One-Dimensional Relaxations and LP Bounds for Orthogonal Packing

G. Belov, V. Kartak, H. Rohling, G. Scheithauer

MATH-NM-03-2009
February 2009
## Contents

1 Introduction  
   1.1 Formulations of OPP  
   1.2 Relaxations and bounds  
   1.3 Our contributions  

2 A New ILP Model of OPP  

3 1D Relaxations of OPP  
   3.1 The non-preemptive (contiguous) relaxation of OPP  
   3.2 The preemptive bar and slice relaxations  
   3.3 Non-preemptive vs. bar and slice relaxations  

4 Probing and Column Generation  

5 Experiments  
   5.1 2D OPP instances of Clautiaux et al. (2008)  
   5.2 2D SPP instances  
   5.3 3D OPP: comparison of bar and slice relaxations and DFFs  
      5.3.1 Generation of instances  
      5.3.2 The bounds tested  
      5.3.3 The results  
      5.3.4 Interpretation of the results  

6 Conclusions
ONE-DIMENSIONAL RELAXATIONS AND LP BOUNDS FOR ORTHOGONAL PACKING

G. Belov, V. Kartak, H. Rohling, G. Scheithauer

Submitted to ITOR in May 2008; revised in February 2009

Abstract

We consider the feasibility problem in $d$-dimensional orthogonal packing ($d \geq 2$), called Orthogonal Packing Problem (OPP): given a set of $d$-dimensional rectangular items, decide whether all of them can be orthogonally packed in the given rectangular container without item rotation. We review two kinds of 1D relaxations of OPP. The first kind is non-preemptive cumulative-resource scheduling, equivalently 1D contiguous stock cutting. The second kind is simple (preemptive) 1D stock cutting. In three and more dimensions we distinguish the so-called bar and slice preemptive relaxations of OPP. We review some models of these problems and compare the strength of their LP relaxations with regard to a certain OPP instance, theoretically and numerically. Both the theory and computational results in 2D and 3D show the advantage of the bar relaxation. We also compare the LP bounds to the commonly-used volume bounds from dual-feasible functions. Moreover, we test the so-called probing (temporary fixing) of intersection variables of OPP with the aim to strengthen the relaxations.

Keywords: packing, relaxation, modeling, conservative scales, dual-feasible function, probing

1 Introduction

Consider a set of $d$-dimensional items (boxes) that need to be packed into the given container. The input data describes the container sizes $W_k \in \mathbb{Z}_+$, $k = 1, d$, and the sizes of the $n$ items $w^k_i \in \mathbb{Z}_+$, $k = 1, d$ for each item $i \in I = \{1, \ldots, n\}$. Without loss of generality, we assume that each individual box fits into the container, i.e., $w^k_i \leq W_k$ holds for each box $i$ and dimension $k$. The guillotine constraint is not considered. The Orthogonal Packing Problem (OPP) \cite{FS04a, CJC08} asks whether all the boxes can be orthogonally packed into the container without rotations.

OPP is a subproblem in solution methods for orthogonal bin-packing (BPP) and knapsack problems (OKP), cf. \cite{FS04a, BB07, PS07}. OPP is polynomially equivalent to the orthogonal strip-packing problem (SPP) \cite{AVPT09}.

\footnote{Supported by the Dresden University of Technology in September 2007; by the German Academic Exchange Service (DAAD) and by the Russian Ministry of Education in autumn 2008.}
1 INTRODUCTION

1.1 Formulations of OPP

The non-linear natural model \cite{CJCM08, PS07} just states the non-overlapping and containment conditions, as follows:

Find a set of coordinates $x^k_i$, $k = 1, d$, $i = 1, n$, satisfying

\begin{align}
0 &\leq x^k_i \leq W_k - w^k_i & \forall k, i, \\
x^k_i + w^k_i &\leq x^k_j \quad \text{or} \quad x^k_j + w^k_j \leq x^k_i \quad \text{for at least one } k, \quad \forall i < j.
\end{align}

This very simple model can be fed to constraint programming software and, supplemented by bounds such as conservative scales or sweep line, leads to the best results today \cite{CJCM08, SO08, BCP08, PS07}.

There are many ILP models for OPP \cite{Bea85, Pad00, BB07}. Their exact solution is difficult because of weak LP bounds of some models \cite{Pad00} and because of the quadratic number of intersection variables and, in some models, the pseudo-polynomial number of position-indexed variables \cite{Bea85, BB07}.

The interval graph model \cite{FS04a} operates not with coordinates but with overlapping relations of the items along each axis. These relations are modeled by special graphs, namely interval graphs. The complements of such graphs allow transitive orientation leading to orderings of the items along each axis. Thus, both item orderings and coordinates are handled implicitly. This model seems to be the most general one; in particular, it removes some of the symmetries in guillotinely structured layouts.

1.2 Relaxations and bounds

To facilitate the OPP infeasibility decision, we can employ bounds. Bounds relate to relaxations (possibly quickly computable). The volume bound is the simplest one: if the total volume of the items exceeds the container volume then no packing is possible.

A general framework for relaxations of OPP are 1D relaxations. We introduce them on the example of a 2D OPP instance $(W, w) \in \mathbb{Z}_+^2 \times \mathbb{Z}_+^n$. The arguments can be straightforwardly extended to more dimensions (see Section 3). For convenience of the following explanations, denote $(L, B, l, b) = (W_1, W_2, w^1, w^2)$.

In the first step, we relax the item geometry in dimension $B$. This leads to non-preemptive cumulative-resource scheduling, equivalently to 1D Contiguous Cutting-Stock Problem (CCSP) \cite{MMV03, AVPT09, CJCM08}. In terms of scheduling, we have $n$ jobs and a single resource. Job $i$ consumes $l_i$ units of the resource for $b_i$ units of time. Each job should be processed without interruption (non-preemptiveness). The resource can be consumed cumulatively, but in any moment of time up to $L$ units. The goal is to schedule all jobs in $[0, B)$. An
example showing that this is a proper relaxation of 2D OPP is given in Figure 1.
A pseudo-polynomially large ILP model of CCSP was formulated in [Fv05] and

Figure 1: An orthogonal packing and a corresponding non-preemptive cumulative-resource schedule. The numbers denote item lengths \( l_i \), \( i = 1, n \).

in [BB07] model RFP.

A further step is to relax non-preemptiveness. Instead of \( n \) items (jobs) sized \( l_i \times b_i \), we consider \( n \) types of items with sizes \( l_i \times 1 \); for each type \( i = 1, n \), \( b_i \) copies should be packed. This leads to a special 1D Cutting-Stock Problem (CSP) [GG61, AV08]: given stock bars of size \( L \) and \( n \) item types of size \( l_i \) and order demands \( b_i \), \( i = 1, n \), obtain all the items from \( B \) stock bars so that each bar has at most one item of each type.

For CSP, the strongest model seems to be the set-partitioning or Gilmore-Gomory model [GG61] with \( n \) constraints but non-polynomially many variables. It has a very small LP–IP gap, probably always smaller than the absolute value of two units [RST02]. It also provides efficient bounds for 2D packing [Sch99, LA01, BB07].

Conservative scales (CS) [FS04b, CAdC08] are relaxed item sizes in a certain dimension. Every 1D knapsack remains feasible when taking CS instead of the original sizes. Thus, CS represent feasible dual solutions of the set-partitioning model of the corresponding 1D relaxation. The volume bound using CS instead of the original sizes is a valid bound for OPP. Moreover, [CLM05] propose an exact algorithm to find a pair of CS maximizing the volume bound in the 2D case. Other authors compute CS heuristically using dual-feasible functions (DFF). In the 2D case, the volume bound from heuristically computed CS is usually weaker than the bound from the set-partitioning model, cf. [BB07], which is also confirmed by our results. In the 3D case this is the case for instances with small waste, see Section 5.

A useful preprocessing technique in exact algorithms is probing [Mar01]. A binary variable is fixed to 0 or 1 and the bound is recomputed; if the bound reports infeasible, the variable is constantly fixed to the opposite value. This can strengthen the model. Probing is related to strong branching [AKM05]. Probing on item ordering variables in OPP was used in [PS07].
1.3 Our contributions

Section 2 presents a new ILP model of OPP. Like all ILP models known today, it is too large for exact solution. Section 3 discusses one-dimensional relaxations of OPP. First, the OPP model from Section 2 is relaxed and we show that the resulting model (already known from [Fv05]) represents CCSP. Then we propose a new model of CCSP. Section 3.2 investigates two 1D CSP relaxations of OPP, bar and slice relaxations. Section 3.3 theoretically compares the LP relaxations of the first CCSP model and of the set-partitioning formulation of the bar CSP using a certain network flow reformulation. Section 4 introduces probing on item intersection variables and discusses its implementation in the set-partitioning model, in particular column generation. Section 5 gives numerical results for 2D and 3D instances, especially a comparison to DFFs.

2 A New ILP Model of OPP

Let us consider the coordinate of item \( i \) in dimension \( k \) to be the smallest integer position occupied by \( i \) in that dimension. Then for each dimension \( k = 1, d \) and for each item \( i = 1, n \), the set of possible coordinates is

\[
s \in A(k, i) = \{0, \ldots, W_k - w_k^i\}.
\]

Consider the binary variables \( z^k_{is} \) indicating whether the \( k \)-th coordinate of item \( i \) is \( s \). The position variables are similarly defined for the 2D case in [BB07, model FP]. However, the non-overlapping condition will be checked here in another fashion: for each dimension \( k = 1, d \) and for each ordered item pair \( 1 \leq i < j \leq n \), define a binary variable \( \delta^k_{ij} \). The value \( \delta^k_{ij} = 0 \) indicates that the projections of the interior of items \( i \) and \( j \) onto the \( k \)-th axis do not intersect. To correctly constrain the \( \delta^k_{ij} \) variables, note that if \( z^k_{is} = 1 \) then the range of coordinates of item \( j \) where it \( k \)-overlaps with \( i \) is

\[
\alpha(k, s, i, j) = \left\{ \max\{s - w^k_j + 1, 0\}, \ldots, \min\{s + w^k_i - 1, W_k - w^k_j\} \right\}.
\]

Thus, \( z^k_{is} = 1 \) and \( z^k_{jt} = 1 \) for some \( t \in \alpha(k, s, i, j) \) imply \( \delta^k_{ij} = 1 \). The model reads as follows:

\[
\begin{align*}
\sum_{s \in A(k, i)} z^k_{is} &= 1 & i = 1, n, & k = 1, d, \\
\sum_{s \in A(k, i)} z^k_{is} + \sum_{t \in \alpha(k, s, i, j)} z^k_{jt} - 1 & \leq \delta^k_{ij} & s \in A(k, i), & \forall i < j, \forall k, \\
\sum_{k=1}^d \delta^k_{ij} & \leq d - 1 & \forall i < j, \\
z^k_{is} & \in \{0, 1\} & s \in A(k, i), & \forall i, k, \\
\delta^k_{ij} & \in \{0, 1\} & \forall i < j, & \forall k.
\end{align*}
\]
3 1D RELAXATIONS OF OPP

Equations (2a) ensure containment of each item in each dimension. Inequalities (2b) guarantee proper definition of item projection overlapping in each dimension. Inequalities (2c) ensure non-overlapping in at least one dimension. Note that inequalities (2b) can be replaced by

$$\delta_{ij}^k \geq \sum_{t=\max\{s-w_i^k+1,0\}}^s z_{it}^k + \sum_{t=\max\{s-w_j^k+1,0\}}^s z_{jt}^k - 1$$

$$s \in \{\min\{w_i^k, w_j^k\} - 1, \ldots, W_k - \min\{w_i^k, w_j^k\}\}, \quad \forall i < j, \forall k.$$  (2f)

While the constraints (2b) register overlapping only in the first position occupied by item $i$, in (2f) it happens on the whole overlapping interval along the $k$-th axis.

With the goal to obtain the so-called normalized (left-justified, gapless) packings, we can add the conditions

$$\sum_i z_{is}^k \geq 1 \quad \forall k,$$  (2g)

$$n \sum_{s:s-w_i^k \geq 0} z_{is}^k \geq \sum_i z_{is}^k \quad s \in \{\min_i w_i^k, \ldots, W_k - \min_i w_i^k\}, \forall k.$$  (2h)

cf. [BB07]; i.e., at least one item should start at coordinate 0 and an item can start only where another one ends. Moreover, to reduce symmetry, we can impose

$$\sum_{s=0}^\left\lfloor (W_k-w_1^k)/2 \right\rfloor z_{1s}^k = 1 \quad \forall k.$$  (2i)

e.i., the center of item 1 should be located in the first half of the bin in each dimension.

The model (2) has $O(n \sum_k W_k + dn^2)$ variables and $O(n^2 \sum_k W_k)$ constraints. The model RF in [BB07] (if generalized to $d$ dimensions) has $O(n \prod_k W_k + n \sum_k W_k)$ variables and $O(\prod_k W_k + n \sum_k W_k)$ constraints.

Note that the number of variables can sometimes be reduced by properly selecting the candidate coordinate sets [the reduced set of raster points, cf. Sch08]. This is a strengthening of the normalized positions [cf. BB07].

The pseudo-polynomial number of variables makes the exact solution of this model difficult. Already for some problems with 10 items and $W_1 = W_2 = 20$, CPLEX 9.0 needed very long time. Especially if the container size is not tight (such that some sizes can be reduced and the instance remains feasible), the solver needed much longer time to find a feasible solution. This can probably be explained by the symmetries in the model. In [BB07], the similar model FP was used, also requiring long running times.

3 1D Relaxations of OPP

This section discusses preemptive and non-preemptive 1D relaxations of OPP. The preemptive relaxation allows for more freedom as to which dimensions from OPP
to aggregate; we distinguish between bar and slice relaxations, although starting with 4D OPP, further combinations are possible. ILP models of the relaxations are discussed. Moreover, LP relaxations of these models with regard to OPP are theoretically compared.

We consider an instance of OPP defined by \((W, w) \in \mathbb{Z}^d \times \mathbb{Z}^d_+\). The 1D relaxations are defined using the data \((L, B, l, b) \in \mathbb{Z}^{1+1+n+n}_+\) given by

\[
(L, B, l, b) = \left( \prod_{k \neq k_0} W_k, \ W_{k_0}, \ (\prod_{k \neq k_0} w_k^{i})_{i=1}^{n}, \ w_{k_0} \right)
\]

for a certain \(k_0 \in \{1, \ldots, d\}\).

### 3.1 The non-preemptive (contiguous) relaxation of OPP

In this section we review an ILP model of the decision version of 1D contiguous stock cutting (CCSP) and propose a new model. Let the CCSP instance be defined by \((L, B, l, b)\) as in (3). The task is to schedule all jobs in the interval \([0, B)\), so that at any moment the resource consumption is at most \(L\). This CCSP is a relaxation of the OPP instance \((W, w)\). An ILP model of CCSP can be obtained as a surrogate relaxation of model (2). The set of possible starting times of job \(i\) is given by

\[
A(i) = A(k_0, i) = \{0, \ldots, B - b_i\} \quad \text{for} \quad i = \overline{1, n}.
\]

Surrogate relaxing the containment conditions (2a) for \(k \neq k_0\) together with the non-intersection conditions (2b), (2c) leads to the 1D knapsack constraints (4b) in the following model:

\[
\begin{align*}
\sum_{s \in A(i)} z_{is} &= 1 & i &= \overline{1, n}, \\
\sum_{i} l_i \sum_{s \geq \max(s - b_i + 1, 0)} z_{it} &\leq L & s \in \{0, \ldots, B - 1\}, \\
&\quad z_{is} \in \{0, 1\} & s \in A(i), \ \forall i.
\end{align*}
\]

Equations (4a) ensure containment of each job in the interval \([0, B)\). The 1D knapsack condition (4b) ensures that for each time point \(s \in [0, B)\), the total resource consumption does not exceed \(L\). This model was already proposed in [Fv05] and in [BB07] model RFP.

We can observe that the OPP model (2) is a combination of \(d\) CCSP models (4) and the non-overlapping conditions. Thus, any OPP layout can be seen as a system of \(d\) CCSP schedules satisfying non-overlapping conditions. This observation has been used in approaches, e.g., [CJCM08].

Another CCSP model can be constructed as follows. The variables \(z_{is} \in \{0, 1\}, s \in A(i), \ \forall i\) remain the same, i.e., denote that job \(i\) starts at time point
s. Let variables \( a_{is} \in \{0, 1\} \), \( s \in \{0, \ldots, B - 1\} \), \( \forall i \) denote that a part of job \( i \) is executed at time period \([s, s + 1)\). The knapsack condition (4b) is relaxed to consider only what is happening at time point \( s \). To ensure contiguity, we impose condition (5d) in the following model:

\[
\sum_{s \in A(i)} z_{is} = 1 \quad \forall i,
\]

\[
\sum_{i} l_i a_{is} \leq L \quad s \in \{0, \ldots, B - 1\}, \forall i,
\]

\[
a_{i0} = z_{i0} \quad \forall i,
\]

\[
a_{is} \leq a_{i,s-1} + z_{is} \quad s \in \{1, \ldots, B - 1\}, \forall i,
\]

\[
\sum_{s=0}^{B-1} a_{is} = b_i \quad \forall i,
\]

\[
z_{is} \in \{0, 1\} \quad s \in A(i), \forall i,
\]

\[
a_{is} \in \{0, 1\} \quad s \in \{0, \ldots, B - 1\}, \forall i.
\]

Again, equalities (5a) ensure containment in the \( B \)-direction, inequalities (5b) ensure containment in the \( L \)-direction, equalities (5c) guarantee that job \( i \) starts at 0 iff \( z_{i0} = 1 \), inequalities (5d) ensure that if not started at 0, job \( i \) starts exactly at \( s \) with \( z_{is} = 1 \). Moreover, the jump from \( a_{i,s-1} = 0 \) to \( a_{is} = 1 \) occurs exactly once which ensures contiguity. Finally, equalities (5e) guarantee proper length of job \( i \). This is a Kantorovich-type model, cf. [Val02].

Note that in both models (4) and (5) we can restrict the number of open stacks (i.e., number of started and not yet finished jobs at any moment, cf. [BS07]) to any positive integer \( K \):

\[
\sum_{i} \sum_{t=\max\{s-b_i+1,0\}}^{s} z_{it} \leq K \quad s \in \{\min_i b_i - 1, \ldots, B - \min_i b_i\} \quad (6)
\]

or

\[
\sum_{i} a_{is} \leq K \quad s \in \{0, \ldots, B - 1\}, \quad (7)
\]

for the models (4) and (5), respectively. To reduce symmetry, we can add constraints (2g)–(2i).

### 3.2 The preemptive bar and slice relaxations

Let \((L, B, l, b)\) describe an instance of the decision version of the following 1D cutting-stock problem (CSP): the task is to produce \( b_i \) items of width \( l_i \), \( i = 1, n \), from \( B \) bars of width \( L \), so that each bar contains at most one item of each type. As in the previous section, let the data \((L, B, l, b)\) be defined as in (3):

\[
(L, B, l, b) = \left( \prod_{k \neq k_0} W_k, \quad W_{k_0}, \quad \left( \prod_{k \neq k_0} w_k^i \right)_{i=1}^n, \quad w_{k_0} \right) \quad (8)
\]

for some \( k_0 \in \{1, \ldots, d\} \). The CSP instance \((L, B, l, b)\) is a relaxation of the OPP instance \((W, w)\). Let us call it the \( k_0 \)-th slice relaxation because the packing of
each stock bar of length $L$ represents a 1D relaxation of a $(d-1)$-dimensional slice which is orthogonal to axis $k_0$.

Now consider the “transposed” CSP instance $(B, L, b, l)$. It is as well a relaxation of $(W, w)$. Let us call it the $k_0$-th bar relaxation [cf. Sch99] because the packing of each stock bar of length $B$ represents a possible 1D stitch through an OPP layout along the axis $k_0$. A scheme for slice and bar relaxations is shown in Figure 2.

Starting with the four-dimensional OPP, we can construct further dimension combinations in (8). This will not be considered here.

**Example 1.** Consider the 3D OPP instance with container $W = (5, 5, 5)$ and nine equal items with sizes $w_1 = \cdots = w_9 = (2, 2, 2)$ [PS04b]. The 1st (2nd, 3rd) slice relaxation is $(L, B, l, b) = (25, 5, (4, \ldots, 4), (2, \ldots, 2))$, see (8). The 1st (2nd, 3rd) bar relaxation is $(B, L, b, l) = (5, 25, (2, \ldots, 2), (4, \ldots, 4))$.

Let us describe the set-partitioning model of CSP [GG61]. Let the CSP instance be given by $(L, B, l, b)$. We introduce the notion of 1D cutting pattern. This is any binary vector $a \in \{0, 1\}^n$ satisfying the knapsack constraint

$$\sum_{i=1}^{n} l_i a_i \leq L. \quad (9)$$

Note that in the standard CSP, the patterns $a$ are general non-negative integer vectors [GG61]. Binary patterns are meaningful when CSP is considered as a relaxation of OPP.

Let the columns of matrix $A = A(L, l) \in \{0, 1\}^{n \times \eta}$ represent all possible cutting patterns (9). Note that matrix $A$ is a constant. The variables are given by vector $x \in \mathbb{Z}^\eta_+$ whose elements represent the usage of the corresponding columns in the solution. The set-partitioning model of CSP is as follows:

$$1x \leq B, \quad (10a)$$

s.t.

$$Ax = b, \quad (10b)$$

$$x \in \mathbb{Z}^\eta_+. \quad (10c)$$
Constraint (10a) says that the total number of stock bars $\sum_{j=1}^{\eta} x_j$ should not exceed $B$. Constraints (10b) say that the exact number of copies of each item type is produced.

Using model (10), we can model the $k_0$-th slice relaxation for $(W, w)$ according to (8) as

$$\Omega(L, B, l, b) = \{ x \in \mathbb{Z}^{\eta+n} : A(L, l)x = b, \ 1x \leq B \} \quad (11)$$

and the $k_0$-th bar relaxation for $(W, w)$ as

$$\Omega(B, L, b, l) = \{ x \in \mathbb{Z}^{n+b} : A(B, b)x = l, \ 1x \leq L \}. \quad (12)$$

Consequently, if $\Omega(L, B, l, b) = \emptyset$ or $\Omega(B, L, b, l) = \emptyset$ for some $k_0 \in \{1, \ldots, d\}$, the OPP $(W, w)$ is infeasible. However, these models can be hard to solve and in practice we would use their LP relaxations which are very strong both in the 1D case [RST02] and as bounds for orthogonal packing [BB07]. Moreover, in the next subsection we compare the bar LP relaxation to the LP relaxation of the CCSP model (4). The computation features, including column generation, are discussed in Section 4.

**Example 1 (continued).** Consider again the 3D OPP instance $W = (5, 5, 5), w_i = (2, 2, 2), i = 1, 9$. The slice relaxation is $(25, 5, (4, \ldots, 4), (2, \ldots, 2))$. Its stock-minimal solution uses 3 bars of width 25, i.e., the total stock consumption is 75 units. The bar relaxation is $(5, 25, (3, \ldots, 3), (9, \ldots, 9))$. Its stock-minimal solution uses 18 bars of width 5, i.e., the total stock consumption is 90. We can say that the bar relaxation is stronger in this example.

**Example 2.** Consider another 3D OPP instance with bin $W = (4, 4, 4)$ and two items $w_i = (3, 3, 3), i = 1, 2$. The slice relaxation is $(16, 4, (9, 9), (3, 3))$. Its stock-minimal solution uses 6 bars of width 16, i.e., the total stock consumption is 96 units. The bar relaxation is $(4, 16, (3, 3), (9, 9))$. Its stock-minimal solution uses 18 bars of width 4, i.e., the total stock consumption is 72. Here slice relaxation is stronger than bar relaxation.

Examples 1 and 2 show:

**Observation 1.** In general, the bar and slice relaxations are incomparable.

In Section 5 we compare them numerically.

### 3.3 Non-preemptive vs. bar and slice relaxations

In this section we show that, for a certain OPP instance, the bar LP relaxation is stronger than the CCSP LP relaxation using CCSP model (4).

CSP is a relaxation of CCSP. This concerns the slice relaxation:
Lemma 1. For any \( k_0 \in \{1, \ldots, d\} \), the \( k_0 \)-th slice relaxation is feasible if the \( k_0 \)-th CCSP relaxation is feasible.

Proof. For any data \((L, B, l, b) \in \mathbb{Z}^{1+1+n+n}_+\), the feasibility of the models (4) or (5) implies that of the set-partitioning model (10). \qed

However, in practical algorithms we would use LP relaxations of models of CCSP and CSP (unless in the root node of enumeration, cf. [BB07]). Let us compare the LP relaxations of (4) and (12). For that, we introduce the so-called standard 1D cutting-stock problem (sCSP), while CSP described by model (10) shall be called the binary CSP (just CSP). The difference is that sCSP allows general integer patterns, cf. [GG61]: each pattern \( a \) is element of \( \mathbb{Z}^n_+ \) satisfying the same knapsack condition as in (9):

\[
\sum_{i=1}^n l_i a_i \leq L. \tag{13}
\]

Let the data \((L, B, l, b)\) be defined as in (8). Using the previous subsection, we define the \( k_0 \)-th (binary / integer) slice LP relaxation for \((W, w)\) as

\[
\Omega^\text{LP}_{u}(L, B, l, b) = \{ x \geq 0 : A_{u}(L, l)x = b, \ 1x \leq B \} \tag{14}
\]

and the \( k_0 \)-th (binary / integer) bar LP relaxation for \((W, w)\) as

\[
\Omega^\text{LP}_{u}(B, L, b, l) = \{ x \geq 0 : A_{u}(B, b)x = l, \ 1x \leq L \} \tag{15}
\]

with \( u \in \{\text{bin, int}\} \) for binary or integer patterns and a corresponding matrix \( A \). Finally, the \( k_0 \)-th CCSP LP relaxation of \((W, w)\) (based on model (4)) is

\[
\Phi^\text{LP}(L, B, l, b) = \{ z_{is} \in [0, 1], \forall i, \forall s \in A(i) \text{ satisfying (4a), (4b)} \} . \tag{16}
\]

Theorem 2. For any \( k_0 \in \{1, \ldots, d\} \), the \( k_0 \)-th CCSP LP relaxation is feasible if and only if the \( k_0 \)-th integer bar LP relaxation is feasible. More generally, for each data set \((L, B, l, b)\) (e.g., obtained by (8)), we have the equivalence

\[
\Phi^\text{LP}(L, B, l, b) \neq \emptyset \iff \Omega^\text{LP}_{\text{int}}(B, L, b, l) \neq \emptyset.
\]

Idea of the proof. In Figures [1, 3] we see that the CCSP schedule can be decomposed into horizontal 1D patterns, i.e., columns of \( A_{\text{bin}}(L, l) \). This is the correspondence to the slice relaxation, see Lemma [1]. In fact it can also be transformed into a set of vertical patterns (columns of \( A_{\text{bin}}(B, b) \)) by splitting the items in \( L \)-direction, Figure [3]. The knapsack condition with regard to \( B \) remains valid.

We are going to show that for the LP relaxations, the latter transformation is an equivalence relation: any solution of the LP (16) can be transformed into vertical integer patterns (columns of \( A_{\text{int}}(B, b) \)) and, vice versa, any solution of the “transposed” LP (15) can be transformed into a solution of (16). The proof uses the arc-flow model of sCSP [Val02].
Proof. Consider $\Phi^{LP}(L, B, l, b)$ (16). Because $z_{is}$ are continuous variables, we can substitute them by

$$y_{is} = l_i z_{is} \quad \forall i, \forall s \in A(i).$$

Moreover, we introduce slack variables

$$r_s \in \mathbb{R}_+ \quad \forall s \in \{0, \ldots, B-1\}$$

for the knapsack constraint (14b), which denote the amount of unused resource at time point $s$. Thus, the LP model $\Phi^{LP}(L, B, l, b)$ is equivalent to this one:

$$\sum_{s \in A(i)} y_{is} = l_i \quad i = 1, n,$$

$$\sum_i \sum_{t=\max\{s-b_i+1,0\}}^s y_{it} + r_s = L \quad s \in \{0, \ldots, B-1\},$$

$$y_{is} \in [0, l_i] \quad s \in A(i), \forall i,$$

$$r_s \in \mathbb{R}_+ \quad s \in \{0, \ldots, B-1\}.$$  

The model (19) can be seen as a network flow problem in the following network: let the node set be $\{0, \ldots, B\}$, the arc set $(s, s+b_i)$ for $i = 1, n$, $s \in A(i)$ (regular arcs) and $(s, s+1)$ (slack arcs). Variable $y_{is}$ represents flow in the arc $(s, s+b_i)$ and $r_s$ represents flow in the arc $(s, s+1)$. Equations (19a) state that the total flow in all arcs $(s,s+b_i)$ equals $l_i$ for any $i$. Equations (19b) demand that, for each $s \in \{0, \ldots, B-1\}$, the flow “passing over” the time period $[s,s+1)$ equals $L$. This includes

- the flow arriving at $s+1$: $\delta_+(s+1) = \sum_{i:s-b_i+1\geq0} y_{i,s-b_i+1} + r_s$;
- the flow arriving later: $\bar{\delta}(s+1) = \sum_i \sum_{t=\max\{s-b_i+2,0\}} y_{it}$.

Let us ask how much flow departures at time $s+1$ for $s \in \{0, \ldots, B-2\}$. It is

- the flow leaving from $s+1$: $\delta_-(s+1) = \sum_i y_{i,s+1} + r_{s+1}$. 

Figure 3: A schedule represented by horizontal patterns (left) and its decomposition into vertical patterns (right). Numbers denote item lengths $l_i$, $i = 1, n$, see Figure 1.
Because of (19b) we have $\sum \sum s \ y_{it} + \sum y_{i,s+1} + r_{s+1} = L$ or

$$\delta(s + 1) + \delta_-(s + 1) = L, \quad s \in \{0, \ldots, B - 2\},$$

which means

$$\delta_+(s) = \delta_-(s), \quad s \in \{1, \ldots, B - 1\},$$

i.e., the flow conservation property for all inner nodes, see Figure 4.

Figure 4: The flow conservation property $\delta_+(s + 1) = \delta_-(s + 1)$ in model (19).

Thus, model (19) is the LP relaxation of the arc-flow model [Val02] of the sCSP instance $(B, L, b, l)$ with integer patterns. Equations (19a) can be seen as the demand constraints. In particular, any solution of (19) can be decomposed into paths representing integer patterns of $A_{int}(B, b)$ (by the flow decomposition property). And vice versa, any LP solution of the sCSP $(B, L, b, l)$ transforms into a solution of $\Phi_{LP}(L, B, b, l)$ (16).

Because of the obvious dominance

$$\Omega^{LP}_{bin}(B, L, b, l) \neq \emptyset \quad \Rightarrow \quad \Omega_{LP}^{int}(B, L, b, l) \neq \emptyset,$$

we obtain

**Corollary 3.** For any $k_0 \in \{1, \ldots, d\}$, the $k_0$-th binary bar LP relaxation dominates the $k_0$-th CCSP LP relaxation, i.e., for each data set $(L, B, l, b)$ we have

$$\Omega^{LP}_{bin}(B, L, b, l) \neq \emptyset \quad \Rightarrow \quad \Phi^{LP}(L, B, b, l) \neq \emptyset.$$

Thus, the bar LP relaxation dominates the non-preemptive LP relaxation (based on model (4)). Moreover, Theorem 2 suggests a method to solve model (4) in integers, namely by branch-and-price similarly to [Val02, AV08].
4 Probing and Column Generation

A useful preprocessing technique in exact algorithms is probing [Mar01]. A binary variable is fixed to 0 or 1 and the bound is recomputed; if the bound reports infeasible, the variable is constantly fixed to the opposite value. To be useful, the technique needs a bound which can be strengthened by such fixings.

We can apply probing to the variables \( \delta_{ij}^k \), \( k = 1, d \), \( i < j \) from model (2). Their meaning is the following: \( \delta_{ij}^k = 0 \) implies that the projections of the interior of items \( i \) and \( j \) on axis \( k \) are disjunct. We can temporarily fix some \( \delta_{ij}^k \) to 0 and recompute the bound; if infeasible, it is permanently fixed to 1. Note that if we permanently fix to 1 all \( \delta_{ij}^k \) for given \( i, j \) and all \( k = 1, d \), the non-overlapping constraint (2c)

\[
\sum_{k=1}^d \delta_{ij}^k \leq d - 1 \quad \forall i < j
\]

implies infeasibility of the OPP instance. We use this constraint even earlier: if for some \( i, j \), \( \delta_{ij}^k = 1 \) is fixed for \( d - 1 \) values of \( k \), then we can fix \( \delta_{ij}^{k_0} = 0 \) for the remaining dimension \( k_0 \). This can be seen as constraint propagation or preprocessing [Mar01] or feasible augmentations [FSvdV07].

Based on Corollary 3 we are going to use the bar LP relaxation for bounds. To strengthen it when fixing some \( \delta_{ij}^k \) to 0, let us note that if \( \delta_{ij}^k = 0 \) for some \( i, j, k \), then only patterns of the \( k \)-th bar relaxation can have items \( i \) and \( j \) together, see Figure 5.

![Figure 5](image-url)

Figure 5: Suppose \( \delta_{ij}^1 = 0 \), then only patterns of the 1-st bar relaxation (horizontal stitches) can have items \( i \) and \( j \) together.

To strengthen the bar relaxation for certain pairs \((i, j)\) with \( \delta_{ij}^{k_0} = 0 \), we need to exclude patterns containing \( i \) and \( j \) together in all \( k \)-th bar relaxations for \( k \neq k_0 \).

We solve the LP relaxation of the set-partitioning formulation of CSP (10) by column generation. In [BB07], for small problems, it was possible to generate the complete set of patterns a priori. For practically relevant problems this is not acceptable. Then the forbidden pairs \((i, j)\) have to be considered during column
generation. This leads to a 1D binary knapsack problem with forbidden item pairs. In [VBJN94] it was proposed to solve problems with disjunct forbidden pairs by a modified branch-and-bound procedure and otherwise as a general IP problem. We experimented with the CBC IP solver (http://www.coin-or.org) but the fastest appeared a modified branch-and-bound procedure where forbidden pairs are quickly stored in bit arrays. Moreover, we do not look for the best column: after a column with negative reduced cost is found, we exit.

5 Experiments

At first we consider 2D OPP and 2D SPP instances from the literature. Then we use randomly generated 3D OPP instances to compare bar, slice, and volume bound relaxations from dual-feasible functions (DFF). The experiments were performed on modern PCs.

5.1 2D OPP instances of Clautiaux et al. (2008)

At first we considered the 27 infeasible OPP instances from [CJCM08]. (In their paper, only 41 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 \( n \leq 23 \). [CJCM08] solved each instance in a few seconds.

For the 27 infeasible instances, we compared the bar LP relaxation (pure and strengthened by probing) with the volume bound from conservative scales (obtained from DFFs and from the exact algorithm of [CLM05]). We considered the DFFs given in Table 1. The maximal volume was calculated using all pairs of

\[
 v = \max_{g_1, g_2 \in \{id, u_k^i, U^\epsilon, \phi^\epsilon, f_{ik}^1, f_{ik}^2\}} \sum_{i=1}^{n} \frac{g_1(w_{1i})g_2(w_{2i})}{g_1(W_1)g_2(W_2)}. \tag{20}
\]

To compute \( u_k^i, U^\epsilon, \) and \( \phi^\epsilon \), the sizes were scaled so that \( W_k = 1, \forall k \).
5 EXPERIMENTS

The code of [CLM05] 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. The results are given in Table 2.

Table 2: The 27 infeasible instances from [CJCM08]

<table>
<thead>
<tr>
<th>Instance</th>
<th>LP$^{bar}_1$</th>
<th>LP$^{bar}_2$</th>
<th>probing</th>
<th>v (BiLin)</th>
<th>v (DFF)</th>
</tr>
</thead>
<tbody>
<tr>
<td>00N10</td>
<td>21</td>
<td>20</td>
<td>0</td>
<td>1,050</td>
<td>1,05</td>
</tr>
<tr>
<td>00N15</td>
<td>20</td>
<td>20</td>
<td>82</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>00N23</td>
<td>20,1</td>
<td>20</td>
<td>0</td>
<td>1,002</td>
<td>1</td>
</tr>
<tr>
<td>00X23</td>
<td>19,7</td>
<td>20</td>
<td>97</td>
<td>1,083</td>
<td>0,99</td>
</tr>
<tr>
<td>03N10</td>
<td>20,5</td>
<td>20</td>
<td>0</td>
<td>1,125</td>
<td>1</td>
</tr>
<tr>
<td>03N15</td>
<td>20</td>
<td>19,4</td>
<td>80</td>
<td>1</td>
<td>0,97</td>
</tr>
<tr>
<td>03N16</td>
<td>19,8</td>
<td>19,4</td>
<td>−40</td>
<td>1</td>
<td>0,97</td>
</tr>
<tr>
<td>03N17</td>
<td>19,8</td>
<td>20</td>
<td>100</td>
<td>1</td>
<td>0,97</td>
</tr>
<tr>
<td>04N15</td>
<td>20,1</td>
<td>19,7</td>
<td>0</td>
<td>1,009</td>
<td>0,96</td>
</tr>
<tr>
<td>04N17</td>
<td>20,3</td>
<td>20</td>
<td>0</td>
<td>1,037</td>
<td>0,97</td>
</tr>
<tr>
<td>04N18</td>
<td>20,08</td>
<td>20</td>
<td>0</td>
<td>1,005</td>
<td>0,96</td>
</tr>
<tr>
<td>05N15</td>
<td>19,7</td>
<td>19</td>
<td>99</td>
<td>1,071</td>
<td>0,98</td>
</tr>
<tr>
<td>05N17</td>
<td>20</td>
<td>20</td>
<td>−90</td>
<td>1</td>
<td>0,95</td>
</tr>
<tr>
<td>05X15</td>
<td>20,3</td>
<td>19</td>
<td>0</td>
<td>1,017</td>
<td>0,96</td>
</tr>
<tr>
<td>07N10</td>
<td>20</td>
<td>18,8</td>
<td>33</td>
<td>1</td>
<td>0,93</td>
</tr>
<tr>
<td>07N15</td>
<td>21</td>
<td>20</td>
<td>0</td>
<td>1,077</td>
<td>1,05</td>
</tr>
<tr>
<td>07X15</td>
<td>19</td>
<td>18,6</td>
<td>−47</td>
<td>1</td>
<td>0,93</td>
</tr>
<tr>
<td>08N15</td>
<td>19,3</td>
<td>20</td>
<td>90</td>
<td>1</td>
<td>0,92</td>
</tr>
<tr>
<td>10N10</td>
<td>20,5</td>
<td>20</td>
<td>0</td>
<td>1,056</td>
<td>1</td>
</tr>
<tr>
<td>10N15</td>
<td>20,5</td>
<td>20</td>
<td>0</td>
<td>1,031</td>
<td>0,97</td>
</tr>
<tr>
<td>10X15</td>
<td>18,5</td>
<td>18</td>
<td>88</td>
<td>1</td>
<td>0,9</td>
</tr>
<tr>
<td>13N10</td>
<td>19,5</td>
<td>17,6</td>
<td>−48</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>13N15</td>
<td>20</td>
<td>20</td>
<td>−84</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>13X15</td>
<td>20</td>
<td>18,3</td>
<td>79</td>
<td>1,167</td>
<td>1</td>
</tr>
<tr>
<td>15N10</td>
<td>18</td>
<td>19,3</td>
<td>28</td>
<td>1</td>
<td>0,85</td>
</tr>
<tr>
<td>15N15</td>
<td>20</td>
<td>20</td>
<td>88</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

solved: 10 0 21 13 2

The columns LP$^{bar}_1$ and LP$^{bar}_2$ contain the minimal length of 1D solution for the 1st and 2nd bar LP relaxation, respectively. The 1st bar LP relaxation solved 10 instances. The 2nd solved none. This can be explained by the fact that item sizes in the 1st dimension are mostly larger, which makes the set-partitioning formulation stronger. The bar relaxations combined by probing took up to half a second and solved 21 instances (the column contains the numbers of variables δ$^{k}_{ij}$ permanently fixed to 1; if the instance was not solved, the number is negative). The code from [CLM05] solved 13 instances, 12 of them rather quickly and one more (E00N23) after 30 seconds. Note that among the 13 instances are the 10 ones solved by the bar relaxation. The volume bound from DFFs solved 2 instances.
5.2 2D SPP instances

Further, we applied probing to 2D SPP. Given strip width $W$, our goal is to minimize the used length $L$. Starting with the value of $L$ given by the pure bar relaxation, we apply probing to container size $(W, L)$. If it fails, we set $L \leftarrow L + 1$ and iterate.

We considered the 500 SPP instances from [BW87, MV98], classes C01–C10. [Bor06] reports lower bound Alb for these instances. Alb is in fact a lower bound for 1D contiguous stock cutting (see Section 3.1) computed by an exact algorithm from [MMV03]. The exact algorithm had to be interrupted in most cases, that is why the theoretically stronger bound dominates the bar relaxation in only a few instances.

We computed the pure bar relaxation and bar relaxation followed by probing. The per cent increase of the bound compared to [Bor06] is shown in Figure 6.Instances with item numbers $n = 20, 40, 60, 80, 100$ are considered separately.

<table>
<thead>
<tr>
<th>$n$</th>
<th>LP</th>
<th>LP + probing</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>1.34</td>
<td>1.57</td>
</tr>
<tr>
<td>40</td>
<td>0.57</td>
<td>0.66</td>
</tr>
<tr>
<td>60</td>
<td>0.42</td>
<td>0.48</td>
</tr>
<tr>
<td>80</td>
<td>0.37</td>
<td>0.40</td>
</tr>
<tr>
<td>100</td>
<td>0.24</td>
<td>0.27</td>
</tr>
</tbody>
</table>

Figure 6: The increase of the lower bound compared to [Bor06], per cent

In [BSM08] there are cited feasible solutions produced by the heuristic SVC(SubKP). We tested in how many cases the three bounds prove optimality of these solutions. The numbers are given in Figure 7. For each value of $n$ there are 100 instances. We see that almost 50 instances could be proved optimal for each $n$. For others, either the bound is too weak or the solution is not optimal. The details on these results can be obtained on the CaPaD webpage http://www.math.tu-dresden.de/~capad, section Test instances and results.

Finally, in Table 3 we give the times needed for probing. We give the average time needed for an instance of each size $n$. In our straightforward implementation, we tested all candidate pairs and the times are quite high. For example, for an instance with $n = 100$, about 5000 LPs had to be resolved. In an exact algorithm we would perform probing only in the root and on a smaller set of pairs.
5 EXPERIMENTS

<table>
<thead>
<tr>
<th>n</th>
<th>Alb</th>
<th>LP</th>
<th>LP + probing</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>30</td>
<td>41</td>
<td>47</td>
</tr>
<tr>
<td>40</td>
<td>34</td>
<td>43</td>
<td>48</td>
</tr>
<tr>
<td>60</td>
<td>33</td>
<td>43</td>
<td>46</td>
</tr>
<tr>
<td>80</td>
<td>31</td>
<td>43</td>
<td>47</td>
</tr>
<tr>
<td>100</td>
<td>29</td>
<td>40</td>
<td>42</td>
</tr>
</tbody>
</table>

Figure 7: The number of heuristic solutions proved optimal

Table 3: Average probing times for instances of each size $n$, seconds

<table>
<thead>
<tr>
<th>n</th>
<th>time, sec.</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>&lt;1</td>
</tr>
<tr>
<td>40</td>
<td>15</td>
</tr>
<tr>
<td>60</td>
<td>100</td>
</tr>
<tr>
<td>80</td>
<td>360</td>
</tr>
<tr>
<td>100</td>
<td>2000</td>
</tr>
</tbody>
</table>

5.3 3D OPP: comparison of bar and slice relaxations and DFFs

For 2D packing problems, the bar and slice relaxations are identical. To compare them, we generated several classes of 3D instances. Alongside they are compared with the volume bounds from dual-feasible functions.

5.3.1 Generation of instances

We considered the following classes of 3D OPP instances: the containers are cubes with sizes $W_1 = W_2 = W_3 = 1000$. The number of items $n$ was from the set $\{10, 20, 40, 100\}$, approximate waste volume $e$ from $\{0\%, 2\%, ..., 40\\%\}$. The maximal ratio $r_{\text{max}}$ between the sizes of an item in different dimensions was from the set $r_{\text{max}} \in \{1, 3, 20\}$. Obviously, if $r_{\text{max}} = 1$, all items are cubes. For each class with fixed $(n, e, r_{\text{max}})$, 100 instances were generated. Every instance was generated in the following way: the total volume of the items $10^9(1 - e)$ is separated into $n$ intervals by $n - 1$ uniformly distributed numbers $z_1, ..., z_{n-1}$ in $(0, 10^9(1 - e))$. The numbers $z_1, ..., z_{n-1}$ are sorted and item volumes are 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 $i = 2, ..., n - 1$. If a volume is less than 125000, the generation procedure restarts. To obtain the three sides of an item, its volume $v_i$, $i = 1, ..., n$ is factorized with the help of three random numbers $a_j$, $j = 1, 2, 3$ whose sum is 3. These numbers are calculated in the same way as the volumes. The side of item $i$ in dimension $j$ has the length $w_{ij} = v_i^{a_j/3}$. If item $i$ is in one dimension larger than 1000 or $r_{\text{max}}$ times the length of another side, the volume $v_i$ is factorized again. To get cubes, we set $a_j = 1$ for $j = 1, 2, 3$. Finally, the sizes are rounded down.
5 EXPERIMENTS

5.3.2 The bounds tested

We calculated the bar LP relaxation (“$b$”) and slice LP relaxation (“$s$”), Section 3.3, for all three dimensions, as the minimal length of the corresponding 1D CSP solution. The volume bound (“$v$”) was computed, similar to the 2D case, as

$$v = \max_{g_1,g_2,g_3 \in \{id,u_1,f_2,\phi \}} \sum_{i=1}^n g_1(w_1^i)g_2(w_2^i)g_3(w_3^i),$$  

with the DFFs from Table 1. If $b > 10^6$, $s > 1000$, or $v > 1$ then the instance cannot be feasible. By saying that an instance was solved we mean one of these cases.

Table 4: $n = 20$, maximal ratio of sides: $r_{\text{max}} = 20$

<table>
<thead>
<tr>
<th>waste %</th>
<th>max $\frac{b}{v}$</th>
<th>max $\frac{s}{v}$</th>
<th>$v$</th>
<th>b</th>
<th>s</th>
<th>$s \cap b$</th>
<th>b $\cap v$</th>
<th>s $\cap v$</th>
<th>b $\cap s \cap v$</th>
<th>total</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.467</td>
<td>1.090</td>
<td>1.004</td>
<td>1.106</td>
<td>100</td>
<td>19</td>
<td>85</td>
<td>19</td>
<td>85</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>2.458</td>
<td>1.064</td>
<td>0.989</td>
<td>1.088</td>
<td>97</td>
<td>7</td>
<td>70</td>
<td>7</td>
<td>69</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>4.450</td>
<td>1.048</td>
<td>0.982</td>
<td>1.061</td>
<td>74</td>
<td>9</td>
<td>59</td>
<td>9</td>
<td>51</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>6.448</td>
<td>1.022</td>
<td>0.971</td>
<td>1.027</td>
<td>54</td>
<td>7</td>
<td>44</td>
<td>7</td>
<td>36</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>8.443</td>
<td>0.995</td>
<td>0.970</td>
<td>1.014</td>
<td>30</td>
<td>4</td>
<td>36</td>
<td>4</td>
<td>21</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>10.429</td>
<td>0.979</td>
<td>0.970</td>
<td>0.989</td>
<td>28</td>
<td>5</td>
<td>24</td>
<td>5</td>
<td>16</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>12.432</td>
<td>0.951</td>
<td>0.970</td>
<td>0.946</td>
<td>18</td>
<td>3</td>
<td>16</td>
<td>3</td>
<td>7</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>14.425</td>
<td>0.926</td>
<td>0.973</td>
<td>0.934</td>
<td>10</td>
<td>3</td>
<td>17</td>
<td>3</td>
<td>6</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>16.419</td>
<td>0.906</td>
<td>0.967</td>
<td>0.909</td>
<td>9</td>
<td>2</td>
<td>8</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>18.424</td>
<td>0.893</td>
<td>0.968</td>
<td>0.890</td>
<td>6</td>
<td>2</td>
<td>6</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>20.412</td>
<td>0.868</td>
<td>0.969</td>
<td>0.855</td>
<td>6</td>
<td>1</td>
<td>5</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>22.409</td>
<td>0.853</td>
<td>0.966</td>
<td>0.839</td>
<td>4</td>
<td>3</td>
<td>5</td>
<td>3</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>24.394</td>
<td>0.828</td>
<td>0.957</td>
<td>0.819</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>26.391</td>
<td>0.812</td>
<td>0.970</td>
<td>0.796</td>
<td>3</td>
<td>1</td>
<td>4</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>28.387</td>
<td>0.798</td>
<td>0.959</td>
<td>0.762</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>30.390</td>
<td>0.784</td>
<td>0.960</td>
<td>0.741</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>32.372</td>
<td>0.762</td>
<td>0.964</td>
<td>0.723</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>34.370</td>
<td>0.756</td>
<td>0.961</td>
<td>0.700</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>36.360</td>
<td>0.720</td>
<td>0.961</td>
<td>0.671</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>38.359</td>
<td>0.720</td>
<td>0.961</td>
<td>0.661</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>40.352</td>
<td>0.696</td>
<td>0.960</td>
<td>0.626</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

5.3.3 The results

For each class with fixed $(n, e, r_{\text{max}})$, 100 instances were tested. Table 4 reports results for $n = 20$ and maximal side ratio $r_{\text{max}} = 20$. Each line corresponds to a different waste $e \in \{0\%, 2\%, \ldots, 40\%\}$. The columns are: ‘waste %’ is the actual average waste in the class, per cent (which is a little larger than $e$ because all sizes
5 EXPERIMENTS

Table 5: Significance of DFFs for the first class in Table 4

<table>
<thead>
<tr>
<th>DFF</th>
<th>solved instances with vol(DFF)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$id, u^k, U^c, \phi^c, f_2^k, f_1^k$ (all)</td>
<td>85</td>
</tr>
<tr>
<td>without $f_1^k$</td>
<td>69</td>
</tr>
<tr>
<td>without $f_2^k$</td>
<td>85</td>
</tr>
<tr>
<td>without $\phi^c$</td>
<td>85</td>
</tr>
<tr>
<td>without $U^c$</td>
<td>81</td>
</tr>
<tr>
<td>without $u^k$</td>
<td>82</td>
</tr>
<tr>
<td>without $id$</td>
<td>56</td>
</tr>
</tbody>
</table>

Table 6: Computation times

<table>
<thead>
<tr>
<th>time (in s)</th>
<th>number of items $n$</th>
</tr>
</thead>
<tbody>
<tr>
<td>LP relaxations</td>
<td>vol(DFF)</td>
</tr>
<tr>
<td>0.1</td>
<td>0.5</td>
</tr>
<tr>
<td>0.9</td>
<td>1.3</td>
</tr>
<tr>
<td>20</td>
<td>3.2</td>
</tr>
<tr>
<td>400</td>
<td>9</td>
</tr>
</tbody>
</table>

were rounded down); $\max b/10^6$ is the average of the maximum among the three bar LP relaxations divided by $10^6$; $\max s/10^3$ is the average of the maximum among the three slice LP relaxations divided by $10^3$; $v$ is the average volume bound from DFFs; the next three columns give the number of instances solved by each kind of bounds; the next four columns give the number of instances solved by every two and all three kinds of bounds simultaneously; the last column is the total number of instances proved infeasible.

We tested the significance of the different DFFs, Table 5, and the runtime for the bounds dependent on the number of items, Table 6.

5.3.4 Interpretation of the results

First of all these results indicate that the bar LP relaxation is much more effective than the slice LP relaxation. This can be explained by the loss of geometry when relaxing slices to 1D knapsacks. In the classes with large waste, the value $\max s/10^3$ of the slice bound is rather high because of large items.

The bar LP relaxation turns out to be stronger than the used DFFs for instances with small waste, Table 4. This relation shifts to a strong advantage of DFFs when items are cubes (Table 7). The results for $r_{\text{max}} = 3$ (not reported here) are similar to $r_{\text{max}} = 20$. Similar overall observations as for $n = 20$ hold for $n = 40,$
where all bounds are weaker (or more instances feasible), Tables 8 and 9, and for
\( n = 10 \), where all bounds are stronger (not reported here). For \( m = 100 \), not a single instance was solved.

For \( n = 20 \) the calculation of the LP bounds is faster than the volume bound, Table 6. Most of the time spend for \( v \) is needed for the DDFF \( f_1^k \) but this is an effective one, see Table 5. Another effective DFF is the identical function (combined with other DFFs). In [CAdC08], many other DFFs are compared, but these are not included.

### Table 7: \( n = 20 \), maximal ratio of sides: \( r_{\text{max}} = 1 \) (cubes)

<table>
<thead>
<tr>
<th>waste %</th>
<th>( \frac{b}{v} )</th>
<th>( \frac{s}{v} )</th>
<th>( v )</th>
<th>( b \cap s )</th>
<th>( b \cap v )</th>
<th>( s \cap v )</th>
<th>( b \cap s \cap v )</th>
<th>total</th>
</tr>
</thead>
<tbody>
<tr>
<td>0,366</td>
<td>1,056</td>
<td>0,996</td>
<td>1,871</td>
<td>100</td>
<td>0</td>
<td>100</td>
<td>0</td>
<td>100</td>
</tr>
<tr>
<td>2,236</td>
<td>1,030</td>
<td>0,976</td>
<td>1,928</td>
<td>88</td>
<td>0</td>
<td>99</td>
<td>0</td>
<td>100</td>
</tr>
<tr>
<td>4,353</td>
<td>1,004</td>
<td>0,957</td>
<td>1,835</td>
<td>50</td>
<td>0</td>
<td>100</td>
<td>0</td>
<td>100</td>
</tr>
<tr>
<td>6,352</td>
<td>0,984</td>
<td>0,937</td>
<td>1,748</td>
<td>24</td>
<td>0</td>
<td>98</td>
<td>0</td>
<td>100</td>
</tr>
<tr>
<td>8,343</td>
<td>0,960</td>
<td>0,917</td>
<td>1,687</td>
<td>6</td>
<td>0</td>
<td>94</td>
<td>0</td>
<td>94</td>
</tr>
<tr>
<td>10,341</td>
<td>0,938</td>
<td>0,897</td>
<td>1,689</td>
<td>2</td>
<td>0</td>
<td>95</td>
<td>0</td>
<td>95</td>
</tr>
<tr>
<td>12,332</td>
<td>0,919</td>
<td>0,877</td>
<td>1,645</td>
<td>0</td>
<td>0</td>
<td>95</td>
<td>0</td>
<td>95</td>
</tr>
<tr>
<td>14,336</td>
<td>0,895</td>
<td>0,857</td>
<td>1,505</td>
<td>0</td>
<td>0</td>
<td>90</td>
<td>0</td>
<td>90</td>
</tr>
<tr>
<td>16,337</td>
<td>0,878</td>
<td>0,837</td>
<td>1,495</td>
<td>0</td>
<td>0</td>
<td>87</td>
<td>0</td>
<td>87</td>
</tr>
<tr>
<td>18,333</td>
<td>0,854</td>
<td>0,817</td>
<td>1,340</td>
<td>0</td>
<td>0</td>
<td>76</td>
<td>0</td>
<td>76</td>
</tr>
<tr>
<td>20,319</td>
<td>0,837</td>
<td>0,797</td>
<td>1,420</td>
<td>0</td>
<td>0</td>
<td>79</td>
<td>0</td>
<td>79</td>
</tr>
</tbody>
</table>

### Table 8: \( n = 40 \), maximal ratio of sides: \( r_{\text{max}} = 20 \)

<table>
<thead>
<tr>
<th>waste %</th>
<th>( \frac{b}{v} )</th>
<th>( \frac{s}{v} )</th>
<th>( v )</th>
<th>( b \cap s )</th>
<th>( b \cap v )</th>
<th>( s \cap v )</th>
<th>( b \cap s \cap v )</th>
<th>total</th>
</tr>
</thead>
<tbody>
<tr>
<td>0,647</td>
<td>0,999</td>
<td>0,994</td>
<td>0,994</td>
<td>21</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>21</td>
</tr>
<tr>
<td>2,639</td>
<td>0,980</td>
<td>0,981</td>
<td>0,975</td>
<td>5</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>4,681</td>
<td>0,960</td>
<td>0,979</td>
<td>0,955</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>6,621</td>
<td>0,938</td>
<td>0,975</td>
<td>0,934</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>8,616</td>
<td>0,919</td>
<td>0,973</td>
<td>0,915</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>10,610</td>
<td>0,899</td>
<td>0,975</td>
<td>0,895</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>12,597</td>
<td>0,882</td>
<td>0,976</td>
<td>0,875</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
</tbody>
</table>

### 6 Conclusions

We reviewed one-dimensional relaxations of higher-dimensional orthogonal packing feasibility problem (OPP). Among the LP bounds of such relaxations, the strongest seems to be the bar relaxation. Theoretically it dominates the LP bound
Table 9: \( n = 40 \), maximal ratio of sides: \( r_{\text{max}} = 1 \) (cubes)

<table>
<thead>
<tr>
<th>waste %</th>
<th>means ( b \cap v )</th>
<th>( s \cap b )</th>
<th>( b \cap v )</th>
<th>( s \cap v )</th>
<th>( b \cap s \cap v )</th>
<th>total</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.461</td>
<td>0.998</td>
<td>0.995</td>
<td>1.183</td>
<td>14</td>
<td>0</td>
<td>71</td>
</tr>
<tr>
<td>2.458</td>
<td>0.978</td>
<td>0.975</td>
<td>1.168</td>
<td>0</td>
<td>0</td>
<td>65</td>
</tr>
<tr>
<td>4.432</td>
<td>0.958</td>
<td>0.956</td>
<td>1.147</td>
<td>0</td>
<td>0</td>
<td>61</td>
</tr>
<tr>
<td>6.443</td>
<td>0.938</td>
<td>0.936</td>
<td>1.099</td>
<td>0</td>
<td>0</td>
<td>37</td>
</tr>
<tr>
<td>8.439</td>
<td>0.917</td>
<td>0.916</td>
<td>1.102</td>
<td>0</td>
<td>0</td>
<td>43</td>
</tr>
<tr>
<td>10.425</td>
<td>0.897</td>
<td>0.896</td>
<td>1.058</td>
<td>0</td>
<td>0</td>
<td>33</td>
</tr>
<tr>
<td>12.428</td>
<td>0.877</td>
<td>0.876</td>
<td>1.041</td>
<td>0</td>
<td>0</td>
<td>38</td>
</tr>
<tr>
<td>14.423</td>
<td>0.857</td>
<td>0.856</td>
<td>1.010</td>
<td>0</td>
<td>0</td>
<td>13</td>
</tr>
<tr>
<td>16.406</td>
<td>0.837</td>
<td>0.836</td>
<td>0.983</td>
<td>0</td>
<td>0</td>
<td>14</td>
</tr>
<tr>
<td>18.406</td>
<td>0.817</td>
<td>0.816</td>
<td>0.950</td>
<td>0</td>
<td>0</td>
<td>8</td>
</tr>
<tr>
<td>20.404</td>
<td>0.797</td>
<td>0.796</td>
<td>0.966</td>
<td>0</td>
<td>0</td>
<td>11</td>
</tr>
</tbody>
</table>

of the non-preemptive relaxation of OPP (model (4)). Practically, the bar relaxation also dominates the slice relaxation.

The slice relaxation appeared very weak, probably because of the 1D relaxation of the slice geometry. We could consider strengthening of the slice contents, cf. the layer relaxation [Sch99].

A comparison with the commonly-used volume bounds from heuristic dual-feasible functions in the 2D case has shown a strong dominance of the bar relaxation, as already observed, e.g., in [BB07]. Volume bounds from the exact algorithm [CLM05] are competitive, however they also compute the bar relaxations as part of the model.

In the 3D case, the volume bounds from DFFs are competitive on instances with large waste and strongly dominate on cubic instances. Most useful are the following DFFs: the identical function and a data-dependent DFF from [CCM07]. Our future research would consider the combination of DFFs and other conservative scales with 1D relaxations.

Probing on the item intersection variables of the proposed OPP model (2) is a method to make the bounds “higher-dimensional”. It proved very effective for the LP relaxations in the 2D case. This is a first step toward branching.

The ILP model (4) of CCSP was used for the theoretical comparison with the bar relaxation. It would be certainly interesting to see the performance of ILP models for OPP and CCSP in such solvers as CPLEX. But the preliminary results, reported in Section 2, indicate that the models are weak. Moreover, the discussed models for OPP and CCSP are pseudo-polynomially large. This means, for the instances of [CICM08], they would probably behave fine, but for the new 3D instances with \( W_1 = W_2 = W_3 = 1000 \) rather poor. Probably, model (4) can be better solved by branch-and-price using the decomposition shown in Theorem 2.
Acknowledgements

We thank Michele Monaci and Alberto Caprara for kindly providing their code for optimizing conservative scales by bilinear programming. We also thank the anonymous referees for their helpful comments.

References


REFERENCES


REFERENCES


