



## INFORMS Journal on Computing

Publication details, including instructions for authors and subscription information:  
<http://pubsonline.informs.org>

### Dual Inequalities for Stabilized Column Generation Revisited

Timo Gschwind, Stefan Irnich

To cite this article:

Timo Gschwind, Stefan Irnich (2016) Dual Inequalities for Stabilized Column Generation Revisited. INFORMS Journal on Computing 28(1):175-194. <http://dx.doi.org/10.1287/ijoc.2015.0670>

Full terms and conditions of use: <http://pubsonline.informs.org/page/terms-and-conditions>

This article may be used only for the purposes of research, teaching, and/or private study. Commercial use or systematic downloading (by robots or other automatic processes) is prohibited without explicit Publisher approval, unless otherwise noted. For more information, contact [permissions@informs.org](mailto:permissions@informs.org).

The Publisher does not warrant or guarantee the article's accuracy, completeness, merchantability, fitness for a particular purpose, or non-infringement. Descriptions of, or references to, products or publications, or inclusion of an advertisement in this article, neither constitutes nor implies a guarantee, endorsement, or support of claims made of that product, publication, or service.

Copyright © 2016, INFORMS

Please scroll down for article—it is on subsequent pages



INFORMS is the largest professional society in the world for professionals in the fields of operations research, management science, and analytics.

For more information on INFORMS, its publications, membership, or meetings visit <http://www.informs.org>

# Dual Inequalities for Stabilized Column Generation Revisited

Timo Gschwind, Stefan Irnich

Gutenberg School of Management and Economics, Johannes Gutenberg University Mainz, D-55128 Mainz, Germany  
{gschwind@uni-mainz.de, irnich@uni-mainz.de}

Column generation (CG) models have several advantages over compact formulations: they provide better linear program bounds, may eliminate symmetry, and can hide nonlinearities in their subproblems. However, users also encounter drawbacks in the form of slow convergence, also known as the tailing-off effect, and the oscillation of the dual variables. Among different alternatives for stabilizing the CG process, Ben Amor et al. [Ben Amor H, Desrosiers J, Valério de Carvalho JM (2006) Dual-optimal inequalities for stabilized column generation. *Oper. Res.* 54(3):454–463] suggest the use of dual-optimal inequalities (DOIs) in the context of cutting stock and bin packing problems. We generalize their results, provide new classes of (deep) DOIs, and show the applicability to other problems (vector packing, vertex coloring, bin packing with conflicts). We also suggest the dynamic addition of violated dual inequalities in a cutting-plane fashion and the use of dual inequalities that are not necessarily (deep) DOIs. In the latter case, a recovery procedure is needed to restore primal feasibility. Computational results proving the usefulness of the methods are presented.

*Keywords:* integer programming; column generation; stabilization; dual inequalities

*History:* Accepted by Karen Aardal, Area Editor for Design and Analysis of Algorithms; received November 2014; revised May 2015, July 2015; accepted July 2015. Published online February 5, 2016.

## 1. Introduction

Column generation (CG) has been proven a versatile tool for solving large-scale linear programs (LPs), often with more variables than explicitly representable. Such huge models may arise naturally or may result from a reformulation of an integer compact model using a Dantzig-Wolfe decomposition (Dantzig and Wolfe 1960), which then leads to an integer CG approach also known as branch-and-price (Barnhart et al. 1998, Lübbecke and Desrosiers 2005). The resulting extensive formulation (cf. Lübbecke and Desrosiers 2005, p. 1009) has an enormous number of variables corresponding to extreme points and extreme rays of the domain chosen as the subproblem in the decomposition. Although the huge number of variables in the extensive formulation lets the model seem unattractive, it also has some profound advantages. The extensive formulation has a tighter LP-bound if the subproblem does not possess the integrality property. Moreover, if nonlinearities are present in the compact formulation, they may be well managed using the right definition of the subproblems. Additionally, some very successful CG-based algorithms have been invented for applications in, e.g., vehicle and crew routing and scheduling (Desaulniers et al. 1998), cutting and packing (Ben Amor and Valério de Carvalho 2005), and graph coloring (Mehrotra and Trick 1996). Their success can be attributed to the availability of

very powerful subproblem solution algorithms that have reached a high degree of maturity; see Irnich and Desaulniers (2005) for constrained shortest path problems, Kellerer et al. (2004) for knapsack problems, and Östergård (2002) and Carraghan and Pardalos (1990) for cliques/stable sets.

However, CG approaches in practice often suffer from instability difficulties. The dual variables, which drive the generation process, may oscillate heavily before they finally converge to some optimal dual values. On the primal side, extensive formulations tend to be degenerate: In many applications the master program is some (generalized) set-partitioning or set-covering problem, in which each variable represents a crew schedule, vehicle route, packing, or partition, which is often a dense column, so a few columns comprise a solution. However, a primal basis must include several other variables that are then at value zero. As a consequence, primal degeneracy leads to rather small and often nonimproving LP pivots, known as the tailing-off effect (Gilmore and Gomory 1961, Vanderbeck 2005). The process of variable generation then continues over many iterations to produce nearly no improvement in the LP objective. To explicitly stabilize the dual values, algorithmic techniques like the box-step method (Marsten et al. 1975), bundle methods (Hiriart-Urruty and Lemaréchal 1993), and tailored CG stabilization approaches have been

proposed, e.g., by du Merle et al. (1999), Rousseau et al. (2007), and Lee and Park (2011). Furthermore, a branch of the recent literature is especially devoted to tools that can help overcome or even benefit from primal degeneracy when solving huge LPs (Gauthier et al. 2015, Desrosiers et al. 2014).

In this paper, we continue the path originally followed by Valério de Carvalho (2005) and Ben Amor et al. (2006). For the *cutting stock problem* (CS) and the *bin packing problem* (BP), they have shown that the knowledge of the domain of optimal dual values can be used to stabilize the CG process. In particular, any valid inequality known for the optimal dual space can be added as an additional variable to the corresponding primal formulation.

We introduce the concept of dual inequalities in a more formal way now, following the notation of Ben Amor et al. (2006). Consider a pair of primal and dual LPs of the forms

$$\begin{aligned}
 (P) \quad & z_P = \min \quad c^\top \lambda \\
 & \text{s.t.} \quad A\lambda = b \\
 & \quad \quad \lambda \geq 0 \\
 (D) \quad & z_D = \max \quad b^\top \pi \\
 & \text{s.t.} \quad A^\top \pi \leq c
 \end{aligned}$$

with a matrix  $A = (a_{ij}) \in \mathbb{R}^{I \times J}$ ,  $c = (c_j) \in \mathbb{R}^J$ , and  $b = (b_i) \in \mathbb{R}^I$  with row indices  $i \in I$  and column indices  $j \in J$ . Let  $m = |I|$  denote the number of rows. We refer to the  $j$ th column of  $A$  as  $a_j \in \mathbb{R}^I$  such that  $A = (a_j)_{j \in J}$ . For any vector  $a \in \mathbb{R}^I$ , we write  $a \in A$  if and only if  $a = a_j$  for some  $j \in J$ . Rows of  $A$  are denoted by  $a_{i*} \in \mathbb{R}^J$  for  $i \in I$ .

We always assume that the primal formulation  $P$  is the extensive formulation to which a CG algorithm is applied. Therefore, we do not solve  $P$  directly, but work with a restricted version of  $P$ , the *restricted master program* (RMP), for which reoptimization of the LP and the solution of the pricing problem (or subproblem) alternate. The pricing problem asks for the determination of at least one column  $a_j \in A$ , for which  $c_j - \pi^\top a_j < 0$  holds, or the guarantee that no such column exists. In the latter case,  $P$  has been solved to optimality and the CG process terminates.

The idea of Ben Amor et al. (2006) is to add additional constraints  $E^\top \pi \leq e$  to the dual  $D$ . Additional constraints in the dual  $\tilde{D}$  correspond with additional variables in the primal  $\tilde{P}$ , denoted by  $y$  in the following. The extended primal and dual models are

$$\begin{aligned}
 (\tilde{P}) \quad & z_{\tilde{P}} = \min \quad \{c^\top \lambda + e^\top y\} \\
 & \text{s.t.} \quad A\lambda + Ey = b \\
 & \quad \quad \lambda \geq 0, y \geq 0 \\
 (\tilde{D}) \quad & z_{\tilde{D}} = \max \quad b^\top \pi \\
 & \text{s.t.} \quad A^\top \pi \leq c \\
 & \quad \quad E^\top \pi \leq e.
 \end{aligned}$$

The set of additional *dual inequalities* (DIs)  $E^\top \pi \leq e$  cuts part of the dual solution space. Thus,  $\tilde{D}$  is a restriction of  $D$ , whereas the corresponding primal

**Table 1** Some Acronyms Used in the Paper

Acronym	Full name
CG	Column generation
RMP	Restricted master program
DI	Dual inequality; any inequality in the variables of $D$
DOI	Dual-optimal inequality; fulfilled by every dual-optimal solution of $D$
DDOI	(Set of) deep dual-optimal inequality/ies
WSI	Weighted subset inequality
SI	Subset inequality
CS	Cutting stock problem
BP	Bin packing problem
BPC	Bin packing problem with conflicts
VP	Vector packing problem
VC	Vertex coloring problem
KP	Binary knapsack problem
UKP	Unbounded knapsack problem
KPC	Binary knapsack problem with conflicts
DKP	Unbounded version of the $d$ -dimensional knapsack problem
MWIS	Maximum weight independent set

model  $\tilde{P}$  is a relaxation of  $P$ . We denote by  $D^*$  the set of optimal solutions to the model  $D$ , i.e., the dual-optimal space. Throughout the paper, we heavily use acronyms; for convenience, Table 1 lists those that are most frequently used.

One can further differentiate two special classes of DIs depending on which part of the dual solution space they cut off (see Ben Amor et al. 2006, p. 455): DIs  $E^\top \pi \leq e$  are called *dual-optimal inequalities* (DOIs) if all dual-optimal solutions  $\pi^* \in D^*$  also satisfy the DIs, i.e.,  $D^* \subseteq \{\pi: E^\top \pi \leq e\}$ . If DIs  $E^\top \pi \leq e$  are satisfied by at least one dual-optimal solution  $\pi^* \in D^*$ , i.e.,  $D^* \cap \{E^\top \pi \leq e\} \neq \emptyset$ , they are called a set of *deep dual-optimal inequalities* (DDOIs). Note that deep dual-optimal is a property referring to a set of DIs, meaning that DDOIs cannot be tested individually for each inequality in  $E^\top \pi \leq e$ . In contrast, for a set of DOIs, every single DI  $E_{j*} \pi \leq e_j$ , where  $E_{j*}$  refers to the  $j$ th row of  $E$ , must qualify as a DOI. This is fulfilled if all dual-optimal solutions  $\pi^* \in D^*$  respect  $E_{j*} \pi^* \leq e_j$ .

Our paper contributes at least five new findings: First, we generalize Propositions 1 and 2 of Ben Amor et al. (2006) and Proposition 1 of Valério de Carvalho (2005), providing insights into the relations of optimal solution values and optimal solutions to models  $P$ ,  $D$ ,  $\tilde{P}$ , and  $\tilde{D}$  when  $E^\top \pi \leq e$  are DOIs or DDOIs. Second, we define several new classes of properties for integer and binary valued matrices  $A$  and unit costs  $c = 1$  that enable the identification of new DOIs and DDOIs. For example, the coefficient matrices of some packing and coloring problems having a set covering or capacitated covering formulation fulfill these matrix properties. Herewith, we derive additional classes of DDOIs for BP and DOIs for CS. The latter can be extended also to the *vector packing problem* (VP), which

can be seen as the multidimensional generalization of CS.

The third aspect is the extension of the DI-based stabilization to several new classes of problems: We address the *vertex coloring problem* (VC), in which the task is to color the vertices of a given undirected graph with a minimum number of colors so that adjacent vertices receive different colors. The synthesis of VC and BP is the *bin packing problem with conflicts* (BPC). In all cases, the derivation of DOIs and DDOIs is based on showing that the above mentioned new matrix properties are valid.

Fourth, we show that a kind of over-stabilization of CG procedures can result in an overall faster convergence. There exist cases in which one can suspect that  $E^\top \pi \leq e$  are DDOIs, but this may not be true. The property may be fulfilled just for some instances of the problem or just for a proper subset that one is unable to identify. In the negative case, the outcome of the stabilized CG algorithm is an infeasible primal solution and a weaker bound than  $z_P$  (see Proposition 1). We show that there is a constructive way of identifying the DIs that have cut off  $D^*$ . This can be seen as a *recovery procedure*. To accelerate the overall CG approach, we therefore propose to alternate between solving the (over-)stabilized formulation  $\tilde{P}$  and the recovery procedure until a feasible primal solution results.

Fifth and finally, we show that the dynamic generation of DIs during the CG process is another option that often helps reduce the computation time. Note that up to now, DOIs and DDOIs have been used in a static fashion. Indeed, when the number of known DOIs or DDOIs is relatively small, e.g., polynomial in the size of the input instance, then  $\tilde{P}$  is not significantly larger than  $P$ , and the advantage of stability can then outreach the resulting larger RMP reoptimization times. Such a family of DOIs has been used in a branch-and-price approach for the capacitated arc-routing problem (Bode and Irnich 2012). However, in all problems considered here, the families of DIs, DOIs, and DDOIs are exponential in size. For the problems CS, BP, VC, and BPC we demonstrate that often the dynamic generation, sometimes in combination with an a priori addition of the expectedly strongest DIs provides the best trade-off between stabilization, effort of DI generation, and effort for reoptimizing the RMP.

The remainder of this paper is organized as follows: §2 presents the generalized proposition on the relations of solutions of models  $P$ ,  $D$ ,  $\tilde{P}$ , and  $\tilde{D}$ . In §3, we derive properties of the coefficient matrix that, if fulfilled, allow the identification of DIs that are DOIs and/or DDOIs. Along with introducing the new properties, we present their application to different problems. This includes the proper introduction of

the extensive formulations of these problems and the specification of the different new classes of DOIs and DDOIs. Section 4 clarifies the dynamic identification of violated DIs and presents the recovery procedure. In §5, we present aggregated computational results for BP, CS, VC, and BPC. Final conclusions are drawn in §6.

## 2. Equivalence Conditions for the Original and Extended Models

The central question of this section is the following: Under which conditions are the models  $P$  and  $\tilde{P}$  and  $D$  and  $\tilde{D}$  equivalent? Equivalence of the respective models means that they provide identical LP-bounds. Moreover, optimal solutions to the extended models  $\tilde{P}$  and  $\tilde{D}$  can be transformed into optimal solutions of the corresponding original models  $P$  and  $D$ , respectively. In case of equivalence, the stabilized model  $\tilde{P}$  can be solved by CG instead of the nonstabilized model  $P$ . Conditions for the equivalence depend on the constraints  $E^\top \pi \leq e$ .

Proposition 1 generalizes the findings of Ben Amor et al. (2006), including some results of Valério de Carvalho (2005). In essence, if the  $E^\top \pi \leq e$  are DDOIs, then the models  $P, D, \tilde{P}$ , and  $\tilde{D}$  are equivalent, and vice versa.

**PROPOSITION 1.** *The following statements are equivalent:*

- (i)  $E^\top \pi \leq e$  are DDOIs.
- (ii) There exists a  $\pi^* \in D^*$  that is feasible also for  $\tilde{D}$ .
- (iii)  $z_D = z_{\tilde{D}}$ .
- (iv)  $z_P = z_{\tilde{P}}$ .
- (v) For every feasible primal solution  $(\tilde{\lambda}, y)$  to  $\tilde{P}$  there exists a primal feasible solution  $\lambda$  to  $P$  with  $c^\top \lambda \leq c^\top \tilde{\lambda} + e^\top y$ .
- (vi) There exists a primal optimal solution  $(\tilde{\lambda}^*, y^*)$  to  $\tilde{P}$  with  $Ey^* = 0$ . Also,  $\tilde{\lambda}^*$  is an optimal solution to  $P$ .
- (vii) Every optimal dual solution  $\pi^*$  to  $\tilde{D}$  is optimal for  $D$ .

The proof of Proposition 1 and proofs for all other propositions and theorems can be found in SEC.1 of the online supplement (available as supplemental material at <http://dx.doi.org/10.1287/ijoc.2015.0670>).

Ben Amor et al. (2006) have shown the implications (i)  $\Rightarrow$  (iii), (i)  $\Rightarrow$  (iv), and (i)  $\Rightarrow$  (vii). Moreover, for DOIs they showed that if there exists a primal optimal solution  $(\tilde{\lambda}^*, y^*)$  to  $\tilde{P}$  with  $Ey^* = 0$ , then  $\tilde{\lambda}^*$  is optimal for  $P$ . Since DOIs are also DDOIs, Proposition 1 proves the fact that if  $E^\top \pi \leq e$  are DOIs, then this implies (i)–(vii). The reverse does not hold in general.

From a practical point of view, the condition (v) is particularly useful for problems for which a constructive procedure is known to transform solutions  $(\tilde{\lambda}, y)$  of  $\tilde{P}$  so they lead to a solution  $\lambda$  of  $P$  with less or equal cost. A solution to model  $P$  can then

be determined by first solving the relaxed model  $\tilde{P}$  to optimality and then converting this solution into a solution to  $P$ . This approach has been taken by Valério de Carvalho (2005) when solving CS with CG: He first solves the extended master program  $\tilde{P}$ , in which several additional  $y$  variables corresponding to DIs have been added a priori. As a result, the optimal master program may contain positive  $y$  variables. This solution is then transformed into a feasible CS solution with identical cost (our recovery procedure presented in §4.2 is inspired by this transformation). Thus, Valério de Carvalho (2005) was the first to show that the original CS problem is actually solved to optimality, even if additional  $y$  variables are positive. In fact, the DIs that he used for CS are DOIs, as later shown by Ben Amor et al. (2006).

Even if no direct transformation from solutions  $(\tilde{\lambda}, y)$  of  $\tilde{P}$  into solutions  $\lambda$  of  $P$  is known, the model  $P$  can still be solved by solving a stabilized variant of the model  $\tilde{P}$ . Ben Amor et al. (2006) show that for DOIs it suffices to marginally increase the costs  $e$  of the  $y$  variables to  $e + \varepsilon$ . As a result, all DOIs are inactive and the corresponding primal variables  $y$  must be at 0 (resulting from complementary slackness). Moreover, Ben Amor et al. (2006) suggest a two-phase procedure for DDOIs: In the first phase, they determine an optimal solution to  $\tilde{P}$ . It has a corresponding dual solution  $\pi^*$ . In the second phase, they use a stabilized model  $P$ , in which the dual variables  $\pi$  are stabilized with the help of a trust region around  $\pi^*$ . The primal solution  $\lambda^*$  to this stabilized model is then a feasible solution to  $P$  (see Ben Amor et al. 2006, Prop. 5).

### 3. Matrix Properties for Deriving DOIs and DDOIs

The definition of DOIs and DDOIs refers to the set of dual-optimal solutions  $D^*$ . Up to now, DOIs and DDOIs have been derived directly by analyzing structural properties of the problem at hand, i.e., for CS and BP. In this section, we derive some new results by analyzing the structure of the constraint matrix  $A$  defining the model  $P$ . If these properties are fulfilled, some generic classes of DIs are automatically known to be DOIs or DDOIs. Testing these matrix properties simplifies the identification of DOIs or DDOIs because it makes clear which structural properties of the problem at hand need to be tested. We validate these matrix properties for new and different problems showing that several classes of DIs are DOIs or DDOIs.

Some additional notation is needed: Recall that  $I$  represents the row indices and  $J$  represents the column indices of the coefficient matrix  $A$ . We denote the  $i$ th unit vector for  $i \in I$  by  $u_i \in \mathbb{Z}_+^I$ . Moreover,  $\mathbf{0}$

and  $\mathbf{1}$  are vectors with all entries 0 and 1, respectively, of appropriate size. In the following, we consider an integer valued matrix  $A \in \mathbb{Z}_+^{I \times J}$ , an integer valued, strictly positive right-hand side  $b \in \mathbb{Z}_{>0}^I$ , and unit costs  $c = \mathbf{1} \in \mathbb{R}^J$ .

#### 3.1. Exchange Property

Several results presented in the following use the exchange property.

**DEFINITION 1 (EXCHANGE PROPERTY).** Let  $r, t \in \mathbb{Z}_+^I$  be given. A matrix  $A \in \mathbb{Z}_+^{I \times J}$  has the  $(r, t)$ -exchange property if  $a_j \in A, a_j \geq r$  implies  $a_j - r + t \in A$ .

If one or both vectors  $r$  and  $t$  are unit vectors, e.g.,  $r = u_h$ , we simplify the notation and write  $(h, t)$ -exchange property. If one of the vectors is the null vector, e.g.,  $t = \mathbf{0}$ , we write  $(r, 0)$ -exchange property.

The exchange property means that the columns in  $A$  are related to each other. Any column that is not too small, i.e., that fulfills  $a_j \geq r$ , can be converted into another column by exchanging entries in some rows against entries in other rows (described by  $t - r$ ). In this case, useful relations between optimal values of primal and dual variables can be derived.

**PROPOSITION 2.** We assume a primal model  $P$  with integer valued matrix  $A \in \mathbb{Z}_+^{I \times J}$  and unit costs  $c = \mathbf{1} \in \mathbb{R}^J$ . Let  $r = (r_i), t = (t_i) \in \mathbb{Z}_+^I$  be given. If  $A$  has the  $(r, t)$ -exchange property and  $(\lambda^*, \pi^*)$  is a pair of optimal solutions to  $P$  and  $D$ , then

$$\text{(Exch-D)} \quad \sum_{i \in I} r_i \pi_i^* \geq \sum_{i \in I} t_i \pi_i^* \quad \text{or}$$

$$\text{(Exch-P)} \quad \lambda_k^* = 0 \quad \text{for all } k \in J \text{ with } a_k \geq r.$$

If  $r$  is a unit vector  $u_h$  for  $h \in I$ , the exchange property allows that an entry in row  $h$  is replaced by entries in a subset of other rows given by the positive entries in  $t$ . The following proposition shows that valid DOIs result for the dual variables described by  $h$  and the positive components of  $t$ .

**PROPOSITION 3.** We assume a primal model  $P$  with integer valued matrix  $A \in \mathbb{Z}_+^{I \times J}$ , an integer valued, strictly positive right-hand side  $b \in \mathbb{Z}_{>0}^I$ , and unit costs  $c = \mathbf{1} \in \mathbb{R}^J$ . Let  $h \in I$  and  $t \in \mathbb{Z}_+^I$  be given. If  $A$  has the  $(h, t)$ -exchange property, then any dual-optimal solution  $\pi^* \in D^*$  fulfills

$$\pi_h^* \geq \sum_{i \in I} t_i \pi_i^*.$$

Proposition 3 can directly be applied to identify DOIs for problems whose coefficient matrix  $A$  has the  $(h, t)$ -exchange property. We exemplify this for CS and VP.

*Cutting Stock.* The CS consists of finding cutting patterns for cutting rolls of length  $L$  into items  $i \in I$  of length  $w_i \leq L$  such that the total number of rolls is minimal and demands  $b_i$  of the items  $i \in I$  are fulfilled. A feasible cutting pattern cuts a roll into several of the items with  $\sum_{i \in I} a_{ij} w_i \leq L$ , where  $a_{ij}$  denotes the number of items of length  $w_i$  that are produced by pattern  $j \in J$ . An extensive formulation for CS was first presented by Gilmore and Gomory (1961), and several CG-based algorithms have been presented over the years, e.g., by Valério de Carvalho (1998), Vanderbeck (2000), Ben Amor and Valério de Carvalho (2005). In our notation, model  $P$  for CS consists of columns  $a_j \in \mathbb{Z}_+^I$  forming  $A$ , one for each feasible cutting pattern. The variable  $\lambda_j$  gives the number of rolls that are cut according to pattern  $j$  (cf. Gilmore and Gomory 1963). The pricing problem has to identify a cutting pattern with negative reduced cost. This is an *unbounded knapsack problem* (UKP), which can be solved by dynamic programming algorithms as a *longest path problem* in an acyclic digraph (see Kellerer et al. 2004). We provide some more details in §4.1.

*Vector Packing.* The VP is the  $d$ -dimensional ( $d \geq 2$ ) generalization of CS. CG algorithms for VP have been presented by Caprara and Toth (2001) and Alves et al. (2014). In VP, items  $i \in I$  have a size  $w_{i1}, w_{i2}, \dots, w_{id}$  in each of the  $d$  dimensions (e.g., weight, volume, cost, etc.), and *cutting patterns* or *packings* are restricted not only by a single capacity, but by  $d$  capacities  $L_1, L_2, \dots, L_d$ . A straightforward extensive model for VP is analog to the one for CS: Columns in the master program correspond to feasible packings, whereas the subproblem is the *unbounded* version of a *d-dimensional knapsack problem* (DKP) (see Kellerer et al. 2004, Chap. 9).

A direct consequence of Proposition 3 is shown in Theorem 1.

**THEOREM 1.** *Let  $d = 1$  or  $d \geq 2$ . Moreover, let  $w_i, i \in I$  (for  $d = 1$ ) or  $w_{ip}, i \in I, p \in \{1, 2, \dots, d\}$  (for  $d \geq 2$ ) be the  $d$ -dimensional weights defining CS and VP, and let  $h \in I, S \subseteq I \setminus \{h\}$ , and  $t \in \mathbb{Z}_{>0}^S$  be given. We distinguish between  $d = 1$  and  $d \geq 2$ :*

- (i) *If  $w_h \geq \sum_{s \in S} t_s w_s$ , then the inequality  $\pi_h \geq \sum_{s \in S} t_s \pi_s$  is a DOI for CS.*
- (ii) *If  $w_{hp} \geq \sum_{s \in S} t_s w_{sp}$  for all  $p = 1, \dots, d$ , then the inequality  $\pi_h \geq \sum_{s \in S} t_s \pi_s$  is a DOI for VP.*

In the following, we refer to DIs of the form  $\pi_h \geq \sum_{s \in S} t_s \pi_s$  as *weighted subset inequalities* (WSIs). In the primal model  $\tilde{P}$ , the corresponding  $k$ th column to a WSI is  $(E_{ik})_{i \in I} \in \mathbb{Z}^I$  with

$$(E_{ik}) = \begin{cases} t_i & i \in S, \\ -1 & i = h, \\ 0 & \text{otherwise,} \end{cases} \quad (1)$$

and a cost of zero; i.e.,  $e_k = 0$ . We refer to such a WSI and its (so-called) WSI column as  $(h \leftarrow t, S)$  with  $h \in I, S \subseteq I \setminus \{h\}$ , and  $t \in \mathbb{Z}_{>0}^S$  instead of using the column index  $k$ . Moreover, for  $t = \mathbf{1} \in \mathbb{Z}_{>0}^S$ , the DI  $\pi_h \geq \sum_{s \in S} \pi_s$  is a *subset inequality* (SI), and we refer to it and its column as  $(h \leftarrow S)$ .

The WSIs and the associated primal WSI columns have a very intuitive practical interpretation in CS: The quantities  $t_s$  for  $s \in S$  describe a combination of items including copies whenever  $t_s > 1$ . Whenever the total length of the combination is not longer than the length  $w_h$  of item  $h$ , then it is more difficult to include item  $h$  in a cutting pattern than all items (and their copies) of the combination. Hence, the marginal cost  $\pi_h$  of producing item  $h$  should be not smaller than the overall marginal cost of producing the combination. It also means that in any cutting pattern that includes item  $h$  one can safely replace this item  $h$  by the combination, i.e., all items  $s \in S$  in quantities given by  $t_s, s \in S$ . The result is a different, but certainly feasible cutting pattern.

When  $\tilde{P}$  is solved by CG, the presence of a WSI  $(h \leftarrow t, S)$  in the RMP implicitly represents several other cutting pattern columns. For any cutting pattern column  $a_j$  including item  $h$ , i.e.,  $a_j \geq u_h$ , the sum of column  $a_j$  and a WSI column  $(h \leftarrow t, S)$  realizes the cutting pattern (1) in which items  $s \in S$  are cut an additional  $t_s$  times and item  $h$  is cut one time fewer than given by  $a_{hj}$ . Thus, the presence of WSI columns in the RMP prevents having to explicitly generate specific columns because they are already implicitly represented. Moreover, these implicitly represented cutting patterns and other WSIs together represent additional cutting pattern columns.

Note that the WSIs for CS in Theorem 1 are generalizations of the SIs introduced by Valério de Carvalho (2005) and proven to be DOIs by Ben Amor et al. (2006). Both works exclusively consider DOIs of the form  $\pi_h \geq \sum_{s \in S} \pi_s$ ; hence they neglect the possibility of replacing a single item by multiples of one or more items  $s \in S$ .

### 3.2. Covering Property

It seems to be common knowledge or folklore that (generalized) covering formulations are preferable to partitioning formulations, i.e., inequality over equality constraints. In the light of Proposition 3 we can justify this as follows: If the matrix  $A$  has the  $(h, 0)$ -exchange property, it means that matrix entries in the row  $h$  can be decreased as long as a column  $a_j, j \in J$  has a positive entry  $a_{hj}$ . Intuitively, the  $h$ th equality of  $A\lambda = b$  can then be replaced by a covering constraint. The following proposition formalizes the idea from a primal and dual perspective:

**PROPOSITION 4.** *Let a matrix  $A \in \mathbb{Z}_+^{I \times J}$  have the  $(h, 0)$ -exchange property for every  $h \in I$ . Then:*

- (i) For every primal solution  $\lambda$  to the system  $A\lambda \geq b$ ,  $\lambda \geq 0$  there exists another primal solution  $\lambda'$  to  $A\lambda' = b$ ,  $\lambda' \geq 0$  with  $\mathbf{1}^\top \lambda' = \mathbf{1}^\top \lambda$ .
- (ii) Any dual-optimal solution  $\pi^* \in D^*$  fulfills  $\pi_h^* \geq 0$ .

This result can be used to explain why some (generalized) partitioning problems are equivalent to their covering versions, which is a well-known fact for many problems. A prerequisite is that the coefficient matrix  $A$  of the problem has the  $(h, 0)$ -exchange property for every  $h \in I$ . Practically speaking, a subset of a feasible structure is again a feasible structure of the same type with no greater cost (note the unit costs assumption made at the beginning of §3). The feasibility of subsets is known as the concept of *heredity* (see, e.g., Pattillo et al. 2013). For CS, e.g., removing an item from a cutting pattern clearly results in another feasible cutting pattern, unless it becomes empty.

Note that for solving these equivalent formulations by CG the covering formulation (with  $A\lambda \geq b$ ) is significantly more stable than the one with equality ( $A\lambda = b$ ). In the latter, the feasible dual space is in  $\mathbb{R}^I$ , whereas feasible dual solutions for covering come from the positive quadrant  $\mathbb{R}_+^I$ . Hence, the dual space is reduced by a factor of  $2^m$ .

Hereditary problems are VP and BP. Additional examples are VC, which is equivalent to partitioning with independent sets and partitioning with cliques in the complement graph, the *edge coloring problem*, and partitioning with matchings. In the context of truck routing, most tour minimization problems are hereditary. In all these problems (some are considered and formally introduced later in this article), subsets of packings, independent sets, cliques, and matchings are clearly feasible subsets and, hence, partitioning and covering are equivalent formulations.

For other problems, however, covering and partitioning differ: Covering and partitioning a graph with certain types of subgraphs are different problems if the subgraphs are, e.g., cycles or  $k$ -clubs. Here the subgraph formed by a subset of the vertices of a cycle or a  $k$ -club is generally not a cycle or  $k$ -club (see Pattillo et al. 2013). The same is true for routing problems with a maximal waiting time constraint, since a sub-tour of a feasible tour may violate the waiting time constraint.

### 3.3. Row Interchange Properties

We now consider LPs with only binary coefficients so that  $A \in \mathbb{B}^{I \times J}$  and  $b = \mathbf{1}$  holds. Moreover, the primal model is assumed to have unit costs  $c = \mathbf{1}$ . For these we define a modified version of the exchange property:

**DEFINITION 2 (ROW REPLACEMENT PROPERTY).** Let  $h, i \in I$  be given. A binary matrix  $A \in \mathbb{B}^{I \times J}$  has the  $(h, i)$ -row replacement property if  $a_j \in A$ ,  $a_j \geq u_h$  and  $a_j = 0$

implies  $a_j - u_h + u_i \in A$ . In this case, the pair  $(h, i) \in I \times I$  is called a *valid replacement* for  $A$ .

The analog to Proposition 3 for LPs  $P$  whose coefficient matrix  $A$  satisfies the row replacement property is:

**PROPOSITION 5.** We assume a primal model  $P$  with binary matrix  $A \in \mathbb{B}^{I \times J}$ , right-hand side  $b = \mathbf{1} \in \mathbb{R}^I$ , and unit costs  $c = \mathbf{1} \in \mathbb{R}^J$ . Let  $E^\top \pi \leq e$  be the set of all inequalities  $\pi_i - \pi_h \leq 0$  for valid replacements  $(h, i)$  for  $A \in \mathbb{B}^{I \times J}$ . Then  $E^\top \pi \leq e$  are DDOIs.

Using Proposition 5, we can show that specific classes of DIs for BP, VC, and BPC are DDOIs. We briefly introduce these problems now.

**Bin Packing.** The BP is a restricted CS in which all items  $i \in I$  have unit demand  $b_i = 1$ . Consequently, an item cannot appear more than once in a feasible cutting pattern, called *bin* in BP. Thus,  $A$  is a binary matrix ( $A \in \mathbb{B}^{I \times J}$ ) and the CG pricing problem is a *binary knapsack problem* (KP) (see Kellerer et al. 2004). CG algorithms for BP have been suggested by Gilmore and Gomory (1963), Vance et al. (1994), and Valério de Carvalho (1999). Clearly, for BP,  $(h, i)$  is a valid replacement for  $A$  if  $w_h \geq w_i$ .

**Vertex Coloring.** The VC is defined for an undirected graph  $G = (I, \mathcal{E})$  with vertex set  $I$  and edge set  $\mathcal{E}$ . VC is the problem of feasibly coloring the vertices with a minimum number of colors such that any two adjacent vertices receive different colors. We denote by  $N(i)$  the set of vertices adjacent to vertex  $i \in I$ . An *independent set* in  $G$  is a subset  $S \subseteq I$  such that no two vertices of  $S$  are adjacent. Clearly, in VC all vertices of the same color form an independent set. Extensive models for VC were analyzed, e.g., in Mehrotra and Trick (1996), Malaguti et al. (2011), Gualandi and Malucelli (2012), and Held et al. (2012), and they are defined as follows: The columns  $a_j \in \mathbb{B}^I$  of  $A$  are incidence vectors of independent sets, and the task is to properly partition  $I$  into independent sets; i.e., the right-hand side is  $b = \mathbf{1}$ . The pricing problem in this formulation consist of finding a *maximum-weight independent set* (MWIS).

**Bin Packing with Conflicts.** The BPC is the synthesis of BP and VC: Items  $i \in I$  with unit demand  $b_i = 1$  and weights  $w_i$  have to be packed into a minimum number of bins each with a capacity of  $L$ . Moreover, a conflict graph  $G = (I, \mathcal{E})$  with vertex set  $I$  and edge set  $\mathcal{E}$  is given, where an edge  $\{i, j\}$  indicates that the items  $i$  and  $j$  are in conflict. Conflicting items cannot be packed into the same bin. An extensive formulation for BPC is analog to the formulations for BP or VC and has been presented by Fernandes Muritiba et al. (2010) and Sadykov and Vanderbeck (2013). Columns can equivalently be seen as *conflict-free bins* or *capacity constrained-independent sets*.

We refer to a DI of the form  $\pi_h \geq \pi_i$  as *pair inequality*. Clearly, the pair inequalities are a special case of the SIs.

**THEOREM 2.** (i) *The pair inequalities  $\{\pi_h \geq \pi_i: w_h \geq w_i\}$  are DDOIs for BP. The pair inequality  $\pi_h \geq \pi_i$  is a DOI for BP, if in addition  $w_h + w_i > L$  holds. The SI  $\pi_h \geq \sum_{s \in S} \pi_s$  is a DOI for BP, if  $w_h \geq \sum_{s \in S} w_s$  and  $w_h + \min_{s \in S} w_s > L$ .*

(ii) *The pair inequalities  $\{\pi_h \geq \pi_i: N(h) \cup \{h\} \supseteq N(i)\}$  are DDOIs for VC. The pair inequality  $\pi_h \geq \pi_i$  is a DOI for VC, if in addition  $h \in N(i)$  holds. The SI  $\pi_h \geq \sum_{s \in S} \pi_s$  is a DOI for VC, if  $N(h) \cup \{h\} \supseteq N(s)$  and  $h \in N(s)$  for all  $s \in S$ , and  $S$  is an independent set.*

(iii) *The pair inequalities  $\{\pi_h \geq \pi_i: w_h \geq w_i, N(h) \cup \{h\} \supseteq N(i)\}$  are DDOIs for BPC. The pair inequality  $\pi_h \geq \pi_i$  is a DOI for BPC, if in addition  $w_h + w_i > L$  or  $h \in N(i)$  holds. The SI  $\pi_h \geq \sum_{s \in S} \pi_s$  is a DOI for BPC, if  $w_h \geq \sum_{s \in S} w_s$ ,  $N(h) \cup \{h\} \supseteq N(s)$  for all  $s \in S$ ,  $S$  is an independent set, and  $w_h + \min_{s \in S} w_s > L$  or  $h \in N(s)$ ,  $s \in S$  holds.*

These pair inequalities of Theorem 2(a) are DOIs for CS and VP, but they are only DDOIs for BP. Note that for BP the equality constraints  $\pi_h = \pi_i$  for items  $h, i \in I$  with  $w_h = w_i$  used by Ben Amor et al. (2006) are a special case of the DDOIs of Theorem 2(a). Equality constraints are analyzed in a more general form in §3.4.

The pair inequalities for VC (see Theorem 2(b)) were earlier stated by Malaguti et al. (2011, p. 180), but not found useful, maybe because they were added only a priori and were not used in combination with SIs.

Also note that for BP and CS a linear-sized subset of the DIs of Theorem 2 is sufficient to enforce all  $\mathcal{O}(m^2)$  pair inequalities: one can assume that all items are sorted by nondecreasing weights so that  $w_1 \leq w_2 \leq \dots \leq w_m$ . Then the  $m - 1$  ranking inequalities  $\pi_1 \leq \pi_2 \leq \dots \leq \pi_m$  imply all pair inequalities (cf. Valério de Carvalho 2005, Ben Amor et al. 2006, for CS). For VP, VC, and BPC such an ordering of the vertices/items is generally not possible, and no linear system of DIs can impose all pair inequalities.

### 3.4. Constraint Aggregation and Elimination

It is well known for BP that equally sized items  $i_1, \dots, i_p \in I$  with  $w_{i_1} = \dots = w_{i_p}$  can be aggregated (Vanderbeck 1999, Ben Amor et al. 2006). It means that the  $p$  rows  $i_1, \dots, i_p$  are dropped and a new row, say row  $k \in I$ , is introduced instead. The result is a master program, in which the aggregated row has right-hand side  $b_k = p$ , and the new entry of a column  $a_j$  is the sum  $\sum_{l=1}^p a_{i_l, j}$  so that  $a_{kj}$  can take the values from  $\{0, 1, \dots, p\}$ . Consequently, the pricing problem of a CG approach to BP with aggregation is a *bounded*

*knapsack problem*; see Kellerer et al. (2004). Note that a similar aggregation is also possible for CS and VP.

In the context of DIs, Ben Amor et al. (2006) have characterized this principle of aggregation for BP in more detail. They have shown that the set of equality constraints  $\pi_h = \pi_i$  for all  $h, i \in I$  with  $w_h = w_i$  is a set of DDOIs for BP. Furthermore, they proposed two different ways to enforce the equality constraints in a CG algorithm: explicitly adding the corresponding primal columns to the RMP or using constraint aggregation. Their computational results indicated that constraint aggregation is by far superior, which can be attributed to the following facts (see Ben Amor et al. 2006): The size of the RMP decreases significantly in terms of both the number of constraints and the number of feasible packings and the size of the subproblem is also reduced.

With the following proposition we generalize the simple addition of two rows.

**PROPOSITION 6.** *Let  $\alpha \in \mathbb{R}$ ,  $h, i \in I$  be two row indices with the primal constraints  $a_{h*} \lambda = b_h$  and  $a_{i*} \lambda = b_i$  and associated dual variables  $\pi_h$  and  $\pi_i$ , respectively.*

(i) *The following constraints are equivalent:*

$$(a_{h*} + \alpha a_{i*}) \tilde{\lambda} = b_h + \alpha b_i \Leftrightarrow \begin{pmatrix} a_{h*} \\ a_{i*} \end{pmatrix} \tilde{\lambda} + \begin{pmatrix} \alpha \\ -1 \end{pmatrix} y = \begin{pmatrix} b_h \\ b_i \end{pmatrix},$$

$y \in \mathbb{R}$ . (2)

(ii) *Let  $\pi^* \in D^*$  be a dual-optimal solution with  $\alpha \pi_h^* = \pi_i^*$ , which fulfills  $E^T \pi^* \leq e$  for a set of given DDOIs. (This is equivalent to stating that  $\alpha \pi_h = \pi_i$  together with  $E^T \pi \leq e$  are DDOIs.)*

*Then  $P$  and  $\tilde{P}$  are equivalent to an aggregated formulation  $\tilde{P}'$  in which the  $h$ th and  $i$ th equalities are replaced by the lhs of (2). The dual solution  $\pi^*$  defined by*

$$\pi_k^* = \begin{cases} \pi_k^* & \text{for } k \in I \text{ with } k \neq h, i, \\ \pi_h^* & \text{for the new aggregated constraint} \end{cases}$$

*is an optimal solution to the dual of  $\tilde{P}'$ .*

Proposition 6 gives rise to the following interesting result for instances of CS with no loss.

**REMARK 1.** Consider an instance of CS that has a solution without loss. An optimal solution then uses the minimum number of rolls given by  $\sum_{i \in I} w_i b_i / L$ . Ben Amor et al. (2006) have shown that the DIs  $\pi_i^* = w_i / L, i \in I$  are a set of DDOIs. Using this information about optimal dual values, the repeated aggregation of rows results in a corresponding LP with a single row, right-hand side  $b' = \sum_{i \in I} b_i w_i / L$ , and coefficients  $a'_j \leq 1$  for all  $j \in J$ . Those aggregated columns with entry  $a'_j = 1$  correspond to original columns  $a_j, j \in J$ , which represent a cutting pattern without



loss. In the aggregated master problem, exactly one of these columns forms the basic solution. This variable takes the value  $\sum_{i \in I} w_i b_i / L$ .

The fact that the aggregated model can be solved by finding a cutting pattern without loss seems appealing. However, this result is not useful for computing a primal feasible LP solution for the original model  $P$ . By disaggregation one can find a solution to  $\tilde{P}$ , where the above DDOIs are generally no WSIs. Thus, the recovery algorithm is not applicable (see §4.2). The only tool to transform the solution to  $\tilde{P}$  with some active DDOIs is running a stabilized CG algorithm, as discussed at the end of §2.

The above example of CS instances without loss is an extreme example in the sense that all constraints can be aggregated into a single constraint. In contrast, if only a partial aggregation but with  $\alpha \neq 1$  is possible, one can benefit from Proposition 6 and the aggregated formulation because CG can then be applied to the corresponding nontrivial, but smaller, instance. The result, the LP-bound and the optimal dual values, both for the aggregated formulation and here-with also for the original formulation, may be rather helpful: Faster bounding procedures can be developed, and optimal dual values can stabilize the CG algorithm for the original disaggregated model  $\tilde{P}$ ; see again §2.

We now consider the special case of  $\alpha = 0$  in Proposition 6. It means that the  $i$ th constraint is completely eliminated. The elimination of dominated constraints has long been used to reduce the size of set-covering/set-partitioning instances (e.g., Balinski 1965, Garfinkel and Nemhauser 1969, Balas and Padberg 1976): If for two rows  $h, i \in I$  the inequality  $a_{ij} \geq a_{ij}$  holds for all  $j \in J$ , then row  $i$  is dominated by row  $h$  and can be eliminated. Using this property, it is possible to eliminate specific rows in the extensive formulation for VC, as earlier observed and exploited in preprocessing by Méndez-Díaz and Zabala (2006, p. 831) and Malaguti et al. (2011, p. 181):

**PROPOSITION 7.** *Consider two vertices  $h, i \in I$  in VC. If  $N(h) \supseteq N(i)$ , row  $i$  is dominated by row  $h$  and can therefore be eliminated.*

The positive effects of constraint elimination on a CG approach to VC are similar as those of aggregation for BP, CS, and VP. One can solve VC for the graph induced by the noneliminated vertices, which leads to a row- and column-reduced RMP as well as smaller subproblem instances.

Following Proposition 6, the elimination of row  $i$  is one possibility to enforce the DI  $\pi_i = 0$ . Let  $P'$  be the primal model resulting from eliminating all dominated rows  $I' \subseteq I$  from  $P$ . Because  $z_{P'} = z_P$ , the set of DIs  $\pi_i = 0, i \in I'$  is a set of DDOIs (Proposition 1).

An alternative way of enforcing the DI  $\pi_i = 0$  is to explicitly include the corresponding column, i.e., the  $i$ th unit vector with cost zero, in the RMP. As for constraint aggregation and equality constraints, constraint elimination is computationally superior to an extended formulation with the DDOIs.

## 4. Separation and Recovery Algorithms

Our approach differs in several aspects from the approaches of Valério de Carvalho (2005) and Ben Amor et al. (2006) for stabilizing CG with DOIs or DDOIs. From an application point of view, we cover additional problems (VP, VC, and BPC) and provide general insights on how DOIs and DDOIs may be determined for alternative problems. The focus of this section is, however, on the differing algorithmic parts.

First, because the number of known DOIs and DDOIs is exponential in size, Valério de Carvalho (2005) and Ben Amor et al. (2006) suggested using only a linear-sized subset of the SIs with  $|S| \leq 2$  when solving CS and BP. The associated primal variables  $y$  are here added as additional columns to  $\tilde{P}$  before solving the RMP for the first time. We do not limit our approach in this way, but consider exponential classes of DOIs and DDOIs, which makes it necessary to identify violated DIs dynamically in the CG process. Indeed, this is a separation problem, and the first part of this section presents effective separation algorithms for CS, BP, VC, and BPC. The result of separation is the identification of a violated DI for  $\tilde{D}$ , which is then added as a new column to the primal formulation  $\tilde{P}$ . In this sense, separation of DIs is also column generation.

Second, we do not restrict our approaches to DIs that have been proven to be DOIs or DDOIs for the problem at hand. Instead, we may perform a kind of over-stabilization by also using classes of DIs that are generally neither DOIs nor DDOIs. As a result, we might end up with an optimal solution to the over-stabilized primal  $\tilde{P}$  that cannot be transformed into a feasible solution to  $P$  (as in Proposition 1). In these cases, we apply a recovery procedure to restore primal feasibility. Such a procedure is presented in the second part of this section.

In the following, all classes of DIs under consideration are in fact WSIs so that we present separation algorithms and the recovery algorithm for these. Recall that WSIs and SIs are denoted by  $(h \leftarrow t, S)$  and  $(h \leftarrow S)$ , respectively. Clearly, one should only consider such subclasses of WSIs that are likely to be DDOIs at least for some instances. For pure binary problems such as BP, VC, and BPC, we therefore restrict ourselves to SIs  $(h \leftarrow S)$ .

A WSI  $(h \leftarrow t, S)$  or SI  $(h \leftarrow S)$  is probably no DDOI if it violates the weight inequality for CS and BP, if the

defining set  $S$  is no independent set for VC, and both for BPC. Therefore, we require  $h$ ,  $t$ , and  $S$  to meet the following conditions:

- For CS and VP, the weights inequality  $w_h \geq \sum_{s \in S} t_s w_s$  must hold.
- For BP, we use the criterion  $w_h \geq \sum_{s \in S} w_s$ .
- For VC,  $h$  and  $S$  need to fulfill  $N(h) \cup \{h\} \supseteq \bigcup_{s \in S} N(s)$ , and  $S$  must constitute an independent set.
- For BPC, we require both of the above conditions of BP and VC.

#### 4.1. Dynamic Separation of Violated Dual Inequalities

The careful dynamic generation and addition of violated WSIs prevents the RMP from being stuffed with useless DOIs as may happen with an a priori addition of many DIs that one suspects helps stabilize the CG process. Clearly, mixing both strategies, the a priori integration of expectedly helpful DIs and the dynamic generation of violated DIs is a viable strategy that we analyze later in the computational experiments.

For the identification of violated WSIs, a separation procedure is needed. Let  $\bar{\pi}_i, i \in I$  be the dual values in a CG iteration after the reoptimization of the RMP. The task of the separation procedure is to identify one or several violated WSIs ( $h \leftarrow t, S$ ) if any, i.e., with  $\bar{\pi}_h < \sum_{s \in S} t_s \bar{\pi}_s$ . In the following, we describe the individual separation algorithms that we later use for CS, BP, VC, and BPC in our computational study.

*Cutting Stock.* For CS, we consider a given item  $h \in I$  for which a violated WSI of the form ( $h \leftarrow t, S$ ) requires the identification of the subset  $S$  and the quantities  $t_s \in \mathbb{Z}_{>0}^S$ . Such a violated WSI exists if and only if  $\bar{\pi}_h < z_h$ , where  $z_h = \sum_{i \in I} \bar{\pi}_i t_i$ , such that  $\sum_{i \in I} w_i t_i \leq w_h$ . This is, for each item  $h \in I$ , an UKP with identical coefficients, as in the pricing problem, except for a smaller capacity of  $w_h$  instead of  $L$ . Note that the WSI ( $h \leftarrow t, S$ ) with  $t$  and  $S = \{i \in I : t_i > 0\}$  for the optimal solution to the prior UKP is the most violated WSI for this specific item  $h$ .

From a worst case point of view, the best known algorithms for solving the UKP pricing problem

are based on dynamic programming (DP), and they require  $\mathcal{O}(mL)$  time (see Kellerer et al. 2004). Figure 1 shows the state graph, in which states correspond with residual capacities  $0, 1, 2, \dots, L$ . For each item  $i \in I$ , arcs  $(p, p + w_i)$  are present for  $0 \leq p$  and  $p + w_i \leq L$ . They all have an associated profit  $\bar{\pi}_i$ . Additional arcs  $(p, p + 1)$  with profit 0 model possible slack in a solution. UKP can now be interpreted as a longest 0- $L$ -path problem in this directed acyclic graph. When solved with DP, the value  $z_h$  for  $h \in I$  can be found at state  $w_h$ . If  $\bar{\pi}_h < z_h$ , then the associated longest 0- $w_h$ -path defines the subset of selected items  $S \subseteq I$  and their quantities  $t \in \mathbb{Z}_+^S$ . Thus, the exact dynamic separation of violated WSIs is a byproduct of solving the pricing problem for CS. The additional effort for identifying a most violated WSI is  $\mathcal{O}(m)$  and, therefore, the separation comes almost for free. Note that the approach provides not only a most violated WSI, but several other violated WSIs can also be separated.

For VP, the multidimensional extension of CS, the pricing problem is an unbounded DKP (Kellerer et al. 2004, Chap. 9). Finding a most violated WSI requires the solution of  $m$  smaller DKPs, which are identical to the pricing problem, but the multidimensional capacity  $(L_1, \dots, L_d)$  is set to  $(w_{h1}, \dots, w_{hd})$  for each  $h \in I$ . The literature reports different exact solution algorithms for DKP (see Ozden 1988, Fréville 2004). It seems that none of these approaches allows the direct retrieval of the solution to the WSI separation problem in a straightforward way.

*Bin Packing.* The pricing problem and the SI separation problem for BP are KPs. Again, they can be solved efficiently in  $\mathcal{O}(mL)$  pseudopolynomial time using DP (Kellerer et al. 2004, Chap. 5). The state graph, however, differs from the one of UKP. There is one stage for each item  $i \in I = \{1, \dots, m\}$  plus one start stage with the single state 0. At stage  $i \in I$ , the states  $0, 1, 2, \dots, L$  are the possible residual capacities. Stage  $i - 1$  and stage  $i$  for  $i \in I$  are connected in the following way: The arcs  $(p, p + w_i)$  with profit  $\pi_i$  for  $0 \leq p \leq p + w_i \leq L$  ( $p = 0$  for  $i = 1$ ) model the inclusion of item  $i$  in the respective solution, whereas the

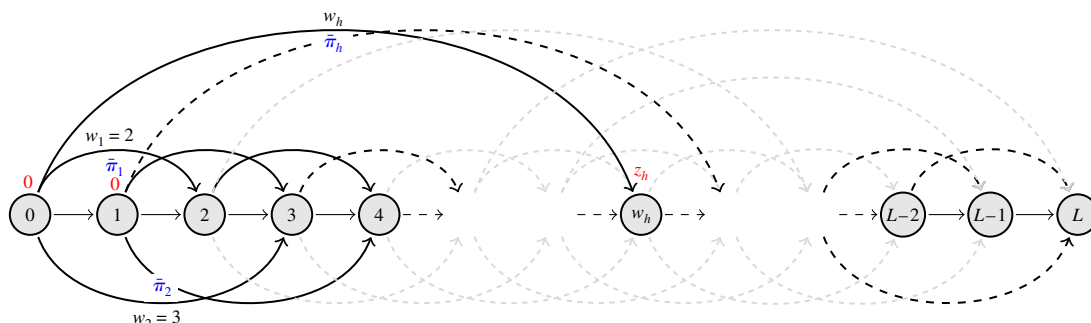


Figure 1 (Color online) Dynamic Programming State Graph for the Unbounded Knapsack Problem

arcs  $(p, p)$  with profit 0 for  $0 \leq p \leq L$  ( $p = 0$  for  $i = 1$ ) model the exclusion of item  $i$ . Using forward DP, the values at the states  $w_h$  at stage  $m$  are  $z_h = \max_{i \in I} \bar{\pi}_i t_i$ , such that  $\sum_{i \in I} w_i t_i \leq w_h$  and  $t \in \{0, 1\}^I$ , i.e., the values of the same KP instance as the pricing problem, but for smaller residual capacities. For each  $h \in I$ , the corresponding path from 0 to  $w_h$  (connecting stages 0 and  $m$ ) defines a set  $S$  of items to include. The respective SI ( $h \leftarrow S$ ) is violated if and only if  $z_h > \bar{\pi}_h$  holds. Summing up, as for CS, if the BP pricing problem is already solved with DP, the additional effort needed to identify violated SIs is  $\mathcal{O}(m)$ . Thus, separation is a byproduct of pricing.

**Vertex Coloring.** For VC, we consider again a fixed vertex  $h \in I$ . For some  $S \subseteq I \setminus \{h\}$ , ( $h \leftarrow S$ ) is a possible SI if  $S$  is an independent set and  $N(h) \cup \{h\} \supseteq N(s)$  holds for all  $s \in S$ . Denote by  $S_h = \{s \in I: N(h) \cup \{h\} \supseteq N(s)\}$  the set of all vertices that have a subset of the conflicts of vertex  $h$ . Then the SI ( $h \leftarrow S$ ) is violated if and only if  $\bar{\pi}_h < z_h$ , where  $z_h = \max_{S \subseteq S_h} \sum_{s \in S} \bar{\pi}_s$  such that  $S$  is an independent set. This is an MWIS problem (Balas and Xue 1991). Again, the separation problem is identical to the pricing problem, but defined on the vertex-induced subgraph  $G[S_h]$ . As such, it is strongly  $\mathcal{NP}$ -hard and no byproduct of solving the pricing problem.

For the separation of violated SIs ( $h \leftarrow S$ ), we solve an MWIS for each vertex  $h \in I$  using the same exact algorithm as for the pricing problem; see §5. The respective sets  $S_h$  for each vertex  $h \in I$  are computed once and prior to the CG process. Typically, the sets  $S_h$  are very small compared to  $I$  in the VC benchmark instances. As a result, separation times are negligible. Analog to CS and BP, the separation procedure provides not only the most violated SI, but also several others.

**Bin Packing with Conflicts.** Since the BPC can be seen as the synthesis of BP and VC, it can be expected that ideas for the separation of DIs can be adapted from these problems. First note that the CG subproblem is a binary *knapsack problem with conflicts* (KPC). For general conflicts, i.e., defined by arbitrary conflict graphs  $(I, \mathcal{E})$ , the KPC is strongly  $\mathcal{NP}$ -hard and also very hard to solve (Hifi and Michrafy 2007, Bettinelli et al. 2014). If the conflict graph is of bounded treewidth or is a chordal graph, an efficient DP algorithm with complexity  $\mathcal{O}(m^2 L)$  has been proposed by Pferschy and Schauer (2009). A substantial part of the CG literature on BPC focuses on an even more restricted class of conflict graphs: interval graphs. Herein, each item  $i \in I$  has an associated interval  $\mathcal{I}_i := (a_i, b_i)$ , and two items  $i, j \in I$  are in conflict if and only if  $\mathcal{I}_i \cap \mathcal{I}_j \neq \emptyset$ . For interval graphs, the KPC subproblem can be solved by dynamic programming in  $\mathcal{O}(mL)$  time and

space with the algorithm of Sadykov and Vanderbeck (2013). The following computational analysis is likewise restricted to the case of interval conflict graphs.

We briefly sketch the DP algorithm for the KPC proposed by Sadykov and Vanderbeck (2013): The state graph consists of the same  $\mathcal{O}(mL)$  states as the one for KP, one for each residual capacity  $0, 1, 2, \dots, L$  and item  $i \in I$  plus one start state at stage 0. The crucial difference, however, is that items  $i \in I$  need to be sorted according to the right-hand side  $b_i$  of their conflict intervals. Let  $pred_i$  denote a last vertex in  $\{0, 1, 2, \dots, i-1\}$  that is not in conflict with a vertex  $i$  (items are sorted in the above order and 0 is an artificial start item not in conflict to any other). Then the arcs  $(p, p + w_i)$  connect stage  $pred_i$  to stage  $i$  (instead of stages  $i-1$  and  $i$ , as for KP). They represent the selection of the item  $i$  and result in the profit  $\bar{\pi}_i$ . Similar to KP, the arcs  $(p, p)$  connect stages  $i-1$  and  $i$ . They model that item  $i$  is not part of the KPC solution with resulting profit of 0. By construction, paths in the state graph correspond to conflict free packings.

As before, for the separation of SIs ( $h \leftarrow S$ ), we consider the possible items  $h \in I$  one by one. The most violated SI ( $h \leftarrow S$ ) results from the solution of a smaller KPC subproblem, which is defined by the smaller capacity  $w_h$  ( $\leq L$ ) and on the item subset  $S_h = \{s \in I: w_h \geq w_s, N(h) \cup \{h\} \supseteq N(s)\}$ . This separation problem is *no* byproduct of the BPC subproblem in this case. In fact, the DP value for the last item  $m$  and the residual capacity  $w_h$  results in a solution that is an independent set  $S$  with weight not exceeding  $w_h$ . However, the necessary condition  $S \subseteq S_h$  generally does not hold. Thus, the DI ( $h \leftarrow S$ ) does typically not qualify as a possible SI.

Since separation is nontrivial, the following three procedures for identifying violated SIs are considered:

- pairs Pair inequalities can be separated by explicit inspection, which takes  $\mathcal{O}(m^2)$  time.
- DP Based on the solution of the KPC pricing problem, the following DP-based heuristic can be applied: One inspects all states  $w = 1, 2, \dots, L$  at the final stage  $m$ . Each state reached represents a solution that is an independent set  $S_w$  with weight  $w$ . For this set  $S_w$ , one can find a best item  $h \in I$  (if any) with  $w_h \geq w$  and  $S_h \supseteq S_w$  so that  $h$  and  $S_w$  induce a SI. Section EC.2 in the online supplement explains that the entire procedure can be implemented to run in  $\mathcal{O}(mL)$  time.
- exact The exact separation of SIs ( $h \leftarrow S$ ) must solve a smaller KPC for each vertex  $h \in I$  with capacity  $w_h$  and vertices  $s \in S_h$ . This requires  $\mathcal{O}(m^2 L)$  time, which turns out to be rather time consuming. Thus, we only solve the DP for an

item  $h \in I$ , for which the LP-bound of the KP with vertices  $s \in S_h$  results in a sufficiently high violation compared to the currently most violated SI found. The procedure is still exact, but potentially misses some SIs when separating multiple SIs within certain quality compared to the most violated one.

#### 4.2. Overstabilization and Recovery of Feasible Primal Solutions

For further stabilizing the CG process, we propose to use not only DIs that are proven to be DOIs or DDOIs. It may then happen that all dual-optimal solutions  $\pi^*$  to  $D$  are cut off. From a primal perspective, this means that we have overstabilized the CG process and that we terminate with a primal solution to  $\tilde{P}$  that cannot be transformed into a feasible solution of  $P$  with the same cost; see Proposition 1. The following example illustrates this behavior.

**EXAMPLE 1.** Consider an instance of BP with bin capacity  $L = 10$  and four items with  $w_1 = 5$ ,  $w_2 = 2$ ,  $w_3 = 2$ , and  $w_4 = 2$ . The linear relaxation uses four feasible packings— $(1, 1, 1, 0)^\top$ ,  $(1, 1, 0, 1)^\top$ ,  $(1, 0, 1, 1)^\top$ , and  $(0, 1, 1, 1)^\top$ —with corresponding dual constraints  $\pi_1 + \pi_2 + \pi_3 \leq 1$ ,  $\pi_1 + \pi_2 + \pi_4 \leq 1$ ,  $\pi_1 + \pi_3 + \pi_4 \leq 1$ , and  $\pi_2 + \pi_3 + \pi_4 \leq 1$ , respectively. The unique primal and dual optimal solutions are  $\lambda^* = \pi^* = (\frac{1}{3}, \frac{1}{3}, \frac{1}{3}, \frac{1}{3})^\top$  with  $z(P) = z(D) = \frac{4}{3}$ .

After adding the SI  $\pi_1 \geq \pi_3 + \pi_4$ , the optimal solution  $\pi^*$  is cut off, and a new dual-optimal solution is given by  $(\frac{1}{2}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4})^\top$  with  $z(\tilde{D}) = 1.25$ . Thus, the SI  $\pi_1 \geq \pi_3 + \pi_4$  is no DDOI. The corresponding column in  $\tilde{P}$  is  $(-1, 0, 1, 1)^\top$ , and an optimal solution for  $\tilde{P}$  is given by  $(\tilde{\lambda}^*, y^*) = ((\frac{1}{2}, \frac{1}{2}, \frac{1}{4}, 0)^\top, \frac{1}{4})$ .

Transforming  $(\tilde{\lambda}^*, y^*)$  into a feasible solution to  $P$  means exchanging item 1 for items 3 and 4 in one of the chosen bin columns that include item 1. Doing so, however, results in a new bin column, in which either item 3 or 4 is packed twice, which is infeasible for BP.

Note that for CS,  $(\tilde{\lambda}^*, y^*)$  can be transformed into a feasible solution for  $P$  because columns are allowed to have nonnegative integer coefficients. This also reflects the intuition that WSIs are DOIs for CS and VP, but they are not for BP.

Before we formally state the recovery algorithm, we introduce some additional notation. Recall that any WSI is of the form  $(h \leftarrow t, S)$ , where  $h \in I$  is the row with the  $-1$  entry in the coefficient column vector  $(E_{ik})$  in (1) and  $(t, S)$  are the nonzero entries  $t_i > 0$  for  $i \in S$ . Let  $K$  denote the column indices of  $E$ . Thus, in the primal model  $\tilde{P}$ , these indices refer to the variables  $y_k, k \in K$  and rows of  $E^\top$  with associated WSI  $(h \leftarrow t, S)$ . We write  $k = k(h \leftarrow t, S)$  in this case. Two indices  $j \in J$  and  $k \in K$  are said to be *WSI compatible* if  $a_j - u_h + t \in A$  for  $h \in I$  defined by  $k = k(h \leftarrow t, S)$ .

#### Algorithm 1 (Column generation recovery algorithm)

```

1 repeat
2   Compute solution  $(\lambda^*, y^*)$  to  $\tilde{P}$  using CG
3   while WSI compatible pairs  $(j, k) \in J \times K$ 
4     with  $\lambda_j^* > 0$  and  $y_k^* > 0$  exist do
5     Select a WSI compatible pair  $(j, k) \in J \times K$ 
6     with  $\lambda_j^* > 0$  and  $y_k^* > 0$ 
7     Let  $\delta := \min\{\lambda_j^*, y_k^*\}$ 
8     Let  $j' \in J$  such that  $a_{j'} = a_j - u_h + t$ 
9     Let  $\lambda_{j'}^* := \lambda_j^* - \delta, y_k^* := y_k^* - \delta, \lambda_{j'}^* := \lambda_{j'}^* + \delta$ 
10    if  $y_k^* > 0$  then
11      Select one (or several)  $h \in I$  such that  $y_k^* > 0$ 
12      for  $h$  defined by  $k = k(h \leftarrow t, S)$ 
13      Eliminate all WSIs  $(h \leftarrow t', S')$  for  $S' \subseteq I,$ 
14       $t' \in \mathbb{Z}_{>0}^I$  from the RMP and prohibit
15      their re-generation
16 until  $y^* = 0$ 
17 Result Solution  $\lambda^*$  to  $P$ 
    
```

The recovery algorithm is formally stated by Algorithm 1. First note that by definition of compatible pairs  $(j, k) \in J \times K$  the value of  $\delta$  is strictly positive in step 5. Moreover, the assignments in the step 7 neither invalidate the primal feasibility  $A\lambda^* + E^\top y = b$  nor alter the cost of the solution. This results from the equalities  $a_{j'} = a_j - u_h + t$  and  $c_j = c_{j'} = 1$ . Therefore, the inner loop (steps 4–10) is successful if an equivalent primal solution with  $y^* = 0$  is constructed. Otherwise (step 8), there must exist an active WSI with row index  $h$  for which no compatible  $j \in J$  exists. In this case, replacements to row  $h$  cannot be recovered. The recovery algorithm has failed in this iteration, and therefore some active WSIs are eliminated from the RMP and their reintroduction is made impossible. As a result, the next RMP solution (step 2) does not contain any WSIs that refer to the row index  $h \in I$  selected in step 9. Thus, after a maximum of  $m$  outer loops, the Algorithm 1 must terminate with a solution  $(\lambda^*, 0)$  to  $\tilde{P}$ . Since the recovery procedure is cost preserving, Proposition 1(v) is fulfilled so that the modified  $\lambda^*$  is an optimal primal solution to  $P$ .

Steps 4 and 9 leave different strategies for choices of specific pairs  $(j, k)$  and  $h$  open. For step 4, the selection of a pair  $(j, k)$  leading to a maximal  $\delta$  may help to minimize the number of necessary iterations of Algorithm 1. For the selection of a row  $h$  in step 9, we observed in our computational test that the number of possible choices is always small so that we always choose all  $h \in I$  with  $y_k^* > 0$  and  $k(h \leftarrow t, S)$ .

## 5. Computational Results

In the following, we present computational results for CS, BP, VC, and BPC, showing how the stabilization with DIs affects the CG process. Furthermore, we analyze the effects of using different classes of DIs and

show that the dynamic generation of DIs is often beneficial. All presented results are on the linear relaxation of the respective master programs. Recall that because of a possible over-stabilization, the recovery procedure needs to be applied for BP and BPC. All algorithms described in this paper were implemented in C++ using CPLEX 12.5 as LP solver. The experiments were conducted on two standard PCs, one with an Intel® Core™ i7-3770 (for BP) and one with an Intel® Core™ i7-2600 (for CS, VC, and BPC), each running at 3.4 GHz and with 16.0 GB main memory using a single thread only.

### 5.1. Results for Bin Packing

We use the same basic CG algorithm for each of the different stabilization strategies that we compare. The main features of the CG algorithm are as follows: We use a *best fit decreasing* heuristic (Martello and Toth 1990) to generate an initial set of columns for warm starting the CG process. The subproblems are solved by the DP algorithm described in §4.1, and a single best reduced-cost bin column is generated in each iteration. Pretests indicated that adding a single bin column is generally the best strategy. If DIs are generated dynamically, multiple SI columns with a violation of at least 25% of the most violated SI are added. In preliminary computational tests, this appeared to be marginally superior than values of 50% and 75% and adding a single DI column only.

The first strategy we consider is the standard CG approach without using any DIs referred to by *standard* in the following. In contrast, the strategy denoted by *aggregation* uses constraint aggregation on items with identical weight. This is the approach Ben Amor et al. (2006) followed and can be seen as the state of the art for BP. It is, therefore, used as the baseline strategy for comparisons with all other strategies. Moreover, all following strategies also make use of this aggregation. We also consider using DIs in a static fashion only (*static*), i.e., the addition of a set of expectedly helpful DIs to the RMP as static columns prior to the CG process. Motivated by preliminary computational tests, we use DIs that are similar to what Ben Amor et al. (2006) proposed for CS. More precisely, we use the  $m-1$  ranking inequalities, which are DDOIs for BP, see Theorem 2. Additionally,  $\Theta(m)$  SIs ( $h \leftarrow S$ ) with  $|S| = 2$  are added, namely one for each item  $h \in I$ . Herein, the set  $S = \{i, j\}$  is chosen (if existing) such that the slack in  $w_h \geq w_i + w_j$  is minimal (and nonnegative). The pure dynamic generation of violated SIs using the separation procedure of §4.1 is denoted by *dynamic*. Finally, we consider the combination of the latter two, i.e., the use of static DIs and the dynamic generation of additional violated SIs. This last strategy is referred to as *stat + dyn*.

As test instances we used the benchmark sets by Scholl et al. (1997) and Sim and Hart (2013) comprising a total of 1,210 and 15,830 instances, respectively. Both benchmark sets are subdivided into several subclasses differing with respect to the number of items, the capacity, and the variance of the item weights, resulting in instances with very different characteristics and degrees of complexity. In the following, we present results averaged over all subclasses. However, we include only those instances for which the computation time of *aggregation* is at least one second. This reduces the number of benchmark instances to 207 and 4,032, respectively. The other instances are solved in too little time with the stronger algorithmic strategies. Hence, the consideration of these instances would, if they were taken into account, produce unreliable values for the computation times.

We further differentiate two subgroups for each benchmark set: Group 1 comprises the instances with computation times between one and 10 seconds for *aggregation*, and it consists of 157 and 3,352 instances for the sets of Scholl et al. (1997) and Sim and Hart (2013), respectively. The hardest instances in terms of solution time ( $>10$  seconds for *aggregation*) are in Group 2, consisting of the remaining 50 and 680 instances.

Table 2 summarizes the results for BP, where the table columns have the following meaning:

$m$	The average number of rows in the RMP relative to <i>aggregation</i>
<i>Solution time</i>	The time for solving the linear relaxation of the RMP relative to <i>aggregation</i> ; we present average, maximum, and minimum values over the instances
# CG Iterations	The number of CG iterations relative to <i>aggregation</i> ; we present average, maximum, and minimum values over the instances
<i>SP/RMP</i>	The ratio of the times spent for solving the subproblem and reoptimizing the RMP, averaged over the instances.
# Dynamic DIs	The number of DIs that are generated dynamically during the CG process; we present average, maximum, and minimum values over the instances
<i>Recovery</i>	The total number of instances for which a recovery is needed (# inst.) and the maximum number of iterations of the recovering algorithm for one of the instances (max it.).

**Table 2** Computational Results for the Bin Packing Problem

Instances	Algorithm	$m$	Solution time (avg./max/min)	# CG Iterations (avg./max/min)	SP/ RMP	# Dynamic DIs (avg./max/min)	Recovery (# inst./max it.)
Sim and Hart (2013) all instances ( $n = 4,032$ )	<i>standard</i>	2.50	3.26/31.46/0.84	2.52/20.47/0.96	2.66	—	—/—
	<i>aggregation</i>	1.00	1.00/1.00/1.00	1.00/1.00/1.00	21.24	—	—/—
	<i>static</i>	1.00	0.97/1.39/0.67	0.66/1.02/0.47	30.03	—	7/1
	<i>dynamic</i>	1.00	0.40/1.39/0.07	0.35/1.11/0.08	11.20	1,462/9,852/26	54/2
	<i>stat + dyn</i>	1.00	0.38/1.30/0.06	0.24/1.02/0.03	14.96	865/7,351/0	54/2
Sim and Hart Group 1 ( $n = 3,352$ )	<i>standard</i>	2.65	3.40/31.46/0.87	2.67/20.47/0.96	2.41	—	—/—
	<i>aggregation</i>	1.00	1.00/1.00/1.00	1.00/1.00/1.00	20.18	—	—/—
	<i>static</i>	1.00	0.96/1.35/0.67	0.65/1.02/0.47	28.74	—	7/1
	<i>dynamic</i>	1.00	0.46/1.39/0.09	0.39/1.11/0.10	12.00	1,007/7,029/26	37/2
	<i>stat + dyn</i>	1.00	0.44/1.30/0.08	0.28/1.02/0.04	15.43	529/5,531/0	39/2
Sim and Hart Group 2 ( $n = 680$ )	<i>standard</i>	1.88	2.64/8.55/0.84	1.91/3.70/1.05	4.33	—	—/—
	<i>aggregation</i>	1.00	1.00/1.00/1.00	1.00/1.00/1.00	26.81	—	—/—
	<i>static</i>	1.00	1.02/1.39/0.79	0.71/0.87/0.59	36.57	—	0/0
	<i>dynamic</i>	1.00	0.19/1.21/0.07	0.21/0.81/0.08	8.18	3,704/9,852/669	17/2
	<i>stat + dyn</i>	1.00	0.18/1.20/0.06	0.11/0.82/0.03	13.09	2,520/7,351/0	15/2
Scholl et al. (1997) all instances ( $n = 207$ )	<i>standard</i>	2.28	3.24/28.77/0.91	2.41/20.26/0.96	2.36	—	—/—
	<i>aggregation</i>	1.00	1.00/1.00/1.00	1.00/1.00/1.00	18.06	—	—/—
	<i>static</i>	1.00	0.97/1.22/0.76	0.65/1.00/0.47	26.87	—	1/1
	<i>dynamic</i>	1.00	0.31/1.07/0.08	0.28/1.03/0.10	6.81	2,252/11,567/35	6/2
	<i>stat + dyn</i>	1.00	0.28/1.12/0.07	0.17/1.00/0.04	10.47	1,465/7,270/0	2/1
Scholl et al. Group 1 ( $n = 157$ )	<i>standard</i>	2.48	3.41/28.77/1.08	2.62/20.26/1.06	2.24	—	—/—
	<i>aggregation</i>	1.00	1.00/1.00/1.00	1.00/1.00/1.00	18.25	—	—/—
	<i>static</i>	1.00	0.97/1.19/0.76	0.64/1.00/0.47	27.17	—	1/1
	<i>dynamic</i>	1.00	0.37/1.07/0.13	0.33/1.03/0.14	7.79	1,372/4,472/35	4/2
	<i>stat + dyn</i>	1.00	0.34/1.12/0.09	0.21/1.00/0.07	11.58	871/3,443/0	2/1
Scholl et al. Group 2 ( $n = 50$ )	<i>standard</i>	1.77	2.75/8.11/0.91	1.86/3.57/0.96	2.80	—	—/—
	<i>aggregation</i>	1.00	1.00/1.00/1.00	1.00/1.00/1.00	17.52	—	—/—
	<i>static</i>	1.00	0.99/1.22/0.86	0.68/0.79/0.55	26.02	—	0/0
	<i>dynamic</i>	1.00	0.17/1.07/0.08	0.18/0.70/0.10	4.61	5,015/11,567/936	2/1
	<i>stat + dyn</i>	1.00	0.15/1.08/0.07	0.09/0.70/0.04	8.03	3,329/7,270/0	0/0

Note that the average values in the first four cases are geometric means because each data point is a ratio. In contrast, the average for # *Dynamic DIs* is an arithmetic average.

Table 2 shows that the dynamic generation of DIs results in an average speedup of approximately factor two compared to *aggregation*. For the hard Group 2 instances, the average speedup even reaches factor three to four. Thereby, *stat + dyn* is slightly faster than the pure use of dynamic DIs. There are also instances for which *stat + dyn* and *dynamic* perform worse than *aggregation* in terms of computation times. However, computation times never go beyond 139% and 130% of *aggregation* for *dynamic* and *stat + dyn*, respectively. On the contrary, the biggest speedups exceed factor 15, and *standard* CG is at least one order of magnitude slower than all strategies applying *aggregation*.

The static use of DIs results in a small speedup of approximately 3% on average. Recall that ranking inequalities were not yet known as DDOIs in the work of Ben Amor et al. (2006). Therefore, a static DI-based CG algorithm was not considered for BP at that time.

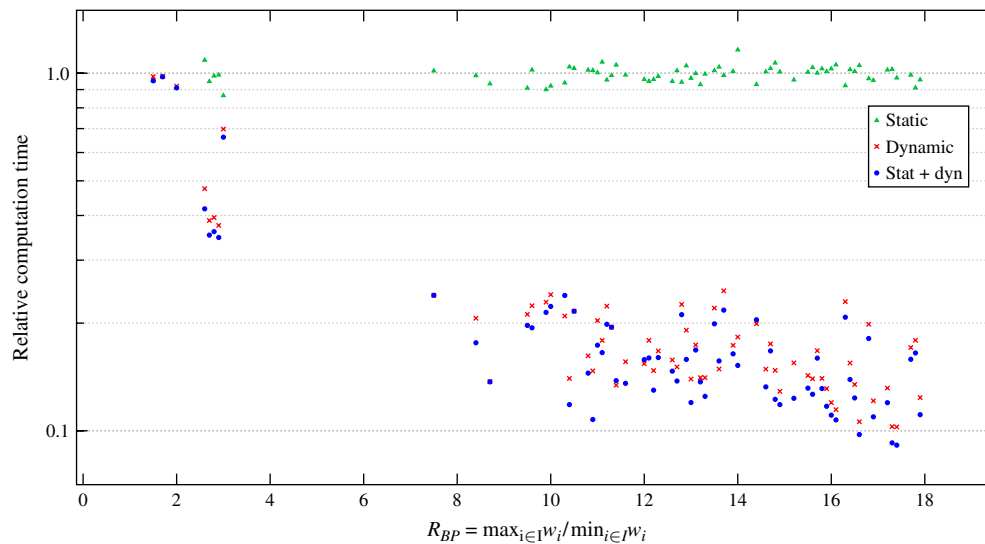
Regarding the number of iterations, the differences between the strategies using dynamic DIs and *aggregation* are even more pronounced than for the

computation times. Also, *static* performs considerably better, as it requires only about two-thirds of the iterations of *aggregation*.

Another interesting result is that a recovery is necessary only for a small number of instances. Even more, the maximum number of iterations of the recovery algorithm is two (cf. Algorithm 1). This demonstrates that the SIs we add to the RMP (static or dynamic) are indeed DDOIs for most of the instances in the benchmark.

Table 2 also indicates that there are instances for which no dynamic DIs are generated when static DIs are added a priori. These are instances in which the item sizes are very similar, so no three items  $h, i, j \in I$  with  $w_h \geq w_i + w_j$  exist. It is therefore clear that no SIs ( $h \leftarrow S$ ) with  $|S| \geq 2$  exist in these cases so no additional speedup can be achieved. Our results include a total of 1,427 and 39 such instances for the sets of Sim and Hart and Scholl et al., respectively. In summary, the overall average speedup for the instances in which SIs with  $|S| \geq 2$  exist actually exceeds factor two significantly.

In Figure 2, we display more generally the influence of the number of possible SIs on the performance of the different CG strategies. The ratio



**Figure 2** (Color online) Average Computation Times Relative to *Aggregation* (Ordinate, in Logarithmic Scale) Depending on the Ratio  $\max_{i \in I} w_i / \min_{i \in I} w_i$  (Abscissae) for all Considered Sim and Hart Instances for the Bin Packing Problem

$R_{BP} = \max_{i \in I} w_i / \min_{i \in I} w_i$  serves as an approximate measure of how many SIs potentially exist. Clearly, this ratio reveals no information on the distribution of the items' weights. In fact, the number of SIs is generally not computable in closed form, but  $R_{BP}$  can be determined easily for each BP instance. Note that a ratio  $R_{BP} < 2$  reflects the case described earlier, in which no SI with  $|S| \geq 2$  exists. The relationship between the ratio  $R_{BP}$  and the speedups using DIs is depicted in Figure 2: It shows the average computation times of *static*, *dynamic*, and *stat + dyn* relative to *aggregation* for all 4,032 considered Sim and Hart instances depending on  $R_{BP}$ . Note that the abscissae groups instances according to  $R_{BP}$  (rounded to one decimal) and that the ordinate shows the relative computation times in logarithmic scale. Figure 2 clearly demonstrates that the dynamic generation of SIs, i.e., implicitly considering the set of all SIs, works better the bigger the ratio  $R_{BP}$ . One can therefore suspect that there is a highly significant positive correlation between the number of SIs and the resulting acceleration of the CG process. For larger values of  $R_{BP}$ , average speedups reach factors between five and ten.

## 5.2. Results for Cutting Stock

For CS, the basic CG algorithm and the different stabilization strategies are similar to BP. We only describe the differences. The UKP subproblems are solved with the DP approach of §4.1. For CS, the basic CG algorithm already makes use of constraint aggregation. Thus, the *standard* and *aggregation* strategies coincide, and results are solely reported as *standard*. Strategy *static* as proposed by Ben Amor et al. (2006) is the

state of the art for CS. Therefore, *static* is the baseline strategy, meaning that results for all other strategies are presented relative to it. Recall that the WSIs are DOIs for CS and no recovery is needed; see Theorem 1(i).

Generally, the BP benchmark instances of the previous section can also be interpreted as CS instances, in which items of identical length need to be aggregated. In preliminary experiments, we found that the UKP subproblems of CS are solved significantly faster than the (binary) KP subproblems of BP for all the benchmark instances. We suspect that this is a result of the smaller state space of the DP of the UKP compared to the one of KP. Recall that the state graph of the UKP (depicted in Figure 1) has only  $L + 1$  states, whereas the state graph of KP has  $1 + (L + 1)m = \mathcal{O}(mL)$  states. We would like to stress that for both UKP and KP we implemented state-of-the-art DP algorithms: In particular, for KP we do not explicitly build the  $\mathcal{O}(mL)$  states but use a list-based implementation, as discussed in Kellerer et al. (2004, § 3.4). In contrast, for UKP we found that a straightforward array-based implementation of the DP approach is faster than the list-based approach. We suspect that on a modern CPU, the smaller-state graph of UKP can be accessed much faster (because of caching techniques) so the solution of the UKP subproblems as they occur in the BP benchmark instances is possible in almost no (measurable) time.

As a result, of the 17,040 benchmark instances from the sets of Scholl et al. and Sim and Hart, over 99% are solved by *standard* CG in less than one second, most of them in only a few milliseconds, meaning that a large number of UKP can be solved in this short time. Therefore, we considered these

**Table 3** Computational Results for the Cutting Stock Problem

Instances	Algorithm	Solution time (avg./max/min)	# CG Iterations (avg./max/min)	SP/ RMP	# Dynamic DIs (avg./max/min)
All instances ( $n = 240$ )	<i>standard</i>	1.43/8.73/0.93	1.63/29.00/1.31	14.91	—
	<i>static</i>	1.00/1.00/1.00	1.00/1.00/1.00	11.91	—
	<i>dynamic</i>	0.98/4.80/0.44	0.81/9.00/0.41	5.01	5,590/18,100/247
	<i>stat + dyn</i>	0.81/1.53/0.29	0.74/1.00/0.31	8.45	1,120/3,742/0

CS instances of limited interest and generated new and harder CS instances. These are characterized by huge values for the capacity (to complicate the subproblems) and larger numbers of items with distinct lengths. The instances we generated can be grouped into different subclasses of problems. Capacities take values  $L \in \{500,000, 1,500,000\}$  and integer item lengths are uniformly drawn from  $[(2/15)L, (2/3)L]$  and  $[(1/150)L, (2/3)L]$ , resulting in items for which the ratios  $w_i/L$  are analog to those in the benchmark instances used by Ben Amor et al. (2006). The number of items with distinct item length takes values  $m = \{125, 250, 500\}$ . Demands are uniformly distributed in the range  $[1, 20]$ . For each subclass, 20 instances were generated. The entire benchmark set comprises 240 instances and is available at <http://logistik.bwl.uni-mainz.de/Dateien/CS.zip>.

Table 3 presents our results for CS. First and foremost, the overall speedups of using stabilized CG are significantly smaller for CS than BP. The *static* use of DIs results in an average speedup of about one-third. The additional use of dynamically generated DI saves an extra 17% of the computation time. The pure *dynamic* generation even results in a slightly slower overall algorithm than the *static* version.

The results regarding the number of iterations are more in favor of using stabilized CG. Iterations are on average reduced by a larger factor than the computation times. The results by subclass are similar to the overall results, as can be seen in §EC.3 of the online supplement.

### 5.3. Results for Vertex Coloring

Analog to BP and CS, we use one basic CG algorithm for our experiments and run it with different strategies regarding the stabilization with DIs. To warm start the CG process, a simple greedy coloring is performed. The MWIS pricing problem is solved using the branch-and-bound algorithm and code by Held et al. (2012). In each iteration, a single best reduced-cost column is added to the RMP. If dynamic generation of DIs is used, violated SIs are separated using the procedure described in §4.1 and multiple SI columns with a violation of at least 25% of the most violated SI are generated.

*Standard* denotes the basic CG algorithm without stabilization. The algorithm using constraint elimination is denoted by *elimination*, and all following strategies also make use of constraint elimination. The setup *elimination* serves as the baseline algorithm for the other strategies. For an approach that uses DIs in a static fashion, recall that no linear system of DIs can impose all pair inequalities; see §3.3. Still, we suggest a strategy that uses a linear-sized subset of the pair inequalities similar to the ranking inequalities in CS and BP. For each vertex  $h \in I$ , we determine a vertex  $i \in \arg \max_{s \in S_h} |N(s)|$ . This is a vertex in the candidate set  $S_h$  (see §4.1) that has the most conflicts, and we add the corresponding SI column to initialize the RMP. The intuition is that this vertex  $i$  is the hardest one to cover of all the vertices that are easier to cover than  $h$ . Therefore, the repetition of this SI selection for each vertex  $h \in I$  should result in a near ranking of all the duals. We refer to this approach as *static*, and no SIs with  $|S| > 1$  are used in this case. The strategies *dynamic* and *stat + dyn* are analog to CS and BP.

Note that because constraint elimination is performed in each strategy that uses DIs, the candidate sets  $S_h$  for all vertices  $h \in I$  comprise only vertices  $s \in S_h$  for which  $h \in N(s)$  holds. Consequently, by Theorem 2(b) all SIs generated by our separation procedure are in fact DOIs, and no recovery is needed.

As benchmark problems, we used the instances from the graph coloring benchmark Web page <https://sites.google.com/site/graphcoloring/home> of Gualandi and Chiarandini (2014). The entire benchmark comprises 136 instances. We restrict our analysis to those instances in which at least one possible SI exists. This eliminates approximately half of the instances, so 74 VC instances remain. Furthermore, we exclude from our computational analysis those instances for which the linear relaxation of the RMP cannot be solved within a time limit of one hour using all strategies (15 instances), and those instances, for which the solution time is less than 0.1 second for the *standard* strategy (34 instances). This leaves 25 instances for the test set, which we further subdivided into two groups: Group 1 comprises all instances for which the ratio  $R_{VC} = (\sum_{h \in I} |S_h|)/m^2 > 0.001$ ; the remaining instances form Group 2.  $R_{VC}$  measures the number of possible replacements of items relative to the instance size, thus giving an approximation on the number of possible SIs relative to the instances size. One can expect



**Table 4** Computational Results for the Vertex Coloring Problem

Instances	Algorithm	$m$	Poss. repl. %	Solution time (avg./max/min)	# CG Iterations (avg./max/min)	SP/RMP	# Dynamic DIs (avg./max/min)
All instances ( $n = 25$ )	<i>standard</i>	1.33	0.00	1.18/2.89/0.86	1.09/1.93/0.89	5.87	—
	<i>elimination</i>	1.00	0.00	1.00/1.00/1.00	1.00/1.00/1.00	8.06	—
	<i>static</i>	1.00	0.64	0.55/1.35/0.06	0.72/1.05/0.16	6.51	—
	<i>dynamic</i>	1.00	0.64	0.51/1.85/0.04	0.69/1.12/0.09	4.95	349/5,314/0
	<i>stat + dyn</i>	1.00	0.64	0.51/1.75/0.06	0.66/1.05/0.09	5.03	121/2,015/0
Group 1 ( $n = 11$ )	<i>standard</i>	1.06	0.00	0.99/1.11/0.86	1.00/1.14/0.89	10.52	—
	<i>elimination</i>	1.00	0.00	1.00/1.00/1.00	1.00/1.00/1.00	11.44	—
	<i>static</i>	1.00	1.45	0.28/0.79/0.06	0.52/0.90/0.16	6.89	—
	<i>dynamic</i>	1.00	1.45	0.23/0.68/0.04	0.47/0.92/0.09	4.81	765/5,314/26
	<i>stat + dyn</i>	1.00	1.45	0.24/0.64/0.06	0.43/0.90/0.09	4.56	270/2,015/0
Group 2 ( $n = 14$ )	<i>standard</i>	1.59	0.00	1.35/2.89/0.96	1.17/1.93/0.97	3.71	—
	<i>elimination</i>	1.00	0.00	1.00/1.00/1.00	1.00/1.00/1.00	6.12	—
	<i>static</i>	1.00	0.01	0.93/1.35/0.66	0.93/1.05/0.79	6.26	—
	<i>dynamic</i>	1.00	0.01	0.93/1.85/0.63	0.93/1.12/0.81	5.06	23/288/0
	<i>stat + dyn</i>	1.00	0.01	0.92/1.75/0.60	0.93/1.05/0.79	5.43	4/52/0

that (dynamic) stabilization generally performs better for Group 1 than for Group 2.

The results of the computational tests are summarized in Table 4. The additional column *Poss. repl. %* shows the ratio  $R_{VC}$  in percent. A general observation is that with stabilized CG an average speedup of approximately factor two can be achieved compared to *elimination*. Thereby, all three strategies using DIs perform comparably well, although the biggest gains can be achieved with a strategy that dynamically adds DOIs. For larger values of  $R_{VC}$ , stabilization with SIs is even more attractive and the average speedup reaches factor four for Group 2.

Table 4 reveals that the biggest speedups exceed factors 15, 25, and 15 for *static*, *dynamic*, and *stat + dyn*, respectively. Expectedly, *elimination* works better the more vertices can be eliminated. The insights regarding the number of iterations are evident: stabilization by DIs can help reduce the number of CG iterations and is rarely detrimental.

#### 5.4. Results for Bin Packing with Conflicts

Analog to §5.1–5.3, we run one basic CG algorithm for BPC with different stabilization strategies to analyze their impact on the CG approach. An initial set of columns to warm start the process is obtained by performing a *first fit decreasing* heuristic several times with different orderings of the items. The KPC subproblems, for which the conflicts are defined on interval graphs in all benchmark instances, are solved with the DP algorithm of Sadykov and Vanderbeck (2013), described in §4.1. Again, a single best reduced-cost bin and multiple SI columns with a minimum violation of 25% of the most violated SI are generated in each iteration.

*Standard* CG without stabilization serves as the baseline strategy for comparisons with the other strategies. Similar to VC, no linear system of DIs can

impose all pair inequalities; see §3.3. For an approach that uses DIs in a *static* fashion, we therefore follow a strategy using the same basic idea as for VC: For each item  $h \in I$  we identify the item  $i \in S_h$  (if any) that is the hardest to cover of all the vertices that are easier to cover than  $h$  and add the respective pair inequality to the RMP. For BPC, however, the hardness to cover an item  $i$  depends on both its weight  $w_i$  and its conflicts  $N(i)$ . Therefore, as item  $i$  we choose one that maximizes the sum of  $w_i/w_h$  and  $|N(i)|/|N(h)|$ . SIs with  $|S| > 1$  are not used in this *static* approach. Strategies *dynamic* and *stat.dyn* are analog to the other problems. Both strategies are performed with the different separation procedures for the SIs, as described in §4.1. The exact separation of pair inequalities by inspection is indicated by the suffix *.pairs*, whereas the suffixes *.DP* and *.exact* refer to the DP-based and exact separation of SIs, respectively.

As test problems, we used the benchmark instances of Fernandes Muritiba et al. (2010). The complete set comprises 800 instances with differing characteristics regarding the number of items, the capacity, the item lengths, and the number of conflicts of the items. Similar to BP and VC, we include in our computational analysis only those 169 instances for which the solution of the RMP took more than one second of computation time in the *standard* CG. We also distinguish two subgroups of instances. Group 1 comprises instances from the classes 1–4 of the benchmark set, and Group 2 comprises instances from classes 5–8. The latter are instances for which the item weights are such that no three items  $i, j, h$  with  $w_h \geq w_i + w_j$  exist, i.e., no SIs with  $|S| > 1$  are possible. Furthermore, we report separate results for the hardest instances in terms of computation time. This includes 49 instances for which the solution time of *standard* CG was more than 10 seconds.

**Table 5** Computational Results for the Bin Packing Problem with Conflicts

Instances	Algorithm	Solution time (avg./max/min)	# CG Iterations (avg./max/min)	SP/ RMP	# Dynamic DIs (avg./max/min)	Recovery (# inst./max it.)
All instances ( $n = 169$ )	<i>standard</i>	1.00/1.00/1.00	1.00/1.00/1.00	0.1	—	—/—
	<i>static</i>	0.24/0.58/0.03	0.33/0.73/0.06	0.2	—	—/—
	<i>dyn.pairs</i>	0.30/2.71/0.07	0.13/0.59/0.02	0.1	12,281/162,521/328	—/—
	<i>dyn.DP</i>	0.30/1.33/0.08	0.10/0.59/0.01	0.2	16,715/227,607/1,317	0/0
	<i>dyn.exact</i>	2.00/12.13/0.48	0.16/0.60/0.05	5.2	2,968/15,235/494	0/0
	<i>stat + dyn.pairs</i>	0.17/0.63/0.02	0.12/0.55/0.02	0.2	3,294/23,661/35	—/—
	<i>stat + dyn.DP</i>	0.18/0.54/0.02	0.09/0.55/0.01	0.3	7,201/60,899/342	0/0
	<i>stat + dyn.exact</i>	1.35/11.25/0.18	0.11/0.58/0.03	10.2	1,991/15,919/272	0/0
Group 1 ( $n = 92$ )	<i>standard</i>	1.00/1.00/1.00	1.00/1.00/1.00	0.1	—	—/—
	<i>static</i>	0.23/0.48/0.15	0.33/0.64/0.20	0.1	—	—/—
	<i>dyn.pairs</i>	0.29/2.71/0.12	0.09/0.58/0.02	0.1	15,741/162,521/328	—/—
	<i>dyn.DP</i>	0.29/1.33/0.15	0.05/0.58/0.01	0.1	23,886/227,607/1,317	0/0
	<i>dyn.exact</i>	1.01/6.32/0.48	0.12/0.50/0.06	1.4	3,701/15,235/494	0/0
	<i>stat + dyn.pairs</i>	0.16/0.63/0.07	0.08/0.53/0.02	0.1	3,738/23,661/35	—/—
	<i>stat + dyn.DP</i>	0.16/0.53/0.07	0.05/0.47/0.01	0.1	10,914/60,899/342	0/0
	<i>stat + dyn.exact</i>	0.53/5.60/0.18	0.07/0.50/0.03	3.1	2,946/15,919/281	0/0
Group 2 ( $n = 77$ )	<i>standard</i>	1.00/1.00/1.00	1.00/1.00/1.00	0.4	—	—/—
	<i>static</i>	0.25/0.58/0.03	0.34/0.73/0.06	0.8	—	—/—
	<i>dyn.pairs</i>	0.30/0.73/0.07	0.20/0.59/0.03	0.3	8,147/27,207/1,517	—/—
	<i>dyn.DP</i>	0.32/0.79/0.08	0.20/0.59/0.03	0.4	8,147/27,207/1,517	0/0
	<i>dyn.exact</i>	4.48/12.13/0.87	0.24/0.60/0.05	24.9	2,092/4,452/663	0/0
	<i>stat + dyn.pairs</i>	0.19/0.56/0.02	0.19/0.55/0.02	0.5	2,763/10,439/589	—/—
	<i>stat + dyn.DP</i>	0.21/0.54/0.02	0.19/0.55/0.02	0.6	2,763/10,439/589	0/0
	<i>stat + dyn.exact</i>	4.09/11.25/0.41	0.20/0.58/0.03	42.4	850/2,496/272	0/0
Hard instances ( $n = 49$ )	<i>standard</i>	1.00/1.00/1.00	1.00/1.00/1.00	0.1	—	—/—
	<i>static</i>	0.15/0.38/0.03	0.23/0.56/0.06	0.1	—	—/—
	<i>dyn.pairs</i>	0.22/2.71/0.07	0.05/0.42/0.02	0.1	27,629/162,521/1,824	—/—
	<i>dyn.DP</i>	0.20/1.33/0.08	0.03/0.36/0.01	0.1	37,699/227,607/2,852	0/0
	<i>dyn.exact</i>	0.93/2.97/0.48	0.09/0.29/0.05	2.2	5,255/15,235/1,629	0/0
	<i>stat + dyn.pairs</i>	0.10/0.63/0.02	0.04/0.41/0.02	0.1	6,322/23,661/975	—/—
	<i>stat + dyn.DP</i>	0.10/0.32/0.02	0.03/0.37/0.01	0.2	15,838/60,899/1,497	0/0
	<i>stat + dyn.exact</i>	0.46/2.71/0.18	0.06/0.27/0.03	4.6	3,949/15,919/416	0/0

Table 5 summarizes the results for BPC. It reveals that a significant improvement can be achieved by using stabilized CG. A high gain is already obtained from the *static* use of pair inequalities, leading to a speedup of about factor four. The additional, dynamic separation of violated SIs further decreases computation times, resulting in an average speedup of almost factor six over all considered instances. The pure *dynamic* generation of DIs is inferior to the other stabilization strategies. Still, a speedup of approximately factor three compared to *standard* CG can be achieved. For the hard instances, stabilized CG is even more beneficial and leads to an average speedup of factor ten for *stat + dyn.pairs* and *stat + dyn.DP*. The results for Group 1 are also more in favor for the stabilization strategies than those over all instances. There, average speedups exceed factors between four and six for *static* and *stat + dyn*, respectively; for Group 2, speedups reach factors four and five, respectively. Also, Table 5 indicates that for Group 2 the separation of pair inequalities and the DP-based heuristic dynamically generate exactly the same number of DIs.

This can be explained by the fact that no SIs other than pair inequalities exist for these instances.

Regarding the different separation strategies for the dynamic separation of violated DIs, the separation of pair inequalities by inspection and the separation with the DP-based heuristic perform comparably well. It is obvious from Table 5 that the exact separation of violated SIs is too time consuming for BPC and does not pay off.

As for the other problems, the results regarding the number of iterations are similar to those for the computation times: stabilized CG results in a significant reduction of the iterations needed for the optimization of the RMP. Moreover, the dynamic addition of SIs yields an additional, considerable gain compared to using DIs in a static fashion only. Table 5 also reveals that the exact separation of violated SIs does not result in a decrease in the number of iterations compared to the other separation strategies. This can be explained by the use of the KP bound (see §4.1) in the separation and the potential miss of some SIs with a violation of at least 25% of the most violated SI. Indeed, preliminary tests showed that without using

the KP bound the exact separation of violated DIs leads to a slight decrease in the number of iterations relative to the other separation strategies. However, computation times increase disproportionately.

Overall, stabilization with DIs works well for our CG approach to BPC. Although speedups can reach a factor of 100, the most successful strategies *static*, *stat + dyn.pairs*, and *stat + dyn.DP* are never slower than *standard* CG. In contrast, the worst performance of these strategies still results in a speedup of almost factor 1.5. Finally, Table 5 indicates that all SIs that we generated in the test problems were DDOIs, since a recovery was never necessary for any of the instances and strategies.

## 6. Conclusions

Ben Amor et al. (2006) introduced DOIs and DDOIs as a general concept to accelerate the CG process, which was then tested for two well-structured problems, CS and BP. As they state, the restriction of the dual space by DIs leads to fewer possible intermediate values for the dual multipliers, so a dual-optimal solution is computed faster. In this paper, we extend their results in theory, applications, algorithms, and computational results.

On the theoretical side, we generalize the characterization of a system of DDOIs as stated in Proposition 1. Moreover, we introduce several new matrix properties referring to the coefficient matrix of the underlying primal problem. These properties can be checked directly for the problem under consideration. One result is the new comprehensive class of the so-called WSIs, which is shown to constitute DOIs for any problem whose coefficient matrix fulfills the exchange property. A direct consequence is that WSIs also hold for VP, and they generalize the subset inequalities prior known to be DOIs for CS. With the help of the row replacement property, the pair inequalities are proven to form a new class of DDOIs for BP.

On the application side, the same pair inequalities are also valid for VC and BPC. For these two problems, DIs have not been successfully applied before (to the best of our knowledge). Furthermore, DOIs and DDOIs can also be used for problems with a general cost-minimization objective, which is only partly covered in this paper (because of the unit-cost assumption made starting from §3). In a companion paper (Gschwind and Irnich 2014), we show that the CG algorithm of Caprara et al. (2013) for the temporal knapsack problem benefits from the integration of DOIs. We agree with Bianchessi et al. (2014) that the split commodities mixed routing problem is an excellent candidate to apply DIs for a vehicle routing problem. We suspect that an observable acceleration of the branch-and-price algorithm can be gained.

We see the most important algorithmic innovation in the *dynamic* generation of DIs. This strategy can either replace or complement the integration of DIs in the form of additional primal columns into the initial RMP. Several exact and heuristic algorithms for the dynamic generation (=separation) of DIs have been presented. The additional effort for the implementation of separation procedures is small, as shown for CS, BP, and VC. In fact, the most violated WSI in CS and BP are an algorithmic byproduct, since they result from intermediate solutions that must be computed when the knapsack-type pricing problem is solved by DP (currently the best known approach in this case). Even some tailored separation heuristics like the ones proposed for BPC are relatively simple to implement.

The possible over-stabilization of the CG process with potential DDOIs is another new idea first coined and analyzed in this paper. The idea goes hand in hand with the dynamic generation of DIs because WSIs form an exponential class of DIs and can therefore not entirely be added to the initial RMP. On the downside, the dynamic addition of non-DDOIs, i.e., DIs that may cut off the entire dual-optimal polyhedron, requires the use of a recovery algorithm, as the one presented in §4.2. The required iterative application of the recovery algorithm may lead to many additional CG iterations. Fortunately, this never happened in our computational studies. It is indeed reverse; only in rare cases was more than one recovery needed, and most of the time none was required at all. Overall, over-stabilization turns out to be advantageous on average for BP, VC, and BPC. For BP, we show that much larger speedups can result when DIs are added dynamically.

The extensive computational tests on many known and some additional harder benchmark problems allow some clear statements: for all problems analyzed (CS, BP, VC, and BPC), the newly identified and potential DDOIs together reduce the number of CG iterations when added before proving optimality of the RMP, sometimes significantly. Since the LP-reoptimization of the RMP becomes slightly more time consuming with additional DI columns, the overall reduction in computation time is generally smaller than the reduction in CG iterations. However, for all studied problems the average computation times are reduced compared to state-of-the-art CG algorithms. For the BP example, we compare with an implementation already using constraint aggregation as suggested by Ben Amor et al. (2006). Here we are able to confirm their substantial and impressive speedups of factors between two and three for many BP instances. With a mix of statically and dynamically added DIs together with over-stabilization we gain another factor of approximately 2.5 for the extensive benchmark set of Sim and Hart (2013) and a factor of 3.5 for

the widely used benchmark by Scholl et al. (1997). Notably, although for this strategy (*stat + dyn*) the maximum slowdown was by factor 1.3, the maximum speedup was more than factor ten on some instances. Thus, the use of DIs seldom and only gently hampers the CG process, but very often accelerates it. For future applications we expect the dynamic addition of DIs to work best for those problems in which the solution of the pricing almost completely occupies the overall computation time: any reduction in the number of CG iterations should linearly contribute to a reduction in the solution time.

The results presented in this paper all refer to the solution of the linear relaxation of the extensive formulation. Clearly, the same techniques should also be tested when CG is used in branch-and-price. Such a comparison is, however, a nontrivial task: the overall computation times very much depend on the branching scheme applied, which because of the degeneracy of the primal solution may make completely different decisions for the alternative DI strategies, leading to actually incomparable search trees. Even more severe is the fact that some branching decisions are incompatible with some DIs; see, e.g., the branch-and-price algorithm of Alves and Valério de Carvalho (2008) for the multiple length CS. Generic strategies to integrate DIs and branching constitute an interesting avenue for future research.

## References

Alves C, Valério de Carvalho JM (2008) A stabilized branch-and-price-and-cut algorithm for the multiple length cutting stock problem. *Comp. Oper. Res.* 35(4):1315–1328.

Alves C, Valério de Carvalho J, Clautiaux F, Rietz J (2014) Multidimensional dual-feasible functions and fast lower bounds for the vector packing problem. *Eur. J. Oper. Res.* 233(1):43–63.

Balas E, Padberg MW (1976) Set partitioning: A survey. *SIAM Rev.* 18(4):710–760.

Balas E, Xue J (1991) Minimum weighted coloring of triangulated graphs, with application to maximum weight vertex packing and clique finding in arbitrary graphs. *SIAM J. Comput.* 20(2):209–221.

Balinski ML (1965) Integer programming: Methods, uses, computation. *Management Sci.* 12(3):253–313.

Barnhart C, Johnson EL, Nemhauser GL, Savelsbergh MWP, Vance PH (1998) Branch-and-price: Column generation for solving huge integer programs. *Oper. Res.* 46(3):316–329.

Ben Amor H, Desrosiers J, Valério de Carvalho JM (2006) Dual-optimal inequalities for stabilized column generation. *Oper. Res.* 54(3):454–463.

Ben Amor H, Valério de Carvalho J (2005) Cutting stock problems. Desaulniers G, Desrosiers J, Solomon M, eds. *Column Generation* (Springer, New York), 131–161.

Bettinelli A, Cacchiani V, Malaguti E (2014) Bounds and algorithms for the knapsack problem with conflict graph. Technical Report OR-14-16, DEIS—University of Bologna, Bologna, Italy.

Bianchessi N, Archetti C, Speranza MG (2014) An exact solution approach for the split commodities mixed routing problem. Presentation, VeRoLog Conference Oslo, Norway.

Bode C, Irnich S (2012) Cut-first branch-and-price-second for the capacitated arc-routing problem. *Oper. Res.* 60(5):1167–1182.

Caprara A, Toth P (2001) Lower bounds and algorithms for the 2-dimensional vector packing problem. *Disc. Appl. Math.* 111(3):231–262.

Caprara A, Furini F, Malaguti E (2013) Uncommon Dantzig-Wolfe reformulation for the temporal knapsack problem. *INFORMS J. Comput.* 25(3):560–571.

Carraghan R, Pardalos PM (1990) An exact algorithm for the maximum clique problem. *Oper. Res. Lett.* 9(6):375–382.

Dantzig GB, Wolfe P (1960) Decomposition principle for linear programs. *Oper. Res.* 8(1):101–111.

Desaulniers G, Desrosiers J, Ioachim I, Solomon M, Soumis F, Villeneuve D (1998) A unified framework for deterministic time constrained vehicle routing and crew scheduling problems. Crainic T, Laporte G, eds. *Fleet Management and Logistics* (Kluwer Academic Publisher, Boston), 57–93.

Desrosiers J, Gauthier JB, Lübbecke ME (2014) Row-reduced column generation for degenerate master problems. *Eur. J. Oper. Res.* 236(2):453–460.

du Merle O, Villeneuve D, Desrosiers J, Hansen P (1999) Stabilized column generation. *Disc. Math.* 194(1–3):229–237.

Fernandes Muritiba AE, Iori M, Malaguti E, Toth P (2010) Algorithms for the bin packing problem with conflicts. *INFORMS J. Comput.* 22(3):401–415.

Fréville A (2004) The multidimensional 0-1 knapsack problem: An overview. *Eur. J. Oper. Res.* 155(1):1–21.

Garfinkel RS, Nemhauser GL (1969) The set-partitioning problem: Set covering with equality constraints. *Oper. Res.* 17(5):848–856.

Gauthier JB, Desrosiers J, Lübbecke ME (2015) Tools for primal degenerate linear programs: IPS, DCA, and PE. *EURO J. Transportation Logist.*, ePub ahead of print March 27, <http://dx.doi.org/10.1007/s13676-015-0077-5>.

Gilmore PC, Gomory RE (1961) A linear programming approach to the cutting-stock problem. *Oper. Res.* 9(6):849–859.

Gilmore PC, Gomory RE (1963) A linear programming approach to the cutting stock problem—Part II. *Oper. Res.* 11(6):863–888.

Gschwind T, Irnich S (2014) Stabilized column generation for the temporal knapsack problem using dual-optimal inequalities. Technical Report LM-2014-06, Gutenberg School of Management and Economics, Johannes Gutenberg University Mainz, Mainz, Germany.

Gualandi S, Chiarandini M (2014) Graph coloring benchmarks. Accessed January 25, 2016, <https://sites.google.com/site/graphcoloring/home>.

Gualandi S, Malucelli F (2012) Exact solution of graph coloring problems via constraint programming and column generation. *INFORMS J. Comput.* 24(1):81–100.

Held S, Cook W, Sewell EC (2012) Maximum-weight stable sets and safe lower bounds for graph coloring. *Math. Programming Comput.* 4(4):363–381.

Hifi M, Michrafy M (2007) Reduction strategies and exact algorithms for the disjunctively constrained knapsack problem. *Comp. Oper. Res.* 34(9):2657–2673.

Hiriart-Urruty J-B, Lemaréchal C (1993) *Convex Analysis and Minimization Algorithms, Part 2: Advanced Theory and Bundle Methods, Grundlehren der Mathematischen Wissenschaften*, Vol. 306 (Springer, Berlin).

Irnich S, Desaulniers G (2005) Shortest path problems with resource constraints. Desaulniers G, Desrosiers J, Solomon M, eds. *Column Generation* (Springer, New York), 33–65.

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

Lee C, Park S (2011) Chebyshev center based column generation. *Disc. Appl. Math.* 159(18):2251–2265.

Lübbecke ME, Desrosiers J (2005) Selected topics in column generation. *Oper. Res.* 53(6):1007–1023.

Malaguti E, Monaci M, Toth P (2011) An exact approach for the vertex coloring problem. *Disc. Opt.* 8(2):174–190.

Marsten RE, Hogan WW, Blankenship JW (1975) The BOXSTEP method for large-scale optimization. *Oper. Res.* 23(3):389–405.

- Martello S, Toth P (1990) *Knapsack Problems: Algorithms and Computer Implementations*, Wiley Interscience Series in Discrete Mathematics and Optimization (Wiley, New York).
- Mehrotra A, Trick MA (1996) A column generation approach for graph coloring. *INFORMS J. Comput.* 8(4):344–354.
- Méndez-Díaz I, Zabala P (2006) A branch-and-cut algorithm for graph coloring. *Disc. Appl. Math.* 154(5):826–847.
- Östergård PRJ (2002) A fast algorithm for the maximum clique problem. *Disc. Appl. Math.* 120(1–3):197–207.
- Ozden M (1988) A solution procedure for general knapsack problems with a few constraints. *Comp. Oper. Res.* 15(2):145–155.
- Pattillo J, Youssef N, Butenko S (2013) On clique relaxation models in network analysis. *Eur. J. Oper. Res.* 226(1):9–18.
- Pferschy U, Schauer J (2009) The knapsack problem with conflict graphs. *J. Graph Algorithms Appl.* 13(2):233–249.
- Rousseau L-M, Gendreau M, Feillet D (2007) Interior point stabilization for column generation. *Oper. Res. Lett.* 35(5):660–668.
- Sadykov R, Vanderbeck F (2013) Bin packing with conflicts: A generic branch-and-price algorithm. *INFORMS J. Comput.* 25(2):244–255.
- Scholl A, Klein R, Jürgens C (1997) Bison: A fast hybrid procedure for exactly solving the one-dimensional bin packing problem. *Comp. Oper. Res.* 24(7):627–645.
- Sim K, Hart E (2013) Generating single and multiple cooperative heuristics for the one dimensional bin packing problem using a single node genetic programming island model. *Proc. 15th Ann. Conf. Genetic Evolutionary Comput.* (ACM, New York), 1549–1556.
- Valério de Carvalho JM (1998) Exact solution of cutting stock problems using column generation and branch-and-bound. *Internat. Trans. Oper. Res.* 5(1):35–44.
- Valério de Carvalho JM (1999) Exact solution of bin-packing problems using column generation and branch-and-bound. *Ann. Oper. Res.* 86:629–659.
- Valério de Carvalho JM (2005) Using extra dual cuts to accelerate column generation. *INFORMS J. Comput.* 17(2):175–182.
- Vance P, Barnhart C, Johnson E, Nemhauser G (1994) Solving binary cutting stock problems by column generation and branch-and-bound. *Comput. Optim. Appl.* 3(2):111–130.
- Vanderbeck F (1999) Computational study of a column generation algorithm for bin packing and cutting stock problems. *Math. Programming* 86(3):565–594.
- Vanderbeck F (2000) On Dantzig-Wolfe decomposition in integer programming and ways to perform branching in a branch-and-price algorithm. *Oper. Res.* 48(1):111–128.
- Vanderbeck F (2005) Implementing mixed integer column generation. Desaulniers G, Desrosiers J, Solomon M, eds. *Column Generation* (Springer, New York), 331–358.