Conservative scales in packing problems[1]

G. Belov,[*2] V. M. Kartak,[**3] H. Rohling,[***] G. Scheithauer[*]

[*] *Technische Universität Dresden, Institute of Numerical Mathematics*
[**] *Ufa State Aviation Technical University, Department of Numerical Mathematics*
[***] *Technische Universität Dresden, OncoRay*

Abstract

Packing problems (sometimes also called cutting problems) are combinatorial optimization problems concerned with placement of objects (items) in one or several containers. Some packing problems are special cases of several other problems such as resource-constrained scheduling, capacitated vehicle routing, etc. In this paper we consider a bounding technique for one- and higher-dimensional orthogonal packing problems, called *conservative scales (CS)* (in the scheduling terminology, *redundant resources*). CS are related to the possible structure of resource consumption: filling of a bin, distribution of the resource to the jobs, etc. In terms of packing, CS are modified item sizes such that the set of feasible packings is not reduced. In fact, every CS represents a valid inequality for a certain binary knapsack polyhedron.

CS correspond to dual variables of the set-partitioning model of a special 1D cutting-stock problem. Some CS can be constructed by *(data-dependent) dual-feasible functions ((D)DFFs)*. We discuss the relation of CS to DFFs: CS assume that at most 1 copy of each object can appear in a combination, whereas DFFs allow several copies. The literature has investigated so-called *extremal maximal DFFs (EMDFFs)* which should provide very strong CS. Analogously, we introduce the notions of *maximal CS (MCS)* and *extremal maximal CS (EMCS)* and show that EMDFFs do not necessarily produce (E)MCS. We propose fast greedy methods to "maximize" a given CS. Using the fact that EMCS define facets of the binary knapsack polyhedron, we use lifted cover inequalities as EMCS. For higher-dimensional orthogonal packing, we propose a *Sequential LP (SLP)* method over the set of CS and investigate its convergence. Numerical results are presented.

Keywords: packing, resource-constrained problems, capacitated problems, conservative scales, dual-feasible functions, knapsack inequalities, lifting, multilinear programming, linearization, sequential linear programming, convergence

1 Introduction

The introductory section states the considered problems, defines the conservative scales and describes their usage, and reviews the paper's contributions.

[1]The final publication is available at www.springerlink.com, DOI: 10.1007/s00291-011-0277-9
[2]Supported by the German Research Foundation (DFG), project BE 4433/1-1.
[3]Supported by the Russian Foundation of Basic Research (RFBR), project 10-07-91330-NNIO_a and by the German Academic Exchange Service (DAAD), programme "Michail Lomonosov II" (2010).

1.1 Problems considered

Cutting and packing (C&P) problems are a classical field of study in combinatorial optimization [WHS07]. Some packing problems represent special cases of other resource-constrained problems, e.g., resource-constrained scheduling [CN07]. The bounding technique investigated in this paper is connected to the feasible solution set of the following well-known combinatorial optimization problem:

1. *The 1D binary knapsack problem (1D BKP)* [MT90, KPP04]. It is described by the following data: knapsack capacity $W \in \mathbb{R}_+$, item sizes $w \in [0, W]^n$, and item profits $\alpha \in \mathbb{R}^n$. The goal is to select a subset of items fitting in the given capacity and having the maximal total profit. The optimal objective value is defined as follows:
$$\max\{\sum_{i=1}^{n} \alpha_i x_i : \sum_{i=1}^{n} w_i x_i \leq W,\ x \in \{0,1\}^n\}.$$

Our 'target interest' are the following problems, which are however interrelated:

2. *The 1D cutting-stock problem (1D CSP)* [CAVR11]. It operates with n *item types*. For each item type $i = \overline{1,n}$ we know its size $w_i \in \mathbb{R}_+$ and its order demand (number of items) $b_i \in \mathbb{Z}_+$. The containers are bins of capacity $W \in \mathbb{R}_+$. The task is to use the minimal number of bins to pack all the items. Without loss of generality we assume that each item fits into a bin, i.e., $w_i \leq W$ holds for each item i.

   We shall use 1D CSP as a "bar relaxation" of orthogonal packing problems (Section 2.2). For that reason we impose an additional restriction that each bin should contain at most one item of each type (*binary patterns*). This is a valid restriction because in orthogonal packing we consider exactly one copy of each item. This restriction is not relevant when solving 1D CSP on its own, cf. Section 5.3.

   The special case with unit demands ($b_i = 1,\ \forall i$) is called *the 1D bin-packing problem (1D BPP)* [GJ79, CAV10]. We shall consider this problem on its own in the experiment. Obviously, we automatically obtain only binary patterns here.

   1D CSP is the so-called *preemptive relaxation* of the resource-constrained scheduling problem, cf. [CN07].

3. *The d-Dimensional Orthogonal Packing Problem (OPP-d)* [FS04a, FSvdV07, BCP08, CJCM08]. Let be given a set of d-dimensional items (boxes) that need to be packed into a fixed container. The input data describe the container sizes $W_k \in \mathbb{R}_+$, $k = \overline{1,d}$, and the sizes of the n items $w_i^k \in \mathbb{R}_+$, $k = \overline{1,d}$ for each item $i \in I = \{1, \ldots, n\}$. OPP-$d$ is a decision problem. It asks whether all the boxes can be orthogonally packed into the container without rotations. The guillotine constraint is not considered. Without loss of generality, we assume that each item fits into the container, i.e., $w_i^k \leq W_k$ holds for each box i and dimension k.

   OPP-2 is a generalization of the decision version of 1D BPP. Given the strong $\mathcal{NP}$-hardness of the latter [GJ79], we conclude the strong $\mathcal{NP}$-completeness of OPP-d for $d \geq 2$. Moreover, OPP is polynomially equivalent to the orthogonal *strip-packing problem* [AVPT09, BM10]. OPP is related to orthogonal *bin-packing* and *knapsack problems* [cf. FS04a, CM04, BB07, PS07].

1.2 Conservative scales: definition and relation to 1D CSP

Conservative scales [FS04b] are modified item sizes of 1D BPP (CSP) such that the set of feasible packings is not reduced. This instrument allows us to obtain non-trivial bounds for 1D CSP (see below) and OPP (see the next subsection).

Definition 1. Let $(W, w) \in \mathbb{R}_+ \times \mathbb{R}_+^n$ be an instance of the 1D bin-packing problem. Vector $\widetilde{w} \in \mathbb{R}^n$ is a *conservative scale (CS) for* (W, w) if any bin filling of (W, w) stays feasible for $(W, \widetilde{w})$:

$$\forall a \in \{0,1\}^n : \qquad \sum_{i=1}^n w_i a_i \le W \quad \Rightarrow \quad \sum_{i=1}^n \widetilde{w}_i a_i \le W. \tag{1}$$

Thus, conservative scales define a new 1D BPP instance whose set of patterns is a superset of that for the original instance (if it is the same, we obtain a so-called *equivalent instance*, cf. [RST02, Dow84]). CS correspond to valid inequalities of a certain binary knapsack polyhedron, see Section 2.1 for a closer discussion. CS allow us to obtain bounds for 1D BPP and CSP as follows.

CS are the LP dual variables of the following set-partitioning model of 1D CSP, published by Kantorovich and Zalgaller [KZ51] and by Gilmore and Gomory [GG61]. The model combines a solution from pre-defined bin fillings. The filling of a bin, also called *packing pattern*, is described by a binary vector $a \in \{0,1\}^n$ satisfying

$$w^\top a \le W. \qquad \text{(Knapsack condition)}$$

As mentioned in the problem statement, in the context of orthogonal packing we allow only binary patterns, which will be further motivated in Section 2.2 ("bar relaxation"). Exceptions from this convention will be stated explicitly.

The model defines an *activity variable* x_j for each pattern a^j, $j = \overline{1,\eta}$, where η is the number of all possible patterns. The *set-partitioning model of 1D CSP* reads

$$\min\{W \cdot \textstyle\sum_{j=1}^\eta x_j : \sum_{j=1}^\eta a^j x_j = b,\ x \in \mathbb{Z}_+^\eta\}. \tag{2}$$

Because η is not bounded by any polynomial in n, this model is solved by column generation in practice. The column generation subproblem is a 1D BKP. Model (2) has a strong LP bound, cf. [RD08].

A dual model to (2) is the following:

$$\max\{b^\top \widetilde{w} : \widetilde{w}^\top a^j \le W\ \forall j = \overline{1,\eta},\ \widetilde{w} \in \mathbb{R}^n\}. \tag{3}$$

Because of the weak duality, any feasible $\widetilde{w}$ provides a lower bound $b^\top \widetilde{w}$ for (2). In fact, any such $\widetilde{w}$ is a conservative scale for (W, w), cf. Definition 1. To obtain strong CS, we can solve the LP (3); however, when fast bounds are needed, we can construct CS heuristically, e.g., by dual-feasible functions, see Sections 1.4 and 2.4.

1.3 Conservative scales and bounds for OPP

A simple bound for OPP is the *volume bound*: if the total volume of the items exceeds that of the container then the instance is infeasible. We can also use CS to obtain a valid bound for OPP, as stated by the following

Theorem 1 ([FS04b]). *Let $\mathcal{I} = (W_1, \dots, W_d, w^1, \dots, w^d)$ be an instance of OPP-d. Let $\widetilde{w}^k$ be a CS for (W_k, w^k), $k = \overline{1,d}$. Then any feasible packing for $\mathcal{I}$ can be transformed into one for the instance $\widetilde{\mathcal{I}} = (W_1, \dots, W_d, \widetilde{w}^1, \dots, \widetilde{w}^d)$.*

Thus, the volume bound for the modified instance is valid for the original instance. Often it is stronger, which is heavily used in algorithms. Our goal is to maximize this *modified volume bound*:

$$\begin{array}{ll} \text{maximize} & \sum_{i=1}^{n} \prod_{k=1}^{d} \widetilde{w}_i^k \\ \text{subject to} & \widetilde{w}^k \text{ is a conservative scale for } (W_k, w^k), \quad k = \overline{1,d}. \end{array} \tag{4}$$

Because CS are feasible points of the LP (3), problem (4) is a *multilinear programming problem with disjoint constraints (MLPP)* [MF09, Dre92]. Caprara and Monaci [CM09] investigate an exact method for (4) for the case $d = 2$. They give absolute and asymptotic worst-case bounds on the gaps between some problems related to (4) and some packing problems.

Fixing some $d-1$ conservative scales in (4) leads to a linear program. This constitutes our LP-based heuristic, Section 4. It can be related to *Successive Linear Programming (SLP)* methods which are well-known for non-linear programming problems [Kel60, PGLE82, ZKL85, SW10], also in combination with SQP methods [BGNW04]. SLP methods are rather efficient in practice [MF09]. Some convergence results are known for Trust-Region-like variants [ZKL85].

1.4 Conservative scales' generation tools: (D)DFFs

A long-standing tradition in the cutting & packing community has been the implicit and explicit usage of *(data-dependent) dual-feasible functions ((D)DFFs)* [Joh73, Lue83, MT90, FS04b, FS01, BM03, CAV10, RAV10, RAV11, CMC09, KCT10] to construct CS. We define them and start their discussion in Section 2.3. As we saw in Definition 1, CS assume that each item appears at most once in a combination; DFFs do not. This means that CS can only be used in problems with binary item combinations in patterns, but there they can lead to stronger bounds. To distinguish the subset of CS produced by DFFs, we introduce the subclass of *unbounded CS (UCS)*, Definition 2 in Section 2.2.

When using (D)DFFs to construct CS, it is advantageous to use strong (D)DFFs; [CAV10, RAV10] review the classes of *maximal* and *extremal maximal DFFs*. We will show that EMDFFs do not necessarily produce *maximal CS* (to be defined in Section 3); [CN07] already proposed an enumerative method to find all *extremal UCS*.

1.5 Our contributions

The next section introduces the optimization tools needed by our heuristics. It defines the subclass of UCS and highlights their correspondence to DFFs. We extend the maximality and extremality notions to CS and DDFFs in Section 3, in particular we characterize maximal CS and propose fast construction heuristics based on tightening and lifting of valid inequalities. However, the Sequential LP method of Section 4 always converges to a stationary point and produces extremal maximal CS in non-degenerate cases. The new and old approaches are numerically compared in Section 5.

2 Properties of conservative scales and (D)DFFs

This section discusses some properties of CS, in particular, the sufficiency to consider only non-negative CS. To apply CS to OPP, we consider CS as duals of a certain 1D CSP relaxation of OPP. We distinguish a subclass of CS, *unbounded CS (UCS)* which correspond to DFFs. The relations between (U)CS and (D)DFFs are discussed. Usage principles and examples of (D)DFFs are given.

2.1 Interpretations of conservative scales and non-negativity

Interpretation as valid inequalities. According to Definition 1, consider some constant CS $\widetilde{w}$ for $(W, w) \in \mathbb{R}_+ \times \mathbb{R}^n_+$. Now, the inequality

$$a^\top \widetilde{w} \le W$$

is a valid inequality for the binary knapsack polyhedron

$$P(W, w) = \operatorname{conv}\big\{a \in \{0,1\}^n : a^\top w \le W\big\}. \tag{5}$$

Some classes of these inequalities are widely used both in packing problems [BB07, CAV10] as well as in general integer programming, cf. [NW88, CM09]. This interpretation motivates our interest in the so-called *extremal maximal CS* which represent facets of (5), refer to Section 3.

The CS polyhedron is the set of all possible CS for a given 1D BPP instance. Furthermore, it is the feasible set of the LP (3) which is the LP dual of the Gilmore-Gomory model of 1D CSP. Given all extreme points $a^j \in \{0,1\}^n$, $j = \overline{1,\eta}$, of the knapsack polyhedron (5), some $\widetilde{w} \in \mathbb{R}^n$ is a CS *iff* it satisfies the constraints

$$a^{j\top} \widetilde{w} \le W, \qquad j = \overline{1,\eta}, \tag{6}$$

i.e., all CS for a certain (W, w) are given by the polyhedron

$$D'(P(W, w)) = \big\{\widetilde{w} \in \mathbb{R}^n : a^{j\top} \widetilde{w} \le W,\ j = \overline{1,\eta}\big\} \tag{7}$$

which is a cross-section through the polar of (5). Compare (7) to the LP (3).

Example 1. Let $(W, w) = (10, (2, 3, 4))$. Then the convex hull $P(W, w)$ of knapsack points is the unit cube and $D'(P(W, w)) = W \operatorname{conv}\{e_1, e_2, e_3\} \oplus \mathbb{R}^3_-$ ($\oplus$ is the Minkowski sum and e_i is the i-th unit vector). The three extreme points $\{We_1, We_2, We_3\}$ of $D'(P(W, w))$ represent facets of $P(W, w)$ and are the extremal maximal CS, Section 3; the three remaining facets $-x_i \le 0$, $i = \overline{1,3}$, do not correspond to any CS.

Lemma 2. *All extreme points of $D'(P(W, w))$ are non-negative.*

Proof. To see this, let $\widetilde{w}^0 \in D'(P(W, w))$ and $\widetilde{w}^0_{i_0} < 0$. Consider two vectors $\widetilde{w}^1, \widetilde{w}^2 \in \mathbb{R}^n$ with $\widetilde{w}^1_i = \widetilde{w}^2_i = \widetilde{w}^0_i$, $\forall i \ne i_0$, and the negative components $\widetilde{w}^1_{i_0} = \frac{1}{2}\widetilde{w}^0_{i_0}$ and $\widetilde{w}^2_{i_0} = \frac{3}{2}\widetilde{w}^0_{i_0}$. The vectors $\widetilde{w}^1, \widetilde{w}^2$ are valid CS because of Definition 1: for any $a \in \{0,1\}^n$ with $a^\top \widetilde{w}^0 \le W$, the vector a' with $a'_i = a_i$ $\forall i \ne i_0$ and $a'_{i_0} = 0$ also satisfies the knapsack condition $a'^\top \widetilde{w}^0 \le W$, because $\widetilde{w}^0$ was assumed a valid CS. Thus, $a^\top \widetilde{w}^k \le a'^\top \widetilde{w}^k = a'^\top \widetilde{w}^0 \le W$, $\forall k \in \{1, 2\}$. Because of $\widetilde{w}^0 = \frac{1}{2}(\widetilde{w}^1 + \widetilde{w}^2)$, the CS $\widetilde{w}^0$ is not an extreme point. □

Lemma 2 implies that there are no facets of (5) with negative components among those that correspond to extreme points of $D'(P(W,w))$.

Section 3 arguments that the modified volume bound (4) over CS has at least one extremal maximum. Thus, we can restrict our attention to the non-negative subset of $D'(P(W,w))$,

$$D(P(W,w)) \;=\; D'(P(W,w)\cap\mathbb{R}^n_+ \;=\; \big\{\widetilde{w}\in\mathbb{R}^n_+ : {a^j}^\top\widetilde{w}\le W,\; j=\overline{1,\eta}\big\}. \tag{8}$$

As can be expected, this restriction leads on average to better numerical results and prevents some technical difficulties in the SLP methods (Section 4).

A linear program. We can maximize a linear objective function over $D(P(W,w))$:

$$\max\big\{h^\top\widetilde{w} : \widetilde{w}\in\mathbb{R}^n_+,\; {a^j}^\top\widetilde{w}\le W,\; j=\overline{1,\eta}\big\} \tag{9}$$

with some $h\in\mathbb{R}^n_+$. Note the similarity between (9) and (3). The LPs (3) and (9) can be solved by a cutting-plane algorithm, cf. [CM09]. This provides the basis for our SLP heuristic, Section 4.

2.2 The bar relaxation of OPP and unbounded conservative scales

In the introductory Section 1.2 we showed that CS correspond to dual variables of the set-partitioning model of 1D CSP. When applying CS to OPP, this correspondence holds for a special 1D CSP which is a relaxation of OPP. A yet weaker relaxation helps us to introduce so-called *unbounded CS* which correspond to DFFs (closer discussed below).

The LP dual to (9) is

$$\min\big\{W\cdot\textstyle\sum_{j=1}^{\eta}x_j : \sum_{j=1}^{\eta}a^jx_j\ge h,\; x\in\mathbb{R}^\eta_+\big\}. \tag{10}$$

Compared to (2), the differences are the '$\ge$' constraints and the non-integral variables. Model (10) is the LP relaxation of the set-covering model of the 1D CSP instance (W,w,h). In [CM09] model (10) is called *(bounded) fractional bin packing*. For some values of the input data, the 1D CSP instance (W,w,h) is a relaxation of the OPP instance $\mathcal{I}=(W_1,\dots,W_d,w^1,\dots,w^d)$. For example, for some $k\in\{1,\dots,d\}$, define $(W,w)=(W_k,w^k)$ and $h=(h_i)_{i=1}^n=(\prod_{k'\ne k}w_i^{k'})_{i=1}^n$. With this data, (10) is the *k-th (binary) bar LP relaxation* of OPP [cf. BKRS09] which relaxes the OPP by dividing each d-dimensional item i sized $(w_i^1,\dots,w_i^d)$ into $\prod_{k'\ne k}w_i^{k'}$ one-dimensional bars of length w_i^k, Figure 1. Note that each 1D cut in an OPP packing "goes through" an item at most once, that is why the additional restriction to binary patterns in the 1D CSP is valid. Moreover, any CS $\widetilde{w}$ for (W_k,w^k) is a feasible dual solution of (10) with $(W,w)=(W_k,w^k)$.

The papers [BKRS09, CM09] also discuss a relaxed version of (10) which uses unbounded integer knapsack vectors $\overline{a}^j\in\mathbb{Z}^n_+$, $j=\overline{1,\overline{\eta}}$, in the constraint matrix:

$$\min\big\{W\cdot\textstyle\sum_{j=1}^{\overline{\eta}}x_j : \sum_{j=1}^{\overline{\eta}}\overline{a}^jx_j\ge h,\; x\in\mathbb{R}^{\overline{\eta}}_+\big\}. \tag{11}$$

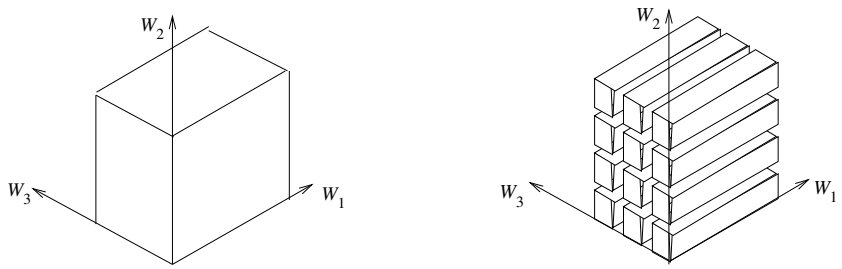

Figure 1: A 3D item and the 1^{st} bar relaxation (along W_1)

It is called *integer bar LP relaxation* of OPP in [BKRS09] and *unbounded fractional bin packing* in [CM09]. Its LP dual is

$$\max\{h^\top \widetilde{w} : \widetilde{w} \in \mathbb{R}^n_+,\ \overline{a}^{j\top} \widetilde{w} \le W,\ j = \overline{1,\overline{\eta}}\}. \tag{12}$$

The paper [CM09] distinguishes two types of modified volume bounds: *2D strengthened DFF problem (2SDP)* corresponds to our model (4) and *2D DFF problem (2DP)* to the same model but with CS from the tighter feasible set of (12). To distinguish this set, we define the corresponding subclass of CS (Definition 1) as follows:

Definition 2. Let $(W, w) \in \mathbb{R}_+ \times \mathbb{R}^n_+$ be an instance of the 1D bin-packing problem. Vector $\widetilde{w} \in \mathbb{R}^n$ is an *unbounded conservative scale (UCS)* for (W, w) if any *unbounded knapsack filling* of (W, w) stays feasible for $(W, \widetilde{w})$:

$$\forall \overline{a} \in \mathbb{Z}^n_+ : \qquad \textstyle\sum_i w_i \overline{a}_i \le W \quad \Rightarrow \quad \sum_i \widetilde{w}_i \overline{a}_i \le W. \tag{13}$$

Obviously, UCS are a subset of CS for a given tuple (W, w). In experiments UCS are usually weaker than CS. The class of UCS is implicitly used, e.g., in [CN07, CM09].

2.3 Definitions of (D)DFFs and relations to (U)CS

This subsection reviews the relationships between (U)CS (Definitions 1 and 2) and certain classes of their heuristic generating functions. We shall establish the following correspondences (which are not always bijective) and inclusions:

$$\begin{array}{ccc} \boxed{\text{UCS}} & \longleftrightarrow & \boxed{\text{DFFs}} \\ \cap\mid & & \cap\mid \\ \boxed{\text{CS}} & \longleftrightarrow & \boxed{\text{DDFFs}} \end{array}$$

and closer discuss the fact that UCS are a subset of CS.

Dual-feasible functions (DFFs) are special functions that produce UCS. The name 'DFFs' is motivated by the fact that UCS are feasible dual solutions of (11), see Lemma 3 below.

Definition 3 ([FS04b, FS01, CAV10])**.** Let there be given positive real numbers C, C'. A function $f : [0, C] \to [0, C']$ is called *dual-feasible* if for any finite set of non-negative real numbers $(x_1, \ldots, x_m) \in \mathbb{R}^m_+$ holds:

$$\textstyle\sum_{i=1}^m x_i \le C \quad \Rightarrow \quad \sum_{i=1}^m f(x_i) \le f(C) = C'.$$

Note that traditional definitions of DFFs use $C = C' = 1$ [FS04b, FS01]. The more general case $C, C' \in \mathbb{Z}_+$ was reviewed in [CAV10], being called there *discrete DFF*. The exact relation of UCS and DFFs is described by the following

Lemma 3 (Lemma 1 in [CM09])**.** *Assume* $0 \le w_1 \le w_2 \le \cdots \le w_n \le W$*. For every DFF* $f : [0, W] \to [0, W']$*, the vector* $\frac{W}{W'} \cdot (f(w_1), \dots, f(w_n))$ *is a UCS for* (W, w)*. Conversely, for any UCS* $\widetilde{w}$*, the function* $f : [0, W] \to [0, W]$ *defined by* $f(x) = \widetilde{w}_i$ *for* $x \in [w_i, w_{i+1})$ *and* $i = \overline{0, n}$ *(letting* $\widetilde{w}_0 = w_0 = 0$*,* $w_{n+1} = W$ *and* $f(W) = W$*) is a DFF.*

Remark 4. Allowing equal item sizes in Lemma 3 (i.e., $w_i = w_{i+1}$ for some i) makes it possible that some UCS $\widetilde{w}$ with $\widetilde{w}_i \ne \widetilde{w}_{i+1}$ cannot be produced by any DFF as defined above. Thus, exact correspondence between UCS and DFFs can be established only for all-different item sizes.

It is easy to see that DFFs always produce a valid UCS (using the method of Lemma 3). They cannot give a CS which is not a UCS, as demonstrated by the following

Example 2. For $(W, w) = (10, (2, 3, 4))$, the CS $\widetilde{w} = \frac{10}{9}(2, 3, 4)$ cannot be obtained by a DFF, as easily seen from Definitions 2 and 3: suppose there exists a DFF $f : [0, 10] \to [0, C']$ with $f(2) > C'/5$. Then $5 \cdot f(2) > C'$ which is a contradiction to Definition 3 because $5 \cdot 2 \le 10$.

This example can be generalized to the following consequence of Definition 3:

Corollary 5. *Any DFF* $f : [0, C] \to [0, C']$ *satisfies* $f(x) \le \frac{C'}{\lfloor C/x \rfloor}$*,*

which is a well-known bound on the duals of the 1D cutting-stock model (11) [CAVR11], cf. also Fact 9 in [CM09].

The broader class of conservative scales corresponds to another class of functions, *data-dependent dual-feasible functions (DDFFs)*, which were proposed in [CCM07]. We slightly modify their definition by allowing real values and by taking the item index as argument:

Definition 4. Let there be given positive real values C, C', and $c_i \le C$, $i \in I = \{1, \dots, n\}$. A mapping $g : I \cup \{0\} \to [0, C']$ is called a *data-dependent dual-feasible function associated with* $(C, (c_1, \dots, c_n))$ if the following holds:

$$\forall\ I_1 \subseteq I, \quad \textstyle\sum_{i \in I_1} c_i \le C \quad \Rightarrow \quad \sum_{i \in I_1} g(i) \le g(0) = C'.$$

We have to define the item index as argument because some functions from the literature implicitly use it, see Section 2.4 and Remark 7 below. Note the correspondence between Definitions 4 (DDFFs) and 1 (CS). This correspondence can be stated precisely in the following

Lemma 6. *For every DDFF* $g : I \cup \{0\} \to [0, W']$ *associated with* $(W, (w_1, \dots, w_n))$*, the vector* $\frac{W}{W'} \cdot (g(1), \dots, g(n))$ *is a CS. Conversely, for any CS* $\widetilde{w}$*, any function* $g(\cdot)$ *satisfying* $g(i) = \widetilde{w}_i$*,* $i = \overline{1, n}$ *and* $g(0) = W$ *is a DDFF.*

Remark 7. Taking the item index as argument for DDFFs enables exact correspondence between CS and DDFFs, as opposed to Remark 4 for DFFs.

Remark 8. Using Lemmas 3 and 6 as well as Fact 9 from Section 3 we can restrict us to integer-valued (D)DFFs as in [CCM07] because (8) and its UCS counterpart are rational polyhedra.

Lemmas 3 and 6 and the fact that UCS are a proper subset of CS imply that DFFs are a proper subset of DDFFs. This relation can be seen on the above example: consider the data $(W, w) = (10, (2, 3, 4))$ and an associated DDFF $g : \{1, 2, 3\} \cup \{0\} \to [0, 9]$ with $(g(1), g(2), g(3), g(0)) = (2, 3, 4, 9)$. As pointed out above, the corresponding conservative scale $\frac{10}{9}(2, 3, 4)$ cannot be obtained by any DFF. When applying a DFF, removing an item may decrease and cannot increase the volume bound. When DDFFs are used, this observation does not hold anymore since the value of other items may be increased [CCM07, KCT10], see the example DDFF below.

The observations of this subsection can be summarized as follows:

1. CS and DDFFs can provide stronger lower bounds than DFFs for our main maximization problems (4) and (3).
2. However, CS are applicable only to problems with *binary item combinations in each pattern*; problems with integer patterns, such as (11), can only be bounded using UCS and DFFs.
3. In general, DFFs cannot produce all UCS; exact correspondence can be established only for all-different item sizes. This applies to all common DFFs. In contrast, some known DDFFs use the item index as argument which allows exact correspondence to CS.

Traditional bounding techniques for (4) and (3), which use (D)DFFs, are closer considered in the next subsection.

2.4 Usage of (D)DFFs in packing problems and examples

Many authors [Lue83, FS01, FS04b, CAV10, CAVR11, KCT10] use (D)DFFs to obtain bounds for OPP and 1D CSP. They heuristically select some (D)DFFs to construct (U)CS (according to Lemmas 3 and 6) and compute a lower bound $\mathcal{L}$ for the maximization problem (4) or (3). For OPP, if $\mathcal{L}$ exceeds the container volume, the container cannot pack all the items.

The objective functions of (4) and (3) can hardly be considered during selection of (D)DFFs. Nevertheless, some research has been invested in finding strong DFFs, e.g., *maximal (dominant)* and *extremal* ones, see the survey [CAV10] and the paper [RAV10]. The next section extends these notions to CS / DDFFs.

Below we present three families of DFFs. They are all maximal, as discussed in [CAV10]. Moreover, they are all extremal, except for f_0^λ with $\lambda \in (1/4, 1/2)$ [RAV10]. The first family of DFFs $f_0^\lambda(x) : [0, 1] \to [0, 1]$ can be called classical because it was used implicitly already in, e.g., [MT90]. In [FS04b, FS01] this family is denoted $U^{(\epsilon)}$. It has a parameter $\epsilon = \lambda \in \left[0, \frac{1}{2}\right]$:

$$f_0^\lambda(x) = \begin{cases} 0, & \text{if } x < \lambda, \\ x, & \text{if } \lambda \le x \le 1 - \lambda, \\ 1, & \text{if } x > 1 - \lambda. \end{cases} \tag{14}$$

Function $f_0^\lambda(\cdot)$ changes the arguments below λ to 0 and those above $1-\lambda$ to 1.

The following family $f^p_{FS,1}(x) : [0, C] \to [0, Cp]$, parameterized by $p \in \mathbb{N}$, was proposed in [FS04b, FS01] (there in the non-discrete version $u^{(p)} : [0, 1] \to [0, 1]$):

$$f^p_{FS,1}(x) = \begin{cases} xp, & \text{if } \frac{x(p+1)}{C} \in \mathbb{Z}, \\ \lfloor \frac{x(p+1)}{C} \rfloor C, & \text{otherwise.} \end{cases} \tag{15}$$

Numerically most advantageous for 1D bin packing [CAV10] appeared the family $f^p_{CCM,1} : [0, C] \to \left[0, 2\lfloor \frac{C}{p} \rfloor\right]$ with parameter $p \in \left[1, \frac{C}{2}\right]$:

$$f^p_{CCM,1} \;=\; \begin{cases} 2\big(\lfloor \frac{C}{p} \rfloor - \lfloor \frac{C-x}{p} \rfloor\big), & \text{if } x > \frac{C}{2}, \\ \lfloor \frac{C}{p} \rfloor, & \text{if } x = \frac{C}{2}, \\ 2\lfloor \frac{x}{p} \rfloor, & \text{if } x < \frac{C}{2}. \end{cases} \tag{16}$$

As we saw above, every DFF is also a DDFF, but not vice versa. The paper [KCT10] proposes the following class of DDFFs. Let $J \subseteq I = \{1, \dots, n\}$ be an index set. Let the function $KP(C, J, \alpha)$ denote the optimal value of the binary knapsack problem with capacity $C \in \mathbb{R}_+$, item sizes $c \in \mathbb{R}^J_+$, and objective coefficients $\alpha \in \mathbb{R}^J$:

$$KP(C, J, \alpha) = \max\big\{\textstyle\sum_{j\in J} \alpha_j x_j : \sum_{j\in J} c_j x_j \le C,\ x \in \{0, 1\}^J\big\}. \tag{17}$$

Let $J \supseteq \{i : c_i \le \frac{C}{2}\}$ be a subset of items including at least all those sized up to half of the bin. The function $g^\alpha_1 : I \cup \{0\} \to [0, KP(C, J, \alpha)]$ defined by

$$g^\alpha_1(j) \;=\; \begin{cases} KP(C, J, \alpha) - KP(C - c_j, J, \alpha), & \text{if } c_j > \frac{1}{2}C, \\ \alpha_j, & \text{if } c_j \le \frac{1}{2}C, \end{cases} \tag{18}$$

is a DDFF associated with $(C, (c_1, \dots, c_n))$. Note that g^α_1 can have different values for equal original sizes, see Remark 7. The best results for the 1D bin-packing problem were obtained in [KCT10] with $\alpha = c$. The authors call g^α_1 a general framework for DDFFs. Below we render this statement more precisely.

3 Extremal and maximal CS and DDFFs

This section extends the discussion of maximality and extremality from DFFs to CS and DDFFs. These notions are defined similarly to those for valid inequalities, cf. Section 2.1 and [NW88, II.1]: a DFF is *maximal* if it is not strictly dominated and it is *extremal* if it is not a linear combination of two different DFFs. We define these notions for CS analogously (Definition 7 below). It is obvious that extremal CS are enough to solve the MLPP (4) to optimality:

Fact 9. *There exists an optimum* $(\widetilde{w}^1, \dots, \widetilde{w}^d)$ *of* (4)*, where each* $\widetilde{w}^k$ *is a vertex of* $P(W_k, w^k)$.

We show by an example that an extremal DFF is not guaranteed to always produce even a maximal CS. We propose a characterization and heuristics to obtain maximal and extremal CS.

Note that [CN07] define maximal and extremal UCS (called there *non-dominated maximal redundant functions*) and propose a method to enumerate all of the latter. We are primarily interested in (E)MCS, so we mention only a few words on extremal and maximal UCS at the end of the section.

3.1 Definitions and motivation

The paper [CAV10] reviews *maximal DFFs (MDFFs)*, i.e., those not dominated by any others. The motivation to consider maximal CS and maximal (D)DFFs is the following. When using (D)DFFs to lower-bound (4) or (3), we heuristically solve

$$\max\{h^\top \widetilde{w} : \widetilde{w} \text{ is a CS for } (W, w)\}. \tag{19}$$

In OPP, $h_i = \prod_{k' \neq k} w_i^{k'}$ is the product of the item sizes in the other dimensions. In 1D BPP, $h_i \equiv 1$. Thus, for every index i with $h_i > 0$, the component $\widetilde{w}_i$ should be "as large as possible". Here is the definition of maximality for DFFs:

Definition 5 (cf. [CAV10])**.** A DFF $f : [0, C] \to [0, C']$ is an MDFF if there does not exist any other DFF $f' : [0, C] \to [0, C'']$ such that $\frac{f(x)}{f(C)} \leq \frac{f'(x)}{f'(C)}$ for all $x \leq C$ and a value y such that $\frac{f(y)}{f(C)} < \frac{f'(y)}{f'(C)}$.

A characterization of MDFFs is known:

Theorem 10 ([CCM07])**.** *A DFF $f : [0, C] \to [0, C']$ is an MDFF if and only if the following conditions hold:*

1. *f is non-decreasing,*
2. *f is superadditive, i.e., $f(x) + f(y) \leq f(x + y)$,*
3. *f is symmetric, i.e., $\forall x \in [0, C]$, $f(x) + f(C - x) = f(C)$,*
4. *$f(0) = 0$.*

The sufficiency criterion can be simplified, see [RAV10]. The same paper investigates extremal maximal DFFs:

Definition 6 ([RAV10])**.** An MDFF f is an *extremal maximal DFF (EMDFF)* if for any two DFFs g, h with $2f(x) = g(x) + h(x)$ $\forall x \in [0, 1]$ it follows that $f \equiv g$.

These notions can be analogously defined for CS:

Definition 7. A CS $\widetilde{w}$ for $(W, w) \in \mathbb{R}_+ \times \mathbb{R}_+^n$ is a *maximal CS (MCS)* if there does not exist any other CS $\widetilde{w}'$ such that $\widetilde{w} \leq \widetilde{w}'$ and an index $i^* \in \{1, \ldots, n\}$ such that $\widetilde{w}_{i^*} < \widetilde{w}'_{i^*}$.

Definition 8. An MCS $\widetilde{w}$ for $(W, w) \in \mathbb{R}_+ \times \mathbb{R}_+^n$ is an *extremal maximal CS (EMCS)* if for any two CS $\widetilde{w}^1, \widetilde{w}^2$ with $2\widetilde{w} = \widetilde{w}^1 + \widetilde{w}^2$ it follows that $\widetilde{w} = \widetilde{w}^1$.

Note that MCS belong to the boundary of the CS polyhedron $D(P(W, w))$ (8) and EMCS are exactly the extreme points of the full polyhedron $D'(P(W, w))$ (7). At the same time, the latter are a subset of the facet-defining inequalities of the binary knapsack polyhedron $P(W, w)$ (5).

Example 3 (see also Example 1)**.** Let $(W, w) = (10, (2, 2, 3, 4))$. Then $P(W, w)$ is the convex hull of all the corners of the 4D unit cube without the point (1,1,1,1). The full CS polyhedron is $D'(P(W, w)) = W \operatorname{conv}\{e_1, e_2, e_3, e_4, \frac{1}{3}(e_1+e_2+e_3+e_4)\} \oplus \mathbb{R}^4_-$ ($\oplus$ is the Minkowski sum). The 5 extreme points $\{We_1, We_2, We_3, We_4, \frac{1}{3}W(e_1 + e_2 + e_3 + e_4)\}$ of $D'(P(W, w))$ represent facets of $P(W, w)$ and are the extremal maximal CS; the 4 remaining facets $-x_i \leq 0$, $i = \overline{1, 4}$, do not correspond to any CS.

Fact 9 above has shown that at least one optimum of the MLPP (4) is extremal. The same holds for the 1D CSP dual (3). Thus, it is enough to consider only extremal CS in these models. In any case, we should try to obtain only maximal CS, as motivated in the beginning of the subsection.

Using the traditional bounding technique for (4) and (3), namely DFFs (Section 2.4), we would prefer EMDFFs as being the strongest. However, EMDFFs do not guarantee to produce even maximal CS, see the following example:

Example 4. Consider again $(W, w) = (10, (2, 3, 4))$ and the EMDFF f_0^λ defined in (14) with $\lambda \leq 0.2$. Applying f_0^λ according to Lemma 3, we obtain the UCS $\widetilde{w} = (2, 3, 4)$ which is dominated by the MCS $\widetilde{w}' = \frac{10}{9}(2, 3, 4)$ (which is not extremal, cf. Example 1).

Another possibility is to try to use DDFFs to obtain (E)MCS. For that, we put

Definition 9. A DDFF g, associated with certain data $(C, c) \in \mathbb{R}_+ \times \mathbb{R}_+^n$, is called a *maximal DDFF (MDDFF) associated with* (C, c) if it produces a maximal CS (using Lemma 6).

3.2 A characterization of maximal CS and DDFFs

We characterize maximal CS with the help of knapsack functions and give a geometric interpretation.

Let $KP(C, I, \gamma) = \max\{\sum_{i\in I} \gamma_i x_i : \sum_{i\in I} c_i x_i \leq C,\ x \in \{0,1\}^I\}$ as in (17). If not stated otherwise, we denote $I = \{1, \dots, n\}$. A simple consequence of Definition 1 of CS is the following characterization of CS:

Observation 11. *Let there be given a tuple $(W, w) \in \mathbb{R}_+ \times \mathbb{R}_+^n$. A vector $\widetilde{w} \in \mathbb{R}^n$ is a CS for (W, w) if and only if the following inequality holds:*

$$KP(W, I, \widetilde{w}) \ \leq \ W. \tag{20}$$

Maximal CS can be characterized as follows.

Theorem 12. *Let there be given a tuple $(W, w) \in \mathbb{R}_+ \times [0, W]^n$. A vector $\widetilde{w} \in \mathbb{R}^n$ is a maximal CS for (W, w) if and only if the following equalities hold:*

$$\widetilde{w}_i \ = \ W - KP(W - w_i, I \setminus \{i\}, \widetilde{w}), \quad \forall i \in I. \tag{21}$$

Proof. "⇒"

Suppose (21) is wrong, i.e.,

$$\exists i_0 \in I : \ \widetilde{w}_{i_0} < W - KP(W - w_{i_0}, I \setminus \{i_0\}, \widetilde{w}).$$

Then we can increase this component up to

$$\widetilde{w}_{i_0} \ = \ W - KP(W - w_{i_0}, I \setminus \{i_0\}, \widetilde{w}).$$

"⇐" Let (21) hold but $\widetilde{w}$ be non-maximal. Then there exist a CS $\widetilde{w}'$ so that $\widetilde{w}' \geq \widetilde{w}$ $(^*)$ and $i_0 \in I$: $\widetilde{w}'_{i_0} > \widetilde{w}_{i_0}$. We obtain the contradiction

$$\begin{aligned} \widetilde{w}_{i_0} \ < \ \widetilde{w}'_{i_0} \ &\leq \ W - KP(W - w_{i_0}, I \setminus \{i_0\}, \widetilde{w}') \\ &\overset{(^*)}{\leq} \ W - KP(W - w_{i_0}, I \setminus \{i_0\}, \widetilde{w}). \end{aligned}$$

□

Corollary 13. *Let there be given a tuple $(W, w) \in \mathbb{R}_+ \times [0, W]^n$. If $\widetilde{w} \in \mathbb{R}^n$ is a maximal CS for (W, w) then the following equality holds:*

$$KP(W, I, \widetilde{w}) \;=\; W. \tag{22}$$

Interpretation. Theorem 12 can be interpreted as follows: $\widetilde{w} \in \mathbb{R}^n$ is an MCS for (W, w) exactly when each item i participates in some optimal solution of value W for objective coefficients $\widetilde{w}$, i.e., when holds:

$$\forall i,\ \exists x^{(i)} \in \{0,1\}^n :\ x_i^{(i)} = 1 \ \wedge\ w^\top x^{(i)} \le W \ \wedge\ \widetilde{w}^\top x^{(i)} = W.$$

Geometrically this means that the hyperplane $\widetilde{w}^\top x = W$ touches at least one feasible knapsack solution (Corollary 13) and, for each $i \in \{1, \dots, n\}$, the i-th component has value 1 in at least one of the touched points, see Figure 2. In the left subfigure, the

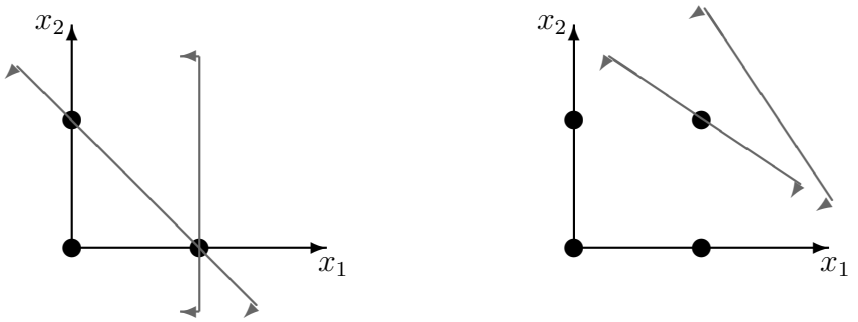

Figure 2: Maximal and non-maximal inequalities geometrically

maximal inequality is satisfied with equality in two points and both x_1 and x_2 take value 1 in different ones. In the right subfigure, the maximal inequality is satisfied with equality in a feasible point where both $x_1 = x_2 = 1$ simultaneously.

Using Theorem 12 and Lemma 6 we can easily prove a similar characterization of maximal DDFFs:

Theorem 14. *Let $I = \{1, \dots, n\}$. Let be given a DDFF $g : I \cup \{0\} \to [0, C']$ associated with some $(C, c) \in \mathbb{R}_+ \times [0, C]^n$. Let $\gamma = (g(1), \dots, g(n))$ be the vector of images of item sizes. Function g is an MDDFF if and only if the following equalities hold:*

$$g(i) \;=\; \gamma_i \;=\; KP(C, I, \gamma) - KP(C - c_i, I \setminus \{i\}, \gamma), \qquad \forall i \in I. \tag{23}$$

Now we can make precise the statement about the generality of the DDFF family g_1^α defined by (18).

Corollary 15. *For a given data tuple $(C, c) \in \mathbb{R}_+ \times [0, C]^n$, consider the family of DDFFs g_1^α (18) with the full subset of items $J = I = \{1, \dots, n\}$ and all $\alpha \in \mathbb{R}_+^n$. This family contains all maximal DDFFs.*

Proof. Let $\widetilde{w}$ be a maximal CS for (C, c). Then it satisfies (21). Select $\alpha = \widetilde{w}$. We obtain

$$g_1^\alpha(j) \;=\; \left.\begin{cases} KP(C, J, \alpha) - KP(C - c_j, J, \alpha) = \alpha_j, & \text{if } \ c_j > \frac{1}{2}C, \\ \alpha_j, & \text{if } \ c_j \le \frac{1}{2}C, \end{cases}\right\} = \alpha_j, \quad \forall j.$$

Because (23) holds, the statement follows. □

However, the family g_1^α contains not only maximal CS, as is shown by

Example 5. Let the capacity and sizes be as follows: $C = 10$, $c = (3, 1, \dots, 1) \in \mathbb{Z}_+^n$ with $n \geq 11$. For g_1^α, choose $J = I$ and the weighting vector $\alpha = (1, 1, \dots, 1) \in \mathbb{R}_+^n$.
For the mapped capacity we obtain $C' = g_1^\alpha(0) = KP(10, J, \alpha) = 10$.
The CS obtained by g_1^α is the vector $\widetilde{c} = \alpha = (1, 1, \dots, 1)$, which is non-maximal.

There arises the question about the existence of a 'closed-form' maximal DDFF for given data. Up to now we have not found such a general function. We propose to compute MCS by tightening of non-maximal CS.

3.3 Computing maximal CS by tightening

There arises the question how to compute maximal CS. It is obvious that any exact solution of the LP (19) for any $h > \mathbf{0}$ is an MCS. It has the advantage that it is a "best" CS for the given objective h. Such an LP-based method will be a topic of Section 4. In this section we are going to construct MCS which heuristically solve (19).

To obtain diversified and strong MCS, we suggest to compute some good UCS by (E)MDFFs (Section 2.4) and to tighten them. As shown in Section 2.3, Example 2, some CS cannot be produced by a DFF, including MCS.

We propose the following general scheme to construct an MCS. Let there be given the input data $(W, w) \in \mathbb{R}_+ \times [0, W]^n$.

1. Construct an UCS $\widetilde{w}$ using some (E)MDFF (Section 2.4) and Lemma 3.
2. Randomize $\widetilde{w}$ by multiplying each $\widetilde{w}_i$ by $(1 - \mathrm{rand}(0, \bar{\nu}))$, where $\mathrm{rand}(a, b)$ is a uniformly distributed random number in $[a, b)$ and $\bar{\nu} \in [0, 1]$ is a parameter.
3. Multiply $\widetilde{w}$ by $\frac{W}{KP(W,I,\widetilde{w})}$ to satisfy (22).
4. In some order, set
$$\widetilde{w}_i \;=\; W - KP(W - w_i, I \setminus \{i\}, \widetilde{w}) \tag{24}$$
for all $i \in I$ to satisfy (21).

We need to specify the order of tightenings in Step 4. Keeping in mind that we are heuristically solving (19), we propose two greedy variants of item sorting.

Greedy 1 (dynamic sorting) Let us estimate the increase of the objective function of (19) when lifting each item:

$$i_0 \;\in\; \arg\max_{i \in I} \Big\{ h_i \cdot \Big[W - KP(W - w_i, I \setminus \{i\}, \widetilde{w}) - \widetilde{w}_i \Big] \Big\}.$$

If the maximal increase is positive, perform the update (24) for this $i = i_0$ and iterate. The complexity of this variant is $O(Wn^3)$ in the naive implementation and can be improved to $O(Wn^2)$ as follows. Denote by $I_{<i} = \{1, \dots, i-1\}$ and $I_{>i} = \{i+1, \dots, n\}$ the sets of item indexes smaller and larger than i, respectively. Note that

$$KP(W - w_i, I \setminus \{i\}, \widetilde{w}) \;=\; \max_{0 \leq p \leq W - w_i} \Big\{ KP(p, I_{<i}, \widetilde{w}) + KP(W - w_i - p, I_{>i}, \widetilde{w}) \Big\} \tag{25}$$

requires $O(W)$ times for computation, where the values of both summands under the maximum are known from the direct and reverse dynamic programming recursions for the binary knapsack problem [MT90], respectively.

Greedy 2 (static sorting) is obtained by simplification: we sort the items in I only once according to non-increasing "item volumes" $h_i c_i$, where $c_i = w_i^k$ is the original size. This sorting is done at the outset of the algorithm. In this ordering, all elements of I are tightened by (24). The complexity of this variant is $O(Wn^2)$ in the naive implementation and can be improved to $O(Wn)$ similar to Greedy 1: after updating some $\widetilde{w}_i$ in Step 4 using (25), the dynamic table with values $KP(p, I_{<i+1}, \widetilde{w})$, $0 \le p \le W$, can be updated in $O(W)$ time.

Example 6. Like above, consider $(C, c) = (10, (2, 3, 4))$, the EMDFF f_0^λ (14) with $\lambda \le 0.2$, and the resulting UCS $\widetilde{c} = (2, 3, 4)$. In Step 3 of the above procedure we obtain the MCS $\widetilde{c}' = \frac{10}{9}(2, 3, 4)$.

Example 7. Let $(C, c) = (10, (\frac{3}{2}, 2, 3, 4))$. Take the EMDFF f_0^λ (14) with $\lambda \le 0.15$, which gives the UCS $\widetilde{c} = (\frac{3}{2}, 2, 3, 4)$. In Step 3 we obtain the CS $\widetilde{c} = \frac{10}{9}(\frac{3}{2}, 2, 3, 4)$. Step 4 gives the MCS $\widetilde{c}' = \frac{10}{9}(2, 2, 3, 4)$.

All items whose value is positively lifted in Step 4 cannot be known a priori in general. However, this is in any case a subset of the items which originally do not satisfy condition (21).

The proposed CS tightening technique generalizes some preprocessing techniques for item sizes, see, e.g., [CCM07, BM03, BM10], and the tightening procedure for conservative scales under search information [FS04b].

3.4 Computing extremal MCS as lifted covers

As argued in the beginning of the section, it is enough to consider only EMCS in the models (4) and (3). We can generate facets of the binary knapsack polyhedron as EMCS. For that, it is possible to adapt the method for extremal UCS from [CN07], however it is rather time-consuming. Our choice was to construct the well-known lifted cover inequalities [cf. NW88, chapter II.2].

A *cover* for a binary knapsack instance $(W, w) \in \mathbb{R}_+ \times \mathbb{R}_+^n$ is a subset $C \subseteq I = \{1, \dots, n\}$ of items with the property $\sum_{i \in C} w_i > W$. Thus, all feasible solutions satisfy the *cover inequality* $\sum_{i \in C} x_i \le |C| - 1$. If the cover is *minimal*, i.e., no item can be removed, then the corresponding inequality represents a facet of the set

$$P(W, w) \cap \{x_i = 0,\ i \notin C\}$$

with $P(W, w)$ being the convex hull of the knapsack fillings, defined by (5). The inequality can be *lifted* to a facet of $P(W, w)$ using the following

Proposition 16 (Sequential lifting, cf. [NW88])**.** *Suppose* $S \subseteq \mathbb{B}^n$, $S^\delta = S \cap \{x \in \mathbb{B}^n : x_1 = \delta\}$ *for* $\delta \in \{0, 1\}$, *and*

$$\textstyle\sum_{j=2}^n \pi_j x_j \le \pi_0 \tag{26}$$

is valid for S^0. *If* $S^1 = \emptyset$, *then* $x_1 \le 0$ *is valid for* S. *If* $S^1 \ne \emptyset$, *then*

$$\textstyle\alpha_1 x_1 + \sum_{j=2}^n \pi_j x_j \le \pi_0 \tag{27}$$

is valid for S *for any* $\alpha_1 \le \pi_0 - \zeta^0$, *where* $\zeta^0 = \max\{\sum_{j=2}^n \pi_j x_j : x \in S^1\}$. *Moreover, if* $\alpha_1 = \pi_0 - \zeta^0$ *("maximal lifting") and* (26) *gives a face of dimension* k *of* $\operatorname{conv}(S^0)$, *then* (27) *gives a face of dimension* $k + 1$ *of* $\operatorname{conv}(S)$.

Note the similarity between the condition (27) and the maximality condition of Theorem 12.

Like in the previous subsection, we have to decide which items should constitute a cover and in which order the remaining items should be lifted. To diversify the search, we employed the UCS produced by (E)MDFFs as hints. Namely, let $\widetilde{w}$ be such a UCS, randomized similar to Step 2 of the MCS heuristic (see previous subsection). We sorted the items according to non-increasing values $h_i\widetilde{w}_i/w_i$, where h_i are the objective coefficients of (19). In this order, we first added the items to a cover C. In the reverse order, the items were removed from C to minimize it. Again in the direct order, the remaining items were lifted.

3.5 A few words about UCS

The unbounded conservative scales, Definition 2, are products of DFFs; the notions of maximality and extremality for them can be defined identically to those for CS. We can use the principles of this section to tighten UCS as well. A characterization of maximal UCS, whose proof is left to the reader, has the form similar to Fact 10 of [CM09]:

Theorem 17. *Let there be given a tuple $(W, w) \in \mathbb{R}_+ \times \mathbb{R}^n_+$. A vector $\widetilde{w} \in \mathbb{R}^n$ is a maximal UCS for (W, w) if and only if the following holds:*

$$\widetilde{w}_i \;=\; \min_{k\in\mathbb{N}}\left\{\frac{W - \max_{\overline{a}\in\mathbb{Z}^n_+}\left\{\sum_{j\neq i}\widetilde{w}_j\overline{a}_j \,:\, w^\top\overline{a}\le W,\, \overline{a}_i = k\right\}}{k}\right\}, \qquad \forall i \in I \tag{28}$$

The paper [CN07] describes a method to enumerate extremal maximal UCS. They discovered that the number of (useful) facets is extremely small, which will be confirmed by our experiments with lifted covers as EMCS.

4 An SLP heuristic for CS in orthogonal packing

For orthogonal packing, specifically for the feasibility problem OPP, the conservative scales in all d dimensions should be optimized simultaneously to provide the best possible volume bound. This exact approach appears too difficult in practice already in 2D [CM09]. The middle way, to select CS in each dimension under consideration of CS in the other dimensions, was approached in the previous section heuristically. Here we discuss a more thorough method that can be classified as Sequential Linear Programming. We also discuss its convergence properties which are used to derive anti-stalling measures.

4.1 The basic variant of the heuristic

Our goal is to construct a set of conservative scales in d dimensions producing the maximal volume bound, see model (4). Using definition (8) of the CS polyhedron $D(P(W, w))$, model (4) can be re-written as

$$\begin{aligned}
\text{maximize} \quad & \textstyle\sum_{i=1}^n \prod_{k=1}^d \widetilde{w}_i^k && && && (29a)\\
\text{subject to} \quad & a^\top\widetilde{w}^k \le W_k && \forall a \in P(W_k, w^k), && k = \overline{1,d} && (29b)\\
& \widetilde{w}^k \in \mathbb{R}^n_+ && k = \overline{1,d}. && && (29c)
\end{aligned}$$

The linear constraints (29b) are disjoint for each $k = \overline{1,d}$. Problem (29) is a *multilinear programming problem with disjoint constraints (MLPP)* [MF09, Dre92]. In the case $d = 2$, (29) is a special bilinear programming problem [cf. CM09]. Generalizing Conjecture 2 of [CM09] we put the following

Conjecture 18. *Problem* (29) *is strongly* $\mathcal{NP}$*-hard.*

Obviously, when the conservative scales $\widetilde{w}^k$ for some $d-1$ dimensions $k \in \{1,\dots,d\} \setminus \{k_0\}$ are fixed, (29) simplifies to the following LP (cf. (9) and (19)):

$$\max\left\{h^\top \widetilde{w} : \widetilde{w} \in \mathbb{R}^n_+,\ a^\top \widetilde{w} \le W_{k_0},\ \forall a \in P(W_{k_0}, w^{k_0})\right\} \tag{30}$$

with $h_i = \prod_{k \ne k_0} \widetilde{w}^k_i$, $i = \overline{1,n}$. The LP (30) looks for a "best" CS $\widetilde{w}^{k_0}$ to combine with the fixed vectors. Such LPs can be considered iteratively, which is the idea for our heuristic.

This heuristic can be related to SLP methods which are well-known for non-linear programming problems [Kel60, PGLE82, ZKL85, SW10, BGNW04]. Convergence results are known only for Trust Region-like variants of the method, see [ZKL85].

Starting with the original sizes w^k, $k = \overline{1,d}$ as conservative scales, the LP (30) is solved many times, each time for a different k_0, and each time the k_0-th CS $\widetilde{w}^{k_0}$ is replaced by the LP's solution. A formal description is given in Figure 3.

Input: OPP instance $(W_1,\dots,W_d,w^1,\dots,w^d)$; iteration limit $t_{\max}$
Output: status 'infeasible' or 'unknown'

S0. Let $\widetilde{w}^k = w^k$, $\forall k = \overline{1,d}$; (initialize CS with the original sizes)

S1. **for** $t = \overline{1,t_{\max}}$
  for $k_0 = \overline{1,d}$
    Solve (30) with $h_i = \prod_{k \ne k_0} \widetilde{w}^k_i$, $\forall i = \overline{1,n}$;
    Replace $\widetilde{w}^{k_0}$ with the corresponding solution vector;

S2. **if** $h^\top \widetilde{w}^{k_0} > \prod_k W_k$ (the LP value exceeds the container volume)
    then Stop; (the OPP instance is infeasible)
  end for k_0
end for t

Figure 3: The basic variant of the LP-based CS-heuristic

We illustrate the heuristic using the 2D case. Let $d = 2$, item lengths $l = w^1$, heights $h = w^2$, container sizes $L = W_1$, $H = W_2$. The heuristic would proceed as follows: first, we look for a CS $\widetilde{l}^{(1)}$ for (L,l) which "best combines" with h:

$$\text{(i)}\quad \widetilde{l}^{(1)} \in \arg\max\{h^\top \widetilde{l} : \widetilde{l} \text{ is a CS for } (L,l)\}.$$

Note that this is a bar relaxation of the OPP instance (Section 2.2). Then we look for a CS $\widetilde{h}^{(1)}$ for (H,h) which "best combines" with the recently found $\widetilde{l}^{(1)}$:

$$\text{(ii)}\quad \widetilde{h}^{(1)} \in \arg\max\{{\widetilde{l}^{(1)}}^\top \widetilde{h} : \widetilde{h} \text{ is a CS for } (H,h)\}$$

... and so on:

$$\begin{array}{rrcl} \text{(iii)} & \widetilde{l}^{(2)} & \in & \arg\max\{{\widetilde{h}^{(1)}}^{\top}\widetilde{l} : \widetilde{l} \text{ is a CS for } (L,l)\}; \\ \text{(iv)} & \widetilde{h}^{(2)} & \in & \arg\max\{{\widetilde{l}^{(2)}}^{\top}\widetilde{h} : \widetilde{h} \text{ is a CS for } (H,h)\}; \\ \ldots & & & \end{array}$$

(which converges to a stationary point $(\widetilde{l}^{(t_0)}, \widetilde{h}^{(t_0)})$ of (29), see the next subsection).

On the other hand, we could have started by looking for a CS $\widetilde{h}^{(1)}$ for (H,h) which "best combines" with the original lengths l:

$$\begin{array}{rrcl} \text{(i)} & \widetilde{h}^{(1)} & \in & \arg\max\{l^{\top}\widetilde{h} : \widetilde{h} \text{ is a CS for } (H,h)\}; \\ \text{(ii)} & \widetilde{l}^{(1)} & \in & \arg\max\{{\widetilde{h}^{(1)}}^{\top}\widetilde{l} : \widetilde{l} \text{ is a CS for } (L,l)\}; \\ \text{(iii)} & \widetilde{h}^{(2)} & \in & \arg\max\{{\widetilde{l}^{(1)}}^{\top}\widetilde{h} : \widetilde{h} \text{ is a CS for } (H,h)\}; \\ \text{(iv)} & \widetilde{l}^{(2)} & \in & \arg\max\{{\widetilde{h}^{(2)}}^{\top}\widetilde{l} : \widetilde{l} \text{ is a CS for } (L,l)\}; \\ \ldots & & & \end{array}$$

(which converges to a, generally different, stationary point $(\widetilde{l}^{(t_1)}, \widetilde{h}^{(t_1)})$ of (29)). In fact, we are going to compute both such sequences together, see Subsection 4.3.

The presented heuristic has the following nice property:

Lemma 19. *The sequence of values of the LP* (30) *solved by the heuristic of Figure 3 is monotone non-decreasing (and hence, the sequence of lower bounds for the OPP instance).*

4.2 A local optimality criterion and convergence

According to Lemma 19, the sequence of modified volume bounds produced by the heuristic is non-decreasing. On the other side, it is bounded by the optimum of (29). We can ask: does the heuristic converge to a local or even a global optimum?

First we investigate a necessary criterion for the iterates to represent a local optimum:

Theorem 20. *Let there be given a set of conservative scales $(\widetilde{w}^1, \ldots, \widetilde{w}^d)$. If, for each $k_0 = \overline{1,d}$, the CS $\widetilde{w}^{k_0}$ is an optimum of the LP* (30) *with the objective coefficients h^{k_0} satisfying $h^{k_0}_i = \prod_{k \neq k_0} \widetilde{w}^k_i$, $i = \overline{1,n}$ then $(\widetilde{w}^1, \ldots, \widetilde{w}^d)$ is a stationary point of* (29).

Proof. Let us perturb the point $(\widetilde{w}^1, \ldots, \widetilde{w}^d)$ by $(\Delta^1, \ldots, \Delta^d)$ so that $\widetilde{w}^k + \Delta^k$ remains a feasible CS for (W_k, w^k), $k = \overline{1,d}$. Let us compute the volume bound in the new point:

$$\textstyle\sum_i \prod_k (\widetilde{w}^k_i + \Delta^k_i) = \sum_i \prod_k \widetilde{w}^k_i + \sum_i \sum_k \Delta^k_i \prod_{j \neq k} \widetilde{w}^j_i + o(\max_k \|\Delta^k\|). \qquad (31)$$

The assumed optimality of $\widetilde{w}^k$ in (30) for each k implies ${\Delta^k}^{\top} h^k \leq 0$ or, equivalently,

$$\textstyle\sum_i \Delta^k_i \prod_{j \neq k} \widetilde{w}^j_i \;\leq\; 0, \qquad \forall\ k.$$

Together with (31) this shows that $(\widetilde{w}^1, \ldots, \widetilde{w}^d)$ satisfies the first-order necessary optimality conditions. □

Our convergence proof needs some assumptions on the implementation, which are provided free of charge when the LPs (30) are solved by the simplex method:

Theorem 21. *Consider the heuristic of Figure 3. Let us assume the following specialization of the algorithm:*

(i) Only basic solutions are computed for (30).

(ii) If the current CS $\widetilde{w}^{k_0}$ *is an optimum of the LP* (30) *then it is not replaced. This is sure to happen if* $\widetilde{w}^{k_0}$ *is used as a starting point in the simplex method for* (30).

Then the set of iterates $(\widetilde{w}^1, \dots, \widetilde{w}^d)$ *maintained by the heuristic arrives at some stationary point of* (29) *in a finite number of steps.*

Proof. The sequence of values of the LP (30) is non-decreasing and bounded. By selecting always basic solutions, we obtain a finite number of possible objective values. Thus, the sequence of LP values achieves a maximum in a finite number of steps.

Moreover, after this has happened, the iterates $(\widetilde{w}^1, \dots, \widetilde{w}^d)$ will not change, thus satisfying the assumptions of Theorem 20 showing that they are a stationary point. □

The restriction to consider only basic solutions in the heuristic is not at all disadvantageous, which is shown by Fact 9 in Section 3.

As it will be seen from the experiment, the heuristic does not always converge to a global optimum.

4.3 Implementation and anti-stalling features

Here we discuss the implemented version of the heuristic whose basic variant was given in Figure 3. The full version computes several CS sequences in parallel, which are all monotone in the sense of Lemma 19; it performs anti-stalling checking as suggested by Theorem 21. However, some operations were simplified, compared to theory, without diminishing practical efficiency. The pseudo-code is given in Figure 4.

In Subsection 4.1 we noticed that it makes generally some difference, in which dimension $k_{00} \in \{1, \dots, d\}$ we start to compute a new CS at first. The order of dimensions, for which CS are computed, determines the sequence of the produced CS. We propose to maintain d sequences, differing by the dimension k_{00} of the first modified CS ($k_{00} = \overline{1, d}$). The conservative scales obtained in iteration $t \in \mathbb{N}$ (Step S3 of Figure 4) are stored in the matrix $(\widetilde{w}^{1,t}, \dots, \widetilde{w}^{d,t})$; they all belong to different sequences. For example, the sequence started at $k_{00} = 1$ is $(\widetilde{w}^{1,1}, \widetilde{w}^{2,2}, \dots, \widetilde{w}^{d,d}, \widetilde{w}^{1,d+1}, \dots)$; each element of the sequence is computed so as to maximize the lower bound on (29) when combined with the last $d-1$ elements. To be precise, the CS $\widetilde{w}^{k_0,t}$ belongs to the sequence started at $k_{00}(k_0, t) = ((k_0 - t) \bmod d) + 1 \in \{1, \dots, d\}$. Vector $\widetilde{w}^{k_0,t}$ is computed in Step S3 by the LP (30) with the objective coefficients

$$h^{k_0} = \left(h_i^{k_0}\right)_{i=1}^{n} = \left(\prod_{p=1}^{d-1} \widetilde{w}_i^{((k_0-p-1) \bmod d)+1, t-p} \right)_{i=1}^{n} \qquad \text{(assuming } \widetilde{w}^{k,t} = w^k,\ \forall t \le 0.)$$

Input: OPP instance $(W_1, \dots, W_d, w^1, \dots, w^d)$; iteration limit $t_{\max}$
Output: status 'infeasible' or 'unknown'

S0. Assume $\widetilde{w}^{k,t} = w^k$, $\forall k = \overline{1,d}$, $\forall t \le 0$; (initialize CS with the original sizes)
for $t = \overline{1, t_{\max}}$
  for $k_0 = \overline{1,d}$
S1. Compute the objective function for the k_0-th LP:

$$h^{k_0} = \left(h_i^{k_0}\right)_{i=1}^{n} = \left(\prod_{p=1}^{d-1} \widetilde{w}_i^{((k_0-p-1) \bmod d)+1,\,t-p}\right)_{i=1}^{n}$$

S2. **if** $h^{k_0} = 0$ **then**
    Compute $\widetilde{w}^{k,t}$, $k = \overline{1,d}$ using (E)MDFFs;
    Set $t \leftarrow t+1$ and $t_{\max} \leftarrow t_{\max} + 1$;
  end for k_0
  for $k_0 = \overline{1,d}$
S3. Solve $\widetilde{w}^{k_0,t} \in \arg\max\{{h^{k_0}}^\top \widetilde{w} : \widetilde{w}$ is a CS for $(W_{k_0}, w^{k_0})\}$,
    using $\widetilde{w}^{k_0,t-1}$ as a start point in the simplex algorithm;
S4. **if** $h^\top \widetilde{w}^{k_0,t} > \prod_k W_k$ (the LP value exceeds the container volume)
    then Stop; (the OPP instance is infeasible)
S5. Anti-stalling feature (simple variant):
    if $\widetilde{w}^{k_0,t} = \widetilde{w}^{k_0,p}$ for some $p < t$
    then Replace $\widetilde{w}^{k_0,t}$ using (E)MDFFs;
  end for k_0
S6. Compute the volume bounds from all known CS:
  if $\max\left\{\sum_i \prod_k \widetilde{w}_i^{k,p_k} : p_k \in \{0,\dots,t\},\ \forall k;\ \exists k : p_k = t\right\} > \prod_k W_k$
    then Stop; (the OPP instance is infeasible)
end for t

Figure 4: The full version of the SLP heuristic.

Note that we have to use expressions of the form $(\dots \mod d)+1$ for dimension indexes because the first index is 1 and not 0. Then the LP value of the sequence started at certain $k_{00} \in \{1,\dots,d\}$ and computed in iteration $t \in \mathbb{N}$ equals (with $k_0 = k_0(k_{00}, t) = ((k_{00}+t-2) \bmod d) + 1$)

$$z_{k_{00},t} = {h^{k_0}}^\top \widetilde{w}^{k_0,t} = \sum_{i=1}^{n} \prod_{p=0}^{d-1} \widetilde{w}_i^{((k_{00}+t-p-2) \bmod d)+1,\,t-p}. \tag{32}$$

Because of Lemma 19, the following monotonicity holds:

$$z_{k_{00},t} \le z_{k_{00},t+1}, \qquad \forall k_{00} = \overline{1,d}, \quad t \in \mathbb{N}.$$

Following Theorem 21, we employ the simplex method for LPs (30). In order to guarantee convergence, we should use $\widetilde{w}^{k_0,t-d}$ as the starting point. Instead, we used $\widetilde{w}^{k_0,t-1}$, i.e., just the last iterate. This was done because of simplicity: storing basis information over several iterations caused numerical difficulties because of the expanding

master problem.[4]

Theorem 20 provides a stalling criterion: if the iterates of one sequence repeat for one cycle, i.e., if $\widetilde{w}^{((k_0-p-1) \bmod d)+1,t-p} = \widetilde{w}^{((k_0-p-1) \bmod d)+1,t-p-d}$, $p = \overline{0, d-2}$ holds for some k_0 and t then we have a stationary point. Then it makes sense to restart the sequence from a different starting point (there is no global convergence in general). We tested several variants of anti-stalling measures. Numerically best appears the following simplified variant: if the computed CS $\widetilde{w}^{k_0,t}$ is already known from before, we replace it by a new CS constructed by DFFs, see below and Step S5. However, this variant does not guarantee convergence to a stationary point because it can quit the monotone sequence prematurely.

In the basic version of the heuristic, because of the monotonicity (Lemma 19), the objective coefficients $(h_i)_{i=1}^n$ are always not altogether zero. When replacing repeated CS by new ones obtained from DFFs (Step S5), it can happen that in the next iterations, the objective coefficients $(h_i^{k_0})_{i=1}^n$ are altogether zero for some k_0 (because for each index $i = \overline{1,n}$, some of the CS in other dimensions has a zero component). In this case we perform one additional iteration with DFFs, i.e., all CS $(\widetilde{w}^{1,t}, \dots, \widetilde{w}^{d,t})$ are computed by DFFs and the iteration counter t is incremented (Step S2). For new starting points we used the EMDFF $f^p_{FS,1}$ (15) with parameters p starting with value 1 and increasing by 1 with each usage. This scheme and this function proved by far the best for the SLP approach.

All conservative scales computed by the heuristic in Steps S2, S3, and S5 of Figure 4 are saved and used to compute alternative volume bounds in Step S6.

5 Results

First, in Section 5.1 we specify the set and parameters of (E)MDFFs. (E)MDFFs were used both on their own as well as to give start vectors for (E)MCS heuristics (Section 3).

In Section 5.2 we tested our approaches on randomly generated 3D OPP instances. We show the effects of some parameters and tune the approaches. In particular, we tested bounds from pure (E)MDFFs, (E)MCS, and SLP (Section 4).

In Section 5.3 we tested (E)MCS, (E)MDFFs, and set-partitioning LP bounds (3) on 1D BPP instances from [CAV10].

In Section 5.4 we tested the same approaches on known and new randomly generated 2D OPP instances as well as on some known 2D BPP instances. Here we additionally took the bilinear programming method from [CM09] for comparison.

As we observed in Section 2.3, the set of DFFs (resp., UCS) is in general a true subset of DDFFs (resp., CS) for a given instance. This does not hold for instances where each item size in each dimension occurs frequently enough to cover the full container size, i.e., where the binary occurrence restriction is not relevant for DFF. For such instances the effect of CS maximization is investigated at the end of Section 5.4.

[4]Note that when the starting point was not used at all, i.e., the LP was always solved completely from scratch, the results were significantly worse. The reason is that some components of the objective function h may become 0 (see Section 4.3), which can lead to non-maximal CS. To improve them in this variant, we found the following modification helpful: for any $i = \overline{1,n}$, if $h_i = 0$, set it to 10^{-6}. This variant produced results similar to the default and is not reported in the experiment.

The experiments were performed on an Intel Xeon X5670 2.93 GHz processor. The algorithms were implemented in GNU C++ 4.1.2 as a single-threaded application. We used IBM ILOG CPLEX 12.3 Academic license [IBM10] as a linear programming library. To solve binary knapsack problems, including column generation, we used simple dynamic programming recursion [MT90]. All running times are reported in seconds.

The test instances and results are available on the CaPaD web page *http://math.tu-dresden.de/~capad*. The software can be provided on request and will be published on the same site after publication of the paper.

5.1 The (extremal) maximal DFFs applied

Table 1 specifies the eight (E)MDFFs reviewed in [CAV10, RAV10], which were used here on their own (as described in Section 2.4) as well as in Step 1 of the MCS heuristic and as sorting hints for EMCS, Sections 3.3 and 3.4. From each DFF except the identity, at most $n^{UCS} = 20$ UCS vectors were produced by Lemma 3 in the default setting.

Table 1: The (extremal) maximal dual-feasible functions used

Function	Parameters*	References
identity	-	-
$f^p_{FS,1}$	$p \in \{1, \dots, n^{UCS}\}$	(15), [FS01, FS04b]
f^λ_0	$\lambda \in \bigcup_{i=1}^n \{w_i, W - w_i\} \cap [0, \frac{W}{4}]$ uniformly	(14), [FS01, FS04b]
$f^p_{CCM,1}$	$p \in [1, \frac{W}{2})$ uniformly	(16), [CAV10, RAV10]
$f^p_{VB,2}$	$p \in \{2, \dots, W\}$ uniformly	cf. [CAV10]
$f^\lambda_{BJ,1}$	$\lambda \in \{2, \dots, W-1\}$ unif., $W \bmod \lambda \neq 0$	[CAV10]
$f^{p,\lambda}_{LL,2}$	λ as above, $p = \left\lceil \text{rand}(0,20) + \frac{\lambda}{W \bmod \lambda} - 1 \right\rceil$	[CAV10]
$f^{p,\lambda}_{DG,1}$	λ, p as for $f^{p,\lambda}_{LL,2}$	[CAV10]

*: $\text{rand}(a, b)$ returns a uniformly distributed random number in $[a, b)$

5.2 3D orthogonal packing feasibility (3D OPP)

Since we do not know any 3D OPP instances in the literature, we had to generate own instances. Below we describe the generation procedure in detail and give test results for various parameter settings.

5.2.1 Generation of instances

To generate test instances, we applied an idea from [BKRS09] which is similar to that used for the 2D OPP instances from [CJCM08]: given the percentage e of container volume which should be waste, distribute the remaining volume among the items. While in [CJCM08] this was done exactly using integer factorization (which has the interesting effect that the items are more often larger in width than in height), we computed the sizes as real numbers and always rounded them down. Due to the relatively large integer container sizes (the containers are cubes with sizes $W_1 = W_2 = W_3 = 1000$), the

deviation from the nominal waste percentage e was small (below 0.5%). We considered $e \in \{0\%, 2\%, \ldots, 40\%\}$.

In addition to the nominal waste percentage e, we also parameterized the maximal item side aspect ratio $r_{\max}$ which means the upper bound on the ratio of some two sides of an item. Obviously, if $r_{\max} = 1$, all items are cubes. We considered $r_{\max} \in \{1, 3, 20\}$, where $r_{\max} = 3$ can be characterized as moderate and $r_{\max} = 20$ as pretty bulky.

The exact generation procedure was as follows. For each tuple $(n, e, r_{\max})$, 100 instances were generated. Every instance was generated in the following way: the nominal total volume of the items, $10^9(1-e)$, was separated into n intervals by $n-1$ uniformly distributed numbers $z_1, ..., z_{n-1}$ in $(0, 10^9(1-e))$. The numbers $z_1, ..., z_{n-1}$ were sorted and item volumes were set as follows: $v_1 = z_1$, $v_n = 10^9(1-e) - z_{n-1}$, and $v_i = z_i - z_{i-1}$ for $i \in \{2, ..., n-1\}$. If the ratio v_i/v_j of the volumes of some two items $i, j \in I$ was higher than 8000, the generation procedure restarted. To obtain the three sides of an item $i \in I$, its volume v_i was factorized with the help of three random numbers a_k, $k \in \{1, 2, 3\}$, whose sum was 3. These numbers were calculated in the same way as the volumes. The side of item i in dimension k had the length $w_i^k = \left\lfloor {v_i}^{a_k/3} \right\rfloor$. If item i was in one dimension larger than 1000 or $r_{\max}$ times the length of another side, the volume v_i was factorized again. To get cubes, we set $a_k = 1$ for $k \in \{1, 2, 3\}$.

5.2.2 Tuning algorithm parameters

First, we played with some parameters of the new algorithms, in order to tune them. Table 2 shows the results for instances of size $n = 15$. For each value of the maximal item side aspect ratio $r_{\max} \in \{1, 3, 20\}$, 2100 randomly generated instances were considered, namely 100 instances for each nominal waste percentage $e \in \{0\%, 2\%, \ldots, 40\%\}$. The column groups of Table 2 give the following results: pure (E)MDFFs (Section 5.1), MCS by Greedy 1 (dynamic), MCS by Greedy 2 (static) (both Section 3.3), EMCS by lifted covers (Section 3.4), and SLP (Section 4). For each of the (E)MCS heuristics, we show variants with and without randomization (parameter $\bar{\nu}$). Especially for (E)MDFFs and Greedy 1 (which works best among (E)MCS) we investigate the effect of the parameter n^{UCS} (maximal number of UCS produced by each of the (E)MDFFS, Section 5.1, default value 20). For SLP we show the anti-stalling effect of CS repetition checking (Step S5 of Figure 4), always setting the iteration limit $t_{\max} = 10$. The result obtained in the first SLP iteration in Step S4 (bar relaxation, Section 2.2) is reported in the column LP[0].

For the lines $r_{\max} \in \{1, 3, 20\}$ and the mentioned columns, Table 2 shows the percentage of solved (i.e., proved infeasible) instances. Lines t_{Lift}, t_{All}, and nCS give the following averages: running time to construct (M)CS, overall time, and the number of different CS obtained in each approach, respectively.

The EMCS heuristic produces very few different CS, still giving good results, which corresponds to the observations in [CN07]. Moreover, EMCS give these competitive results in very short time.

For the maximal number of UCS produced by each (E)MDFF from Table 1, the value $n^{UCS} = 20$ seems to be the best compromize between time and quality. Probably we could improve the (E)MDFF and (E)MCS results by changing the set of (E)MDFFs used.

Randomization proved a very effective technique in the (E)MCS heuristics.

Table 2: Tuning the parameters: 3D OPP, $n = 15$

	(E)MDFFs			MCS-1 (dynamic)					MCS-2		EMCS		LP[0] (S4)	SLP (S6)	
$\bar{\nu} =$				0	0.8	1	1	1	0	1	0	1			
$n^{UCS} =$	10	20	50	20	20	10	20	50	20	20	20	20		S5 off	S5 on
% solved:															
$r_{\max} = 20$	27.6	28.9	30.1	59.4	75.0	74.2	75.3	76.1	57.7	69.8	67.9	74.4	36.9	74.9	76.2
$r_{\max} = 3$	33.8	36.7	38.4	59.7	60.5	59.8	60.6	61.0	59.4	60.4	49.1	55.3	28.6	60.0	60.9
$r_{\max} = 1$	77.2	80.6	82.7	86.2	86.2	86.2	86.2	86.2	86.2	86.2	86.2	86.2	19.3	65.6	85.5
Means:															
% solved	46.2	48.7	50.4	68.4	73.9	73.4	74.0	74.4	67.7	72.1	67.7	72.0	28.3	66.8	74.2
t_{Lift}	0.00	0.00	0.00	0.08	0.12	0.06	0.12	0.31	0.06	0.06	0.04	0.04			
t_{All}	0.03	0.16	2.24	0.14	0.22	0.08	0.22	1.82	0.12	0.17	0.04	0.04	0.00	0.01	0.02
nCS	168	317	788	276	352	178	352	888	273	351	11	38			

Table 3: 3D OPP: varying number of items n

	(E)MDFF					MCS-1					MCS-2					EMCS					LP[0] (S4)		SLP (10 iter.)			
	%	s	t_{Lift}	t_{All}	nCS	%	s	$t_{Lift}^{\text{naïve}}$	t_{Lift}	t_{All}	nCS	%	s	$t_{Lift}^{\text{naïve}}$	t_{Lift}	t_{All}	nCS	%	s	t_{Lift}	t_{All}	nCS	%	s	%	s	t_{All}	t_{ColGen}	nCols
$n = 15$																												
$r_{\max} = 20$	28.9	0.00	0.20		313	75.3	0.23	0.11	0.21		360	69.8	0.08	0.06	0.18		359	74.4	0.04	0.04		43	36.9	0.00	76.2	0.02	0.02	112
$r_{\max} = 3$	36.7	0.00	0.17		318	60.6	0.30	0.16	0.32		355	60.4	0.08	0.07	0.24		354	55.3	0.04	0.04		43	28.6	0.01	60.9	0.02	0.02	116
$r_{\max} = 1$	80.6	0.00	0.10		320	86.2	0.33	0.14	0.20		340	86.2	0.08	0.06	0.12		340	86.2	0.04	0.04		28	19.3	0.00	85.5	0.01	0.01	108
$n = 40$																												
$r_{\max} = 20$	0.0	0.00	0.33		314	4.5	1.11	0.09	0.72		381	2.0	0.13	0.03	0.67		382	8.6	0.06	0.10		144	0.4	0.10	10.5	0.24	0.22	524
$r_{\max} = 3$	0.0	0.00	0.32		312	0.1	1.52	0.10	0.60		355	0.1	0.14	0.03	0.53		355	0.1	0.07	0.09		129	0.0	0.07	0.1	0.25	0.22	571
$r_{\max} = 1$	18.2	0.00	0.27		298	21.9	1.47	0.09	0.41		324	21.9	0.13	0.02	0.34		324	22.3	0.06	0.07		84	0.0	0.05	12.7	0.22	0.19	474
$n = 100$																												
$r_{\max} = 20$	0.0	0.00	0.61		321	0.0	17.18	0.49	1.62		390	0.0	0.74	0.08	1.22		391	0.2	0.38	0.66		245	0.0	0.92	0.5	1.83	1.69	1227
$r_{\max} = 3$	0.0	0.00	0.53		308	0.0	24.15	0.58	1.41		352	0.0	0.78	0.07	0.89		352	0.0	0.40	0.63		228	0.0	0.43	0.0	2.01	1.84	1340
$r_{\max} = 1$	0.0	0.00	0.36		269	0.0	24.45	0.54	0.99		289	0.0	0.63	0.05	0.50		289	0.0	0.29	0.39		174	0.0	0.28	0.0	2.12	1.95	1361

Table 4: Number of solved instances by nominal waste and SLP iterations: 3D OPP, $n = 15$, $r_{\max} = 20$

e, %	(E)MDFF	MCS-1	MCS-2	EMCS	LP[0] (S4)	SLP: Step S6 of Figure 4, iteration ...									
						1	2	3	4	5	6	7	8	9	10
0	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100
2	95	100	100	100	100	100	100	100	100	100	100	100	100	100	100
4	84	100	100	100	99	100	100	100	100	100	100	100	100	100	100
6	69	100	100	100	88	100	100	100	100	100	100	100	100	100	100
8	55	100	100	97	84	100	100	100	100	100	100	100	100	100	100
10	47	98	97	97	76	98	98	98	98	99	99	99	99	99	99
12	33	99	97	94	53	93	95	97	97	99	99	99	99	99	99
14	26	98	95	93	45	89	93	95	97	98	98	98	98	98	98
16	32	95	90	98	33	81	95	96	96	96	96	97	97	98	98
18	17	96	86	89	28	76	84	88	91	92	92	93	93	93	93
20	12	73	66	70	17	56	66	71	73	73	73	74	74	74	74
22	8	78	74	79	15	53	72	75	77	78	78	78	79	80	80
24	14	74	69	76	15	60	67	70	71	73	74	75	75	75	75
26	4	72	60	70	7	53	68	72	72	73	75	75	77	77	78
28	3	63	55	62	3	38	57	61	64	64	64	64	64	64	64
30	3	53	46	50	3	38	49	52	54	55	55	55	55	55	55
32	2	50	34	50	3	35	43	48	49	50	51	51	51	51	51
34	0	35	26	38	2	24	33	36	37	37	37	37	38	38	38
36	2	38	29	40	1	22	33	34	38	38	39	39	39	39	39
38	0	31	24	31	2	17	28	30	30	30	30	31	31	31	31
40	0	28	17	29	1	15	28	29	29	29	29	29	29	29	29
Means	28.9	75.3	69.8	74.4	36.9	64.2	71.9	73.9	74.9	75.4	75.7	75.9	76.1	76.2	76.2

Table 5: 1D BPP, instances from [CAV10]

$(w_{\min}, n)$	[CAV10]	LP (3)			(E)MDFFs				MCS-1		MCS-2				EMCS	
					$n^{UCS=20}$		$n^{UCS=500}$		$n^{UCS=20}$		$n^{UCS=20}$		$n^{UCS=500}$		$n^{UCS=20}$	
		$\lceil$LP$\rceil$	t_{LP}	t_{ColGen}	$\lceil$val$\rceil$	t	$\lceil$val$\rceil$	t	$\lceil$val$\rceil$	t	$\lceil$val$\rceil$	t	$\lceil$val$\rceil$	t	$\lceil$val$\rceil$	t
(1,1000)	507.811	508.273	0.57	0.007	507.309	0.001	508.015	0.018	507.322	1.29	507.325	0.25	508.033	8.05	501.152	5.01
(1, 100)	52.534	52.712	0.33	0.005	52.425	0.001	52.621	0.012	52.6	0.04	52.597	0.02	52.672	0.54	51.220	0.06
(1, 500)	255.506	255.853	0.58	0.008	255.165	0.001	255.644	0.018	255.189	0.26	255.184	0.11	255.655	2.78	251.774	1.30
(20,1000)	626.712	627.277	0.21	0.001	625.978	0.001	626.956	0.015	626.157	0.79	626.155	0.25	627.032	7.29	617.065	3.13
(20, 100)	64.434	64.676	0.20	0.002	64.286	0.000	64.552	0.011	64.5	0.04	64.492	0.02	64.614	0.52	63.259	0.04
(20, 500)	315.027	315.479	0.30	0.001	314.562	0.001	315.209	0.015	314.695	0.27	314.696	0.11	315.265	2.72	311.462	0.81
(35,1000)	768.816	768.816	0.10	0.000	766.990	0.001	768.816	0.012	767.914	0.65	767.911	0.24	768.816	6.87	762.135	1.71
(35, 100)	78.624	78.635	0.09	0.000	78.252	0.001	78.635	0.010	78.551	0.03	78.548	0.02	78.635	0.49	77.469	0.03
(35, 500)	385.782	385.782	0.10	0.001	384.550	0.001	385.782	0.013	385.225	0.23	385.226	0.11	385.782	2.56	383.719	0.45
average	339.472	339.723	0.28	0.003	338.835	0.001	339.581	0.014	339.128	0.40	339.126	0.12	339.612	3.54	335.473	1.39

For the "bulky" instances ($r_{\max} = 20$), the LP[0] value mostly dominates the bound from (E)MDFFs. This dominance noticeably increases in the subsequent iterations of the SLP heuristic. For cubic and moderately bulky instances, the LP[0] value is very weak compared to (E)MDFFs, which becomes the converse after a few SLP iterations. The weakness of LP[0] for cubes can be explained by a greater loss of geometric information. The stalling prevention feature (Step S5 in Figure 4) proved very effective.

5.2.3 Varying number of items

Table 3 is similar to Table 2 but also considers instances with $n = 40$ and $n = 100$. It reports some more data: the average time for column generation (t_{ColGen}) and the final number of columns (nCols) for SLP and the running time of MCS-1,2 heuristics in the naive implementation ($t_{Lift}^{\text{naïve}}$). For (E)MCS heuristics we accepted parameter values $\bar{\nu} = 1$ and $n^{UCS} = 20$. We see that with larger n, the advantage of (E)MDFFs and (E)MCS over SLP on cubic instances grows. The worst-case-optimal implementation of MCS-1,2 heuristics is much faster than the naive variant.

5.2.4 Some detailed results

Table 4 details the results of Table 2, line $r_{\max} = 20$, for each nominal waste percentage $e \in \{0\%, 2\%, \ldots, 40\%\}$ and for each SLP iteration. We see that SLP converges rather quickly.

5.3 1D bin packing

The test bed from [CAV10] includes 9 instance classes. They all have the following parameters: bin size $W = 100$ and n items with sizes in the set $\{w_{\min}, \ldots, 100\}$. Several values of n (100, 500, 1000) and $w_{\min}$ (1, 20, 35) are used. For each pair $(n, w_{\min})$ there are 1000 instances. Our comparison is reported in Table 5.

We observe that (E)MDFFs produce good results in small time (we do not know the (E)MDFF parameters and n^{UCS} for the excellent results in [CAV10]). The chosen EMCS heuristic is very weak on these instances. Note that we set the (E)MCS randomization parameter $\bar{\nu} = 0$ here. The time to compute the exact LP bound is rather small. This can be explained by the small absolute value of bin capacity $W = 100$: the number of distinct integer item sizes was below 100 and we could group equal items, modeling each packing pattern as a feasible point of the *bounded knapsack problem* [MT90, KPP04]. Without grouping, the LP time was prohibitively large.

5.4 2D packing

For 2D packing we considered published instances of OPP and BPP and new instances of OPP. Some of the new instances were generated in such a way that the sets of UCS and CS coincided. The exact bilinear programming method of [CM09] was compared.

5.4.1 2D OPP instances of Clautiaux et al.

At first we considered the 27 infeasible OPP instances from [CJCM08]. (In their paper, only 26 infeasible instances are cited, but the complete set has 15 feasible and 27 infeasible instances). The container size is 20×20 and the number of items $n \leq 23$.

The code of [CM09] was kindly provided to us by the authors. It performs a branch-and-cut algorithm to find a pair of conservative scales maximizing the volume bound (4). We used the constrained version of this algorithm (2SDP). 2SDP is the version where only binary knapsack solutions are considered. This corresponds to Definition 1.

The results are given in Table 6. The first column contains instance names; in each

Table 6: The 27 infeasible instances from Clautiaux et al. [CJCM08]

	Clautiaux et al.			Graphs, FS		BiLin		EMDFF	MCS-1		SLP		
inst	fails	chpts	t	nodes	t	vol	t	solved	solved	t	solved	t	iter.
00N10	1	0	0	1	0	1.05	0.01	0	1	0.01	1	0.01	0
00N15	33	32	0.03	127	0	1	$> 1h$	0	0	0.01	0	0.02	> 10
00N23	105	104	0.03	-	-*	1.002	31.01	0	0	0.01	1	0.02	0
00X23	11904	11903	2.38	-	-	1	$> 1h$	0	0	0.01	0	0.04	> 10
02N20	2	1	0.02	1	0	1.08	0.01	0	0	0.01	1	0.02	2
03N10	2	1	0	1	0	1.13	0.01	0	1	0.01	1	0	0
03N15	70	69	0.03	13	0	1	$> 1h$	0	0	0.01	0	0.01	> 10
03N16	2706	2705	0.66	9891	2.00	1	$> 1h$	0	0	0.01	0	0.02	> 10
03N17	3	2	0.02	431	0	1	$> 1h$	0	0	0.01	0	0.02	> 10
04N15	996	995	0.17	35	0	1.01	0.03	0	0	0.01	1	0.02	0
04N17	17	16	0.03	1	0	1.04	0.02	0	0	0.01	1	0.01	0
04N18	626	625	0.08	24593	10.00	1.01	0.02	0	0	0.01	1	0.02	0
05N15	2558	2557	1.08	1	0	1.07	0.02	0	0	0.01	1	0.01	0
05N17	67	66	0.02	1	0	1	$> 1h$	0	0	0.01	0	0.02	> 10
05X15	989	988	0.25	18369	2.00	1.02	0.02	0	0	0.01	1	0.01	0
07N10	1	0	0.02	17	0	1	$> 1h$	0	0	0	0	0.01	> 10
07N15	1	0	0	61	0	1.08	0.01	0	1	0.01	1	0.01	0
07X15	167	166	0.06	1	0	1	$> 1h$	0	0	0.01	0	0.01	> 10
08N15	1	0	0	1	0	1	$> 1h$	0	0	0.01	0	0.01	> 10
10N10	1	0	0.03	5	0	1.06	0.01	0	1	0.01	1	0.01	0
10N15	589	588	0.22	7	0	1.03	0.01	0	0	0.02	1	0.01	0
10X15	298	297	0.05	77	0	1	$> 1h$	0	0	0.01	0	0.02	> 10
13N10	1	0	0	17	0	1	$> 1h$	0	0	0.01	0	0	> 10
13N15	2	1	0.03	1	0	1	$> 1h$	0	0	0.02	0	0.01	> 10
13X15	43	42	0.02	1	0	1.17	0.01	0	0	0.01	1	0.01	1
15N10	no data			no data		1	$> 1h$	0	0	0.01	0	0	> 10
15N15	2	1	0.02	1	0	1	$> 1h$	0	0	0.01	0	0.01	> 10
mean	815	814	0.20	2236	0.58	$\sum$ 13		$\sum$ 0	$\sum$ 4	0.01	$\sum$ 13	0.01	0

*: Instances not solved by the original graphs algorithm, time limit unknown [CJCM08].

name, the first number is the percentage of waste and the second is the number of items n. Further columns report the results of various algorithms. At first we report two exact algorithms for OPP: the schedule-or-postpone exact algorithm for 2D OPP [CJCM08][5] and the graph-theoretical exact algorithm from [FSvdV07] (data taken from [CJCM08]), executed on a Pentium M 1.8 GHz. For the schedule-or-postpone algorithm, 'fails' is the number of backtracks and 'chpts' is the number of choice points (position fixings). Then we report results of 2SDP [CM09] (column BiLin), EMDFF, MCS-1, and SLP. The results of 2SDP report the proved lower bound on (4), denoted by "vol". The results of

[5]We report here new values given to us by F. Clautiaux in private correspondence.

SLP report the number of iterations; when it is 0, it means that the infeasibility proof was done by pure bar relaxation (step S4).

(E)MDFFs could not solve any instance, even when setting $n^{UCS} = 2000$. The 2SDP code from [CM09] solved 13 instances, 12 of them rather quickly and one more (instance 00N23) after 30 seconds. The same 13 instances were solved by SLP, 11 of them in iteration 0 (by the bar relaxation).

5.4.2 2D BPP instances from [CM09]

We tested the 83 2D BPP instances considered in [CM09]. Similarly to the results of the previous subsection, our heuristic in a few iterations obtained exactly the same bounds as the exact bilinear programming method 2SDP.

5.4.3 Random 2D OPP instances

Table 7: Randomly generated instances with $n = 20$

	(E)MDFF	MCS-1	MCS-2	EMCS	LP[0] (S4)	SLP (S6)	BiLin
% solved:							
$r_{\max} = 20$	0.5	11.4	7.1	11.0	13.3	14.8	12.4
$r_{\max} = 3$	0.0	0.0	0.0	0.0	2.4	3.8	2.4
$r_{\max} = 1$	0.0	1.0	1.0	0.0	0.0	3.8	3.8
Means:							
% solved	0.2	4.1	2.7	3.7	5.2	7.5	6.2
t_{Lift}	0.000	0.120	0.064	0.078			
t_{All}	0.002	0.122	0.066	0.079	0.035	0.075	1.67*
nCS	196	211	210	71			

*: For BiLin, only the time for solved instances was considered (time limit 60 seconds).

In the two previous subsections we observed similar behavior of SLP and BiLin. To compare these methods on a broader basis, we generated random 2D OPP instances in a similar way to 3D OPP (Section 5.2.1), 10 instances for each maximal item side aspect ratio $r_{\max} \in \{1, 3, 20\}$ and for each waste percentage $e \in \{0\%, 2\%, \ldots, 40\%\}$, in total 630 instances. The number of items was $n = 20$. Their results are reported in Table 7. We observe a slight advantage of SLP both in time and quality. For (E)MCS heuristics, almost all the running time was spent for CS construction, as opposed to the 3D case, where the computation of the modified volume bounds takes a long time.

5.4.4 Random instances with UCS ≡ CS

As we observed in Section 2.3, the set of DFFs (resp., UCS) is in general a true subset of DDFFs (resp., CS) for a given instance. This does not hold for instances where each item size in each dimension occurs frequently enough to almost cover the full container size, i.e., where the binary occurrence restriction is not relevant for DFF. For example, the following instance has this property:

$$L = H = 100, \quad 4 \text{ items } 18 \times 31, \quad 3 \text{ items } 18 \times 41, \quad 2 \text{ items } 28 \times 31, \quad 2 \text{ items } 28 \times 41.$$

We generated several classes of such instances in the following way. Let container sizes be $L = H = 1000$. Let $c^L_{\min}$, $c^L_{\max}$, $c^H_{\min}$, $c^H_{\max}$ denote the minimum and maximum item size along the L and H axis, respectively. Let k^L and k^H denote the number of distinct item sizes in dimensions L and H, respectively. We generated these sizes in each dimension randomly uniformly and combined them to 2D items taking care that each size occurs frequently enough to (at least almost) cover the container size, i.e.,

$$\left|\{i : l_i = l_{i_0}\}\right| \;\geq\; \left\lfloor L/l_{i_0} \right\rfloor, \qquad \forall i_0 = \overline{1,n}$$

(and similarly for H). Additional items with sizes from this set were added until the total item area was approximately $10^6(1-e)$, where $e \in \{0\%, 2\%, \ldots, 40\%)$ is the nominal percentage of waste as in Section 5.2.1. For each value of e, 10 instances were generated. The generation program can be downloaded with the instances.

Table 8 reports the results for nine test classes whose parameters $(c^L_{\min}, c^L_{\max}, k^L, c^H_{\min}, c^H_{\max}, k^H)$ are given in the first column. For eight of the classes, 210 instances were generated per class. The other class, (50,400,3,15,100,6), contains only 160 instances, because starting with waste percentage $e = 32\%$, no instance could be generated with reasonable effort so that total item area would not exceed $10^6(1-e)$. The number of instances per class is given in the second column. Further columns report: the average number of items n in the class and the results of the methods (E)MDFF, MCS-2, LP[0], SLP, and BiLin. We observe a slight advantage of the

Table 8: Randomly generated instances with CS ≡ UCS

Instance class	N inst.	mean n	(E)MDFF		MCS-2				LP[0] (S4)		SLP (S6)		BiLin	
					$\bar\nu = 0$		$\bar\nu = 1$							
			% s	t	% s	t_{All}	% s	t_{All}	% s	t	% s	t	% s	t
50,400,2,50,400,2	210	22.8	20.0	0.00	24.3	0.04	19.5	0.04	19.0	0.02	24.3	0.07	24.3	0.02
50,400,3,50,400,3	210	25.9	4.8	0.00	11.0	0.05	6.2	0.05	9.0	0.04	11.0	0.12	11.0	0.05
50,400,4,50,400,4	210	31.6	1.9	0.00	7.1	0.07	1.4	0.07	6.2	0.07	8.6	0.20	8.6	0.26
50,400,3,50,400,6	210	36.6	1.4	0.00	2.9	0.07	0.5	0.06	2.9	0.09	2.9	0.29	3.3	0.21
50,400,3,15,100,6	160	155.7	1.9	0.00	5.6	0.33	0.0	0.26	5.6	4.07	5.6	20.95	10.0	0.58
50,400,3,20,150,4	210	72.9	3.3	0.00	6.7	0.13	1.4	0.11	6.7	0.58	7.1	2.08	8.1	0.14
50,400,3,20,150,6	210	104.1	1.9	0.00	3.3	0.20	0.0	0.17	3.8	1.38	3.8	5.78	3.8	0.52
15,200,3,15,200,3	210	105.6	0.5	0.00	2.9	0.19	0.0	0.15	2.4	3.82	3.3	58.20	3.8	0.15
25,200,3,20,150,4	210	112.3	1.0	0.00	2.9	0.18	0.0	0.14	3.3	2.56	3.3	9.62	3.3	0.20
Means		72.0	4.1	0.00	7.4	0.13	3.3	0.11	6.6	1.33	7.8	10.54	8.4	0.23

method BiLin. It should be noticed that their program uses additional lower and upper bounds for (4), which explains the short running times. For MCS-2, we see that only the non-randomized version ($\bar\nu = 0$) has advantages over (E)MDFFs. This is in a big contrast to all previous 2D and 3D test classes and can probably be explained by the structure of the instances (many equal sizes).

We see that even on such instances, MCS are much stronger than (E)MDFFs and are comparable with SLP.

6 Conclusions

We reviewed the classical lower-bounding tools for packing problems, (data-dependent) dual-feasible functions. We proposed to strengthen their values (conservative scales)

by maximization techniques. Furthermore, we proposed an SLP heuristic to construct extremal maximal conservative scales and investigated its convergence properties. The SLP heuristic is based on a simplification of the original multilinear optimization problem to a linear one by fixing the CS in all dimensions but one to constants.

The main theoretical and experimental observations of the paper are the following:

- All known DFFs can have just a single value for one original size, while (U)CS can have different values for equal original sizes.
- Some DDFFs from the literature can have different values for equal original sizes, which establishes exact correspondence between DDFFs and CS.
- (Extremal) maximal DFFs do not always produce maximal (U)CS.
- A maximal (U)CS can be characterized as follows: each knapsack variable takes a positive value in some of the feasible points satisfying the corresponding valid inequality with equality.
- The optimized implementations of the MCS greedy heuristics are rather fast. Their running time would further decrease for smaller absolute container sizes.
- Randomization often proved a very effective technique in the (E)MCS heuristics.
- For 1D packing, (E)MDFFs and maximized CS have similar performance, however weaker than LP.
- For 2D packing, SLP and bilinear programming are comparable. (E)MDFFs and (E)MCS are often weaker.
- For 3D packing, (E)MDFFs and (E)MCS proved especially efficient for cubic instances. (E)MCS are competitive with SLP. EMCS by lifted covers are fast; only very few of them are required.
- The set of UCS (resp., DFFs) is in general a true subset of CS (resp., DDFFs) for a given instance. Maximization of CS obtained from (E)MDFFs proved efficient even on instances where the sets of CS and UCS are equivalent (where each item size in each dimension occurs frequently).

As outlook we can pose the following topics:

- Till now we do not know any 'closed-form' maximal DDFF.
- The chosen EMCS heuristic (lifted covers) is very weak for large-scale 1D bin packing. This result poses the question about the relative efficiency of facet-defining and other maximal cuts constructed from relaxed polyhedra.

Acknowledgements

We thank Michele Monaci and Alberto Caprara for kindly providing us with their code for optimizing conservative scales by bilinear programming; anonymous referees for their fruitful comments; François Clautiaux for the 1D test instances and further suggestions for the experiment. We appreciate the Academic Initiative of IBM which enables many researchers all over the world to compare their methods using state-of-the-art IBM ILOG Optimization Software.

References

[AVPT09] R. Alvarez-Valdes, F. Parreño, and J. M. Tamarit. A branch and bound algorithm for the strip packing problem. *OR Spectrum*, 31(2):431–459, 2009.

[BB07] R. Baldacci and M. A. Boschetti. A cutting-plane approach for the two-dimensional orthogonal non-guillotine cutting problem. *European Journal of Operational Research*, 183(3):1136–1149, 2007.

[BCP08] N. Beldiceanu, M. Carlsson, and E. Poder. New filtering for the *cumulative* constraint in the context of non-overlapping rectangles. In L. Perron and M.A. Trick, editors, *CPAIOR*, volume 5015 of *LNCS*, pages 21–35. Springer, 2008.

[BGNW04] R. H. Byrd, N. I. M. Gould, J. Nocedal, and R. A. Waltz. An algorithm for nonlinear optimization using linear programming and equality constrained subproblems. *Mathematical Programming*, 100:27–48, 2004.

[BKRS09] G. Belov, V. Kartak, H. Rohling, and G. Scheithauer. One-dimensional relaxations and LP bounds for orthogonal packing. *International Transactions on Operational Research*, 16(6):745–766, 2009.

[BM03] M. A. Boschetti and A. Mingozzi. The two-dimensional finite bin packing problem. Part I: New lower bounds for the oriented case. *4OR: A Quarterly Journal of Operations Research*, 1:27–42, 2003.

[BM10] M. A. Boschetti and L. Montaletti. An exact algorithm for the two-dimensional strip-packing problem. *Operations Research*, 58(6):1774–1791, 2010.

[CAV10] F. Clautiaux, C. Alves, and J. Valério de Carvalho. A survey of dual-feasible functions for bin-packing problems. *Annals of Operations Research*, 179(1):317–342, 2010.

[CAVR11] F. Clautiaux, C. Alves, J. Valério de Carvalho, and J. Rietz. New stabilization procedures for the cutting stock problem. *INFORMS JoC*, 23(4):530–545, 2011.

[CCM07] J. Carlier, F. Clautiaux, and A. Moukrim. New reduction procedures and lower bounds for the two-dimensional bin packing problem with fixed orientation. *Computers & Operations Research*, 34(8):2223–2250, 2007.

[CJCM08] F. Clautiaux, A. Jouglet, J. Carlier, and A. Moukrim. A new constraint programming approach for the orthogonal packing problem. *Computers & Operations Research*, 35(3):944–959, 2008.

[CM04] A. Caprara and M. Monaci. On the two-dimensional knapsack problem. *Operations Research Letters*, 32:5–14, 2004.

[CM09] A. Caprara and M. Monaci. Bidimensional packing by bilinear programming. *Mathematical Programming*, 118(1):75–108, 2009.

[CMC09] F. Clautiaux, A. Moukrim, and J. Carlier. New data-dependent dual-feasible functions and lower bounds for a two-dimensional bin-packing problem. *International Journal of Production Research*, 47(2):537–560, 2009.

[CN07] J. Carlier and E. Néron. Computing redundant resources for the resource constrained project scheduling problem. *European Journal of Operational Research*, 176(3):1452 – 1463, 2007.

[Dow84] K. A. Dowsland. The three–dimensional pallet chart: An analysis of the factors affecting the set of feasible layouts for a class of two–dimensional packing problems. *Journal of the Operational Research Society*, 35(10):895–905, 1984.

[Dre92] R. F. Drenick. Multilinear programming: Duality theories. *Journal of Optimization*

Theory and Applications, 72:459–486, 1992.

[FS01] S. P. Fekete and J. Schepers. New classes of lower bounds for the bin packing problem. *Mathematical Programming*, 91(1):11–31, 2001.

[FS04a] S. P. Fekete and J. Schepers. A combinatorial characterization of higher-dimensional orthogonal packing. *Mathematics of Operations Research*, 29(2):353–368, 2004.

[FS04b] S. P. Fekete and J. Schepers. A general framework for bounds for higher-dimensional orthogonal packing problems. *Mathematical Methods of Operations Research*, 60(2):311–329, 2004.

[FSvdV07] S. P. Fekete, J. Schepers, and J. van der Veen. An exact algorithm for higher-dimensional orthogonal packing. *Operations Research*, 55(3):569–587, 2007.

[GG61] P. C. Gilmore and R. E. Gomory. A linear programming approach to the cutting-stock problem (Part I). *Operations Research*, 9:849–859, 1961.

[GJ79] M. R. Garey and D. S. Johnson. *Computers and intractability.* Freeman, 1979.

[IBM10] IBM Software. IBM ILOG CPLEX optimizer. Data sheet, IBM Corporation, 2010. URL: *http://www.ibm.com.*

[Joh73] D. S. Johnson. *Near-optimal bin packing algorithms.* PhD thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, 1973.

[KCT10] A. Khanafer, F. Clautiaux, and E.G. Talbi. New lower bounds for bin packing problems with conflicts. *European Journal of Operational Research*, 206(2):281–288, 2010.

[Kel60] J. E. Kelley. The cutting–plane method for solving convex programs. *J. of the Soc. for Industrial and Applied Mathematics*, 8(4):703–712, 1960.

[KPP04] H. Kellerer, U. Pferschy, and D. Pisinger. *Knapsack Problems.* Springer, Berlin, 2004.

[KZ51] L. V. Kantorovich and V. A. Zalgaller. *Calculation of Rational Cutting of Stock.* Lenizdat, Leningrad, 1951.

[Lue83] G. S. Lueker. Bin packing with items uniformly distributed over intervals [a,b]. In *Proc. 24th Ann. Symp. of Comp. Sci., Tucson*, pages 289–297, 1983.

[MF09] R. Misener and C. A. Floudas. Advances for the pooling problem: Modeling, global optimization, and computational studies. *Applied and Computational Mathematics*, 8(1):322, 2009.

[MT90] S. Martello and P. Toth. *Knapsack Problems: Algorithms and Computer Implementations.* John Wiley & Sons, Chichester-New York, 1990. URL: *http://www.or.deis.unibo.it/knapsack.html.*

[NW88] G. L. Nemhauser and L. A. Wolsey. *Integer and Combinatorial Optimization.* John Wiley and Sons, New York, 1988.

[PGLE82] F. Palacios-Gomez, L. Lasdon, and M. Engquist. Nonlinear optimization by successive linear programming. *Management science*, 28(10):1106–1120, 1982.

[PS07] D. Pisinger and M. M. Sigurd. Using decomposition techniques and constraint programming for solving the two-dimensional bin-packing problem. *INFORMS Journal on Computing*, 19(1):36–51, 2007.

[RAV10] J. Rietz, C. Alves, and J.M. Valério de Carvalho. Theoretical investigations on maximal dual feasible functions. *Operations Research Letters*, 38(3):174 – 178, 2010.

[RAV11] J. Rietz, C. Alves, and J.M. Valério de Carvalho. Worst-case analysis of maximal

dual feasible functions. *Optimization Letters*, 2011.

[RD08] J. Rietz and S. Dempe. Large gaps in one-dimensional cutting stock problems. *Discrete Applied Mathematics*, 156(10):1929 – 1935, 2008.

[RST02] J. Rietz, G. Scheithauer, and J. Terno. Families of non-IRUP instances of the one-dimensional cutting stock problem. *Discrete Applied Mathematics*, 121:229–245, 2002.

[SW10] C. Still and T. Westerlund. A linear programming-based optimization algorithm for solving nonlinear programming problems. *European Journal of Operational Research*, 200(3):658 – 670, 2010.

[WHS07] G. Wascher, H. Haußner, and H. Schumann. An improved typology of cutting and packing problems. *European Journal of Operational Research*, 183(3):1109–1130, 2007.

[ZKL85] J. Zhang, N.-H. Kim, and L. Lasdon. An improved successive linear programming algorithm. *Management science*, 31(10):1312–1331, 1985.