Stabilized Branch-and-Price Algorithms for Vector Packing Problems

Katrin Heßler[*,a], Timo Gschwind[a], Stefan Irnich[a]

[a]*Chair of Logistics Management, Gutenberg School of Management and Economics, Johannes Gutenberg University Mainz, Jakob-Welder-Weg 9, D-55128 Mainz, Germany.*

Abstract

This paper considers packing and cutting problems in which a packing/cutting pattern is constrained independently in two or more dimensions. Examples are restrictions with respect to weight, length, and value. We present branch-and-price algorithms to solve these vector packing problems (VPPs) exactly. The underlying column-generation procedure uses an extended master program that is stabilized by (deep) dual-optimal inequalities. While some inequalities are added to the master program right from the beginning (static version), other violated dual-optimal inequalities are added dynamically. The column-generation subproblem is a multidimensional knapsack problem, either binary, bounded, or unbounded depending on the specific master problem formulation. Its fast resolution is decisive for the overall performance of the branch-and-price algorithm. In order to provide a generic but still efficient solution approach for the subproblem, we formulate it as a shortest path problem with resource constraints (SPPRC), yielding the following advantages: (i) Violated dual-optimal inequalities can be identified as a by-product of the SPPRC labeling approach and thus be added dynamically; (ii) branching decisions can be implemented into the subproblem without deteriorating its resolution process; and (iii) larger instances of higher-dimensional VPPs can be tackled with branch-and-price for the first time. Extensive computational results show that our branch-and-price algorithms are capable of solving VPP benchmark instances effectively.

Key words: Cutting, Vector packing, Shortest path problem with resource constraints, Dual-optimal inequalities, Stabilization

1. Introduction

In this paper, we analyze different covering formulations and column-generation-based solution approaches for packing and cutting problems with two or more dimensions. While the 1-dimensional case includes the classical *bin packing problem* (BPP) and *cutting stock problem* (CSP) that are both well studied and recently surveyed by Delorme *et al.* (2016), we focus on the multidimensional case where packings/cutting patterns are constrained independently in all dimensions. Examples are restrictions with respect to weight, length, and value. In the following, we refer to these problems as *vector packing problems* (VPPs). The literature uses different names for VPPs such as p-dimensional vector (bin) packing (Buljubašić and Vasquez, 2016; Spieksma, 1994; Brandao and Pedroso, 2016), vector bin packing (Panigrahy *et al.*, 2011), or (for two dimensions) two-constraint bin packing (Monaci and Toth, 2006). In contrast to VPPs, there is another family of multidimensional packing problems where dimensions are not independent, e.g., when packing rectangles (Huang and Korf, 2012) or 3-dimensional items (Martello *et al.*, 2000). These problems are not addressed here.

VPPs have various practical applications. For example, consider a logistics company that has to transport items with different lengths and weights. The smallest amount of vehicles possible should be used for

[*]Corresponding author.

Email addresses: khessler@uni-mainz.de (Katrin Heßler), gschwind@uni-mainz.de (Timo Gschwind), irnich@uni-mainz.de (Stefan Irnich) transportation. How can the items be packed into the vehicles taking into account the length of the vehicles and the maximum loading weight? Another example is the static resource allocation problem. Given a set of servers with known capacities and a set of services with known demands the aim is to minimize the number of required servers (Panigrahy *et al.*, 2011).

In the literature, several heuristic and exact methods have been introduced to solve VPPs. The first exact approach was proposed by Spieksma (1994). The author incorporated lower bounds into a branch-and-bound algorithm and, additionally, introduced a heuristic based on the first-fit decreasing heuristic. The approach was improved by Caprara and Toth (2001) who exploited on the one hand lower bounds combined with heuristics and on the other hand a branch-and-bound algorithm to find exact solutions for VPPs. Alves *et al.* (2014) further enhanced the approach by calculating lower bounds based on dual-feasible functions. Recently, Brandao and Pedroso (2016) adapted the arc-flow formulation with side constraints (Valério de Carvalho, 1999) to packing problems including VPPs. The solution was accelerated by means of graph compression to reduce symmetry and to combine sub-graphs. In contrast, Hu *et al.* (2017) used the covering formulation of Gilmore and Gomory (1961) and introduced a branching strategy based on dominance relations between cutting patterns. Besides exact approaches, heuristics have been applied to solve larger instances. Buljubašić and Vasquez (2016) used a consistent neighborhood search to find heuristic solutions. Variants of the first-fit decreasing algorithm were presented by Panigrahy *et al.* (2011), yielding good heuristic VPP solutions.

We formally define the VPP as follows: Let $D \geq 1$ be an integer specifying the dimension. A given set $\mathcal{I}$ of items needs to be packed into a minimum number of equally sized bins. Bins are characterized by capacities given by a D-dimensional vector $\mathbf{W} = (W^1, \ldots, W^D)$. Items $i \in \mathcal{I}$ are characterized by weights/seizes which are given by D-dimensional vectors $\mathbf{w}_i = (w_i^1, \ldots, w_i^D) \leq \mathbf{W}$. Although classified differently as 1/V/I/R and 1/V/I/M in (Dyckhoff, 1990), the BPP and the CSP are essentially the same problem as pointed out in (Ben Amor and Valério de Carvalho, 2005, p. 132). The relationship between BPP and CSP is the following: In the BPP, all items are modeled as *individual objects*. As a consequence, each and every item $i \in \mathcal{I}$ has a demand of $q_i = 1$ and a model must include a corresponding covering/packing constraint. In contrast, in the CSP all items of the same weight are *aggregated*. We formally introduce the set of aggregated items as follows: Let $\mathcal{I}[\mathbf{w}]$ be the equivalence classes of items with identical weight. Then, the set I of *aggregated items* is the coset $\mathcal{I}/\mathcal{I}[\mathbf{w}]$ that contains one representative item for each of the different weights. As a consequence, any two different aggregated items $i_1, i_2 \in I$ have different weights $\mathbf{w}_{i_1} \neq \mathbf{w}_{i_2}$. The demand of an aggregated item $i \in I$ is $q_i = |\mathcal{I}[\mathbf{w}_i]| = |\{j \in \mathcal{I} : \mathbf{w}_j = \mathbf{w}_i\}| \geq 1$, and at least some aggregated items have a demand greater than 1.

The models that we compare are all variations of the famous covering formulation of Gilmore and Gomory (1961) for CSP that is based on cutting patterns. For the VPP, a *pattern* (or *packing*) describes how a subset of the items is packed into a bin. Assuming $I = \{1, 2, \ldots, m\}$, i.e., m is the number of aggregated items, the set of feasible patterns for the VPP, with different levels of aggregation, can be defined as

$$P = \left\{ (a_1, \ldots, a_m)^\top \in \mathbb{Z}_+^m : \sum_{i=1}^m a_i w_i^d \leq W^d \text{ for all } 1 \leq d \leq D \right\}.$$

In order to uniquely refer to a pattern $p \in P$ we write its coefficients as $\mathbf{a}^p = (a_1^p, \ldots, a_m^p)^\top$. Using integer decision variables x_p for the number of times pattern $p \in P$ is used, the VPP can be formulated as

$$\text{(VPP)} \qquad \begin{aligned} z^{\text{VPP}} = \min & \sum_{p \in P} x_p && && \text{(1a)} \\ \text{s.t.} & \sum_{p \in P} a_i^p x_p = q_i, && i \in I && \text{(1b)} \\ & x_p \geq 0 \text{ integer}, && p \in P. && \text{(1c)} \end{aligned}$$

The objective (1a) is the minimization of the patterns/bins that are used. Constraints (1b) ensure that all items are packed as often as their demand requires. As already mentioned by Gilmore and Gomory (1961),

Table 1: Comparison of covering formulations of the VPP

	01-VPP		B-VPP		U-VPP
Aggregation	none		partial or full		full
(Aggregated) Items I	$I = \mathcal{I}$		I		$I = \mathcal{I}/\mathcal{I}[\mathbf{w}]$
Weights $\mathbf{w}_i$	can be identical		can be identical		all different
Demand q_i	$q_i = 1$		$q_i \geq 1$		$q_i \geq 1$
Bound on Pattern Coefficients u_i	$a_i^p \leq 1 = u_i^{01}$		$a_i^p \leq u_i^B \leq q_i$		$a_i^p \leq u_i^U = \min_d \lfloor \frac{W^d}{w_i^d} \rfloor$
Pattern Set	P_{01}		P_B		$P_U = P$
Domain of Variables	$x_p \in \{0,1\}$		$x_p \in \mathbb{Z}_+$		$x_p \in \mathbb{Z}_+$
Number of Constraints m	$m_{01} = \lvert\mathcal{I}\rvert$	$>$	m_B	$\geq$	$m_U = \lvert\mathcal{I}/\mathcal{I}[\mathbf{w}]\rvert$
LP Bound z_{LP}	$z_{LP}^{\text{01-VPP}}$	$=$	$z_{LP}^{\text{B-VPP}}$	$\geq$	$z_{LP}^{\text{U-VPP}}$
Subproblem	01-MKP		B-MKP		U-MKP
Branching	(Ryan and Foster, 1981)		(Vanderbeck, 1999)		(Vanderbeck, 1999)
Dual Inequalities					
PIs $\pi_h \geq \pi_i$	DDOIs		DDOIs		DOIs
PIs with $\mathbf{w}_h + \mathbf{w}_i \not\leq \mathbf{W}$	DOIs		DOIs		DOIs
SIs $\pi_h \geq \sum_{i \in S} \pi_i$	can be invalid		can be invalid		DOIs
SIs with $\mathbf{w}_h + \mathbf{w}_i \not\leq \mathbf{W}$ for all $i \in S$	DOIs		DOIs		DOIs
WSIs $\pi_h \geq \sum_i t_i \pi_i$	invalid		can be invalid		DOIs

Note: PIs=Pair Inequalities, SIs=Subset Inequalities, and WSIs=Weighted Subset Inequalities

(1b) can be replaced by covering constraints, i.e., $\sum_{p \in P} a_i^p x_p \geq q_i$ for all $i \in I$. The domain of the decision variables is given by (1c).

The quality of the linear relaxation bound is crucial to solve mixed-integer problems. As in the BPP (i.e., $D = 1$ and $q_i = 1$ for all $i \in I$), patterns can be restricted to only have binary coefficients $\mathbf{a}^p \in \{0,1\}^m$ when modeling individual items. While such a constraint does not impact the validity of integer VPP solutions to (1) (also for $D \geq 2$), the linear relaxation is generally tighter than without the binary requirement. It means that a proper subset of patterns is used:

$$P_{01} = \left\{ (a_1, \ldots, a_m)^\top \in P : a_i \leq 1 \text{ for all } i \in I \right\}$$

We denote formulation (1) using only binary patterns P_{01} as *binary* VPP (01-VPP). In this case, optimal solutions to the 01-VPP have binary x_p-variables.

Also for VPP with non-unit demand, the patterns' coefficients can be further constrained. Whenever the demand q_i of some item $i \in I$ is smaller than $\lfloor W^d / w_i^d \rfloor$ for all $d \in D$, the pattern set

$$P_B = \left\{ (a_1, \ldots, a_m)^\top \in P : a_i \leq q_i \text{ for all } i \in I \right\} \tag{2}$$

is a proper subset of P. We denote formulation (1) using only bounded patterns P_B as *bounded* VPP (B-VPP). For completeness, formulation (1) with no additional bounds on pattern coefficients is referred to as *unbounded* VPP (U-VPP) and the pattern set is explicitly denoted by $P_U = P$ in the following.

Table 1 summarizes differences between the pure binary formulation 01-VPP, the bounded formulation B-VPP, and the unbounded formulation U-VPP. Note that for a given set of individual items $\mathcal{I}$, the formulations 01-VPP and U-VPP are unique. They represent the two extremes of aggregation (completely disaggregated vs. fully aggregated), while B-VPP is a family of formulations resulting from different types of aggregation and exploitation/disregarding of the coefficient bounds $a_i^p \leq q_i$ in (2).

Due to the huge number of variables, formulation (1) is typically solved with the help of a column-generation algorithm (Desaulniers *et al.*, 2005). One starts with a (small) subset of patterns $P' \subset P$ and the linear relaxation of (1) using only variables x_p with $p \in P'$. This so-called restricted master program (RMP) is then optimized. Let $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_m)^\top$ be the dual solution w.r.t. the covering constraints (1b). The task of the pricing subproblem is then to provide (at least) one new pattern p with negative reduced cost $\tilde{c}_p = 1 - \boldsymbol{\pi}^\top \mathbf{a}^p$ or to prove that no such pattern exists. In the former case, the resulting RMP (with the generated pattern/s) is re-optimized and the column-generation process is repeated. In the latter case, the linear relaxation of (1) is solved providing a lower bound z_{LP}^{VPP} on z^{VPP}. An optimal integer solution to (1) requires the integration of column generation into a branch-and-bound scheme a.k.a. branch-and-price (Lübbecke and Desrosiers, 2005).

The pricing problems of the different formulations 01-VPP, B-VPP, and U-VPP seek to find a pattern with negative reduced cost $\tilde{c}_p$ that is feasible for the pattern set considered in the respective formulation. They are equivalent to solving one of the following integer programs:

$$\max \sum_{i=1}^{m} \pi_i y_i \tag{3a}$$

$$\text{s.t.} \sum_{i=1}^{m} w_i^d y_i \leq W^d, \qquad d \in D \tag{3b}$$

$$\text{(01-MKP)} \quad y_i \in \{0,1\}, \qquad \text{(B-MKP)} \quad y_i \in \{0,1,\dots,u_i^B\}, \qquad \text{(U-MKP)} \quad y_i \in \mathbb{Z}_+, \qquad i \in I \tag{3c}$$

where $u_i^B = \min\{q_i, \min_d \lfloor \frac{W^d}{w_i^d} \rfloor\}$. The pricing subproblems are thus variants of *multidimensional knapsack problems* (MKPs, Kellerer *et al.*, 2004), either binary (01-MKP) for 01-VPP, bounded (B-MKP) for B-VPP, or unbounded (U-MKP) for U-VPP. For U-MKP, it is valid to bound the variables y_i in (3c) from above by $u_i^U = \min_d \lfloor \frac{W^d}{w_i^d} \rfloor$. For convenience, we define a corresponding upper bound $u_i^{01} = 1$ for all $i \in I$ for 01-MKP.

We now briefly discuss the main advantages and disadvantages of the different covering formulations. The master program of 01-VPP is the largest of the three in terms of number of constraints, while the master of U-VPP is the smallest. Regarding the number of variables, i.e., the number of different feasible patterns, there is no general order between the three formulations, since it depends on the demands q_i and the share of items with identical weights in $\mathcal{I}$. To be precise, the number of feasible patterns for 01-VPP is $\mathcal{O}\left(\sum_{i=1}^{\min\{u,|\mathcal{I}|\}} \binom{|\mathcal{I}|}{i}\right)$, with $u = \max_i \min_d \lfloor \frac{W^d}{w_i^d} \rfloor$. For U-VPP, it is $\mathcal{O}\left(\sum_{i=1}^{u} \binom{|\mathcal{I}/\mathcal{I}[\mathbf{w}]|+i-1}{i}\right)$. In the special case of complete aggregation, i.e., $I = \mathcal{I}/\mathcal{I}[\mathbf{w}]$, the number of B-VPP patterns is by definition not greater than that of U-VPP, i.e., $|P_B| \leq |P_U|$.

Concerning the linear relaxation bounds, $z_{LP}^{\text{01-VPP}} = z_{LP}^{\text{B-VPP}} \geq z_{LP}^{\text{U-VPP}}$ holds. The equality $z_{LP}^{\text{01-VPP}} = z_{LP}^{\text{B-VPP}}$ results from the constraint aggregation approach discussed in (Ben Amor *et al.*, 2006). The following Example 1 shows a small instance with strict inequality between $z_{LP}^{\text{B-VPP}}$ and $z_{LP}^{\text{U-VPP}}$. Moreover, Rietz (2003) has shown that the 01-VPP with $D \geq 2$ does not posses the *modified integer round up property* (MIRUP) (see also Scheithauer and Terno, 1995; Caprara *et al.*, 2014). The property is also unclear for CSP, i.e., U-VPP with $D = 1$, and herewith also for U-VPP in higher dimension. Still, linear relaxation bounds seem to be rather tight for most instances and all three formulations (see Section 2.4.4).

Example 1. The following VPP example has the same LP bound for the covering formulations 01-VPP and B-VPP but different LP bounds for 01-VPP/B-VPP and U-VPP. Consider a two-dimensional VPP with $\mathbf{W} = (6,6)^\top$ and items $\mathcal{I} = \{1,2,3\}$ with weights $\mathbf{w_1} = (3,3)^\top$ and $\mathbf{w_2} = \mathbf{w_3} = (2,2)^\top$. For 01-VPP, $z_{LP}^{\text{01-VPP}} = 1.5$. An optimal solution is 0.5 times $(1,1,0)^\top$, 0.5 times $(1,0,1)^\top$, and 0.5 times $(0,1,1)^\top$. For B-VPP, consider $I = \{1,[2]\}$ with $q_1 = 1$ and $q_2 = 2$. Then, $z_{LP}^{\text{B-VPP}} = 1.5$. An optimal solution is 1 time $(1,1)^\top$ and 0.5 times $(0,2)^\top$. For U-VPP, $I = \{[1],[2]\}$ and $z_{LP}^{\text{U-VPP}} = 7/6$. An optimal solution is 0.5 times $(2,0)^\top$ and 2/3 times $(0,3)^\top$.

Recall that the subproblems of formulations 01-VPP, B-VPP, and U-VPP are 01-MKPs, B-MKPs, and U-MKPs with $m_{01} \geq m_B \geq m_U$ items, respectively. Recall also that in the 1-dimensional case, all three versions can be solved in $\mathcal{O}(mW)$ time, where W is the 1-dimensional capacity. Moreover, the space requirement is (applying storage reduction techniques) $\mathcal{O}(m+W)$ for binary and bounded knapsack problems, while it is $\mathcal{O}(W)$ in the unbounded case (Kellerer *et al.*, 2004). In higher dimensions, the solution of the MKPs with a straightforward dynamic-programming algorithm requires $\mathcal{O}(m\mathcal{W})$ time, where $\mathcal{W} = \prod_d W^d$.

To obtain integer solutions, the well known branching rule of Ryan and Foster (1981), which has proven to be quite effective for set-partitioning in general, can be applied to 01-VPP. For B-VPP and U-VPP, however, Ryan-Foster branching is not applicable and one has to resort to alternative branching rules such as the branching scheme presented by Vanderbeck (1999). Note that Ryan-Foster branching is a special case of the Vanderbeck branching. All branching decisions taken in any of the three formulations have a non-trivial impact on the subproblem and have to be accounted for in its solution. Overall, branching is much more complex in the B-VPP and U-VPP cases compared to 01-VPP. On the other hand, formulation 01-VPP suffers from severe symmetry issues with respect to items with identical weights.

Another difference of the three formulations originates from efforts to stabilize the column-generation process using *(deep) dual-optimal inequalities* ((D)DOIs, see Section 2.3). This technique has already been successfully applied to BPP and CSP by Ben Amor *et al.* (2006) and Valério de Carvalho (2005). Additional classes of valid DOIs and DDOIs as well as the dynamic separation of violated (D)DOIs have been proposed and applied to different coloring and packing problems by Gschwind and Irnich (2016). It is shown in Section 2.3 that the dual inequalities used in (Gschwind and Irnich, 2016), namely *weighted subset inequalities* (WSIs) and *pair inequalities* (PIs), are DOIs for U-VPP. For B-VPP and 01-VPP, however, the PIs are DDOIs whereas the WSIs are generally neither DOIs nor DDOIs.

The contribution of the paper at hand is the following: We present a unified modeling and solution approach for the three MKP subproblems (binary, bounded, and unbounded) based on dynamic programming and the *shortest path problem with resource constraints* (SPPRC, Irnich and Desaulniers, 2005). From a worst-case complexity point of view, its temporal effort of $\mathcal{O}(m\mathcal{W})$ is equivalent to the above-mentioned dynamic-programming approaches. It exploits that many of the dynamic-programming states are not reached in practice, giving it an attractive time and space consumption in the average case. Even more, the SPPRC-based solution approach can cope with the Ryan-Foster and the Vanderbeck branching schemes. In addition, the approach allows to identify violated (D)DOIs as a by-product of the pricing. Computational results indicate that our unified approach for solving 01-VPP, B-VPP, and U-VPP is very competitive with the most recent exact VPP approaches presented in the literature.

This paper is organized as follows. Section 2 presents the unified solution approach with subsections on the SPPRC formulation of the MKP subproblems, their resolution via dynamic-programming labeling, the discussion of stabilization using (D)DOIs, and branching. Section 3 presents the computational results. Final conclusions are drawn in Section 4.

2. Unified Branch-and-Price

We now describe the components of the branch-and-price algorithm. In Section 2.1 the column-generation pricing problem, the MKP, is modeled as an SPPRC. Two alternative labeling algorithms for its solution are presented in Section 2.2. The stabilization of the column-generation process via (D)DOIs is described in Section 2.3. Branching rules including the resulting modified MKP subproblem, its representation as an SPPRC, and the impact of branching decisions on (D)DOIs are discussed in Section 2.4.

2.1. SPPRC Formulation for MKP Subproblems

All three variants of the MKP (U-MKP, B-MKP, and 01-MKP) can be formulated as SPPRCs with $D+1$ resources as follows: The underlying digraph $G=(V,E)$ consists of $m+1$ vertices $V=\{0,\dots,m\}$ and $m+\sum_{i\in I} u_i$ arcs E. For each item $i\in I$, there exist u_i+1 parallel arcs denoted by $E(i)$ connecting vertex $i-1$ with vertex i. Recall that, depending on the MKP variant, the number u_i is defined differently, see above. We assume that the arcs $e\in E(i)$ are indexed by $a_i(e)\in\{0,1,\dots,u_i\}$. Moreover, we define the weight of the $a_i(e)$th arc e of $E(i)$ as $\mathbf{w}(e):=a_i(e)\mathbf{w}$ and its profit as $\pi(e):=a_i(e)\pi_i$.

An example of the digraph G is shown in Figure 1. In general, every 0-m-path $(0,e_1,1,e_2,2,\dots,m-1,e_m,m)$ in G defines a pattern $(a_1(e_1),a_2(e_2),\dots,a_m(e_m))^\top$. This pattern is feasible if and only if $\sum_{i=1}^m \mathbf{w}(e_i)\le\mathbf{W}$ holds. Hence, the MKP pricing problem is the problem of finding a maximum profit 0-m-path in G with a D-dimensional weight not exceeding $\mathbf{W}$. If the profit $\pi=\sum_{i=1}^m \pi(e_i)$ exceeds 1, then the pattern $(a_1(e_1),a_2(e_2),\dots,a_m(e_m))^\top$ has negative reduced cost $\tilde{c}=1-\pi$. This is an SPPRC with m independent capacity constraints.

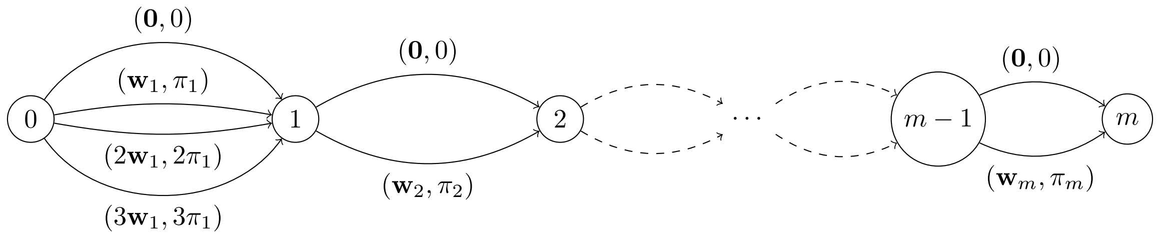


Figure 1: SPPRC digraph for a MKP with $u_1 = 3, u_2 = 1$, and $u_m = 1$. The arcs $e \in E$ are shown with their D-dimensional weight and profit component $(w(e), \pi(e)))$.

2.2. Subproblem Solution via Dynamic-Programming Labeling

Dynamic-programming labeling for solving SPPRCs offers several algorithmic choices such as mono-directional (pure forward or pure backward) or bidirectional labeling, and the dominance rules used to discard partial paths that provably lead to only non-optimal 0-m-paths. Forward dynamic programming starts at the trivial forward path $F = (0)$ with the initial label $(\mathbf{0}, 0)$ and labels are extended iteratively to $1, 2, \ldots, m$ to finally form feasible 0-m-paths. Accordingly, every forward partial path $F = (0, e_1, 1, \ldots, e_j, j)$ from 0 to some $j \in V$ is represented by the label $L(F) = (\sum_{i=1}^{j} \mathbf{w}(e_i), \sum_{i=1}^{j} \pi(e_i)) = (\mathbf{w}, \pi)$. Forward extension along an arc $e_{j+1} \in E(j+1)$ produces the label $L(F, e_{j+1}, j+1) = (\mathbf{w} + \mathbf{w}(e_{j+1}), \pi + \pi(e_{j+1}))$.

Conversely, backward dynamic programming starts with the trivial backward path $B = (m)$ with the initial label $(\mathbf{0}, 0)$ and labels are extended iteratively to $m-1, m-2, \ldots, 1, 0$ to finally form feasible 0-m-paths. Here, every backward partial path $B = (j, e_{j+1}, j+1, \ldots, e_m, m)$ from some $j \in V$ to m is represented by the label $L(B) = (\sum_{i=j+1}^{m} \mathbf{w}(e_i), \sum_{i=j+1}^{m} \pi(e_i)) = (\mathbf{w}, \pi)$. Backward extension along an arc $e_j \in E(j)$ produces the label $L(j-1, e_j, B) = (\mathbf{w} + \mathbf{w}(e_j), \pi + \pi(e_j))$.

Forward and backward labels are feasible if their weight component does not exceed $\mathbf{W}$. Every forward label at vertex m and every backward label at vertex 0 implies a negative reduced-cost pattern if and only if $1 - \pi$ is strictly negative.

A forward label $L(F) = (\mathbf{w}, \pi)$ and a backward label $L(B) = (\mathbf{w}', \pi')$ that end and start at the same vertex can be merged to form a complete feasible 0-m-path if $\mathbf{w} + \mathbf{w}' \leq \mathbf{W}$. The reduced-cost of the imposed pattern is $1 - (\pi + \pi')$.

Labeling with Weak Dominance. Our first dynamic-programming algorithm is a pure forward labeling algorithm and the one with the better worst-case time complexity. It uses the following simple dominance rule.

Rule 1. (Weak Dominance) A forward label $L(F) = (\mathbf{w}, \pi)$ of a forward partial path F dominates another forward label $L(F') = (\mathbf{w}', \pi')$ of a forward partial path F' ending at the same vertex if $\mathbf{w} = \mathbf{w}'$ and $\pi \geq \pi'$.

Dominated labels can be discarded except for cases of mutual dominance, i.e., identical costs, in which case one of the labels can be discarded while the other one must be kept.

The time complexity of a forward labeling algorithm that uses Rule 1 depends on the way how labels are stored. In our implementation, we use a hash table for the labels referring to the same vertex. For a label $(\mathbf{w}, \pi)$, $\mathbf{w}$ is the hash key and π the value to be stored in the hash table. Note that the number of possible keys is exactly $K := \prod_{d=1}^{D}(W^d + 1)$. If the hash table is designed appropriately, arbitrary insertions and lookups of key-value pairs can be guaranteed to take amortized constant time per operation (Cormen *et al.*, 2009, Chapter 11). In cases where the dimension D and the capacities $W^1, \ldots, W^D$ are very large, K may become so huge that a hash table of appropriate size does not fit into the main memory of the computer. Then, with a smaller hash table, its load factor becomes critical and constant-time operations can no longer be ensured. Hence, the overall worst-case complexity is bounded by $\mathcal{O}(mK)$ when computer memory of $\mathcal{O}(K)$ is available.

Labeling with Strong Dominance. Our second dynamic-programming algorithm is a bidirectional algorithm. The idea of bidirectional labeling is to alleviate the combinatorial explosion that typically occurs when partial paths grow longer. As a consequence, bidirectional labeling algorithms often, though not always, have a better average-case time effort than their monodirectional counterparts. For (E)SPPRC, the idea was first presented and successfully used by Righini and Salani (2006). They merge labels "in the middle", which in our context is at vertex $\lfloor m/2 \rfloor \in V$ (or $\lceil m/2 \rceil \in V$ if m is odd). Recent studies of Tilk *et al.* (2017) have shown that one can often benefit from choosing the merge point dynamically, comparing the (expected) workload in forward and backward direction.

In the bidirectional algorithm, we use the following stronger dominance rule whose application, however, produces additional effort in the worst case.

Rule 2. (Strong Dominance) A forward (backward) label $L(P) = (\mathbf{w}, \pi)$ of a partial path P dominates another forward (backward) label $L(P') = (\mathbf{w}', \pi')$ of a partial path P' ending at the same vertex if $\mathbf{w} \leq \mathbf{w}'$ and $\pi \geq \pi'$.

Note that there exist better dominance algorithms than straightforward pairwise comparison of labels. For dominance between Δ-dimensional vectors, Bentley (1980) invented a multi-dimensional divide-and-conquer algorithm that requires $\mathcal{O}\left(n \log(n)^{\Delta-1}\right)$ time to determine all Pareto-optimal out of n vectors. In our application, we have $\Delta = D + 1$ (labels have weights and profit). We use the algorithm of Bentley only for 2-dimensional VPPs but not for the 20-dimensional VPPs (see Section 3) even if $log(n)^D$ is in the worst case better than linear.

Figure 2 visualizes the differences between the weak and the strong dominance.

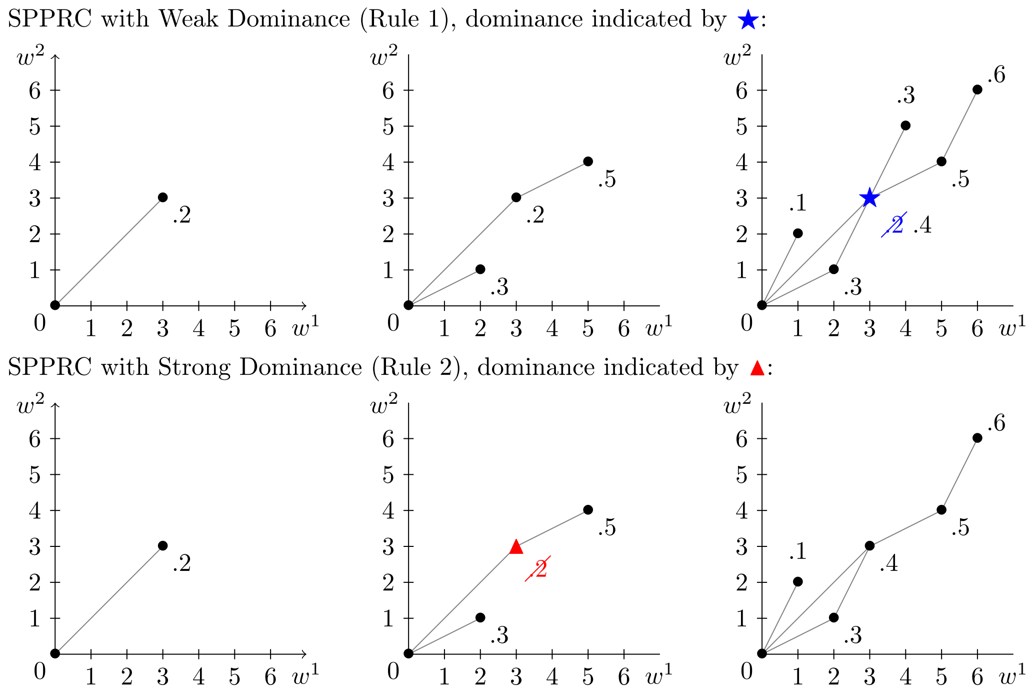


Figure 2: Comparison of two variants for solving 01-MKP with $\mathbf{w}_1 = (3,3), \mathbf{w}_2 = (2,1), \mathbf{w}_3 = (1,2), \pi_1 = 0.2, \pi_2 = 0.3$ and $\pi_3 = 0.1$. For the SPPRC with Weak Dominance, the label consisting of item set $\{1\}$ with profit 0.2 and weight $(3,3)$ is discarded by Rule 1 at the final step. In the end, the undominated labels are $((0,0),0),((1,2),0.1),((2,1),0.3),((3,3),0.4)((4,5),0.3),((5,4),0.5),((6,6),0.6)$ with corresponding item sets $\varnothing, \{3\}, \{2\}, \{2,3\}, \{1,3\}, \{1,2\}$, and $\{1,2,3\}$, respectively. For the SPPRC with Strong Dominance, the label with profit 0.2 and weight $(3,3)$ is dominated by the label with profit 0.3 and weight $(2,1)$ and is therefore discarded at the second step. In the end, the undominated labels are $((0,0),0),((1,2),0.1),((2,1),0.3),((3,3),0.4),((5,4),0.5),((6,6),0.6)$ with corresponding item sets $\varnothing, \{3\}, \{2\}, \{2,3\}, \{1,2\}$, and $\{1,2,3\}$, respectively.

2.3. Dual Inequalities

For the VPP, any linear inequality in the dual variables $\pi_i, i \in I$, is a *dual inequality.* We denote by D^* the set of optimal solutions of the dual model to the linear relaxation of (1), i.e., the dual-optimal space.

Dual Inequalities for U-VPP. Following Ben Amor *et al.* (2006), a dual inequality $\mathbf{t}^\top \boldsymbol{\pi} \leq t$ (with $\mathbf{t} \in \mathbb{Z}^I$ and $t \in \mathbb{Z}$ is a *dual-optimal inequality* if $D^* \subseteq \{\boldsymbol{\pi} : \mathbf{t}^\top \boldsymbol{\pi} \leq t\}$. In (Gschwind and Irnich, 2016, Theorem 1(ii)) it is proven that for any $\mathbf{t} \in \mathbb{Z}^I_+$ and an item $h \in I$ with

$$\mathbf{w}_h \geq \sum_{i \in I} t_i \mathbf{w}_i \qquad \text{it follows that} \qquad \pi_h \geq \sum_{i \in I} t_i \pi_i \tag{4}$$

is a DOI for U-VPP. These DOIs are called *weighted subset inequalities* (WSIs). The interpretation of the precondition is that whenever item h is used (in a solution) this item can always be replaced by one or several occurrences of the items $S = S(\mathbf{t}) := \{i \in I : t_i > 0\}$, where the coefficient t_i specifies the number of occurrences. The DOI then says that covering one unit of demand of item h cannot be cheaper than covering t_i units of demand for all $i \in S$.

WSIs were first introduced in (Gschwind and Irnich, 2016) and generalize the *subset inequalities* (SI) of Ben Amor *et al.* (2006), in which it is required that all coefficients t_i are binary. Hence, dual inequalities $\pi_h \geq \sum_{i \in S} \pi_i$ with $h \in I$ and $S \subset I$ are DOIs for U-VPP if $\mathbf{w}_h \geq \sum_{i \in S} \mathbf{w}_i$ holds.

A special case of the SIs is two items $i, h \in I$ that fulfill $\mathbf{w}_i \leq \mathbf{w}_h$. In this case, the so-called *pair inequality* $\pi_i \leq \pi_h$ is a DOI for U-VPP. It means that covering one unit of demand of item h is not cheaper than one unit of demand of item i.

Dual Inequalities for 01-VPP. The situation of 01-VPP and B-VPP requires a more differentiated analysis. We start by analyzing dual inequalities for the 01-VPP. The formulation (1) for the 01-VPP is a pure binary formulation with binary constraint matrix A, right-hand side $\mathbf{1} \in \mathbb{R}^I$, and cost coefficients $\mathbf{1} \in \mathbb{R}^P$. Moreover, the coefficient matrix A has the (h, i)-row replacement property (Gschwind and Irnich, 2016) for every two items $h, i \in I$ with $\mathbf{w}_h \geq \mathbf{w}_i$. The (h, i)-row replacement property means that any pattern $\mathbf{a} \in P$ with $a_h = 1$ and $a_i = 0$ can be modified into one with $a_h = 0$ and $a_i = 1$ (keeping all other coefficients unchanged) so that again a feasible pattern results. Hence, all preconditions of Proposition 5 of (Gschwind and Irnich, 2016) are fulfilled. Thus, the set of all PIs $\{\pi_h \geq \pi_i : \mathbf{w}_h \geq \mathbf{w}_i\}$ forms a set of *deep dual-optimal inequalities* (DDOIs) for 01-VPP. By definition (Ben Amor *et al.*, 2006) for a set of dual inequalities to be DDOIs, there must exist at least one dual-optimal solution $\boldsymbol{\pi}^* \in D^*$ that fulfills all inequalities of the set. In particular, all DOIs are DDOIs but the reverse is not necessarily true, because DDOIs may cut off some but not all dual-optimal solutions from D^*.

However, for the 01-VPP, specific pair inequalities that fulfill some additional requirements can be proven DOIs. For example, if in addition to $\mathbf{w}_h \geq \mathbf{w}_i$ the condition $\mathbf{w}^d_h + \mathbf{w}^d_i > W^d$ holds for at least one dimension $1 \leq d \leq D$, then this PI is a DOI for 01-VPP. Generalizing, for a SI defined by a subset $S \subset I$ and item $h \in I$, the additional conditions that $\mathbf{w}^d_h + \mathbf{w}^d_i > W^d$ for at least one dimension $1 \leq d \leq D$ for all $i \in S$ guarantee that this SI is a DOI for 01-VPP. These are restricted SIs; general SIs may neither be DOIs nor DDOIs for 01-VPP if only $\mathbf{w}_h \geq \sum_{i \in S} \mathbf{w}_i$ holds.

Dual Inequalities for B-VPP. Now we derive (D)DOIs for B-VPP by discussing their relationship to (D)DOIs of 01-VPP. Recall first that B-VPP is a family of formulations; every B-VPP can be derived from formulation 01-VPP via a (partial) constraint aggregation as already discussed in Section 1. Indeed, let h and i be two items of 01-VPP with identical weights $\mathbf{w}_h = \mathbf{w}_i$, i.e., items from the same equivalence class $\mathcal{I}[\mathbf{w}_h] = \mathcal{I}[\mathbf{w}_i]$. For these two items, the two pair inequalities $\pi_i \leq \pi_h$ and $\pi_h \leq \pi_i$ imply the dual equality $\pi_h = \pi_i$. Hence, $\bigcup\{\pi_h = \pi_i : \mathbf{w}_h = \mathbf{w}_i\} \cup \{\pi_h \geq \pi_i : \mathbf{w}_h \geq \mathbf{w}_i\}$ are DDOIs for 01-VPP. Now, any aggregation over some or all $\mathcal{I}[\mathbf{w}_h]$ with $h \in \mathcal{I}$ leads to a specific choice of aggregated items I and herewith to one specific B-VPP. In any case, Propositions 6 in (Gschwind and Irnich, 2016) ensures that all pair inequalities remain valid DDOIs for the B-VPP.

All theoretical results about the discussed dual inequalities for the three VPP formulations are summarized in the bottom part of Table 1.

Use of Dual Inequalities for Stabilization. Dual inequalities can be added as additional columns of (1) in order to stabilize the column-generation process for solving the linear relaxation. If the set of dual inequalities is DDOIs, this addition does not change the lower bound of the linear relaxation (this is shown in Proposition 1 in (Ben Amor *et al.*, 2006) and in Proposition 1 in (Gschwind and Irnich, 2016)). For a WSI (4) that replaces item $h \in I$ by multiple (aggregated) items $i \in S = S(\mathbf{t})$ each t_i times, the associated column in (1) has a cost of zero and the coefficients $\mathbf{t} - \mathbf{u}_h$, where $\mathbf{u}_h$ is the hth unit vector in $\mathbb{R}^I$.

Note first that in general there is an exponential number of WSIs and SIs, while there can be at most m^2 PIs. In high dimensional VPPs, i.e., $D \gg 2$, it can happen that there are only very few PIs, if any. The question is therefore which dual inequalities one should add to the RMP at which point of the column-generation process. Different strategies have been described in the literature:

1. *Static Approach:* Valério de Carvalho (2005) and Ben Amor *et al.* (2006) pre-select a subset of PIs and SIs (with S restricted to $|S| = 2$) for the BPP and CSP, respectively. These dual inequalities were then added to the initial RMP.
2. *Dynamic Approach:* Gschwind and Irnich (2016) were the first to suggest the addition of (the most) violated dual inequalities in every (or only some) column-generation iterations. This actually requires a separation algorithm. For PIs, separation can be done by inspection, while for SIs in the BPP and WSIs in the CSP they showed that separation is a by-product of solving the pricing problem. In the next paragraph, we describe how separation can also be accomplished efficiently for VPPs using the results of the SPPRC labeling algorithm.
3. *Mixed Approach:* It is obvious that the static and dynamic approach can be combined in the sense that some dual inequalities are initially added to the RMP and others are separated in the column-generation iterations as done in (Gschwind and Irnich, 2016).

If additional columns for PIs, SIs, or WSIs are added to the RMP, the solution to the master program may consist not only of columns of patterns but of a mixture of these and columns of dual inequalities. Such a solution can however be transformed into a pure pattern-columns solution if the dual inequalities are DDOIs (both statements are in fact equivalent as proven in Gschwind and Irnich, 2016, Proposition 1). An iterative transformation procedure in the case that all added dual inequalities are WSIs is described in detail in (Gschwind and Irnich, 2016, Algorithm 1). The basic idea is as follows. In each iteration, it chooses from the current (partly transformed) solution a pattern column with coefficients $\mathbf{a}$ and a WSI column with coefficients $\mathbf{t} - \mathbf{u}_h$ that are compatible, meaning that $\mathbf{a} + \mathbf{t} - \mathbf{u}_h$ represents a feasible pattern. Let x^* and y^* be the solution value of the chosen pattern and WSI column, respectively. Then, a new solution is constructed from the current one by decreasing the solution values of the chosen pattern and WSI columns by $\min\{x^*, y^*\}$ and increasing the solution value of the column corresponding to pattern $\mathbf{a} + \mathbf{t} - \mathbf{u}_h$ by the same amount. The procedure continues until no more compatible pattern and WSI column-pair exists. If all WSIs that have been added are DDOIs, it is guaranteed that none of them has positive value in the current transformed solution so that a pure pattern-columns solution is found.

As described and done in (Gschwind and Irnich, 2016) where, e.g., SIs are used for BPP, it is also possible to add to the RMP dual inequalities that are (generally) no DDOIs. The intention of this *overstabilization* when using appropriate families of dual inequalities is twofold: First, although they are generally not, these inequalities may in fact be DDOIs for most instances. Second, whether they are actually DDOIs or not, their addition has a stabilizing effect on the column-generation process. Moreover, if the added inequalities are no DDOIs for the given instance, i.e., all dual-optimal solutions are cut-off, this defect can be remedied typically without significant additional computational effort. The recovery procedure of Gschwind and Irnich (2016) can be used to detect and handle overstabilization if all added dual inequalities are WSIs. It proceeds by first trying to build from the RMP solution a pure pattern-columns solution. If this is not possible the RMP has been overstabilized. In this case, there exists a WSI column replacing item $h \in I$ with positive value, but no compatible pattern column exists in the solution. The recovery procedure then eliminates all WSIs replacing item h from the RMP, forbids their re-separation/generation, and restarts the column-generation process to optimize the RMP. The process iterates until a pure pattern-columns solution is found.

Separation of Violated Dual Inequalities. Since the number of valid PIs $\pi_h \geq \pi_i$ is at most quadratic, violated PIs can be found by simple inspection. For this purpose, the list of pairs $(h, i) \in I \times I$ with $\mathbf{w}_h \geq \mathbf{w}_i$ can be determined in a preprocessing step prior to the column-generation process and stored requiring $\mathcal{O}(m^2)$ time and space only. Then, in every separation step, this list needs to be parsed and $\pi_h < \pi_i$ is tested.

Direct inspection is clearly not possible for SIs and WSIs due to their exponential number. When solving the pricing problem with the SPPRC labeling algorithm, however, violated SIs and WSIs can be effectively identified. If Rule 2 is applied for dominance, the identification even is a direct by-product of the labeling procedure: Consider for each single item $h \in I$ its associated label $L_h = (\mathbf{w}_h, \pi_h)$ which results from the extension of $(\mathbf{0}, 0)$ along the second arc (with multiplicity 1) between $h-1$ and h. If L_h is dominated using the strong dominance Rule 2, with strict $>$ in the profit component, then a violated SI or WSI in which h is replaced is found. The item set S of the SI or WSI and, in case of the U-VPP the multiplicities $\mathbf{t} = (t_i)$, can be read from the dominating label $(\mathbf{w}, \pi)$. Indeed, the item set S is the item set of the dominating label and (if needed) the multiplicities $\mathbf{t} = (t_i)$ are those counting the number of aggregated items in the dominating label. In Figure 2, item $h = 1$ with label $L_1 = ((3,3), 0.2)$ is dominated by the label $((2,1), 0.3)$. This latter label has item set $S = \{2\}$ (with multiplicity 1). Hence, the PI $\pi_1 \geq \pi_2$ is violated.

If in the example of Figure 2 a fourth item $h = 4$ with weight $\mathbf{w}_4 = (5,5)$ and profit $\pi_4 = 0.5$ existed, then its label $L_4 = ((5,5), 0.5)$ would be dominated by the label $((5,4), 0.5)$, which has item set $S = \{1,2\}$, so that the violated SI would be $\pi_4 \geq \pi_1 + \pi_2$.

When using the weak dominance Rule 1, however, this procedure is not applicable. Furthermore, the procedure does not produce most violated SIs/WSIs. Still, with some additional effort both cases can be effectively handled using information from the SPPRC as follows: Consider the set $\mathcal{P}$ of all Pareto-optimal labels of the final stage m and add the label $L_h = (\mathbf{w}_h, \pi_h)$ for each item $h \in I$ to $\mathcal{P}$. Then, invoke the dominance algorithm using the strong Rule 2 on the set $\mathcal{P}$ and track for each $L_h, h \in I$ the labels that dominate it with strict $>$ in the profit component. Finally, for each $h \in I$ this list needs to be scanned for (the most) violated SIs/WSIs. When using the algorithm of Bentley (1980) for the dominance algorithm, this separation procedure requires $\mathcal{O}(n \log(n)^D)$ and $\mathcal{O}(n)$ time and space, respectively.

2.4. Branching Schemes

Vanderbeck (1999) suggested a branching scheme for bin packing and cutting stock problems that we adapt to the multidimensional case. We first summarize the branching scheme and explain how the branching constraints are considered in the subproblem. Afterwards, we focus on how to treat DOIs along the branch-and-bound tree.

2.4.1. Subproblem Transformation and Branching Rule

The idea of the branching scheme of Vanderbeck (1999) is to formulate the knapsack subproblem as a pure binary problem. For the 01-VPP, the subproblem is already binary. For B-VPP and U-VPP, the subproblem is transformed into a binary multi-class knapsack problem. For a better understanding, we introduce an example of the transformation and present the general transformation afterwards.

Consider an instance of the B-VPP or U-VPP with $u_1 = 3$ and $u_2 = u_3 = u_4 = 1$. The coefficients $a_i^p \in \{0, 1, \ldots, u_i\}$ of patterns for $i \in I = \{1,2,3,4\}$ are now written in binary code using $m_i = \lceil \log_2(u_i + 1) \rceil$ binary digits per item. Some examples are:

$$p_1 = \begin{pmatrix} 0 \\ 1 \\ 1 \\ 1 \end{pmatrix}_{10} = \begin{pmatrix} \boxed{\begin{smallmatrix} 0 \\ 0 \end{smallmatrix}} \\ 1 \\ 1 \\ 1 \end{pmatrix}_2, \; p_2 = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 1 \end{pmatrix}_{10} = \begin{pmatrix} \boxed{\begin{smallmatrix} 1 \\ 0 \end{smallmatrix}} \\ 0 \\ 1 \\ 0 \end{pmatrix}_2, \; p_3 = \begin{pmatrix} 2 \\ 0 \\ 1 \\ 0 \end{pmatrix}_{10} = \begin{pmatrix} \boxed{\begin{smallmatrix} 0 \\ 1 \end{smallmatrix}} \\ 0 \\ 1 \\ 1 \end{pmatrix}_2, \; p_4 = \begin{pmatrix} 3 \\ 1 \\ 1 \\ 0 \end{pmatrix}_{10} = \begin{pmatrix} \boxed{\begin{smallmatrix} 1 \\ 1 \end{smallmatrix}} \\ 1 \\ 1 \\ 0 \end{pmatrix}_2. \tag{5}$$

In these examples, the original 4-dimensional patterns are now represented using a 5-dimensional binary code because $\sum_{i \in I} m_i = 2 + 1 + 1 + 1 = 5$. For convenience, we define the binary places as pairs I_{01} of items and digits, here $I_{01} = \{(1,0), (1,1), (2,0), (3,0), (4,0)\}$. The binary coefficients are $\tilde{a}_{ib}^p$ for $(i,b) \in I_{01}$, e.g., $\tilde{a}_{1,0}^{p_1} = 0$ and $\tilde{a}_{2,0}^{p_1} = 1$.

Vanderbeck considers subsets of patterns that are defined by a pair (I^0, I^1) where I^0 and I^1 are disjoint subsets of I_{01}:

$$\hat{P}(I^0, I^1) = \{p \in P : \tilde{a}_{ib}^p = 0, (i,b) \in I^0 \text{ and } \tilde{a}_{ib}^p = 1, (i,b) \in I^1\}$$

Whenever a solution x^* to (1) is fractional, Vanderbeck has proven that there exists a pair (I^0, I^1) such that

$$\alpha(I^0, I^1) := \sum_{p \in \hat{P}(I^0, I^1)} x_p^*$$

is fractional. Consequently, one can create the two branches

$$\sum_{p \in \hat{P}(I^0, I^1)} x_p \leq \lfloor \alpha(I^0, I^1) \rfloor \tag{6a}$$

and

$$\sum_{p \in \hat{P}(I^0, I^1)} x_p \geq \lceil \alpha(I^0, I^1) \rceil. \tag{6b}$$

Example 2. Consider an RMP with patterns $p_1, \ldots, p_4$ of equation (5), demands $q_1 = 5, q_2 = 2, q_3 = 3, q_4 = 4$ and the optimal solution $x_1^* = 1.25, x_2^* = 2.75, x_3^* = 0$, and $x_4^* = 0.75$. To branch with the pair $(I^0, I^1) = (\{(2,0)\}, \varnothing)$, find all patterns with value 0 at binary place $(2,0) \in I_{01}$, i.e., the pattern set $\hat{P}(\{(2,0)\}, \varnothing) = \{p_2, p_3\}$ and $\alpha(\{(2,0)\}, \varnothing) = 2.75$.

To branch with the pair $(I^0, I^1) = (\varnothing, \{(1,0),(3,0)\})$, find all patterns with value 1 at binary places $(1,0),(3,0) \in I_{01}$, i.e., the pattern set $P(\varnothing, \{(1,0),(3,0)\}) = \{p_2, p_4\}$ and $\alpha(\varnothing, \{(1,0),(3,0)\}) = 3.5$.

For 01-VPP, the demand of items is 1 so that branching on single items is impossible. However, for any two items $i_1, i_2 \in I$, the relation $0 \leq \alpha(\varnothing, \{(i_1,0),(i_2,0)\}) \leq 1$ holds. Branching on $(I^0, I^1) = (\varnothing, \{(i_1,0),(i_2,0)\})$ means that the items are either forced to be packed together (6b) or not allowed to be packed together (6a). This is Ryan-Foster branching.

When a constraint of type (6) is added to the RMP, the subproblem must take the dual price of this constraint into account. More generally, consider a branch-and-bound node n, and let L^n and G^n be sets of branching constraints of the form (6a) and (6b) with corresponding dual variables $\mu_j \leq 0, j \in L^n$ and $\nu_k \geq 0, k \in G^n$, respectively, that are active at n. Note that each inequality is defined on the basis of a unique pair (I_j^0, I_j^1) for $j \in L^n$ and a unique pair (I_k^0, I_k^1) for $k \in G^n$, respectively. Let the corresponding pattern sets be $\hat{P}(I_j^0, I_j^1)$ and $\hat{P}(I_k^0, I_k^1)$. The associated subproblem, i.e., the MKP written with binary pattern coefficients and branching constraints, can now be written as a pure binary problem. Herein, binary variables $\hat{y}_{ib}$ for $(i,b) \in I_{01}$ are binary representations of y-variables in the MKP (3), i.e., $y_i = \sum_{b=0}^{m_i-1} 2^b \hat{y}_{ib}$. The associated pattern to $\hat{y} \in \{0,1\}^{I_{01}}$ is

$$a(\hat{y}) = \left(\sum_{b=0}^{m_1-1} 2^b \hat{y}_{1b}, \sum_{b=0}^{m_2-1} 2^b \hat{y}_{2b}, \ldots, \sum_{b=0}^{m_m-1} 2^b \hat{y}_{mb} \right) \in P. \tag{7}$$

Moreover, additional binary variables $l_j, j \in L^n$, and $g_k, k \in G^n$, are needed to indicate whether the resulting pattern $a(\hat{y})$ occurs in the respective constraints (6a) and (6b). The subproblem reads as follows:

$$\max \sum_{i\in I}\sum_{b=0}^{m_i-1}(2^b\pi_i)\hat{y}_{ib} \quad + \sum_{j\in L^n}\mu_j l_j + \sum_{k\in G^n}\nu_k g_k \tag{8a}$$

$$\text{s.t.} \sum_{i\in I}\sum_{b=0}^{m_i-1}(2^b w_i^d)\hat{y}_{ib} \le W^d, \qquad d\in D \tag{8b}$$

$$\sum_{b=0}^{m_i-1} 2^b\hat{y}_{ib} \le u_i, \qquad i\in I \tag{8c}$$

$$a(\hat{y})\in \hat{P}(I_j^0,I_j^1) \quad \Longrightarrow \quad l_j=1, \qquad j\in L^n \tag{8d}$$

$$a(\hat{y})\notin \hat{P}(I_k^0,I_k^1) \quad \Longrightarrow \quad g_k=0, \qquad k\in G^n \tag{8e}$$

$$\hat{y}_{ib}\in\{0,1\}, \qquad (i,b)\in I_{01} \tag{8f}$$

$$l_j\in\{0,1\}, \qquad j\in L^n \tag{8g}$$

$$g_k\in\{0,1\}, \qquad k\in G^n \tag{8h}$$

The objective (8a) consists of three terms, one for the regular profit achieved for the selected items, similar to (3a), and two other terms that can be interpreted as penalties imposed by the branching constraints (6); note that $\mu_j \le 0, j\in L^n$ and $\nu_k\ge 0, k\in G^n$. The capacity constraints (8b) are reformulations of (3b). The upper bounds for the pattern coefficients are incorporated by (8c). The coupling between the $\hat{y}$-variables and the branching-related indicator variables l and g are given by (8d) and (8e). Their linearization is straightforward and omitted here for the sake of brevity. The domains of all variables are given by (8f)–(8h).

2.4.2. SPPRC Graph Adaption

The main advantage of solving the MKP pricing problems (8) via SPPRC labeling is that the above branching constraints can all be implemented by (i) modifying profits of arcs and/or (ii) aggregating stages of the SPPRC digraph defined in Section 2.1. Such modifications do not alter the general structure of the SPPRC. Hence, the same dynamic-programming labeling approach as presented in Section 2.2 remains applicable.

The type of graph modification needed to implement the given branching decisions depends on how many (original) items interact in the branching. For a single branching decision (I^0,I^1) with $I^0,I^1\subset I_{01}$, the set of affected items is $I^p = proj_1(I^0\cup I^1)$ (projection onto the first component, i.e., the item component). We explain the modifications required by the following three types of branching decisions:
(i) Branching decision (I^0,I^1) for single items, i.e., $|I^p|=1$;
(ii) Branching decision (I^0,I^1) for more than one item, i.e., $|I^p|>1$;
(iii) Several branching decisions (I_j^0,I_j^1) and/or (I_k^0,I_k^1).

In the following, we assume that a branch-and-bound node n with branching constraints $j\in L^n$ and $k\in G^n$ and dual prices μ_j to (6a) and ν_k to (6b) are given, respectively.

Branching Decision for Single Items. The case of branching decisions that each affect only a single item, i.e, $I^p = proj_1(I^0\cup I^1)=\{i\}$ for an item $i\in I$ can be handled by modifying the profits of some arcs. The arcs affected by these constraints are those arcs $e_i=(i-1,i)\in E(i)$ of the stage referring to item i. An arc is affected by a branching constraint if and only if its pattern coefficients $a_i(e_i)$, written as $\tilde{a}_{ib}(e_i)$ in binary coding, fulfill

$$\tilde{a}_{ib}(e_i)=0 \quad \text{for all } (i,b)\in I^0 \qquad \text{and} \qquad \tilde{a}_{ib}(e_i)=1 \quad \text{for all } (i,b)\in I^1.$$

In this case, the dual price μ_j is added if $(I^0,I^1)=(I_j^0,I_j^1)$ for some $j\in L^n$ and/or the dual price ν_k is added if $(I^0,I^1)=(I_k^0,I_k^1)$ for some $k\in G^n$.

Example 3. We assume that there are two branching decisions active at a branch-and-bound node n_A, one of type (6a) with dual price $\mu_A \leq 0$ and one of type (6b) with dual price $\nu_A \geq 0$. We further assume that the (first) $\leq$-condition refers to $(I^0, I^1) = (\{(1,1)\}, \varnothing)$, i.e., to the item $i = 1$ and the first binary digit $b = 1$ with value 2^1. The (second) $\geq$-condition refers to $(I^0, I^1) = (\varnothing, \{(1,0)\})$, i.e., the same item $i = 1$ and the lowest binary digit $b = 0$ with value 2^0. Figure 3 shows the modification of the SPPRC digraph.

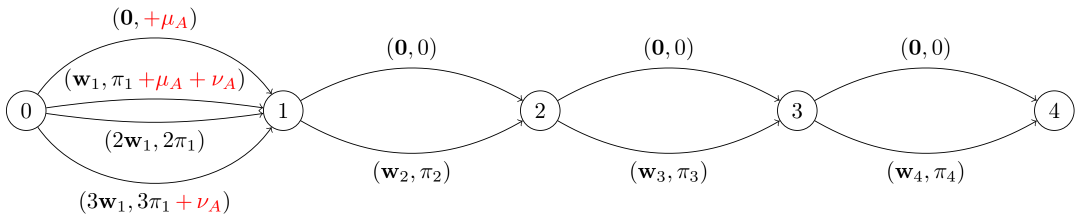


Figure 3: SPPRC digraph for a four-item MKP with $u_1 = 3, u_2 = u_3 = u_4 = 1$ (cf. Figure 1 and equation (5)). The arcs $e \in E$ are shown with their D-dimensional weight and profit component $(w(e), \pi(e))$ after implementing two branching constraints. The depicted situation refers to a branch-and-bound node with a $\leq$-constraint referring to $(I^0, I^1) = (\{(1,1)\}, \varnothing)$ with dual price μ_A and a $\geq$-constraint referring to $(I^0, I^1) = (\varnothing, \{(1,0)\})$ with dual price ν_A.

Branching Decision for Several Items. Branching decisions that affect several items at the same time are handled by item aggregation. If $|I^p| = |proj_1(I^0 \cup I^1)| > 1$ for a branching constraint defined by (I^0, I^1), all items in I^p are aggregated into a single *virtual* item. Assuming $I^p = \{i_1, i_2, \ldots, i_r\}$, new arcs are built as combinations of all possible multiplicities of the items $i_1, i_2, \ldots, i_r$.

Example 4. (continued from Example 3) We assume that a son node n_B of the branch-and-bound node n_A of Example 3 has an additional active branching decision based on $(I^0, I^1) = (\varnothing, \{(3,0), (4,0)\})$ resulting from a $\geq$-condition of the form (6b) with dual price ν_B. Here, $I^p = \{3, 4\}$ requires an aggregation over the two items 3 and 4 with $u_3 = u_4 = 1$. The number of possible combinations is therefore $(u_3 + 1)(u_4 + 1) = 4$, where each of the two items can either be included or excluded. Due to $I^1 = \{(3,0), (4,0)\}$, the only combination affected by the additional branching constraint is the one in which both items 3 and 4 are included, i.e., for weight $\mathbf{w}_3 + \mathbf{w}_4$ and original profit $\pi_3 + \pi_4$. The modified arc now has profit $\pi_3 + \pi_4 + \nu_B$.

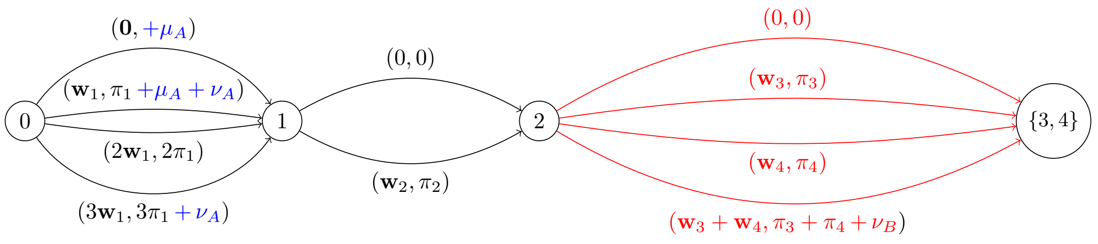
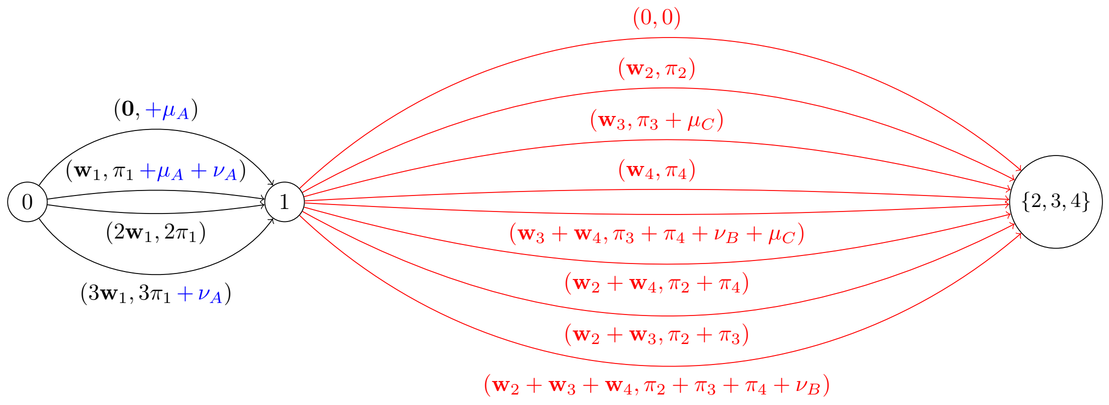

$(\mathbf{0}, +\mu_A)$
$(\mathbf{w}_1, \pi_1 + \mu_A + \nu_A)$
$(2\mathbf{w}_1, 2\pi_1)$
$(3\mathbf{w}_1, 3\pi_1 + \nu_A)$
$(0, 0)$
$(\mathbf{w}_2, \pi_2)$
$(0, 0)$
$(\mathbf{w}_3, \pi_3)$
$(\mathbf{w}_4, \pi_4)$
$(\mathbf{w}_3 + \mathbf{w}_4, \pi_3 + \pi_4 + \nu_B)$
0
1
2
$\{3, 4\}$

Figure 4: Modified SPPRC digraph resulting from three branching decisions. The first two decisions produce a digraph as in Figure 3. The additional third branching constraint, i.e., a $\geq$-condition on $(I^0, I^1) = (\varnothing, \{(3,0), (4,0)\})$ with dual price ν_B leads to the aggregation of the original items 3 and 4 indicated in red.

Several Branching Decisions. It can happen that several branching constraints overlap, i.e., the affected item sets are overlapping. More formally, consider several branching constraints affecting item sets $I^p_1, \ldots I^p_r$. These item sets are overlapping if they can be (re-)ordered such that $(I^p_1 \cup \ldots \cup I^p_{j-1}) \cap I^p_j \neq \varnothing$ for all $j \in \{1, \ldots, r\}$. In this case, each such union $I^p = I^p_1 \cup \ldots \cup I^p_r$ of the items must be aggregated. This works as described before by constructing all possible combinations of items in I^p.

Example 5. (continued from Example 4) We assume now that at a son node n_C of the branch-and-bound node n_B of Example 4 has an additional active branching decision on $(I^0, I^1) = (\{(2,0)\}, \{(3,0)\})$ resulting from a $\leq$-condition of the form (6a) with dual price μ_C. At node n_C the sets of overlapping item sets are $I_1^p = \{1\}$ (resulting from the constraints active at the grandfather node n_A) and $I_2^p = \{3,4\} \cup \{2,3\} = \{2,3,4\}$ (resulting from the constraints defining n_B and n_C). The aggregation of items 2, 3, and 4 allows for eight combinations, each resulting from the inclusion or exclusion of the respective item. The modifications of nodes n_A and n_B remain and, additionally, μ_C is added to the profit of arcs excluding item 2 and including item 3. Figure 5 shows the modification of the SPPRC digraph.

In general, the item sets I^p may consist of non-consecutive items, e.g., $I_1^p = \{1,3\}$ and $I_1^p = \{2,4\}$, so that the items/nodes of the SPPRC digraph have to be re-ordered.



$(0,0)$
$(\mathbf{w}_2, \pi_2)$
$(\mathbf{0}, +\mu_A)$
$(\mathbf{w}_3, \pi_3 + \mu_C)$
$(\mathbf{w}_1, \pi_1 + \mu_A + \nu_A)$
$(\mathbf{w}_4, \pi_4)$
0
1
$\{2,3,4\}$
$(2\mathbf{w}_1, 2\pi_1)$
$(\mathbf{w}_3 + \mathbf{w}_4, \pi_3 + \pi_4 + \nu_B + \mu_C)$
$(\mathbf{w}_2 + \mathbf{w}_4, \pi_2 + \pi_4)$
$(3\mathbf{w}_1, 3\pi_1 + \nu_A)$
$(\mathbf{w}_2 + \mathbf{w}_3, \pi_2 + \pi_3)$
$(\mathbf{w}_2 + \mathbf{w}_3 + \mathbf{w}_4, \pi_2 + \pi_3 + \pi_4 + \nu_B)$

Figure 5: Fourth branching with $(I^0, I^1) = (\{(2,0)\}, \{(3,0)\})$ and $\leq$-condition (6a) with dual price μ_C. The profit of arcs excluding item 2 and including item 3 has to be modified by adding μ_C. The modification of the third branching decision (cf. Figure 4) remains. The items $I_2^p = \{2,3,4\}$ affected by the third and fourth branching decision are aggregated yielding $(u_2+1)(u_3+1)(u_4+1) = 8$ arcs indicated in red.

2.4.3. Handling of Dual-Optimal Inequalities

When using (D)DOIs within a branch-and-price algorithm, it needs to be ensured that the (D)DOIs are compatible with the branching decisions taken. Consider the following example.

Example 6. Consider again the RMP of Example 2 with patterns $p_1, \dots, p_4$ and demands $q_1 = 5, q_2 = 2, q_3 = 3$, and $q_4 = 4$. Additionally, the PI $\pi_2 \geq \pi_1$ with associated primal variable x_5 is added to the RMP and the $\leq$-branching constraint (6a) with $(I^0, I^1) = (\{(2,0)\}, \varnothing)$ and $\lfloor \alpha(\{(2,0)\}, \varnothing) \rfloor = 2$ has been imposed. The resulting constraints are

$$\begin{pmatrix}0\\1\\1\\1\\0\end{pmatrix} x_1 + \begin{pmatrix}1\\0\\1\\1\\1\end{pmatrix} x_2 + \begin{pmatrix}2\\0\\1\\0\\1\end{pmatrix} x_3 + \begin{pmatrix}3\\1\\1\\0\\0\end{pmatrix} x_4 + \begin{pmatrix}1\\-1\\0\\0\\0\end{pmatrix} x_5 \quad \begin{matrix}\geq\\\geq\\\geq\\\geq\\\leq\end{matrix} \quad \begin{pmatrix}5\\2\\3\\4\\2\end{pmatrix}.$$

An optimal solution to the corresponding RMP is $x_1^* = x_2^* = 2, x_3^* = 0, x_4^* = x_5^* = 0.75$. It can be transformed into the pure patterns solution that uses pattern p_1 1.25 times, pattern p_2 2.75 times, and pattern p_4 0.75 times. This solution violates the branching constraint because

$$\sum_{p \in \hat{P}(\{(2,0)\}, \varnothing)} x_p^* = x_2^* + x_3^* = 2.75 \not\leq \lfloor \alpha(\{(2,0)\}, \varnothing) \rfloor = 2.$$

In essence, the issue with branching and the use of (D)DOIs in our case is the following: The branching decisions translate into RMP-constraints that are formulated on a set of pattern columns $\hat{P}(I^0, I^1)$. By replacing items in certain pattern columns $p \in P \setminus \hat{P}(I^0, I^1)$, (D)DOI-columns can then be used to implicitly represent pattern columns $p' \in \hat{P}(I^0, I^1)$, and vice versa, without any impact on the left-hand side of the corresponding RMP-constraint (6a) or (6b) so that a solution violating the branching decisions can result from the RMP.

One strategy to remedy this defect is to eliminate the respective (D)DOIs from the RMP and prevent their regeneration when using (D)DOIs in a dynamic approach, see Section 2.3. This strategy was also followed by Alves and Valério de Carvalho (2008) for solving the multiple length CSP with a stabilized column-generation approach. To the best of our knowledge, they were the first to use (D)DOIs within a fully-fledged branch-and-price algorithm. They derive conditions for the validity of the (D)DOIs that they use with respect to specific branching decisions present at a certain branch-and-bound node. All (D)DOIs that do not meet these conditions are then removed. Note that they use a different branching scheme that is compatible with their solution approach for the subproblem, so that their rules are not applicable in our case.

Consider a branch-and-bound node n in our algorithm. The set of items affected by the corresponding branching decisions is I^p. All (D)DOIs that are not related to any of the affected items, i.e., with $(\{h\} \cup S) \cap I^p = \varnothing$, are clearly valid at n. A straightforward way to ensure compatibility of the RMP solution with the branching decisions is the elimination of all other (D)DOIs that are related to at least one affected item, i.e., with $(\{h\} \cup S) \cap I^p \neq \varnothing$. We can, however, also derive some additional conditions for the validity of (D)DOIs of the latter type. This enhancement is explained in the following.

Note first that given a branching decision on the patterns $\hat{P}(I^0, I^1)$, in the $\leq$-branch (6a) it has to be ensured that no pattern $p' \in \hat{P}(I^0, I^1)$ can be implicitly represented using (D)DOIs and patterns $p \in P \setminus \hat{P}(I^0, I^1)$, while in the $\geq$-branch (6b) no patterns $p' \in P \setminus \hat{P}(I^0, I^1)$ are allowed to be implicitly represented using (D)DOIs and patterns $p \in \hat{P}(I^0, I^1)$. Then, for 01-VPP, where coefficients are binary and the (D)DOIs we use are SIs, SIs that are related to at least one affected item are compatible in the $\leq$-branch if $h \notin proj_1(I^0)$ and $S \cap proj_1(I^1) = \varnothing$ hold. It is easy to verify that with the compatible SIs no patterns $p' \in \hat{P}(I^0, I^1)$ can be implicitly represented. In analogy, SIs are compatible in the $\geq$-branch if $S \cap proj_1(I^0) = \varnothing$ and $h \notin proj_1(I^1)$ hold.

For B-VPP and U-VPP, consider first only one of the affected items $i \in I^p$. A condition similar to the above condition for the binary case is as follows: In the $\leq$-branch, if *all* bits of i are fixed to 0 WSIs with $h \neq i$ are compatible with respect to item i, while if *all* bits of i are fixed to 1 WSIs with $i \notin S$ are compatible with respect to item i. The analog conditions in the $\geq$-branch are $i \notin S$ if all bits of i are fixed to 0 and $h \neq i$ if all bits of i are fixed to 1. Consider now the case that some of the m_i bits of item i are fixed to some value by the branching decisions and let $m'_i < m_i$ be the largest bit that is fixed. Then, WSIs with $i \in S$ and $t_i \mod 2^{m'_i+1} \equiv 0$ are compatible with respect to item i. For a WSI to be compatible with the branching decisions at n, it has to fulfill one of the above conditions for each of the items $i \in (\{h\} \cup S) \cap I^p$.

2.4.4. Branching Strategy

For VPPs, the lower bound $\lceil z_{LP}^{VPP} \rceil$ is often the optimal number of patterns. In fact, this holds for all instances that we solved. The main issue is therefore to find such an integer solution by reducing the fractionality of the master solution. Hence, we conduct a depth-first search strategy and first explore the branch with greater-or-equal constraint (6b).

For 01-VPP, we adopt Ryan-Foster branching forcing to pack items together with $I^0 = \varnothing$ and $I^1 = \{(i_1, 0), (i_2, 0) : i_1, i_2 \in I\}$ (cf. Example 2). The branching priority is to select the most fractional corresponding $\alpha(I^0, I^1)$ value. For B-VPP, our priority is first to keep the subtrees balanced choosing the cardinality $|I^0| + |I^1|$ as low as possible, second forcing to pack items (and not to not-pack items), i.e., prohibit $I^0 = \{(i, 0) : i \in I\}$ and $I^1 = \varnothing$, and third to take the most fractional $\alpha(I^0, I^1)$ value. For U-VPP, it may happen that patterns p with a coefficient $a_{ip} > q_i$ are part of a solution. In this case, we first forbid these patterns creating a single branch. Afterwards, the branching priority of B-VPP is adopted.

3. Computational Results

The branch-and-price algorithms were implemented in C++ and compiled into 64-bit single-thread code with MS Visual Studio 2013. The experiments were conducted on a standard PC with an Intel(R) Core(TM) i7-5930k clocked at 3.5 GHz and 64 GB of RAM, by allowing a single thread for each run. CPLEX 12.6.2 was used to solve the RMP in the column-generation algorithms and as a primal MIP-based heuristic solver called after the solution of each branch-and-bound node using the so far generated columns. In the latter case, computation times are limited to a maximum of 5 seconds per call. CPLEX's default values were kept for all parameters.

3.1. Instances

In order to compare our algorithms with the leading exact approaches (Brandao and Pedroso, 2016; Hu *et al.*, 2017) from the literature, we base the computational experiments on the same set of instances which are those of Caprara and Toth (2001) who generated 2-dimensional VPPs. Their instances are grouped into ten classes with 40 instances each, according to the weights generation scheme. The ten classes are further divided into four subclasses each with ten instances of identical size: In the first nine classes, the instances have 25, 50, 100 and 200 items, respectively. In the tenth class, the instances have 24, 51, 99 and 201 items, respectively. In total, Caprara and Toth (2001) generated 400 benchmark instances.

In addition, Brandao and Pedroso (2016) combined each subclass with ten 2-dimensional instances into a single 20-dimensional instance, yielding another 40 instances. These instances have not been considered in the article of Hu *et al.* (2017).

For all 440 instances, the demand of each item is almost always one. Neither Hu *et al.* (2017) nor Brandao and Pedroso (2016) aggregated the (very few) items with identical weights. Therefore, we also solve these instances only with the 01-VPP model. Results on these binary VPPs are presented in Section 3.3.

Because the demand of the instances is almost always one, we generated new instances with larger demands to compare the three models 01-VPP, B-VPP, and U-VPP. The item sizes of the new instances are the same as already used by Caprara and Toth (2001) and Brandao and Pedroso (2016), respectively, but the demand of the items is larger: We generated the larger demand using a discrete, uniformly distributed random variable between 1 and 100. Section 3.4 discusses our results for these instances. The new instances are available on `http://logistik.bwl.uni-mainz.de/benchmarks.php`.

3.2. Computational Setup

The initial RMP is constructed as follows. First, we add patterns for single items, i.e., unit vectors and columns with only one non-negative coefficient $a_i^p = u_i$ (where $u_i = u_i^{01}$ or $= u_i^B$ or $= u_i^U$ depending on the VPP formulation). Second, additional patterns are constructed with the help of the well-known first-fit (FF) and best-fit (BF) heuristics (Johnson, 1973). Both FF and BF are once run with the items sorted decreasingly by their 1-norm. Moreover, another 99 runs of FF and BF are performed in which the items are sorted randomly.

To measure the impact of (D)DOIs, all instances were solved with and without adding (D)DOIs. Pretests have shown that the *mixed approach* described in Section 2.3 of adding some (D)DOIs to the initial RMP and adding the most violated (D)DOIs later on dynamically on average dominates the other strategies. Since the VPP instances typically do not allow SI and WSI with $|S| > 2$, our DOI selection and separation strategy is straightforward: For the initialization of the first RMP, two different kinds of static (D)DOIs are added. On the one hand, for each item i the PI $\pi_i \geq \pi_j$ with $j = \arg\min_{k \in I}\{||\mathbf{w}_i - \mathbf{w}_k||_1 : \mathbf{w}_i \geq \mathbf{w}_k\}$ is added. These (D)DOIs are the multi-dimensional extension of cuts of type 1 and ranking constraints, respectively (Valério de Carvalho, 2005; Ben Amor *et al.*, 2006). On the other hand, all SIs with $|S| = 2$ are added. There could exist $\mathcal{O}(m^3)$ such inequalities but for the instances of the benchmark, the number is rather small. These (D)DOIs are the multi-dimensional extension of cuts of type 2 (Valério de Carvalho, 2005). Dynamic DOIs are only added while solving the root node. Separation of violated (D)DOIs is done with a straightforward enumeration procedure considering all PIs by inspection. The DOIs remain in the RMP except when becoming invalid due to some active branching decisions, see Section 2.4.3.

3.3. Results for Binary Vector Packing Problems

In this section, we report results of our branch-and-price algorithms when applied to the 01-VPP. According to Section 2.2, we compare two versions of dynamic-programming labeling for the solution of the column-generation subproblems, one is a monodirectional labeling using a weak but fast dominance rule and the other is a more sophisticated bidirectional labeling using the strong dominance rule. As the latter leads to less labels but a more time-consuming dominance algorithm, it is not clear a priori which of the two labeling approaches is superior. Moreover, we conduct experiments with non-stabilized and stabilized branch-and-price algorithms. This gives rise to four variants of branch-and-price.

Comparison with Other Exact Algorithms. We compare our algorithms against the branch-and-price of Hu *et al.* (2017) and the graph compression and arc-flow model based approach of Brandao and Pedroso (2016). While Hu *et al.* (2017) restrict computation times to a maximum of 600 seconds per instance, Brandao and Pedroso (2016) did not impose any time limit. To improve comparability of results, we let our branch-and-price algorithms run for a maximum of 600 as well as 3600 seconds.

The results are summarized in Tables 2 and 3, where the first table reports average results for the 400 instances with 2-dimensional items/bins and the second table reports instance-wise results for the 40 instances of dimension 20. Note that Hu *et al.* (2017) did not consider the latter instances. Hence, we do not report results for a time limit of 600 seconds in Table 3. The table entries have the following meanings:

Class:	class of instances; weights generated using different distributions;
Items:	number of items in the instance;
opt:	number of instances (out of 10) solved to proven optimality within 1 hour (3600 seconds);
$\mathbf{opt}_{10}$**:**	same for a time limit of 10 minutes (600 seconds);
T**:**	computation time in seconds for a single instance;
T_{LP}**:**	computation time in seconds for solving the linear relaxation of the master program for a single instance;
$\bar{T}$**:**	average computation time in seconds over 10 instances; unsolved instances are taken into account with the time limit of 1 hour (3600 seconds);
$\bar{T}_{10}$**:**	same for a time limit of 10 minutes (600 seconds);
LB:	lower bound obtained by linear relaxation or Lagrangian lower bound if master program is terminated prematurely.

For all approaches, the instances of the classes 2, 3, and 8 are well solvable and computation times always remain below 15 seconds. The approach of Brandao and Pedroso is the only one that solves all 88 instances of the classes 1 and 10 (80 of dimension $D = 2$ and 8 of dimension $D = 20$). Instances of the classes 6 and 7 seem to be hard for the branch-and-price algorithm of Hu *et al.* as they solve only 58 of the 80 2-dimensional instances. In contrast, all 88 instances of classes 6 and 7 are solved with the algorithm of Brandao and Pedroso as well as all of our branch-and-price algorithms.

The classes 4, 5, and 9 comprise the most difficult instances for all approaches. The algorithm of Brandao and Pedroso can only cope with smaller-sized instances with 25 items of classes 4 and 5. However, it solves all $30+3 = 33$ instances of class 9 with 25, 50, and 100 items. On the latter instances of class 9, the branch-and-price of Hu *et al.* solves only the 20 instances with 25 and 50 items and our best variant (with stabilization and monodirectional labeling) solves $30+4 = 34$ instances.

Comparing all algorithms on the basis of a reduced computation time of 600 seconds, the approaches of Hu *et al.* and Brandao and Pedroso show a comparable performance solving 321 and 320 instances (of dimension 2). Our four branch-and-price algorithms deliver within the same time between 334 and 351 optimal solutions. With the longer computation time of 3600 seconds the algorithm of Brandao and Pedroso gives $320+33 = 353$ optimal solutions, while our four branch-and-price algorithms provide between 346+32=378 and 368+34=392 optimal solutions. Overall, the four algorithms together provide optima for 370+34=404 of the 440 instances. Together with the results of Hu *et al.* and Brandao and Pedroso, 24+6=30 instances of the benchmark remain open.

Table 2: Results and comparison for 2-dimensional 01-VPP

						Monodirectional with weak dominance						Bidirectional with strong dominance					
Class		Hu et al.		Brandao/P.		Stabilized			Non-Stabilized			Stabilized			Non-Stabilized		
	Items	opt$_{10}$	$\bar{T}_{10}$	opt	$\bar{T}$	opt$_{10}$	opt	$\bar{T}$	opt$_{10}$	opt	$\bar{T}$	opt$_{10}$	opt	$\bar{T}$	opt$_{10}$	opt	$\bar{T}$
1	25	10	1.5	10	0.1	10	10	0.3	10	10	0.3	10	10	0.1	10	10	0.1
	50	9	89.6	10	1.6	10	10	8.8	10	10	8.9	10	10	1.2	10	10	1.5
	100	3	529.0	10	67.0	7	10	418.2	5	10	554.4	10	10	140.1	10	10	182.0
	200	0	600.0	10	7601.4	0	2	3289.7	0	0	3600.0	0	3	2846.3	0	3	3507.7
2	25	10	0.7	10	<0.1	10	10	<0.1	10	10	<0.1	10	10	<0.1	10	10	<0.1
	50	10	1.0	10	<0.1	10	10	<0.1	10	10	<0.1	10	10	<0.1	10	10	<0.1
	100	10	2.8	10	0.2	10	10	<0.1	10	10	0.1	10	10	<0.1	10	10	0.1
	200	10	11.0	10	6.9	10	10	0.1	10	10	0.1	10	10	0.1	10	10	0.1
3	25	10	0.5	10	<0.1	10	10	<0.1	10	10	<0.1	10	10	<0.1	10	10	<0.1
	50	10	1.0	10	<0.1	10	10	<0.1	10	10	<0.1	10	10	<0.1	10	10	<0.1
	100	10	2.5	10	0.1	10	10	<0.1	10	10	<0.1	10	10	<0.1	10	10	<0.1
	200	10	8.1	10	0.2	10	10	0.1	10	10	0.1	10	10	0.1	10	10	0.1
4	25	10	7.9	10	22.0	10	10	13.6	10	10	13.4	10	10	0.6	10	10	0.5
	50	10	10.9	–	–	10	10	48.2	10	10	49.0	10	10	7.0	10	10	6.2
	100	6	419.8	–	–	8	9	660.4	3	3	2602.4	4	10	687.1	2	5	2464.3
	200	0	600.0	–	–	0	0	3600.0	0	0	3600.0	0	0	3600.0	0	0	3600.0
5	25	10	20.3	10	10.6	10	10	16.5	10	10	20.1	10	10	0.3	10	10	0.4
	50	10	31.4	–	–	10	10	60.9	10	10	64.7	10	10	34.5	10	10	39.9
	100	10	93.9	–	–	10	10	367.6	10	10	422.3	0	0	3600.0	0	0	3600.0
	200	7	346.0	–	–	0	7	2468.8	0	6	2879.4	0	0	3600.0	0	0	3600.0

(Table 2 cont'd)

						Monodirectional with weak dominance						Bidirectional with strong dominance					
Class		Hu et al.		Brandao/P.		Stabilized			Non-Stabilized			Stabilized			Non-Stabilized		
	Items	opt$_{10}$	$\bar{T}_{10}$	opt	$\bar{T}$	opt$_{10}$	opt	$\bar{T}$	opt$_{10}$	opt	$\bar{T}$	opt$_{10}$	opt	$\bar{T}$	opt$_{10}$	opt	$\bar{T}$
6	25	10	0.1	10	<0.1	10	10	<0.1	10	10	<0.1	10	10	<0.1	10	10	<0.1
	50	10	0.5	10	0.1	10	10	0.2	10	10	0.1	10	10	0.1	10	10	0.1
	100	6	308.5	10	0.3	10	10	1.4	10	10	1.3	10	10	0.9	10	10	0.9
	200	0	600.0	10	4.8	10	10	16.5	10	10	20.2	10	10	10.3	10	10	21.3
7	25	10	0.1	10	<0.1	10	10	<0.1	10	10	<0.1	10	10	<0.1	10	10	<0.1
	50	10	10.2	10	0.1	10	10	0.2	10	10	0.2	10	10	0.1	10	10	0.1
	100	10	64.8	10	1.7	10	10	1.9	10	10	2.0	10	10	1.4	10	10	1.7
	200	2	481.6	10	14.0	10	10	29.2	10	10	33.4	10	10	31.7	10	10	49.5
8	25	10	<0.1	10	<0.1	10	10	<0.1	10	10	<0.1	10	10	<0.1	10	10	<0.1
	50	10	0.1	10	<0.1	10	10	<0.1	10	10	0.1	10	10	<0.1	10	10	0.1
	100	10	0.2	10	0.1	10	10	0.1	10	10	0.1	10	10	0.1	10	10	0.1
	200	10	0.7	10	0.2	10	10	0.3	10	10	0.2	10	10	0.2	10	10	0.2
9	25	10	1.1	10	0.1	10	10	0.2	10	10	0.2	10	10	0.1	10	10	0.1
	50	10	9.9	10	0.7	10	10	7.5	10	10	7.6	10	10	2.3	10	10	2.2
	100	0	600.0	10	28.1	10	10	342.5	9	10	402.8	6	10	816.4	6	10	1013.1
	200	0	600.0	–	–	0	0	3600.0	0	0	3600.0	0	0	3600.0	0	0	3600.0
10	24	10	<0.1	10	<0.1	10	10	<0.1	10	10	<0.1	10	10	<0.1	10	10	<0.1
	51	10	0.2	10	0.1	10	10	0.1	10	10	0.2	10	10	0.1	10	10	0.1
	99	10	0.5	10	0.7	10	10	2.4	10	10	3.0	10	10	2.2	10	10	2.6
	201	8	222.4	10	155.4	6	10	647.4	7	8	1106.7	8	10	313.2	6	8	1370.8
Total																	
	600 s	321	142.0	320	127.9	351		109.6	344		119.4	338		114.7	334		121.0
	3600 s			320	727.9		368	390.1		357	474.8		353	482.4		346	576.6
	>3600 s			330													

Table 3: Results for 20-dimensional 01-VPP

			Monodirectional with weak dominance				Bidirectional with strong dominance				
Class		Brandao/P.	Stabilized		Non-Stabilized		Stabilized		Non-Stabilized		
	Items	T	T_{LP}	T	T_{LP}	T	T_{LP}	T	T_{LP}	T	LB
1	25	0.1	0.1	0.1	0.1	0.1	<0.1	<0.1	0.1	0.1	8
	50	0.8	0.6	1.1	0.5	0.6	0.7	0.8	1.4	1.5	15
	100	36.3	39.2	39.9	32.8	53.1	463.3	467.4	357.5	358.0	29
	200	1374.2	1685.6	2944.4	1650.3	3600.0	3600.0	3600.0	3600.0	3600.0	57
2	25	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	23
	50	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	48
	100	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	98
	200	<0.1	0.1	0.1	<0.1	<0.1	0.1	0.1	<0.1	<0.1	193
3	25	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	23
	50	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	48
	100	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	98
	200	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	0.1	0.1	193
4	25	50.3	109.9	109.9	112.1	112.1	24.9	24.9	25.5	25.5	4
	50	–	3600.0	3600.0	3600.0	3600.0	3600.0	3600.0	3600.0	3600.0	7
	100	–	3600.0	3600.0	3600.0	3600.0	3600.0	3600.0	3600.0	3600.0	
	200	–	3600.0	3600.0	3600.0	3600.0	3600.0	3600.0	3600.0	3600.0	
5	25	73.2	1958.4	1958.4	1996.7	1996.7	15.5	15.5	15.4	15.4	2
	50	–	3600.0	3600.0	3600.0	3600.0	3600.0	3600.0	3600.0	3600.0	
	100	–	3600.0	3600.0	3600.0	3600.0	3600.0	3600.0	3600.0	3600.0	
	200	–	3600.0	3600.0	3600.0	3600.0	3600.0	3600.0	3600.0	3600.0	
6	25	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	13
	50	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	26
	100	0.1	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	52
	200	0.2	0.1	0.2	0.1	0.7	0.1	0.2	0.1	0.7	102
7	25	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	13
	50	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	26
	100	0.1	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	52
	200	0.2	0.1	0.6	0.1	0.1	0.1	0.7	0.1	0.1	102
8	25	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	21
	50	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	37
	100	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	63
	200	0.1	<0.1	0.6	<0.1	0.6	<0.1	0.7	<0.1	0.6	104
9	25	0.1	<0.1	<0.1	<0.1	0.1	<0.1	<0.1	<0.1	<0.1	8
	50	0.4	0.1	0.4	0.2	0.3	0.1	0.4	0.2	0.5	16
	100	12.8	6.2	7.2	5.5	6.4	19.7	20.5	28.5	29.3	31
	200	–	880.8	3571.9	812.3	2997.0	3600.0	3600.0	3600.0	3600.0	58
10	24	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	10
	51	0.1	0.1	0.1	<0.1	0.1	<0.1	0.1	<0.1	0.1	21
	99	0.9	4.1	4.6	4.5	4.9	11.9	12.3	14.4	14.8	39
	201	14.5	243.3	245.8	274.7	277.2	2310.1	2312.3	2474.8	2476.1	77
Total		669.1	663.2	762.1	628.0	734.7	791.2	791.4	793.0	793.1	
	opt	33		34		32		32		32	

Comparison of the Four Variants of Branch-and-Price. Tables 2 and 3 clearly indicate that the stabilization of the column-generation process using (D)DOIs is always beneficial: (average) computation times with stabilization are always below those of the corresponding variant without stabilization with very few exceptions. In these exceptional cases, the additional time for the approach with stabilization is always below 1 second. On the positive side, the speedup obtained with stabilization can reach a factor of more than 4, see, e.g., class 4 instances with 100 items in dimension 2. It is also obvious that in dimension 20 the acceleration caused by stabilization is significantly smaller than in dimension 2. This can be explained by the fact that there exist fewer valid (D)DOIs, e.g., pair inequalities (PI), in dimension 20 than in dimension 2. As a consequence of all these observations, we restrict the remainder of the comparison to the two variants that use stabilization, i.e., monodirectional with stabilization vs. bidirectional with stabilization.

There is no clear winner between the monodirectional and the bidirectional labeling approach, as can be seen, e.g., from class 1 with up to 100 items where the bidirectional approach is faster compared to class 5 with 100 and 200 items where the monodirectional approach outperforms the bidirectional one. Focussing on instances with 25 (or 24) items only, one can see that the bidirectional approach is always superior to the monodirectional approach. This may be explained by the fact that in these instances, the bidirectional approach guarantees that there are never more than $2^{12} = 4096$ and $2^{13} = 8192$ forward and backward labels, respectively, to consider when reaching the half-way point where the merge of forward and backward labels is performed. Typically, as only a few items fit into a single bin, the number of labels is much smaller than this worst-case estimation so that combinatorial explosion is effectively suppressed. For instances with more items (≥ 50), the number of undominated labels in the bidirectional approach grows quickly. Here, the execution of the more complex dominance algorithm becomes predominant. Even using the multi-dimensional divide-and-conquer algorithm of Bentley (1980) for the dominance, which has an attractive worst-case complexity of $\mathcal{O}\left(n \log(n)^2\right)$ in the number n of labels for $D = 2$ (compared to quadratic with a straightforward pairwise dominance test), does not help here: The reduction in the number of labels through the dominance is eaten up by the time-consuming dominance algorithm. The general recommendation is therefore to use monodirectional labeling with the fast hash-table based weaker dominance Rule 1 for solving SPPRC subproblems whenever the number of items is larger.

Finally, we give some comments on the linear relaxation results that we obtained with our strongest branch-and-price variant, i.e, the one with stabilization and monodirectional labeling. For the unsolved 2-dimensional instances, there are only nine instances for which the column-generation process was terminated before solving the root node (instances `1.200.04`, `4.200.03`, `4.200.04`, `5.200.03`, `5.200.04`, `9.200.03`, `9.200.04`, `9.200.08`, and `9.200.09`; using the short notation `Class.Items.Instance`). For all of these instances, however, a Lagrangian lower bound has been computed. All lower bounds are reported on our website `http://logistik.bwl.uni-mainz.de/benchmarks.php`.

For the 20-dimensional instances, Lagrangian lower bounds are reported in Table 3 in column **LB**. Among the six unsolved instances, a lower bound is only available for the instance `4.50` (notation `Class.Items`).

3.4. Results for Vector Packing Problems with Larger Demands

We now compare the three models 01-VPP, B-VPP, and U-VPP and their solution with different branch-and-price algorithms. Table 4 summarizes the results in a very condensed way, numbers are aggregated over all instances of dimension 2 and all instances of dimension 20, respectively. The meaning of the row entries is analog to the column entries described at the beginning of this section. In addition, the rows labeled with $\bar{N}^{bb}$ give the average number of branch-and-bound nodes solved by the respective algorithm.

Table 4 shows that, for larger demands, the branch-and-price algorithms that are based on formulations B-VPP and U-VPP behave very similarly, and they clearly outperform the algorithms based on the binary formulation 01-VPP. This is not only true for the presented averages but for every single instance in the benchmark. The inferiority of the algorithms based on formulation 01-VPP can be attributed to two characteristics: First, for a larger demand, the 01-VPP model has $\sum_{i \in I} q_i$ constraints making the RMP a really huge IP with dense columns. Hence, RMP reoptimization is really time-consuming, cf. entries $\bar{T}_{LP}$ in Table 4. Second, the 01-VPP-based approaches suffer for the inherent symmetry when it comes to branching, see entries N^{bb} in the last row of Table 4. As a consequence, we discuss and present more detailed results only for our branch-and-price algorithms based on formulation B-VPP in the following.

Table 4: Results for VPP instances with larger demands.

Dimension	Stabilization		01-VPP		B-VPP		U-VPP	
			Monodir.	Bidir.	Monodir.	Bidir.	Monodir.	Bidir.
2	Stab.	opt	107	109	298	293	299	293
		$\bar{T}_{LP}$	1200.3	1234.9	394.2	674.1	393.7	677.7
		$\bar{T}$	2658.9	2654.0	1013.1	1048.6	1020.5	1046.1
		$\bar{N}^{bb}$	40.2	41.8	37.5	37.9	37.6	36.7
	Non-Stab.	opt	106	107	282	285	281	285
		$\bar{T}_{LP}$	1776.2	1658.4	402.9	674.3	401.5	674.9
		$\bar{T}$	2676.0	2668.4	1158.0	1135.9	1160.4	1136.0
		$\bar{N}^{bb}$	37.6	40.4	55.4	51.4	55.6	51.4
20	Stab.	opt	21	25	29	30	29	30
		$\bar{T}_{LP}$	1055.5	1108.6	901.9	1019.8	900.4	1021.0
		$\bar{T}$	1664.8	1664.8	1027.8	1051.1	1026.0	1043.2
		$\bar{N}^{bb}$	20.1	19.7	2.7	1.8	2.8	1.8
	Non-Stab.	opt	21	25	30	29	30	29
		$\bar{T}_{LP}$	1302.2	1293.2	925.6	1015.1	925.9	1014.9
		$\bar{T}$	1664.5	1664.6	962.0	1111.5	964.3	1111.2
		$\bar{N}^{bb}$	18.8	18.2	3.2	1.8	3.2	1.8

Comparing the four variants of branch-and-price using model B-VPP, i.e., results presented in the middle block of Table 4, we see a similar behaviour as already observed for 01-VPP: Stabilization of the column-generation process is clearly beneficial and there is no clear winner between monodirectional and bidirectional labeling for solving the MKP subproblems using the SPPRC model.

Finally, Table 5 does not provide surprising new insights. The general behaviour is similar to the 01-VPP case. Regarding the comparison of the monodirectional and bidirectional approaches, the results for the classes 4 and 5 with 25 and 50 items in dimension 2 differ from those presented in Table 2. In the B-VPP case, the bidirectional labeling is not longer superior. The reason is that in the B-MKP subproblem and the associated SPPRC digraph, the number of arcs per stage is $u_i^B + 1$. For example, this number of arcs is on average 8 for class 4 and 16 for class 5. Already for instances with 25 items, the bidirectional labeling is no longer able to effectively limit the number of labels generated up to the half-way point. The dominance algorithm based on the stronger Rule 2 becomes too slow.

In total, the four branch-and-price algorithms solve $306 + 30 = 336$ of the 400+40=440 instances with larger demands to proven optimality.

4. Conclusions

In this paper, we proposed branch-and-price algorithms for the solution of different formulations of the VPP. To the best of our knowledge, this is the first branch-and-price approach that also tackles non-binary formulations of the VPP, namely the B-VPP and U-VPP variants. To test the impact of using these formulations, we introduced new benchmark instances with demands $\gg 1$ that were generated based on the standard benchmark sets for the VPP from the literature (Caprara and Toth, 2001; Brandao and Pedroso, 2016). Moreover, the presented approach constitutes the first branch-and-price for VPPs with more than two dimensions.

Our column-generation subproblems are variants of the multidimensional knapsack problem, either binary, bounded, or unbounded for 01-VPP, B-VPP, or U-VPP, respectively. In order to obtain a generic approach, the subproblems were formulated as SPPRCs. Two different labeling algorithms (one monodirectional, one bidirectional) with different dominance rules were derived for their solution. Additional dual inequalities (not necessarily (deep) dual optimal) are added to stabilize the column-generation process. This is the first time that such dual inequalities are used in a branch-and-price for the VPP. For specific families of

Table 5: Results for B-VPP instances with larger demands

		2-dimensional								20-dimensional					
Class		Monodir./weak dom.				Bidir./strong dom.				Monodir./weak dom.			Bidir./strong dom.		
	Items	opt	$\bar{T}_{LP}$	$\bar{T}$	$\bar{N}^{bb}$	opt	$\bar{T}_{LP}$	$\bar{T}$	$\bar{N}^{bb}$	T_{LP}	T	N^{bb}	T_{LP}	T	N^{bb}
1	25	10	1.9	2.0	1.0	10	0.9	1.0	1.0	5.5	6.1	1	5.3	5.8	1
	50	10	21.5	22.9	1.0	10	4.4	5.5	1.0	23.4	24.3	1	22.6	23.4	1
	100	1	293.7	3258.2	381.4	1	140.9	3330.1	483.3	399.3	672.9	23	1773.4	2685.4	41
	200	0	1645.2	3600.0	72.8	0	1660.2	3600.0	172.5	3600.0	3600.0	0	3600.0	3600.0	0
2	25	10	0.1	1.0	1.0	10	0.1	1.0	1.0	0.2	0.5	1	0.2	0.5	1
	50	10	0.3	62.9	1.1	10	0.3	70.8	1.1	0.4	2.4	1	0.4	2.4	1
	100	10	0.6	407.0	1.6	10	0.7	419.9	1.6	0.9	0.9	1	1.0	1.0	1
	200	10	1.4	468.3	13.2	10	1.4	465.9	13.2	2.2	2.2	1	2.0	2.0	1
3	25	10	0.1	0.3	1.0	10	0.1	0.3	1.0	0.2	0.5	1	0.2	0.5	1
	50	10	0.3	0.9	1.0	10	0.3	0.9	1.0	0.5	2.6	1	0.5	2.5	1
	100	10	0.7	2.5	1.0	10	0.7	2.5	1.0	1.0	1.0	1	0.9	0.9	1
	200	10	1.4	130.1	3.1	10	1.4	128.4	3.1	1.9	1.9	1	2.0	2.0	1
4	25	10	60.1	141.8	13.6	10	103.0	104.4	1.0	376.0	379.5	1	2632.6	2946.5	4
	50	2	113.0	2913.1	246.9	2	391.7	2929.9	258.3	3600.0	3600.0	0	3600.0	3600.0	0
	100	0	883.6	3600.0	56.9	0	3600.0	3600.0	0.0	3600.0	3600.0	0	3600.0	3600.0	0
	200	0	3475.8	3600.0	1.2	0	3600.0	3600.0	0.0	3600.0	3600.0	0	3600.0	3600.0	0
5	25	7	180.7	1202.0	51.1	1	3258.0	3263.1	0.2	3600.0	3600.0	0	3600.0	3600.0	0
	50	0	172.7	3600.0	82.6	0	3600.0	3600.0	0.0	3600.0	3600.0	0	3600.0	3600.0	0
	100	0	2208.6	3600.0	12.7	0	3600.0	3600.0	0.0	3600.0	3600.0	0	3600.0	3600.0	0
	200	0	3600.0	3600.0	0.0	0	3600.0	3600.0	0.0	3600.0	3600.0	0	3600.0	3600.0	0
6	25	10	0.7	1.9	1.0	10	0.7	2.0	1.0	0.1	0.1	1	0.1	0.1	1
	50	10	11.5	13.8	1.0	10	11.7	14.2	1.0	0.3	0.3	1	0.3	0.3	1
	100	10	46.1	220.4	1.9	10	45.6	47.2	1.2	0.6	0.6	1	0.7	0.7	1
	200	9	486.4	495.5	2.4	10	120.2	135.1	3.4	1.3	1.3	1	1.3	1.3	1
7	25	10	0.4	1.3	1.0	10	0.4	1.3	1.0	0.1	0.1	1	0.1	0.1	1
	50	10	12.3	13.9	1.0	10	11.4	13.0	1.0	0.3	0.3	1	0.3	0.3	1
	100	10	21.9	22.5	1.0	10	21.9	22.6	1.0	0.7	0.7	1	0.7	0.7	1
	200	10	88.4	92.3	1.3	10	90.8	100.1	2.6	1.4	1.4	1	1.4	1.4	1
8	25	10	0.1	0.6	1.0	10	0.1	0.6	1.0	0.2	0.2	1	0.2	0.2	1
	50	10	0.2	3.3	1.0	10	0.2	3.5	1.0	0.4	0.4	1	0.4	0.4	1
	100	10	0.2	651.7	1.2	10	0.2	634.7	1.2	0.8	0.8	1	0.7	0.7	1
	200	8	0.6	1071.1	17.7	8	0.6	1065.8	17.7	1.4	1.4	1	1.5	1.5	1
9	25	10	1.7	1.8	1.0	10	0.9	1.0	1.0	0.9	0.9	1	0.8	0.9	1
	50	10	22.7	23.2	1.0	10	5.0	5.4	1.0	0.4	266.4	2	0.3	5.3	1
	100	6	358.0	1908.9	219.1	6	920.0	1795.8	128.8	112.2	3600.0	44	192.0	198.9	1
	200	0	1588.5	3600.0	76.5	0	2064.2	3600.0	134.2	3600.0	3600.0	0	3600.0	3600.0	0
10	24	10	1.6	2.6	1.0	10	1.1	2.2	1.0	0.1	2.0	1	0.1	2.0	1
	51	10	17.5	20.3	1.1	10	16.5	20.3	1.2	0.3	1.7	1	0.3	2.1	1
	99	10	20.2	21.5	1.0	10	16.3	18.0	1.2	135.3	139.7	1	149.1	153.1	1
	201	5	428.9	2144.0	223.3	5	70.8	2138.2	275.0	2606.2	3600.0	13	3600.0	3600.0	0
Total		298	394.2	1016.5	37.5	293	674.1	1048.6	37.9	901.9	1027.8	2.7	1019.8	1051.1	1.8

dual inequalities we showed that they are (deep) dual optimal inequalities for some of the formulations. The dual inequalities are used in a combined static/dynamic fashion, where the dynamic generation of violated dual inequalities is a by-product of solving the SPPRC representation of the subproblem with our labeling methods. Even more, our solution approach for the subproblem is also compatible with our branching scheme (the branching scheme of Vanderbeck, 1999, which is equal to the Ryan-Foster rule in the 01-VPP case) meaning that branching decisions can be implemented into it without deteriorating the resolution process. In order to use the dual inequalities also in the branch-and-bound tree, different rules for their validity with respect to the branching decisions taken had to be derived.

In an extensive computational study, we demonstrated that our branch-and-price algorithms are competitive with respect to the state of the art. For the 2-dimensional benchmark set, our strongest algorithm provides 368 provably optimal solutions compared to the 330 and 321 optima found by Brandao and Pedroso (2016) and Hu *et al.* (2017), respectively. Of the 20-dimensional instances, we can solve 34 instances vs. 33 optima computed by Brandao and Pedroso (2016). With all branch-and-price variants, we solve to proven optimality 404 out of the 440 standard benchmark instances. Together with the results of Brandao and Pedroso (2016) and Hu *et al.* (2017), 30 instances remain open.

Finally, our computational results indicate the usefulness of all the different algorithmic parts: (i) the stabilized algorithms are clearly superior to their non-stabilized counterparts, (ii) neither of the labeling algorithms for the subproblems completely dominates the other, and (iii) for instances with larger demands one has to resort to the non-binary formulations B-VPP and U-VPP, since the 01-VPP is too large and suffers from severe symmetry issues.

Acknowledgment

This research was funded by the Deutsche Forschungsgemeinschaft (DFG) under grant no. IR 122/6-1.

References

Alves, C. and Valério de Carvalho, J. (2008). A stabilized branch-and-price-and-cut algorithm for the multiple length cutting stock problem. *Computers & Operations Research*, **35**(4), 1315–1328.

Alves, C., Valério de Carvalho, J., Clautiaux, F., and Rietz, J. (2014). Multidimensional dual-feasible functions and fast lower bounds for the vector packing problem. *European Journal of Operational Research*, **233**(1), 43–63.

Ben Amor, H. and Valério de Carvalho, J. (2005). Cutting stock problems. In Desaulniers *et al.* (2005), chapter 5, pages 131–161.

Ben Amor, H., Desrosiers, J., and Valério de Carvalho, J. M. (2006). Dual-optimal inequalities for stabilized column generation. *Operations Research*, **54**(3), 454–463.

Bentley, J. (1980). Multidimensional divide-and-conquer. *Communications of the ACM*, **23**(4), 214–229.

Brandao, F. and Pedroso, J. P. (2016). Bin packing and related problems: general arc-flow formulation with graph compression. *Computers & Operations Research*, **69**, 56–67.

Buljubašić, M. and Vasquez, M. (2016). Consistent neighborhood search for one-dimensional bin packing and two-dimensional vector packing. *Computers & Operations Research*, **76**, 12–21.

Caprara, A. and Toth, P. (2001). Lower bounds and algorithms for the 2-dimensional vector packing problem. *Discrete Applied Mathematics*, **111**(3), 231–262.

Caprara, A., Dell'Amico, M., Díaz-Díaz, J. C., Iori, M., and Rizzi, R. (2014). Friendly bin packing instances without integer round-up property. *Mathematical Programming*, **150**(1), 5–17.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2009). *Introduction to Algorithms*. MIT Press, 3rd edition.

Delorme, M., Iori, M., and Martello, S. (2016). Bin packing and cutting stock problems: Mathematical models and exact algorithms. *European Journal of Operational Research*, **255**(1), 1–20.

Desaulniers, G., Desrosiers, J., and Solomon, M., editors (2005). *Column Generation*. Springer, New York, NY.

Dyckhoff, H. (1990). A typology of cutting and packing problems. *European Journal of Operational Research*, **44**(2), 145–159.

Gilmore, P. and Gomory, R. (1961). A linear programming approach to the cutting-stock problem. *Operations Research*, **9**, 849–859.

Gschwind, T. and Irnich, S. (2016). Dual inequalities for stabilized column generation revisited. *INFORMS Journal on Computing*, **28**(1), 175–194.

Hu, Q., Zhu, W., Qin, H., and Lim, A. (2017). A branch-and-price algorithm for the two-dimensional vector packing problem with piecewise linear cost function. *European Journal of Operational Research*, **260**(1), 70–80.

Huang, E. and Korf, R. E. (2012). Optimal rectangle packing: An absolute placement approach. *Journal of Artificial Intelligence Research*, **46**, 47–87.

Irnich, S. and Desaulniers, G. (2005). Shortest path problems with resource constraints. In Desaulniers *et al.* (2005), chapter 2, pages 33–65.

Johnson, D. S. (1973). *Near-Optimal Bin Packing Algorithms*. Ph.D. thesis, Massachusetts Institute of Technology.

Kellerer, H., Pferschy, U., and Pisinger, D. (2004). *Knapsack Problems*. Springer, Berlin.

Lübbecke, M. and Desrosiers, J. (2005). Selected topics in column generation. *Operations Research*, **53**(6), 1007–1023.

Martello, S., Pisinger, D., and Vigo, D. (2000). The three-dimensional bin packing problem. *Operations Research*, **48**(2), 256–267.

Monaci, M. and Toth, P. (2006). A Set-Covering-Based heuristic approach for Bin-Packing problems. *INFORMS Journal on Computing*, **18**(1), 71–85.

Panigrahy, R., Talwar, K., Uyeda, L., and Wieder, U. (2011). Heuristics for vector bin packing. Technical report, Microsoft Research.

Rietz, J. (2003). *Untersuchungen zu MIRUP für Vektorpackprobleme*. Dissertation, Faculty of Mathematics und Computer Science, Technischen Universität Bergakademie Freiberg, Germany.

Righini, G. and Salani, M. (2006). Symmetry helps: Bounded bi-directional dynamic programming for the elementary shortest path problem with resource constraints. *Discrete Optimization*, **3**(3), 255–273.

Ryan, D. and Foster, B. (1981). An integer programming approach to scheduling. In A. Wren, editor, *Computer Scheduling of Public Transport: Urban Passenger Vehicle and Crew Scheduling*, chapter 17, pages 269–280. Elsevier, North-Holland.

Scheithauer, G. and Terno, J. (1995). The modified integer round-up property of the one-dimensional cutting stock problem. *European Journal of Operational Research*, **84**, 562–571.

Spieksma, F. C. (1994). A branch-and-bound algorithm for the two-dimensional vector packing problem. *Computers & operations research*, **21**(1), 19–25.

Tilk, C., Rothenbächer, A.-K., Gschwind, T., and Irnich, S. (2017). Asymmetry matters: Dynamic half-way points in bidirectional labeling for solving shortest path problems with resource constraints faster. *European Journal of Operational Research*, **261**(2), 530–539.

Valério de Carvalho, J. M. (1999). Exact solution of bin-packing problems using column generation and branch-and-bound. *Annals of Operations Research*, **86**, 629–659.

Valério de Carvalho, J. M. (2005). Using extra dual cuts to accelerate column generation. *INFORMS Journal on Computing*, **17**(2), 175–182.

Vanderbeck, F. (1999). Computational study of a column generation algorithm for bin packing and cutting stock problems. *Mathematical Programming*, **86**(3), 565–594.