0	7.9	8.8	5	6	7	7	6	7	9	0	7	2	3	4	4	3	4	6	0	4
	40	16.9	17.9	16.8	17.0	17.1	16.9	16.9	17.0	17.7	16.3	16.9	3	5	5	4	3	4	10	3	3	2	4	3	3	2	3	8	2	2
	60	23.8	24.6	23.5	23.7	23.9	23.4	23.2	23.8	24.0	21.7	23.1	5	7	9	4	3	8	10	0	2	2	4	5	3	2	5	5	0	0
	80	31.2	32.1	30.5	31.0	31.2	30.7	30.8	31.2	31.5	27.7	30.0	2	6	7	3	4	7	10	0	0	1	2	3	1	1	3	4	0	0
	100	38.9	40.2	38.1	38.6	38.8	38.4	38.4	38.9	39.7	35.9	38.2	1	3	4	2	3	4	10	0	0	1	1	2	1	1	2	5	0	0
Total													110	56	103	113	107	80	259	170	273	78	35	72	84	75	53	183	132	200

Except for the last function, the complexity of these lower bounding procedures is polynomial. This allows to compute lower bounds much faster than by column generation. As a comparison, four of the five lower bounding procedures discussed in Caprara and Toth (2001) have a complexity that ranges from $\mathcal{O}(n)$ to $\mathcal{O}(n^4 ln\ n)$ (their fifth bound relies on column generation), while the complexity of eight of our nine procedures (the ninth is the last one that is pseudopolynomial) ranges from $\mathcal{O}(n)$ to $\mathcal{O}(n^3)$. In the next section, we report on computational experiments that illustrate the efficiency of our procedures.

8. Computational experiments

In this section, we report on two sets of computational experiments that we conducted to evaluate both the quality of the bounds and the performance of the lower bounding procedures related to the VP-DFFs described above. As shown in Section 5, the best bound that a VP-DFF can generate is at most equal to the bound z_{CG} of the LP relaxation of (8)–(10). While this bound is well-known to be strong, computing it by solving (8)–(10) through column generation can be very time consuming. Our first set of computational experiments shows that the VP-DFFs can approximate the bound z_{CG} very efficiently. In practice, it takes usually a fraction of seconds to compute the bounds using VP-DFFs. In our second set of experiments, we focus on the impact that the lower bounding procedures described above may have on the convergence of a branch-and-bound based algorithm for the 2D-VPP.

We conducted our experiments on two sets of instances of the 2D-VPP. The first set (instance set I) was proposed in Caprara and Toth (2001) for the vector-packing problem, the second (instance set II) was proposed in Berkey and Wang (1987) and Martello and Vigo (1998) for the two-dimensional bin-packing problem. Within each group of the first set, the instances are further divided into 4 sets of 10 instances each according to the number of items which belongs to {25,50,100,200}. In the second set of instances, each set contains 50 instances divided in 5 sets of 10 instances with a number of items per instance in the set {20,40,60,80,100}. This set originally contains 10 groups of instances. However, for this set, we only kept the groups where the bound of Spieksma is strictly smaller than the column generation lower bound. Therefore, we only report our results for sets 1, 3, 5, 7, 8, 9, 10 of Berkey and Wang (1987) and Martello and Vigo (1998). The indices of these groups of instances have been updated accordingly. All the experiments were conducted on a PC with an Intel Core i3 CPU with 2.27 gigahertz and 4 gigabytes of RAM.

For some of our VP-DFFs, we used the 1-dimensional DFF $f_{CCM,1}$ proposed in Carlier et al. (2007) and described in Section 3. The results for our first set of experiments are reported in Tables 1–4. The entries in these tables have the following meaning: *Inst.* stands for the group of instances, n for the number of items, z_C represents the lower bound given by the LP relaxation of (2)–(6) (equivalent to the bound of Spieksma (1994)) and z_{CG} the lower bound given by the LP relaxation of (8)–(10). The columns (a)–(i) are associated respectively to the cases (a)–(i) introduced in Section 7.

In Tables 1–4, we report on the quality of the lower bounds provided by our VP-DFFs. In these tables, the results related to the VP-DFFs are divided in three parts. In the first part, we give the average lower bound obtained with the corresponding VP-DFF. The second part (*Best bound*) indicates the number of times the VP-DFFs give the best bound among all the VP-DFFs from cases (a)–(i). The last part of these tables (*Equals* z_{CG}) shows the number of times the bound given by the VP-DFF is equal to the column generation bound z_{CG}.

From Tables 1 and 2, we can observe that the average lower bounds provided by the VP-DFFs are very near from the average

Table 5
Computational results for the branch-and-price algorithm: instance set I.

Inst.	n	*BP*		(*a*)		(*b*)		(*c*)		(*d*)		(*e*)		(*f*)		(*g*)		(*h*)		(*i*)	
		opt.	*t*	*opt.*	*t*	*opt.*	*t*	*opt.*	*t*	*opt.*	*t*	*opt.*	*t*	*opt.*	*t*	*opt.*	*t*	*opt.*	*t*	*opt.*	*t*
1	25	10	33.9	10	0.5	10	0.5	10	0.6	10	0.6	10	0.7	10	0.6	10	0.6	10	0.6	10	0.6
	50	8	343.0	10	19.2	10	19.5	10	19.2	10	20.7	10	24.8	10	19.1	10	19.1	10	19.2	10	19.8
2	25	10	0.4	10	0.3	10	0.4	10	0.4	10	0.4	10	0.6	10	0.4	10	0.3	10	0.3	10	0.2
	50	10	2.5	10	1.4	10	2.6	10	2.4	10	1.7	10	3.6	10	2.5	10	1.3	10	1.4	10	1.8
	100	10	12.3	10	11.3	10	11.4	10	11.3	10	11.9	10	12.9	10	11.4	10	11.5	10	11.2	10	11.7
	200	10	422.2	10	394.4	10	394.7	10	393.3	10	399.5	10	403.2	10	393.4	10	390.3	10	393.8	10	402.2
3	25	10	0.5	10	0.3	10	0.5	10	0.4	10	0.4	10	0.6	10	0.4	10	0.3	10	0.4	10	0.2
	50	10	1.8	10	1.3	10	1.7	10	1.7	10	1.5	10	2.2	10	1.7	10	0.9	10	0.9	10	1.3
	100	10	3.7	10	3.5	10	3.5	10	3.5	10	3.7	10	4.1	10	3.4	10	3.4	10	3.5	10	3.6
	200	10	70.1	10	61.4	10	62.3	10	61.9	10	64.6	10	66.5	10	62.5	10	63.5	10	62.2	10	66.9
4	25	7	75.3	9	2.7	9	2.4	9	2.7	9	2.6	9	3.1	9	2.6	9	2.3	9	2.4	9	2.6
	50	6	371.5	10	0.7	10	0.7	10	0.7	10	0.8	10	0.9	10	0.7	10	0.7	10	0.7	10	0.8
5	25	0	–	10	0.1	10	0.0	10	0.0	10	0.0	10	0.1	10	0.0	10	0.1	10	0.0	10	0.1
	50	0	–	10	0.1	10	0.1	10	0.0	10	0.1	10	0.2	10	0.1	10	0.1	10	0.1	10	0.1
	100	0	–	5	0.3	5	0.2	5	0.3	5	0.2	5	0.6	5	0.3	5	0.3	5	0.2	5	0.4
6	25	9	6.1	9	3.4	9	3.0	9	3.1	9	3.8	9	4.4	9	3.2	9	3.4	9	3.5	9	3.5
	50	9	68.1	10	22.9	9	15.2	10	20.2	10	23.0	10	30.6	10	22.7	10	20.8	9	17.5	9	18.4
	100	8	552.2	9	313.0	8	279.0	7	208.1	7	232.0	7	235.7	7	209.8	7	214.2	6	166.3	6	166.2
7	25	10	8.1	10	1.7	10	2.0	10	1.6	10	2.1	10	2.5	10	2.0	10	2.0	10	1.9	10	1.8
	50	10	87.2	10	31.1	10	30.9	10	29.9	10	32.9	10	38.2	10	32.3	10	30.7	10	33.4	10	30.3
	100	6	607.9	8	356.8	9	370.8	8	357.6	8	360.5	7	325.9	7	319.7	6	269.8	6	271.4	7	326.1
8	25	10	3.2	10	0.5	10	0.3	10	0.3	10	0.6	10	2.7	10	3.2	10	3.1	10	3.4	10	3.2
	50	10	5.7	10	3.2	10	3.3	10	3.0	10	3.6	10	6.2	10	5.5	10	5.5	10	5.7	10	5.5
	100	10	21.8	10	12.8	10	13.5	10	12.1	10	14.4	10	23.9	10	22.5	10	21.9	10	21.8	10	23.2
	200	10	140.0	10	80.0	10	87.8	10	74.9	10	81.7	10	146.4	10	139.5	10	139.3	10	140.9	10	145.6
9	25	8	24.5	10	23.9	10	19.9	10	19.4	10	21.8	10	25.4	10	22.4	10	21.0	9	23.3	9	22.3
	50	9	416.6	10	379.2	10	355.2	8	371.5	10	360.3	10	393.6	10	330.6	10	330.2	10	388.8	9	394.3
10	24	10	1.5	10	1.1	10	1.5	10	0.7	10	1.3	10	1.2	10	1.0	10	0.7	10	1.1	10	1.0
	51	10	5.5	10	5.0	10	10.9	10	17.0	10	4.9	10	4.9	10	12.9	10	4.8	10	4.8	10	4.8
	99	9	30.1	9	30.9	9	36.4	9	28.6	9	30.0	9	29.4	9	31.4	9	34.5	9	29.2	9	30.3

column generation bound. The worst case is for the group 9 of instances with $n = 25$. For this case, the relative gap between the bounds based on VP-DFFs is at least 14%. Note that, for these instances, z_C and z_{CG} are small values. Furthermore, the difference between z_C and z_{CG} for all the instances of this group is always equal to 1. For all the other instances of this set, one of our VP-DFFs gives always a lower bound whose relative gap compared to the column generation bound is at most 7%. For the instance sets 1, 4, 5, and 10, the lower bounds provided by the VP-DFFs are always equal to the column generation bound z_{CG}, which is in turn always equal to z_C.

The performance of our VP-DFFs is better illustrated in the final part of Table 1, where we report on the number of times the bounds obtained with the VP-DFFs are as good as the bounds given by the LP relaxation of the column generation model. Almost all the VP-DFFs provide a bound equal to z_{CG} for a significant number of cases. In fact, for 309 of the 400 instances of set I, one of our VP-DFFs always gives a lower bound that is equal to z_{CG}. For the other instances, the average value of the differences between z_{CG} and the best bounds obtained with a VP-DFF is equal to only 1.87. Note that the bounds obtained with VP-DFFs are computed usually in a few milliseconds, i.e. much more efficiently than using column generation. Furthermore, it is interesting to note that the only group of instances (apart from the sets 1, 4, 5, and 10) where the bound z_{CG} was reached for all the instances with at least one of our VP-DFFs is the group 8, for which the difference between z_C and z_{CG} is large. The best results are obtained for groups 2, 3, and 8. For the two first, the instances were generated uniformly on both dimensions: functions denoted (*g*), (*h*) and (*i*) lead to the best results for them. For group 8, where many items are large on exactly one dimension, functions (*a*) to (*e*) perform better.

Tables 3 and 4 illustrate the results obtained with the instance set II. For each group, there is always one VP-DFF that provides a lower bound with a relative gap of at most 5% compared to z_{CG}. For this set of instances, the best results are achieved with the VP-DFF of Proposition 11 (and the corresponding lower bounding procedure related to the case (*i*) introduced in Section 7). In 273 of 350 instances, the best bound is given by this VP-DFF, while for 200 of these instances, this bound is equal to z_{CG}. The VP-DFF of case (*i*) returns the best results for all instances of groups 1, 2, 3, and 6. For groups 4, 5, and 7, the best results on average are obtained by VP-DFF (*g*). In the three groups where (*g*) performs better, the number of items that are large on both dimensions is small (10% on average). For groups 1, 2, 3, and 6, this ratio is much larger, which hints that (*i*) is well suited to such instances.

For 255 of the 350 instances in set II, there is always one VP-DFF that yields a bound equal to z_{CG}. For the remaining instances, the average value of the differences between the bounds provided by the VP-DFFs and z_{CG} is equal to only 1.2. These results confirm the previous ones obtained for the instance set I. For the majority of the cases, the VP-DFFs presented in this paper lead to lower bounds that are as good as the column generation bound. The advantage is that they require only a very small fraction of the computing time that is needed to compute the column generation bound. For the other instances, the lower bounds achieved with

Table 6
Computational results for the branch-and-price algorithm: instance set II (part 1).

Inst.	n	*BP*		(*a*)		(*b*)		(*c*)		(*d*)		(*e*)		(*f*)		(*g*)		(*h*)		(*i*)	
		opt.	*t*	*opt.*	*t*	*opt.*	*t*	*opt.*	*t*	*opt.*	*t*	*opt.*	*t*	*opt.*	*t*	*opt.*	*t*	*opt.*	*t*	*opt.*	*t*
1	20	10	0.3	10	0.3	10	0.3	10	0.3	10	0.4	10	0.3	10	0.3	10	0.3	10	0.3	10	0.3
	40	10	0.9	10	0.7	10	0.8	10	0.7	10	0.9	10	0.9	10	0.8	10	0.7	10	0.7	10	0.7
	60	10	4.9	10	3.7	10	4.2	10	4.3	10	4.1	10	4.9	10	4.6	10	2.4	10	4.2	10	2.8
	80	10	3.0	10	1.7	10	2.8	10	2.8	10	1.8	10	3.1	10	3.0	10	1.6	10	1.5	10	1.4
	100	10	22.8	10	21.2	10	20.4	10	20.0	10	22.3	10	23.0	10	20.8	10	17.8	10	14.2	10	11.2
2	20	10	0.2	10	0.2	10	0.2	10	0.2	10	0.2	10	0.3	10	0.2	10	0.2	10	0.2	10	0.2
	40	10	4.2	10	3.6	10	3.7	10	3.5	10	4.0	10	4.4	10	3.5	10	3.2	10	3.4	10	2.9
	60	10	14.7	10	8.7	10	10.5	10	9.9	10	9.5	10	12.9	10	11.3	10	10.7	10	10.1	10	5.5
	80	10	23.5	10	19.8	10	23.5	10	23.0	10	21.2	10	24.5	10	23.6	10	19.8	10	20.1	10	17.2
	100	10	29.4	10	27.9	10	28.2	10	28.0	10	28.4	10	29.8	10	29.6	10	29.8	10	28.0	10	24.9
3	20	10	0.2	10	0.2	10	0.2	10	0.2	10	0.2	10	0.3	10	0.2	10	0.2	10	0.2	10	0.2
	40	10	1.8	10	1.6	10	1.6	10	1.7	10	1.9	10	2.1	10	1.7	10	1.7	10	1.7	10	1.5
	60	10	8.2	10	7.9	10	8.1	10	7.8	10	8.6	10	9.3	10	8.1	10	7.9	10	7.8	10	7.5
	80	10	7.2	10	7.3	10	8.0	10	7.9	10	8.3	10	9.0	10	7.5	10	7.7	10	7.4	10	6.6
	100	10	39.6	10	35.5	10	37.0	10	36.1	10	38.2	10	38.9	10	36.8	10	37.5	10	36.3	10	34.7
4	20	10	0.1	10	0.1	10	0.1	10	0.1	10	0.2	10	0.2	10	0.1	10	0.1	10	0.1	10	0.1
	40	10	0.2	10	0.2	10	0.2	10	0.2	10	0.3	10	0.4	10	0.2	10	0.2	10	0.2	10	0.2
	60	10	0.7	10	0.6	10	0.6	10	0.6	10	0.8	10	1.0	10	0.7	10	0.7	10	0.7	10	0.6
	80	10	1.6	10	1.2	10	1.3	10	1.2	10	1.5	10	1.6	10	1.3	10	1.1	10	1.0	10	1.2
	100	10	2.4	10	2.3	10	2.2	10	2.2	10	2.6	10	2.6	10	2.3	10	1.3	10	1.3	10	2.3
5	20	10	0.2	10	0.1	10	0.1	10	0.2	10	0.2	10	0.2	10	0.1	10	0.1	10	0.1	10	0.1
	40	10	0.3	10	0.2	10	0.3	10	0.3	10	0.3	10	0.5	10	0.4	10	0.3	10	0.3	10	0.3
	60	10	0.6	10	0.6	10	0.5	10	0.5	10	0.8	10	0.7	10	0.6	10	0.5	10	0.5	10	0.5
	80	10	0.7	10	0.7	10	0.7	10	0.7	10	0.9	10	0.9	10	0.7	10	0.7	10	0.7	10	0.7
	100	10	2.2	10	2.2	10	2.2	10	2.2	10	2.6	10	2.5	10	2.2	10	1.9	10	1.9	10	2.4

these VP-DFFs remain very near from these column generation bounds.

The objective of our second set of experiments is to evaluate the impact of our lower bounding procedures on the convergence of a branch-and-bound based algorithm for the 2D-VPP. We consider again the cases (*a*)–(*i*) described in the previous section. We compare in particular the impact of our procedures with an alternative approach that relies essentially on column generation. For this purpose, we implemented a branch-and-price algorithm for the 2D-VPP, and we tested two versions of this algorithm: one without the lower bounding procedures related to the cases (*a*)–(*i*) and another that resorts to these procedures. In the latter, we tested each one of the cases (*a*)–(*i*) separately. The results of these experiments are reported in Tables 5–7.

At each node of the branching tree, the LP relaxation of (8)–(10) is solved exactly using column generation. The corresponding restricted master problems are solved using CPLEX 12.2. The pricing subproblems are multidimensional knapsack problems, which are solved also up to optimality using CPLEX 12.2. At each iteration of the column generation procedure, the most attractive column is added to the restricted master problem, which is solved again until there are no more attractive columns. If the solution of the restricted master problem at a given node is not integer, and the node cannot be pruned by bound, two nodes are created using the following branching rule. Let δ_{ij} denote the sum of all the variables λ_p in (8)–(10) that include both the item i and the item j, with $i,j \in \{1,\ldots,n\}$ and $i \neq j$. Among all the fractional δ_{ij} at a given branching node, we choose the one that is closer to 0.5. Assume that a and b are the items related to the variable δ_{ij} that is selected, and let P' be the subset of patterns that include both the items a and b. The branching constraints that are enforced stand respectively as follows: $\sum_{p\in P'} \lambda_p = 0$ and $\sum_{p\in P'} \lambda_p = 1$. Note that this branching scheme works for instances whose items have unit demands $b_i = 1$, $i = 1,\ldots,n$. The branching tree is explored using a depth-first search strategy. This pure branch-and-price algorithm is denoted by *BP* in Tables 5–7.

As an alternative, we implemented and tested a version of this algorithm that resorts to the lower bounding procedures related to the cases (*a*)–(*i*) described above. At each node of the branching tree, we first apply one of these lower bounding procedures. Let z_{VPDFF} denote the resulting bound. If the node cannot be pruned immediately using z_{VPDFF}, then we enforce the constraint $\sum_{p\in P} \lambda_p \geqslant z_{VPDFF}$ in the master problem and we solve it using column generation. All the other features of the branch-and-price algorithm remain unchanged. We tested this version of the branch-and-price algorithm with each one of the cases (*a*)–(*i*) individually. In Tables 5–7, the results related to each case are given in the columns identified with the corresponding index (*a*)–(*i*) of the case.

Column *opt.* in Tables 5–7 represents the number of instances solved up to optimality and t is the corresponding average computing time (in seconds) required to solve these instances. All the experiments were conducted with a time limit of 600 seconds. In these tables, we removed the lines of the instance sets for which none of the approaches were able to find an optimal solution for at least one instance within the time limit.

The results for the instance set I reported in Table 5 shows that the branch-and-price algorithm with our lower bounding procedures clearly outperforms the pure column generation based version of this algorithm. The reduction in the total computing time required to find an optimal solution goes up to nearly 99%. These results are even more impressive for the group 5 of instances. In this case, the pure branch-and-price algorithm failed in finding the optimal solution within the time limit for all the instances, while the branch-and-price algorithm with our lower bounding procedures could find these optimal solutions in much less than

Table 7
Computational results for the branch-and-price algorithm: instance set II (part 2).

Inst.	n	*BP*		(*a*)		(*b*)		(*c*)		(*d*)		(*e*)		(*f*)		(*g*)		(*h*)		(*i*)	
		opt.	*t*	*opt.*	*t*	*opt.*	*t*	*opt.*	*t*	*opt.*	*t*	*opt.*	*t*	*opt.*	*t*	*opt.*	*t*	*opt.*	*t*	*opt.*	*t*
6	20	10	0.1	10	0.1	10	0.1	10	0.1	10	0.2	10	0.2	10	0.1	10	0.1	10	0.1	10	0.1
	40	10	0.2	10	0.2	10	0.2	10	0.2	10	0.2	10	0.4	10	0.2	10	0.2	10	0.2	10	0.2
	60	10	0.3	10	0.3	10	0.3	10	0.3	10	0.4	10	0.5	10	0.3	10	0.3	10	0.3	10	0.3
	80	10	0.6	10	0.5	10	0.6	10	0.5	10	0.7	10	0.7	10	0.6	10	0.6	10	0.6	10	0.5
	100	10	1.3	10	1.1	10	1.1	10	1.1	10	1.3	10	1.5	10	1.2	10	1.1	10	1.1	10	1.3
7	20	10	2.2	10	1.6	10	1.7	10	1.6	10	1.9	10	2.2	10	1.6	10	1.1	10	1.6	10	0.9
	40	10	11.9	10	9.6	10	9.2	10	10.4	10	10.5	10	10.9	10	9.9	10	5.9	10	10.2	10	7.6
	60	9	80.7	9	69.9	9	75.0	9	73.3	9	79.8	9	76.7	9	74.0	9	61.0	9	73.9	9	72.9
	80	10	346.7	10	279.8	10	292.3	10	299.6	10	307.6	10	305.4	10	287.7	9	300.1	10	296.0	10	291.9
	100	5	493.2	5	432.8	5	428.2	5	443.0	5	451.4	5	446.5	5	441.3	6	351.8	5	443.8	5	376.5

1 second for the majority of the instances. As shown in Tables 6 and 7, most of the instances of the set II are solved efficiently with the pure branch-and-price algorithm. Nevertheless, our lower bounding procedures still improve the convergence of the algorithm and reduces the total computing time to reach an optimal solution by up to nearly 63%. Note that the branching constraints $\sum_{p\in P'}\lambda_p = 1$ are used to eventually strengthen the lower bounds provided by the VPDFFs. These constraints force the items i and j to appear in the same pattern. These items are replaced by a single one with the sum of the sizes in both dimensions, which may impact positively in the quality of the lower bound. Indeed, from this node downward the lower bounds computed with the VPDFFs are obtained from the resulting instance with these aggregated items.

9. Conclusions

In this paper, we extended the concept of DFF to the multidimensional case. The functions that were proposed apply directly to the vector packing problem and, as a consequence, we called them vector packing dual-feasible functions. We explored the properties of these functions, and we described different families of such functions. We analyzed also the complexity of the related lower bounding procedures that can be defined from these functions. To evaluate the quality and performance of our approaches, we conducted a set of computational experiments on benchmark instances. Our results show that our functions are able to generate strong lower bounds. The utility of multidimensional dual-feasible functions goes far beyond the computation of lower bounds for vector packing problems. Indeed, these functions can be used to obtain valid inequalities for general mixed integer programs, i.e. any integer programming problem with multidimensional knapsack constraints.

Acknowledgements

This work was supported by FEDER funding through the Programa Operacional Factores de Competitividade – COMPETE and by national funding through the Portuguese Science and Technology Foundation (FCT) in the scope of the project PTDC/EGE-GES/116676/2010. Additionally, the work was supported by FCT through the postdoctoral grant SFRH/BPD/45157/2008 for Jürgen Rietz.

The authors thank the anonymous referee for his/her constructive comments and suggestions, which helped improving the quality of the paper.

References

Bansal, N., Caprara, A., & Sviridenko, M. (2006). A new approximation method for set covering problems with applications to multidimensional bin packing. In *Proceedings of the 47th annual IEEE symposium on foundations of computer science (FOCS 2006)*. IEEE Computer Society.

Berkey, J. O., & Wang, P. Y. (1987). Two-dimensional finite bin-packing algorithms. *Journal of the Operational Research Society, 38*, 423–429.

Burdett, C., & Johnson, E. (1977). A subadditive approach to solve linear integer programs. *Annals of Discrete Mathematics, 1*, 117–144.

Caprara, A. (1998). Properties of some ilp formulations of a class of partitioning problems. *Discrete Applied Mathematics, 87*, 11–23.

Caprara, A., & Toth, P. (2001). Lower bounds and algorithms for the 2-dimensional vector packing problem. *Discrete Applied Mathematics, 111*, 231–262.

Carlier, J., Clautiaux, F., & Moukrim, A. (2007). New reduction procedures and lower bounds for the two-dimensional bin-packing problem with fixed orientation. *Computers and Operations Research, 34*, 2223–2250.

Carlier, J., & Néron, E. (2007). Computing redundant resources for the resource constrained project scheduling problem. *European Journal of Operational Research, 176*(3), 1452–1463.

Chang, S., Hwang, H.-C., & Park, S. (2005). A two-dimensional vector packing model for the efficient use of coil cassettes. *Computers and Operations Research, 32*, 2051–2058.

Chekuri, C., & Khanna, S. (1999). On multidimensional packing problems. In *Proceedings of the 10th annual ACM-SIAM symposium on discrete algorithms (SODA'99)*. New York: ACM Press.

Clautiaux, F., Alves, C., & Valério de Carvalho, J. M. (2010). A survey of dual-feasible and superadditive functions. *Annals of Operations Research, 179*(1), 317–342.

de la Vega, W., & Lueker, G. (1981). Bin packing can be solved within $1+\epsilon$ in linear time. *Combinatorica, 1*(4), 349–355.

Fekete, S., & Schepers, J. (2001). New classes of fast lower bounds for bin packing problems. *Mathematical Programming, 91*, 11–31.

Garey, M., Graham, R., Johnson, D., & Yao, A. (1976). Resource constrained scheduling as generalized bin packing. *Journal of Combinatorial Theory Series A, 21*, 257–298.

Garey, M., & Johnson, D. (1978). Strong NP-completeness results: Motivation, examples, and implications. *Journal of the Association for Computing Machinery, 25*, 499–508.

Johnson, D. (1973). *Near optimal bin packing algorithms*. Dissertation. Cambridge, Massachussetts: Massachussetts Institute of Technology.

Kantorovich, L. (1960). Mathematical methods of organising and planning production (from the report in russian, 1939). *Management Science, 6*, 366–422.

Kellerer, H., & Kotov, V. (2003). An approximation algorithm with absolute worst-case performance ratio 2 for two-dimensional vector packing. *Operations Research Letters, 31*, 35–41.

Lueker, G. (1983). Bin packing with items uniformly distributed over intervals [a, b]. In *Proceedings of the 24th annual symposium on foundations of computer science (FOCS 83)* (pp. 289–297). IEEE Computer Society.

Martello, S., & Vigo, D. (1998). Exact solution of the two-dimensional finite bin packing problem. *Management Science, 44*, 388–399.

Nemhauser, G., & Wolsey, L. (1998). *Integer and combinatorial optimization*. New York: Wiley.

Rietz, J., Alves, C., & Valério de Carvalho, J. (2010). Theoretical investigations on maximal dual feasible functions. *Operations Research Letters, 38*, 174–178.

Spieksma, F. (1994). A branch-and-bound algorithm for the two-dimensional vector packing problem. *Computers and Operations Research, 21*, 19–25.

Woeginger, G. (1997). There is no asymptotic PTAS for two-dimensional vector packing. *Information Processing Letters, 64*, 293–297.

Yao, A. (1980). New algorithms for bin packing. *Journal of the Association for Computing Machinery, 27*, 207–227.