Two-stage column generation *

Matteo Salani † Ilaria Vacca ‡ Michel Bierlaire ‡

November 30, 2010

Report TRANSP-OR 101130
Transport and Mobility Laboratory
Ecole Polytechnique Fédérale de Lausanne
transp-or.epfl.ch

*This research is supported by the Swiss National Science Foundation Grant 200021-125317.

†Dalle Molle Institute for Artificial Intelligence (IDSIA), Galleria 2, CH-6928 Manno-Lugano, Switzerland. Email: matteo.salani@idsia.ch

‡Transport and Mobility Laboratory, Ecole Polytechnique Fédérale de Lausanne (EPFL), CH-1015 Lausanne, Switzerland. Email: ilaria.vacca@epfl.ch, michel.bierlaire@epfl.ch

Abstract

We introduce a new concept in column generation for handling complex large scale optimization problems, called two-stage column generation, where columns for the compact and extensive formulation are simultaneously generated. The new framework is specifically conceived for tackling complex problems that cannot be efficiently solved by standard column generation and exploits the relationship between compact and extensive formulation. In particular, the concept of extensive reduced cost is introduced in order to estimate the contribution of compact formulation variables to the master problem.

A formal description of the proposed framework is provided and major theoretical issues are discussed. An example based on the Resource Constrained Shortest Path Problem illustrates how two-stage column generation works when the pricing subproblem satisfies or not the integrality property. The two-stage scheme is applied to the Discrete Split Delivery Vehicle Routing Problem and extensive computational experiments are provided to validate our framework. In particular, computational results show that two-stage column generation reduces the number of generated columns and the computational time for complex instances. The transferability of the designed framework across applications and future research directions are further discussed.

1 Introduction

In this paper we consider the exact solution of large-scale combinatorial problems. Combinatorial problems arise in many concrete contexts, such as telecommunication, transportation and logistics, and their complexity often represents a major issue.

Over the last two decades, the Dantzig-Wolfe (DW) decomposition (Dantzig and Wolfe, 1960) and its application to integer programs (Vanderbeck, 2000) has been widely studied and applied to a variety of combinatorial problems.

Column generation (Vanderbeck and Wolsey, 1996; Barnhart et al., 1998; Desrosiers and Lübbecke, 2005) is nowadays one of the most successful tools to solve large-scale integer optimization problems arising in real-world applications. Branch-and-price codes are able to solve problems that commercial MIP solvers could never cope with. However, practical problems of growing size and complexity represent a challenge for the research community and the need of further advances in column generation, both theoretically and algorithmically, is well recognized (Lübbecke and Desrosiers, 2005; Lübbecke, 2010).

In the last decade, different research directions have been explored, aiming to design accelerating techniques for the master and the pricing problem, and to cope with instability issues that affect standard column generation.

Stabilized column generation, introduced by du Merle et al. (1999) and Ben Amor (2002), has been designed to overcome drawbacks such as slow convergence and generation of irrelevant columns in the first iterations (Vanderbeck, 2005). The main reason is the unstable behavior of dual variables. A recent review of different stabilization techniques for column generation and numerical comparison on five applications can be found in Briant et al. (2008).

Dynamic constraint aggregation is a method proposed by Elhallaoui et al. (2005; 2008; 2010) to reduce the master problem size and speed up its solution by aggregating set partitioning constraints. Optimality is guaranteed by dynamically adjusting the set of aggregated constraints. The

crucial point of the method is that aggregated dual master variables need to be disaggregated. The methodology has been recently extended to cope with general degenerate linear programs and is referred to as Improved Primal Simplex (Raymond et al., 2010).

Irnich et al. (2010) have proposed an exact method for *arc variable elimination* based on path reduced costs, in the context of column generation with shortest path pricing subproblems. The technique is based on a well known property in integer programming: if the reduced cost of a non-negative integer variable exceeds a given optimality gap, the variable must be zero in any optimal integer solution (Nemhauser and Wolsey, 1988). The authors investigate the relationship between reduced costs of the compact and the extensive formulation, and extend the method proposed by Walker (1969) to compute reduced costs of original formulation variables starting from a dual feasible solution to the master problem. The technique is applied one the root node has been solved. The method allows to significantly reduce the size of the expanded network underlying the pricing, while keeping optimality.

All the mentioned techniques confirm that good dual information is crucial to enhance column generation schemes. Furthermore, the relationship between compact formulation and column generation is worth investigating, as it may represent an important source of information, as observed by Villeneuve et al. (2005); in particular, the authors study branching rules based on the compact formulation variables.

In this paper a novel framework for complex large-scale optimization problems called *two-stage column generation* is proposed, where columns both for the compact and the extensive formulation are generated simultaneously. The approach is particularly suited for problems where the large number of variables in the compact formulation directly affects the pricing problem and its efficiency. Specifically, this structure is present in the Discrete Split Delivery Vehicle Routing Problem with Time Windows (Salani and Vacca, 2009), where the expanded network of the pricing subproblem depends not only on arcs and customers, but also on discrete orders. This structure is common to other different real-world applications, such as the Tactical Berth Allocation Problem (Giallombardo et al., 2010; Vacca, 2010)

and the Field Technician Scheduling Problem (Xu and Chiu, 2001; Cordeau et al., 2010).

The basic idea of two-stage column generation is the following: we first solve the problem on a subset of compact formulation variables via Dantzig-Wolfe decomposition and column generation. At this point, either profitable compact formulation variables are dynamically generated and added to the formulation; or the current solution is proved to be optimal, in the same spirit of standard column generation. The key point of our approach is that the contribution of compact formulation variables is evaluated with respect to the extensive formulation, in order to take advantage of the constraints that have been "convexified" in the DW reformulation. Indeed, the objective is to add compact formulation variables that are profitable for the master problem, regardless of the optimal solution of the linear relaxation of the compact formulation.

Computational results show that two-stage column generation significantly reduces the number of generated columns to prove optimality of the root node with respect to standard column generation. Suboptimal compact formulation variables are detected correctly and a large percentage of variables do not need to be taken into account during the solution process.

Our new framework differs from *nested column generation*, where basically both the master and the pricing problem are solved via column generation and dual information is available for both problems. This approach was proposed by Vanderbeck (2001) for the 3stage 2dimensional cutting stock problem, and applied to the crude oil tanker routing and scheduling in the context of maritime transport (Hennig, 2010).

The paper is organized as follows. Section 2 recalls the basics of standard column generation. A formal description of the new two-stage column generation framework is provided in Section 3. The overall methodology is illustrated on the Resource Constrained Shortest Path Problem (RCSPP) in Section 4, distinguishing between pricing problems with or without the integrality property. Two-stage column generation is further applied to the Discrete Split Delivery Vehicle Routing Problem with Time Windows (DSDVRPTW) in Section 5. The framework is validated in Section 6 by computational tests on the DSDVRPTW. Computational results for dif-

ficult DSDVRPTW instances are provided in Section 7. Future research directions are discussed in Section 8. Appendix A provides an illustrative example based on the RCSPP, while Appendix B reports the complete computational results for the DSDVRPTW.

2 Standard column generation

In this section the terminology for standard column generation is recalled. Consider the following integer linear program, called the *original* or *compact formulation* (CF):

$$z_{IP} = \min \quad c^T x \tag{1}$$
$$\text{s.t.} \quad Ax \geq b \tag{2}$$
$$Dx \geq d \tag{3}$$
$$x \in \mathbb{Z}^n_+. \tag{4}$$

We assume that the constraints $\{Dx \geq d\}$ present a particular structure that can be "convexified". Let $P = conv\{x \in \mathbb{Z}^n_+ : Dx \geq d\} \neq \emptyset$ be a bounded polyhedron. Each $x \in P$ can be represented as a convex combination of extreme points $\{p_q\}_{\{q \in Q\}}$ of P:

$$x = \sum_{q \in Q} p_q \lambda_q, \ \sum_{q \in Q} \lambda_q = 1, \ \lambda \in \mathbb{R}^{|Q|}_+. \tag{5}$$

The equivalent *extensive formulation* (EF) of (1)–(4) is:

$$z_{IP} = \min \quad \sum_{q \in Q} c_q \lambda_q \tag{6}$$
$$\text{s.t.} \quad \sum_{q \in Q} A_q \lambda_q \geq b \tag{7}$$
$$\sum_{q \in Q} \lambda_q = 1 \tag{8}$$
$$\lambda \geq 0 \tag{9}$$
$$x = \sum_{q \in Q} p_q \lambda_q \tag{10}$$
$$x \in \mathbb{Z}^n_+. \tag{11}$$

where $c_q = c^T p_q$ and $A_q = A p_q$ $\forall q \in Q$. Constraints (10) are usually referred to as *coupling constraints*. By relaxing the integrality of x in (11), coupling constraints also become redundant and the resulting *master problem* (MP) is:

$$z_{MP} = \min \quad \sum_{q \in Q} c_q \lambda_q \tag{12}$$

$$\text{s.t.} \quad \sum_{q \in Q} A_q \lambda_q \geq b \tag{13}$$

$$\sum_{q \in Q} \lambda_q = 1 \tag{14}$$

$$\lambda \geq 0. \tag{15}$$

Typically, the number of variables λ is large and they cannot be explicitly enumerated. Column generation allows for implicit enumeration, and profitable variables are dynamically added to the master problem.

Specifically, a *restricted master problem* (RMP) is repeatedly solved on a subset of variables λ and, at each iteration, negative reduce-cost variables not yet in the formulation are added, if any, by solving the *pricing subproblem*:

$$\min_{q \in Q} \tilde{c}_q := \min_{q \in Q} c_q - \pi A_q - \pi_0, \tag{16}$$

where $\pi \geq 0$ is the dual vector associated with constraints (13), $\pi_0 \in \mathbb{R}$ is the dual variable associated with the convexity constraint (14) and $\tilde{c}_q$ denotes the reduced cost of variable λ_q.

3 Two-stage column generation

In this section, the two-stage column generation framework is formally described and some specific notation and terminology are introduced. Furthermore, methods to evaluate the contribution of compact formulation variables to the master problem are discussed.

3.1 General framework

Let X be the set of compact formulation variables, $|X| = n$. The basic idea of our approach is to start with a subset $\bar{X} \subset X, |\bar{X}| = \bar{n}$ such that the linear relaxation of (CF) is feasible; the problem is reformulated and the master problem is solved via column generation. At this point, either profitable variables in $\widehat{X} := X \setminus \bar{X}$ are dynamically added to the problem, or the current solution is proved to be optimal, in the same spirit of standard column generation.

The clear benefit of this approach is that the associated pricing problem is solved over a smaller set of variables. Furthermore, not all the variables $x_i \in \widehat{X}$ will eventually need to be added.

Without loss of generality and for simplicity of notation we assume that $x = [\,\bar{x}\,|\,\hat{x}\,]$, $c = [\,\bar{c}\,|\,\hat{c}\,]$, $A = [\,\bar{A}\,|\,\widehat{A}\,]$ and $D = [\,\bar{D}\,|\,\widehat{D}\,]$. The *partial compact formulation* (PCF) is defined as follows:

$$\begin{aligned} \bar{z}_{IP} = \min \quad & \bar{c}^T\bar{x} && (17)\\ \text{s.t.} \quad & \bar{A}\bar{x} \geq b && (18)\\ & \bar{D}\bar{x} \geq d && (19)\\ & \bar{x} \in \mathbb{Z}_+^{\bar{n}}. && (20) \end{aligned}$$

We remark that $\bar{z}_{IP} \geq z_{IP}$, since the problem is solved on a subset $\bar{X} \subset X$.

Let $\bar{P} = conv\{\bar{x} \in \mathbb{Z}_+^{\bar{n}} \,|\, \bar{D}\bar{x} \geq d\} \neq \emptyset$. Again, each $\bar{x} \in \bar{P}$ can be represented as a convex combination of extreme points $\{p_q\}_{\{q \in \bar{Q}\}}$ of $\bar{P}$:

$$\bar{x} = \sum_{q \in \bar{Q}} p_q \lambda_q, \;\; \sum_{q \in \bar{Q}} \lambda_q = 1, \;\; \lambda \in \mathbb{R}_+^{|\bar{Q}|}. \tag{21}$$

By substituting $\bar{c}_q = \bar{c}^T p_q$ and $\bar{A}_q = \bar{A} p_q \;\forall q \in \bar{Q}$, the equivalent *partial extensive formulation* (PEF) and its linear relaxation, called the

partial master problem (PMP) can be defined:

$$\bar{z}_{MP} = \min \quad \sum_{q \in \bar{Q}} \bar{c}_q \lambda_q \tag{22}$$

$$\text{s.t.} \quad \sum_{q \in \bar{Q}} \bar{A}_q \lambda_q \geq b \tag{23}$$

$$\sum_{q \in \bar{Q}} \lambda_q = 1 \tag{24}$$

$$\lambda \geq 0. \tag{25}$$

The partial master problem is solved via standard column generation, and the resulting pricing subproblem is:

$$\min_{q \in \bar{Q}} \tilde{c}_q := \min_{q \in \bar{Q}} \bar{c}_q - \pi \bar{A}_q - \pi_0. \tag{26}$$

As soon as $\tilde{c}_q \geq 0$ for all $q \in \bar{Q}$, the current partial master problem is proved to be optimal. However, we still have to determine whether the current partial compact formulation is optimal or not. Therefore, compact formulation variables $x \in \widehat{X}$ are "priced out" in order to identify those that are profitable to be added to the (PCF). Indeed, a correct estimation of the contribution of compact formulation variables to the master problem is necessary to make the overall two-stage column generation methodology consistent; this specific issue is addressed in the next section.

Algorithm 1: Two-stage column generation

input set $\bar{X}$

repeat

  repeat

    CG1: generate extensive variables λ for partial master problem (PMP)

  until *optimal partial master problem (PMP)* ;

  CG2: generate compact variables x for partial compact formulation (PCF)

until *optimal master problem (MP)* ;

A sketch of the two-stage column generation procedure is outlined in Algorithm 1. In the inner loop (denoted by CG1) standard column generation is applied to solve the partial master problem; in particular, the dual optimal vector π is known at every iteration and thus reduced costs $\tilde{c}_q := \bar{c}_q - \pi\bar{A}_q - \pi_0$ of λ variables can be computed exactly; negative reduced-cost columns are provided by the pricing subproblem, if any. In the outer loop (denoted by CG2), compact formulation variables $x_i \in \widehat{X}$ are dynamically added to the partial compact formulation in the same spirit of standard column generation, until optimality is reached.

3.2 Contribution of compact formulation variables

Consider the linear relaxation of the compact formulation (1)-(4) and denote by α and β the dual vectors associated with constraints (2) and (3) respectively. The reduced cost of x is defined by:

$$\tilde{c}_{CF}(x) := c^T - b^T\alpha - d^T\beta. \tag{27}$$

Within the two-stage column generation framework, we refer to $\tilde{c}_{CF}(x)$ as *compact-formulation reduced cost*.

In the scientific literature, a few contributions have investigated the computation of reduced cost of the compact formulation variables in the context of Dantzig-Wolfe decomposition and column generation.

Walker (1969) illustrates a method for computing the reduced cost of compact formulation variables that can be applied only if the pricing problem is solved as a pure linear program, since it makes use of the final tableau to read the dual variables of the pricing.

Poggi de Aragão and Uchoa (2003) propose to explicitly keep the coupling constraints in the master problem, introducing an alternative Dantzig-Wolfe reformulation, called Explicit Master. The reduced costs of compact formulation variables are therefore represented by the dual variables associated with the coupling constraints in the master problem.

Two-stage column generation should determine which compact formulation variables $x \in \widehat{X}$ are worth to be added to (PCF). At first, a rule based on compact reduced cost $\tilde{c}_{CF}(x)$ would appear the most intuitive choice;

however, the information provided by $\tilde{c}_{CF}(x)$ is myopic, since it does not take into account the DW decomposition nor the optimal master problem solution, that is the final goal of our two-stage column generation scheme. Indeed, reduced costs $\tilde{c}_{CF}(x)$ would be negative for all variables x that are part of the optimal solution of the linear relaxation of (CF).

Instead, we are interested in evaluating the contribution of compact formulation variables in the extensive formulation. In order to achieve this goal, the concept of *extensive-formulation reduced cost* is introduced for compact formulation variables, denoted by $\tilde{c}_{EF}(x)$, that guides the column generation process in (CG2) according to the optimal master problem solution.

A recent work by Irnich et al. (2010) investigates the relationship in terms of reduced costs between a compact arc-flow formulation and a path-based DW reformulation. In the context of arc-flow variable elimination, the authors prove that arc-reduced costs can be computed from path-reduced costs and propose an efficient bidirectional search technique to compute path-reduced costs.

In our framework, the method for computing the extensive reduced cost $\tilde{c}_{EF}(x)$ exploits the specific structure of the pricing subproblem. In particular, there are two main classes of subproblems:

- **pricing problem with the integrality property**: when the integrality property holds, the pricing can be solved as a linear program and therefore reduced costs of compact formulation variables can be computed using the method by Walker (1969);
- **pricing problem without the integrality property**: when the integrality property doesn't hold, DW reformulations often present a path-based pricing structure; in this case, the extensive reduced cost $\tilde{c}_{EF}(x)$ of compact formulation variables can be computed using and/or adapting the method by Irnich et al. (2010).

4 Illustration of two-stage column generation

In this section, we illustrate how to apply two-stage column generation and how the contribution of compact formulation variables to the extensive formulation is estimated. In particular, we distinguish between pricing subproblems with integrality property, that can be solved as pure linear programs, and pricing subproblems without the integrality property that present a path-based structure. An illustrative example, adapted from the primer example on resource constrained shortest paths introduced by Desrosiers and Lübbecke (2005), is provided in Appendix A.

4.1 Pricing problem with the integrality property

Consider a network $G(N, A, c, t)$, where $s \in N$ denotes the origin and $t \in N$ the destination. Every arc $(i, j) \in A$ has a cost c_{ij} and a resource consumption t_{ij}. The available amount of resource is denoted by T. The Resource Constrained Shortest Path Problem (RCSPP) aims to find the minimum-cost path that satisfies the resource constraint.

The RCSPP can be formulated as an integer program:

$$z_{IP} = \min \sum_{(i,j)\in A} c_{ij} x_{ij} \tag{28}$$

$$\sum_{j:(s,j)\in A} x_{sj} = 1 \tag{29}$$

$$\sum_{j:(j,i)\in A} x_{ji} - \sum_{j:(i,j)\in A} x_{ij} = 0 \quad \forall i \in A,\ i \neq s, t \tag{30}$$

$$\sum_{i:(i,t)\in A} x_{it} = 1 \tag{31}$$

$$\sum_{(i,j)\in A} t_{ij} x_{ij} \leq T \tag{32}$$

$$x_{ij} = \{0, 1\} \quad \forall (i, j) \in A, \tag{33}$$

where x_{ij} is a binary decision variable, equal to 1 if arc (i, j) is in the path and 0 otherwise. Formulation (28)-(33) represents our compact formulation.

The Dantzig-Wolfe reformulation proposed by Desrosiers and Lübbecke (2005) relies on the convexification of constraints (29)-(31). Consider $X = \{x_{ij} \text{ binary } : (29) - (31)\}$. An extreme point x_p of the polytope defined by the convex hull of X corresponds to a path $p \in P$ in the network (Ahuja et al., 1993). Therefore, any arc-flow variable x_{ij} can be expressed as a convex combination of extreme points of P:

$$x_{ij} = \sum_{p \in P} x_{ijp} \lambda_p \, \forall (i,j) \in A, \qquad \sum_{p \in P} \lambda_p = 1, \qquad \lambda_p \geq 0 \, \forall p \in P,$$

where λ_p represents the amount of flow on path $p \in P$ and x_{ijp} is a coefficient equal to 1 if arc $(i,j) \in A$ belongs to path $p \in P$.

Now, by relaxing the integrality requirements on x_{ij}, the coupling constraints $x_{ij} = \sum_{p \in P} x_{ijp} \lambda_p$ become redundant and the resulting master problem is:

$$z_{MP} = \min \sum_{p \in P} c_p \lambda_p \tag{34}$$

$$\sum_{p \in P} t_p \lambda_p \leq T \tag{35}$$

$$\sum_{p \in P} \lambda_p = 1 \tag{36}$$

$$\lambda_p \geq 0 \quad \forall p \in P, \tag{37}$$

where $c_p = \sum_{(i,j) \in A} c_{ij} x_{ijp}$ is the cost associated with path $p \in P$.

Dual variables associated with constraints (35) and (36) are denoted by π_T and π_0 respectively. The pricing subproblem is formulated as a Shortest Path Problem:

$$\min \sum_{(i,j) \in A} (c_{ij} - \pi_T t_{ij}) x_{ij} - \pi_0 \tag{38}$$

$$- \sum_{j:(s,j) \in A} x_{sj} = -1 \tag{39}$$

$$\sum_{j:(j,i) \in A} x_{ji} - \sum_{j:(i,j) \in A} x_{ij} = 0 \quad \forall i \neq s,t \tag{40}$$

$$\sum_{i:(i,t) \in A} x_{it} = 1 \tag{41}$$

$$x_{ij} = \{0,1\} \quad \forall (i,j) \in A. \tag{42}$$

We remark that formulation (38)-(42) has the *integrality property*: it means that even the optimal solution of the linear relaxation of (38)-(42) is an integer solution. Indeed, the shortest path problem can be expressed as a transshipment problem with one origin and one destination: by shipping one unit from the origin to the destination, the solution determines the shortest path throughout the network.

Dual variables associated with the pricing constraints (39)-(41) are denoted by μ_s, μ_i and μ_t respectively.

4.1.1 Reduced costs via Walker's method

In this example, the integrality property holds and the pricing is solved as a pure linear program. Therefore, the (CF) linear relaxation and the master problem provide the same optimal solution: it means that, in this special case, the contribution of compact formulation variables to the master problem corresponds to the compact reduced cost of variable x_{ij}, i.e., $\tilde{c}_{CF}(x_{ij}) \equiv \tilde{c}_{EF}(x_{ij})$.

In this case, the compact reduced cost $\tilde{c}_{CF}(x_{ij})$ can be computed exactly using the method proposed by Walker (1969). This method applies when the pricing problem is linear and it computes exactly the reduced cost of a variable x_{ij} in the compact formulation given an optimal dual solution to the master problem and an optimal dual solution to the pricing subproblem.

The method is briefly recalled. Consider the linear relaxation of (28)-(33) and denote by α_s, α_i, α_t and α_T the dual variables associated with constraints (29), (30), (31) and (32) respectively. The dual problem of the linear relaxation of the compact formulation is:

$$z_{DP} = \max \alpha_t - \alpha_s + T\alpha_T \tag{43}$$

$$\alpha_j - \alpha_i + t_{ij}\alpha_T \leq c_{ij} \quad \forall (i,j) \in A \tag{44}$$

$$\alpha_i \in \mathbb{R} \quad \forall i \in N \tag{45}$$

$$\alpha_T \leq 0. \tag{46}$$

Walker proves that an optimal solution to (43)-(46) is given by $(\mu_s^*, \mu_i^*, \mu_t^*, \pi_T^*)$, where $(\mu_s^*, \mu_i^*, \mu_t^*)$ is an optimal dual solution to the pricing problem and

π_T^* is the optimal dual solution to the master problem associated with constraint (35). Furthermore, the reduced cost $\tilde{c}_{CF}(x_{ij})$ of x_{ij} can be computed as:

$$\tilde{c}_{CF}(x_{ij}) = c_{ij} + \mu_i - \mu_j - t_{ij}\pi_T. \tag{47}$$

Two-stage column generation We start solving the problem on a reduced network $G(N, \bar{A}, c, t)$ such that subset $\bar{A} \subset A$ ensures that a feasible solution exists. The set of arcs that are not taken into account in the solution process is denoted by $\widehat{A}$. The associated formulation is the *partial compact formulation*.

At every iteration, the reduced cost $\tilde{c}_{CF}(x_{ij})$ of variables x_{ij} not in the formulation, associated with arcs $(i, j) \in \widehat{A}$, is computed using Walker's method.

If $\tilde{c}_{CF}(x_{uv}) < 0$ for some arc $(u, v) \in \widehat{A}$, the corresponding "column" x_{uv} is added to the partial compact formulation and $\bar{A} = \bar{A} \cup (u, v)$; otherwise, the current partial compact formulation is proved to be optimal.

Example An illustrative example of how two-stage column generation works when the pricing problem satisfies the integrality property is provided in Appendix A.1.

4.2 Pricing problem without the integrality property

Consider the Resource Constrained Shortest Path Problem introduced in the previous section, with the addition of one resource, that is capacity Q. The compact formulation therefore presents an additional resource con-

straint and becomes:

$$z_{IP} = \min \sum_{(i,j)\in A} c_{ij}x_{ij} \tag{48}$$

$$\sum_{j:(s,j)\in A} x_{sj} = 1 \tag{49}$$

$$\sum_{j:(j,i)\in A} x_{ji} - \sum_{j:(i,j)\in A} x_{ij} = 0 \quad \forall i \in A,\ i \neq s,t \tag{50}$$

$$\sum_{i:(i,t)\in A} x_{it} = 1 \tag{51}$$

$$\sum_{(i,j)\in A} q_{ij}x_{ij} \leq Q \tag{52}$$

$$\sum_{(i,j)\in A} t_{ij}x_{ij} \leq T \tag{53}$$

$$x_{ij} = \{0,1\} \quad \forall (i,j) \in A. \tag{54}$$

Since we are interested in obtaining a pricing problem without the integrality property, we propose to "convexify" the set of constraints (49)-(52), i.e., resource T is handled in the master problem and resource Q is handled in the pricing subproblem. Although this is not the appropriate way to convexify the resource constraints, since they would typically be handled together in the pricing subproblem (combinatorial algorithms such as dynamic programming are particularly suited to take into account resources), this decomposition is introduced for illustration purposes.

The formulation of the master problem is unchanged with respect to the previous example:

$$z_{MP} = \min \sum_{p\in P} c_p\lambda_p \tag{55}$$

$$\sum_{p\in P} t_p\lambda_p \leq T \tag{56}$$

$$\sum_{p\in P} \lambda_p = 1 \tag{57}$$

$$\lambda_p \geq 0 \quad \forall p \in P. \tag{58}$$

On the contrary, the pricing subproblem now takes into account resource Q and is therefore formulated as a Resource Constrained Shortest Path Problem:

$$\min \sum_{(i,j)\in A} (c_{ij} - \pi_T t_{ij}) x_{ij} - \pi_0 \tag{59}$$

$$- \sum_{j:(s,j)\in A} x_{sj} = -1 \tag{60}$$

$$\sum_{j:(j,i)\in A} x_{ji} - \sum_{j:(i,j)\in A} x_{ij} = 0 \quad \forall i \neq s,t \tag{61}$$

$$\sum_{i:(i,t)\in A} x_{it} = 1 \tag{62}$$

$$\sum_{(i,j)\in A} q_{ij} x_{ij} \leq Q \tag{63}$$

$$x_{ij} = \{0,1\} \quad \forall (i,j) \in A. \tag{64}$$

4.2.1 Reduced costs via Irnich's method

The pricing subproblem is based on shortest paths, therefore the contribution $\tilde{c}_{EF}(x_{ij})$ of compact formulation variables x_{ij} to master problem can be estimated using the method by Irnich et al. (2010).

The reduced cost of path $p \in P$ is denoted by $\tilde{c}_p$ and the set of all paths that use arc $(i,j) \in A$ is denoted by $\mathcal{F}_{ij}$. In other words $\mathcal{F}_{ij} = \{p \in P : x_{ijp} = 1\}$.

According to Irnich et al. (2010), the extensive reduced cost $\tilde{c}_{EF}(x_{ij})$ of arc-flow variable x_{ij} can be estimated as follows:

$$\tilde{c}_{EF}(x_{ij}) = \min_{p\in\mathcal{F}_{ij}} \tilde{c}_p. \tag{65}$$

In other words, $\tilde{c}_{EF}(x_{ij})$ is given by the minimum reduced cost of any path passing by arc (i,j). If $\mathcal{F}_{ij} = \{\emptyset\}$, i.e., if there is no path using arc (i,j), then the contribution of arc (i,j) in the master problem solution is null and therefore $\tilde{c}_{EF}(x_{ij})$ can be set to 0.

Two-stage column generation As in the previous example, the initialization is given by a subset of arcs $\bar{A} \subset A$ such that the problem is feasible. We start solving the problem on the reduced network $G(N, \bar{A}, c, t)$; the set of arcs that are not taken into account in the solution process is denoted by $\widehat{A}$. The associated formulation is the *partial compact formulation*.

At every iteration, the extensive reduced cost $\tilde{c}_{EF}(x_{ij})$ of variables x_{ij} associated with arcs $(i, j) \in \widehat{A}$ not in the formulation is estimated according to (65).

If $\tilde{c}_{EF}(x_{uv}) < 0$ for some arc $(u, v) \in \widehat{A}$, the correspondent "column" x_{uv} is added to the partial compact formulation and $\bar{A} = \bar{A} \cup (u, v)$; otherwise, the current partial compact formulation is proved to be optimal.

Example An illustrative example of how two-stage column generation works when the pricing problem does not satisfy the integrality property is provided in Appendix A.2. In particular, the example clearly shows that extensive reduce costs of compact formulation variables guide the optimization process towards the optimal solution of the master problem, and not towards the optimal solution of the linear relaxation of (CF). Alternative methods to obtain reduced costs of compact formulation variables (e.g. Poggi de Aragão and Uchoa, 2003) do not necessarily lead to the same result.

5 Application to DSDVRPTW

In this section, we illustrate the two-stage column generation framework with the Discrete Split Delivery Vehicle Routing Problem with Time Windows (DSDVRPTW) introduced by Salani and Vacca (2009).

The DSDVRPTW aims to design the optimal set of routes to serve a given set of customers at minimum cost, while respecting constraints on vehicles' capacity and customers' time windows. Each customer can be visited by more than one vehicle since each customer's demand, discretized in items, can be split in orders, i.e., feasible combinations of items.

The arc-flow model presented by the authors represents our compact formulation. They apply Dantzig-Wolfe decomposition and obtain a path-flow

model solved by column generation. In particular, the pricing subproblem is a Resource Constrained Shortest Path Problem that is solved by bidirectional bounded dynamic programming (Righini and Salani, 2006; Righini and Salani, 2008).

Computational results show that the problem is complex. In particular, as soon as the number of orders increase, the problem becomes much more difficult to solve, despite the advanced and sophisticated techniques implemented for accelerating the master and the pricing problem.

The problem structure is particularly suited to two-stage column generation and we attempt to overcome the intrinsic complexity of the problem by exploiting our new framework.

The main issue is represented by the estimation of extensive reduced costs in the CG2 step of the two-stage scheme. Given the structure of the pricing problem for the DSDVRPTW, Irnich et al.'s (2010) method can be adapted to estimate the contribution of compact formulation variables to the extensive formulation. Their technique is based on bidirectional dynamic programming *without* bounding and has been proposed and successfully applied to arc-flow variables. For the specific implementation details, we refer the reader to Irnich et al. (2010) and Irnich (2010).

In order to avoid confusion, the dynamic programming algorithm developed for computing extensive reduced costs is referred to as *CG2 dynamic programming*, as this computation occurs at the CG2 step of two-stage column generation. We propose a *relaxed* version of the CG2 DP, mainly motivated by efficiency reasons and reduction of computational effort. Specifically, the elementarity requirement on paths is removed and a RCSPP is solved instead of a RCESPP; as a consequence, we obtain a lower bound to the RCESPP, that is still valid and consistent with our method.

The contribution of variables x_{ij} associated with arcs $(i,j) \in E$ to the master problem solution is computed as follows. Let $\tilde{c}_p$ be the reduced cost of route $p \in P$ as defined in Salani and Vacca (2009), and let $\mathcal{F}_{ij} \subseteq P$ be the set of all routes that make use of arc $(i,j) \in E$. The extensive reduced cost of variable x_{ij} is estimated by:

$$\tilde{c}_{EF}(x_{ij}) = \min_{p \in \mathcal{F}_{ij}} \tilde{c}_p, \tag{66}$$

that is the minimum reduced cost among any route passing by arc (i, j).

Two-stage column generation is also applied to another type of compact formulation variables, namely order-selection variables y_c associated with orders $c \in C$. The method proposed by Irnich et al. (2010) for arc-flow variables is extended to handle variables y_c: the extension requires to modify the domination rule in the CG2 dynamic programming, in order to obtain a minimum reduced cost path for every arc $(i, j) \in E$ and for every order $c \in C$.

The extensive reduced cost of y_c is computed as follows. Let $\mathcal{F}_c \subseteq P$ be the set of all routes that deliver order $c \in C$. The reduced cost of variable y_c is estimated as:

$$\tilde{c}_{EF}(y_c) = \min_{p \in \mathcal{F}_c} \tilde{c}_p, \tag{67}$$

that is the minimum reduced cost among any route delivering order c.

In two-stage column generation, we start considering a subset of compact formulation variables $\bar{X} \subset X$ such that a feasible solution exists. The set of variables not yet taken into account in the solution process is denoted by $\widehat{X} = X \setminus \bar{X}$. During two-stage column generation, variables of set $\widehat{X}$ are dynamically added to the problem according to their extensive reduced cost, until no variable with negative reduced cost exists.

At every step of the algorithm, we assume that an upper bound ub (typically obtained from primal heuristics) and a valid lb are available. The quantity $ub - lb$ is referred to as *optimality gap*. According to their status, compact formulation variables are classified in three groups:

- **active variables**: compact formulation variables that are in the formulation; in particular, active variables are either in the formulation since the initialization, or they have been added during the two-stage column generation process;
- **inactive variables**: compact formulation variables that are not in the formulation; inactive variables present a positive extensive reduced cost, therefore they are not taken into account in the formulation;
- **suboptimal variables**: compact formulation variables that are not in the formulation and that are proved to be suboptimal according to

their extensive reduced cost (cf. Irnich et al., 2010).

The algorithm distinguishes among active, inactive and suboptimal variables throughout the whole solution process. A computational analysis on the number of suboptimal variables detected by the two-stage column generation and standard variable elimination is provided in Section 6.2.

6 Validation on the DSDVRPTW

In this section, we present the computational experiments carried out on the Discrete Split Delivery Vehicle Routing Problem. We mainly present a summary of the computational experiments, while detailed computational results are provided in Appendix B.

In this analysis, we focus on the root node and on the optimal master problem solution, and standard column generation is compared to our two-stage scheme. Results are provided in Section 6.1. The validity of our framework is confirmed by the fact that the same lower bound is obtained at the end of the root node; furthermore, by applying two-stage column generation we expect to generate a lower number of columns (namely, only "good" columns) and to significantly speed-up the algorithm.

Two-stage column generation for the DSDVRPTW is implemented in ANSI C and compiled with gcc 4.1.2. All restricted master problems are solved using ILOG CPLEX version 10.2. Computational experience is run under a linux operating system on a 2Ghz Intel processor equipped with 2GB of RAM.

Both exact and relaxed CG2 dynamic programming have been implemented. Exact CG2 DP has been discarded during preliminary tests: the method was not efficient because of the huge computational effort required. All the computational results are obtained using the relaxed version of CG2 dynamic programming, if not differently specified.

Computational tests are performed on instances based on the well-known Solomon's data set (Solomon, 1983); in particular, a subset of the instances generated by Salani and Vacca (2009) for the DSDVRPTW is analyzed. For a complete description of the instances and their generation,

we refer the reader to the paper.

We focus the analysis of 25 customers on classes R1_25_C_100 and C1_25_C_100, in order to deal with the maximum number of orders (scenario C); for 50 customers, we focus on classes R1_50_A_50 and C1_50_A_100, since the increased number of customers makes the instances complex already with scenario A. The time limit is set to one hour for all tests.

Two initialization strategies for the subset of compact formulation variables are considered:

- **opt_master**: $\bar{E}$ and $\bar{C}$ are initialized according to the arcs and orders that belong to columns with positive flow in the optimal master solution;
- **opt_lp**: $\bar{E}$ and $\bar{C}$ are initialized according to the arcs and orders taken with positive value in the optimal solution of the linear relaxation of the compact formulation.

A third initialization based on the optimal basis of the master problem has been discarded during preliminary tests: surprisingly, it was outperformed by opt_master in terms of number of columns and computational time.

In standard column generation, adding one column per iteration is in general not efficient. Different strategies for adding columns in the CG2 step are considered: "$10ord-10arc$", "$50ord-50arc$", "$10ord-100arc$", "$10ord-150arc$", "$50ord-150arc$". For each strategy, a different number of orders and arcs (encoded in the name) are added to the partial compact formulation at every iteration, among those with the most negative extensive reduced cost. A computational comparison of these strategies is provided is Section 6.3. The default strategy is "$10ord-10arc$" if not differently specified.

6.1 Standard vs two-stage column generation

Table 1 provides a comparison between standard column generation and two-stage column generation at the root node. For each tested class we report the total number of instances (nr) and, for each method, the number

of instances solved (sol), the average number of generated columns (cols) and the average computational time in seconds (t).

Table 2 provides additional aggregated information on two-stage column generation when using the opt_master or opt_lp initialization. For every class of instances the average percentage of active orders (%ord) and active arcs (%arc) as well as the average number of CG2 iterations (it) are reported. The average reduction of the total number of columns (%cols) is computed for instances solved by both standard and two-stage column generation within the time limit.

For instances solved by both methods, the number of generated columns is reduced by 68% and 25% for opt_master and opt_lp respectively. This is a very important reduction, especially for the opt_master initialization. Expectedly, the opt_master initialization performs better than opt_lp, also in terms of number of solved instanced and computational time[1]. Furthermore, opt_lp requires on average a higher number of CG2 iterations than opt_master (20 vs 15), reaching slightly higher percentages of active variables at the end of the root node (96% vs 91% for orders and 18% vs 16% for arcs). The different behavior of orders and arc variables is discussed in Section 6.2.

The opt_master initialization constantly reduces the number of columns in all classes and it allows for time savings in some cases; it appears to be particularly successful for instances with 50 customers, where the computational time is significantly decreased with respect to standard column generation. In particular, opt_master requires on average 30 seconds for solving an instance of class C1_50_A_100 versus 501 seconds required by standard CG.

Although our two-stage approach does not yield time savings for easy instances of class R1_25_C_100, we remark that the root node is always efficiently solved by standard CG in less than one minute for this class. On

[1]We remark that for class C1_25_C_100 the number of solved instances is different for standard and two-stage column generation; the aggregated results on the average computational time in Table 1 may therefore be misleading and cannot be directly compared. We suggest the reader to refer to Table 6 that provides detailed results for every instance of the class.

		Standard CG			Two-stage column generation					
					opt_master			opt_lp		
Class	**nr**	**sol**	**cols**	**t**	**sol**	**cols**	**t**	**sol**	**cols**	**t**
R_25_C_100	12	12	3165	18	12	971	88	12	1619	140
C_25_C_100	9	8	3953	596	**7**	881	326	**3**	1683	215
R_50_A_50	12	12	1789	37	12	667	28	12	1690	192
C_50_A_100	9	8	2910	501	8	852	30	8	2894	268

Table 1: *Standard vs two-stage column generation: summary of results for relaxed CG2 dynamic programming.*

	Two-stage column generation							
	opt_master				opt_lp			
Class	**%cols**	**%ord**	**%arc**	**it**	**%cols**	**%ord**	**%arc**	**it**
R_25_C_100	-68%	85%	23%	15	-48%	88%	24%	15
C_25_C_100	-74%	90%	22%	14	-48%	94%	24%	15
R_50_A_50	-62%	92%	9%	15	-5%	100%	11%	27
C_50_A_100	-67%	96%	9%	17	3%	100%	11%	24

Table 2: *Two-stage column generation:* opt_master *vs* opt_lp *initialization.*

the contrary, applying two-stage column generation to the more difficult instances of class C1_25_C_100 is beneficial: the computational time is reduced for some instances, and the result is encouraging, in addition to the substantial reduction of columns constantly obtained by our approach. Unfortunately, for class C1_25_C_100 one instance less than standard column generation is solved within the time limit.

The behavior of the opt_lp initialization is unstable and the number of generated columns is even increased for some instances with 50 customers (cf. Table 7 in Appendix B). The number of solved instances is the same as standard column generation, except for class C1_25_C_100 where only 3 out of 12 instances are solved within the time limit of one hour. Classes R1_25_C_100 and C1_25_C_100 obtain a higher reduction than classes R1_50_A_50 and C1_50_A_100 (-48% vs -1% only, on average). This

result is explained by the fact that scenario C presents a higher number of order partitions per customer than scenario A; two-stage CG has therefore the possibility to keep out of the formulation a higher number of order partitions and only "good" columns are generated. The computational time is on average much higher than opt_master; as expected, it proves to be a bad initialization for our method. However, for instances belonging to class $C1\_50\_A\_100$, the computational time is halved with respect to standard column generation.

To some up, the results are promising, since in addition to the validation of our framework, the number of generated columns is importantly reduced. Still, two-stage column generation may be faster or slower than standard CG, depending on the chosen initialization and on the specific class of instances. Interestingly, time savings become more evident when the size of the instances increases.

6.2 Variable elimination

This section provides a comparison between standard variable elimination (Irnich et al., 2010) and two-stage column generation in terms of detected suboptimal variables at the end of the root node.

Computational results are summarized in Table 3. For every class, the number of solved instances (sol), the average percentage of suboptimal orders ($\%|C_{sub}|$), suboptimal arcs ($\%|E_{sub}|$), inactive orders ($\%|C_{ina}|$) and inactive arcs ($\%|E_{ina}|$), as well as the average computational time (t) are reported. Detailed results are provided in Tables 8 and 9 in Appendix B.

Analyzing the Variable Elimination method, a different behavior between different type of variables is noticed: while suboptimal arcs are easily detected (27% on average), no orders can be proved suboptimal. This may be due to a different behavior of the reduced costs, since arc-flow variables appear in the objective function and order-selection variables do not.

Concerning arc variables, two-stage column generation cannot prove suboptimality as much as standard variable elimination; however, by considering also the inactive variables, the number of variables not taken into account throughout the solution process is higher (84% for opt_master

Class	Variable Elimination				Two-stage CG - opt_master						Two-stage CG - opt_lp					
	sol	%$\|C_{sub}\|$	%$\|E_{sub}\|$	**t**	**sol**	%$\|C_{sub}\|$	%$\|C_{ina}\|$	%$\|E_{sub}\|$	%$\|E_{ina}\|$	**t**	**sol**	%$\|C_{sub}\|$	%$\|C_{ina}\|$	%$\|E_{sub}\|$	%$\|E_{ina}\|$	**t**
R_25_C_100	12	0%	21%	22	12	4%	11%	1%	77%	88	12	3%	8%	1%	75%	140
C_25_C_100	8	0%	26%	624	**7**	0%	10%	0%	78%	326	**3**	0%	6%	1%	76%	215
R_50_A_50	12	0%	26%	38	12	0%	8%	0%	91%	28	12	0%	0%	0%	89%	192
C_50_A_100	7	0%	34%	108	**8**	0%	4%	0%	91%	30	**8**	0%	0%	1%	89%	268

Table 3: *Comparison between variable elimination (Irnich et al., 2010) and two-stage column generation.*

and 83% for opt_lp vs 27% for variable elimination).

Interestingly, two-stage column generation detects suboptimal orders whereas variable elimination cannot (4% for opt_master and 3% for opt_lp for class R1_25_C_100 vs 0% for variable elimination). Furthermore, this percentage is increased up to 9% when considering also the inactive variables.

Two-stage column generation is faster than variable elimination especially on larger size instances. In particular, all instance of class C_50_A_100 are solved by two-stage column generation, whereas variable elimination fails to solve one instance within the time limit.

6.3 Strategies for adding CG2 columns

In previous results, we have seen that strategy "10ord − 10arc" allows for a huge reduction of the number of columns but requires a high number of CG2 iterations and this is computationally expensive. The objective of this section is therefore to analyze different strategies for adding columns in the CG2 step, in order to reduce the number of CG2 iterations and thus the computational effort. A comparison of different strategies is provided by Table 4. Detailed results are provided in Tables 10, 11, 12 and 13 in Appendix B.

Strategy "50ord − 150arc" generally yields the highest reduction in terms of CG2 iterations and computational time, except for class C1_25_C_100 where "10ord − 10arc" goes faster.

Although "50ord − 150arc" generates a little more columns than strategies that add only 10 orders per iteration, the reduction with respect to standard column generation is still significant: for 25 customers, about 66% of reduced columns when using the opt_master initialization, and about 33% with opt_lp; for 50 customers, about 60% of reduced columns when using the opt_master initialization, whereas opt_lp unfortunately increases the number of generated columns by about 20%.

As a general remark, it is clear that the number of CG2 iterations is strongly affected by the chosen strategy, varying from an average of 5 iterations for the "50ord − 150arc" strategy up to an average of 20 iterations

for the "$10ord - 10arc$" strategy.

Using "$50ord - 150arc$", the time reduction for 50 customers is significant, especially when using the opt_master initialization: the average computational time is decreased already for the easier class $R1\_50\_A\_50$, from 37 seconds of standard CG to 12 seconds of two-stage CG with the "$50ord - 150arc$" strategy. Remarkably, for class $C1\_50\_A\_100$, the average computational time is reduced from 501 seconds to about 18 seconds.

Surprisingly, the "bad" LP initialization also yields to substantial savings for class $C1\_50\_A\_100$ when using "$50ord - 150arc$": the average computational time for these difficult instances decreases from 501 seconds to 102 seconds. Unfortunately, for class $R1\_50\_A\_50$ a slight increase is observed.

To sum up, a good guess of the optimal master solution is required for the initialization, in order to obtain the best performance in terms of time and number of columns. Nevertheless, strategy "$50ord - 150arc$" allows a significant reduction of computational time with respect to standard column generation, especially with difficult instances.

7 Computational results for difficult instances

In this section two-stage column generation is tested on difficult instances with 50 customers and up to 350 orders. In particular, we are interested in analyzing how two-stage column generation performs when the number of orders increases. Computational results are summarized in Table 5. Detailed results are provided in Tables 14 and 15 in Appendix B. Given the additional complexity of the test set, the time limit is increased to 10 hours of computation.

Concerning class R1, expectedly, the computational time increases with the problem size, and so the number of generated columns, both for standard and two-stage CG. Time savings are constant when using the opt_master initialization for scenarios A and B, while an increase of computational time for some instances of scenario C is observed; all instances are solved within the time limit. The column reduction is stable for opt_master, that generates about 60% less columns than standard CG. The opt_lp initialization

		Stand.CG			Two-stage column generation														
					10ord-10arc			50ord-50arc			10ord-100arc			10ord-150arc			50ord-150arc		
Class	**nr**	**sol**	**col**	**t**	**sol**	**col**	**t**	**sol**	**col**	**t**	**sol**	**col**	**t**	**sol**	**col**	**t**	**sol**	**col**	**t**
opt_master																			
R1_25_C_100	12	12	3165	18	12	971	88	12	1402	25	12	648	51	12	689	58	12	1129	24
C1_25_C_100	9	8	3953	596	7	881	326	8	1223	872	7	906	823	7	805	720	7	997	610
R1_50_A_50	12	12	1789	37	12	667	28	12	681	9	12	704	16	12	703	19	12	721	12
C1_50_A_100	9	8	2910	501	8	852	30	8	1034	17	8	892	79	8	872	123	8	1082	18
opt_lp																			
R1_25_C_100	12	12	3165	18	12	1619	140	12	3074	72	12	922	84	12	898	78	12	1855	52
C1_25_C_100	9	8	3953	596	3	1683	215	2	3817	146	3	1226	146	3	1180	142	4	2646	939
R1_50_A_50	12	12	1789	37	12	1690	192	12	1989	91	12	1578	97	12	1525	123	12	1970	89
C1_50_A_100	9	8	2910	501	8	2894	268	8	3610	179	8	2438	267	8	2281	213	8	3991	102

Table 4: *Comparison of different strategies for adding columns in the CG2 step.*

		Stand.CG			Two-stage column generation					
					opt_master			opt_lp		
Class	**nr**	**sol**	**cols**	**t**	**sol**	**cols**	**t**	**sol**	**cols**	**t**
R_50_A_50	12	12	1789	37	12	721	12	12	1970	89
R_50_B_50	12	12	3435	647	12	1247	269	12	3099	1582
R_50_C_50	12	12	4801	1937	12	1738	2254	10	3917	6988
C_50_A_50	9	8	2910	501	8	1082	18	8	3991	102
C_50_B_50	9	8	5352	6200	8	1872	1614	6	5423	4467
C_50_C_50	9	5	6638	2335	5	2272	7450	1	7063	2479

Table 5: *Summary of computational results for difficult instances.*

performs differently: under scenario A, a higher number of columns is generated, whereas for scenarios B and C a reduction that increases with the number of orders is noticed. As mentioned, this behavior is explained by the fact that, as soon as a higher number of order partitions is available, two-stage CG has the possibility to keep out of the formulation a higher number of orders and therefore only "good" columns are generated. As remarked in the previous section, the opt_lp initialization does not allow for time savings on these instances; in particular, a lower number of instances can be solved for scenario C within the time limit.

The opt_master initialization shows similar results for class C1: the number of generated columns is decreased on average by 65% and for scenarios A and B the computational time is decreased by a factor of 28 and 4 respectively. Time savings are achieved also for scenario C, with the exception of instance c108_50_C_100, that prevent to show the time reduction in the aggregated results of Table 5. We refer the reader to Table 15 in Appendix B for more detailed results. Remarkably, two-stage column generation appears is beneficial for the more difficult problems.

The opt_lp initialization presents unstable results: while a time reduction is obtained for scenario A, the computational time suddenly increases for scenarios B and C, resulting in a much lower number of solved instances; furthermore, the number of generated columns is always greater than in standard column generation. We attribute this fact to the bad

quality of the initialization provided by the LP relaxation.

To conclude, computational results show that two-stage column generation is a promising approach to solve more and more complex problems. The method allows for an important reduction of the number of generated columns. Furthemore, time savings become more evident when the size of the instances increases.

8 Conclusions

In this paper a novel framework called two-stage column generation is introduced, that is specifically conceived to tackle complex problems that cannot be efficiently solved by standard column generation. The relationship between compact and extensive formulation is studied, with the main objective of exploiting the information provided by the DW reformulation when dealing with compact formulation variables.

The proposed methodology is applied to the Discrete Split Delivery Vehicle Routing Problem with Time Windows and intensive computational experiments are performed in order to validate our new framework and to analyze the effects especially on very complex instances.

Computational results show that two-stage column generation importantly reduces the number of generated columns to prove optimality of the root node. Furthermore, suboptimal compact formulation variables are detected correctly and a large percentage of variables is inactive and therefore not taken into account during the solution process. The additional effort required by our sophisticated approach makes the method competitive in terms of computational time especially for instances of a certain difficulty.

Furthermore, the proposed framework is transferable across applications; in particular, the Tactical Berth Allocation Problem (Giallombardo et al., 2010; Vacca, 2010) presents a structure that is suitable to be solved by two-stage column generation, since the concept of quay crane profiles is very similar to the orders in the DSDVRPTW.

To conclude, two-stage column generation is a promising new approach to investigate more and more complex problems and many aspects are worth being investigated by future research, such as an alternative way to

compute extensive reduced cost for compact formulation variables, different strategies for adding columns in the CG2 step of the algorithm and initializations for the set of active compact variables.

References

Ahuja, R. K., Magnanti, T. L. and Orlin, J. B. (1993). *Network flows: theory, algorithms and applications*, Prentice Hall.

Barnhart, C., Johnson, E. L., Nemhauser, G. L., Savelsbergh, M. W. P. and Vance, P. H. (1998). Branch-and-price: Column generation for solving huge integer programs, *Operations Research* **46**(3): 316–329.

Ben Amor, H. M. (2002). *Stabilization de l'Algorithme de Génération de Colonnes*, PhD thesis, École Polytechnique de Montréal.

Briant, O., Lemaréchal, C., Meurdesoif, P., Michel, S., Perrot, N. and Vanderbeck, F. (2008). Comparison of bundle and classical column generation, *Mathematical Programming* **113**: 299–344.

Cordeau, J. F., Laporte, G., Pasin, F. and Ropke, S. (2010). Scheduling technicians and tasks in a telecommunications company, *Journal of Scheduling* **13**: 393–409.

Dantzig, G. B. and Wolfe, P. (1960). Decomposition principle for linear programs, *Operations Research* **8**: 101–111.

Desrosiers, J. and Lübbecke, M. E. (2005). A primer in column generation, *in* G. Desaulniers, J. Desrosiers and M. Solomon (eds), *Column Generation*, GERAD, chapter 1, pp. 1–32.

du Merle, O., Villeneuve, D., Desrosiers, J. and Hansen, P. (1999). Stabilized column generation, *Discrete Mathematics* **194**: 229–237.

Elhallaoui, I., Desaulniers, G., Metrane, A. and Soumis, F. (2008). Bidynamic constraint aggregation and subproblem reduction, *Computers & Operations Research* **35**(5): 1713 – 1724.

Elhallaoui, I., Metrane, A., Soumis, F. and Desaulniers, G. (2010). Multi-phase dynamic constraint aggregation for set partitioning type problems, *Mathematical Programming* **123**: 345–370.

Elhallaoui, I., Villeneuve, D., Soumis, F. and Desaulniers, G. (2005). Dynamic Aggregation of Set-Partitioning Constraints in Column Generation, *Operations Research* **53**(4): 632–645.

Giallombardo, G., Moccia, L., Salani, M. and Vacca, I. (2010). Modeling and solving the tactical berth allocation problem, *Transportation Research Part B* **44**(2): 232–245.

Hennig, F. (2010). *Optimization in Maritime Transportation: Crude Oil Tanker Routing and Scheduling*, PhD thesis, Norwegian University of Science and Technology.

Irnich, S. (2010). A new branch-and-price algorithm for the traveling tournament problem, *European Journal of Operational Research* **204**(2): 218 – 228.

Irnich, S., Desaulniers, G., Desrosiers, J. and Hadjar, A. (2010). Path reduced costs for eliminating arcs, *Journal on Computing* **22**(2): 297–313.

Lübbecke, M. E. (2010). Column generation, *in* J. J. Cochran (ed.), *Wiley Encyclopedia of Operations Research and Management Science*, John Wiley and Sons, Chichester, UK.

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

Nemhauser, G. L. and Wolsey, L. A. (eds) (1988). *Integer and Combinatorial Optimization*, John Wiley & Sons, New York, N. Y.

Poggi de Aragão, M. and Uchoa, E. (2003). Integer program reformulation for robust branch-and-price algorithms, *Proceedings of Mathematical Programming in Rio: A conference in honour of Nelson Maculan*, pp. 56–61.

Raymond, V., Soumis, F. and Orban, D. (2010). A new version of the improved primal simplex for degenerate linear programs, *Computers & Operations Research* **37**(1): 91 – 98.

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

Righini, G. and Salani, M. (2008). New dynamic programming algorithms for the resource constrained shortest path problem., *Networks* **51**(3): 155–170.

Salani, M. and Vacca, I. (2009). Branch and price for the vehicle routing problem with discrete split deliveries and time windows, *Technical Report TRANSP-OR 091224*, Transport and Mobility Laboratory, Ecole Polytechnique Fédérale de Lausanne.

Solomon, M. (1983). *Vehicle Routing and Scheduling with Time Windows Constraints: Models and Algorithms*, PhD thesis, University of Pennsylvania.

Vacca, I. (2010). *Container terminal management: integrated models and large-scale optimization algorithms*, PhD thesis, Ecole Polytechnique Fédérale de Lausanne.

Vanderbeck, F. (2000). On dantzig-wolfe decomposition in integer programming and ways to perform branching in a branch-and-price algorithm, *Operations Research* **48**(1): pp. 111–128.

Vanderbeck, F. (2001). A nested decomposition approach to a three-stage, two-dimensional cutting-stock problem, *Management Science* **47**(6): pp. 864–879.

Vanderbeck, F. (2005). *Implementing Mixed Integer Column Generation*, Springer-Verlag, pp. 331–358.

Vanderbeck, F. and Wolsey, L. A. (1996). An exact algorithm for ip column generation, *Operations Research Letters* **19**: 151–159.

Villeneuve, D., Desrosiers, J., Lübbecke, M. E. and Soumis, F. (2005). On compact formulations for integer programs solved by column generation, *Annals of Operations Research* **139**(1): 375–388.

Walker, W. (1969). A method for obtaining the optimal dual solution to a linear program using the Dantzig-Wolfe decomposition, *Operations Research* **17**: 368–370.

Xu, J. and Chiu, S. Y. (2001). Effective heuristic procedures for a field technician scheduling problem, *Journal of Heuristics* **7**(5): 495–509.

A Illustrative example

A.1 Pricing with the integrality property

We consider the problem of finding a least-cost path from $s = 1$ to $t = 6$ such that the total traversal time of the path does not exceed $T = 14$ time units. The network is illustrated in Figure 1.

We can enumerate all the paths of this network:

p	**path**	**cost**	**time**
1	1-2-4-6	3	18
2	1-2-5-6	5	15
3	1-2-4-5-6	14	14
4	1-3-5-6	24	8
5	1-3-4-6	16	17
6	1-3-4-5-6	27	13
7	1-3-2-4-6	13	13
8	1-3-2-4-5-6	24	9
9	1-3-2-5-6	15	10

Some paths are not feasible with respect to the resource constraint ($p = 1, 2, 5$). The minimum cost integer solution for the problem is given by path 13246 with cost $z^*_{IP} = 13$ and resource consumption 13.

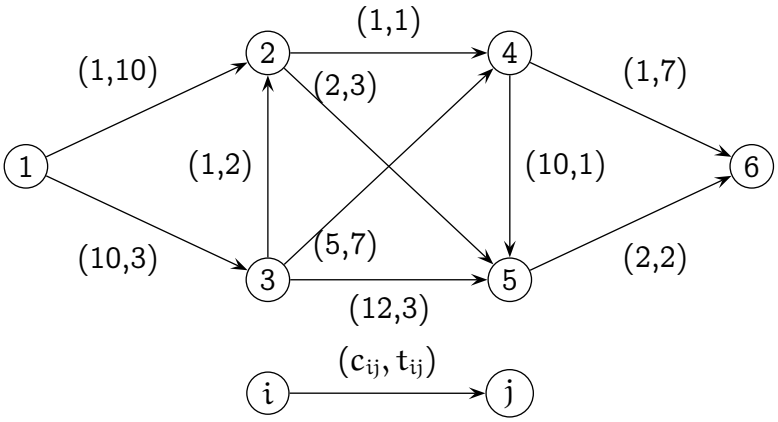

Figure 1: *RCSPP with one resource.*

The optimal value of the linear relaxation of the compact formulation is $z^*_{LP} = 7$ and the optimal fractional solution is:

$$x^*_{12} = 0.8 \qquad x^*_{13} = 0.2 \qquad x^*_{32} = 0.2 \qquad x^*_{25} = 1 \qquad x^*_{56} = 1.$$

A.1.1 Standard column generation

The iterations of standard column generation are reported in the following table:

it	**master obj**	π_T	π_0	**pricing obj**	**added path**
1	100	0	100	-97	1-2-4-6
2	24.555	-5.38889	100	-32.8888	1-3-5-6
3	11.4	-2.1	40.8	-4.8	1-3-2-5-6
4	9	-1.5	30	-2.5	1-2-5-6
5	7	-2	35	0	STOP

The master problem is initialized by an artificial variable y_0 with cost 100. For more details, we refer the reader to Desrosiers and Lübbecke (2005).

Since the integrality property holds, $z^*_{MP} = z^*_{LP} = 7$ and the fractional optimal solutions are equivalent:

$$\lambda^*_{1256} = 0.8 \qquad \lambda^*_{13256} = 0.2.$$

A.1.2 Two-stage CG: contribution of non-optimal arcs

We start solving the problem on the reduced network $G(N, \bar{A}, c, t)$ where $\bar{A} = \{(1,2), (1,3), (2,5), (3,2), (4,5), (4,6), (5,6)\}$.

In this example, the set $\widehat{A} = \{(2,4), (3,4), (3,5)\}$ is composed of arcs that are all *non-optimal* for the linear relaxation of the compact formulation and for the master problem.

We apply column generation to the reduced problem and we solve a (restricted) partial master problem via standard column generation.

The first iteration of CG2 is reported in the following table:

it	master obj	π_T	π_0	pricing obj	added path
1.1	100	0	100	-95	1-2-5-6
1.2	11.3333	-6.333	100	-21.6667	1-3-2-5-6
1.3	7	-2	35	0	STOP

At this point, we have a partial master problem that is optimal. Now, we want to determine whether the current partial compact formulation is optimal. We therefore compute the reduced cost $\tilde{c}_{CF}(x_{ij})$ of compact formulation variables x_{ij} associated with not-yet-considered arcs $(i,j) \in \widehat{A}$ using Walker's method.

The optimal dual solutions associated with the master and pricing problems are:

$$\mu_1^* = -16 \qquad \mu_2^* = 5 \qquad \mu_3^* = 0 \qquad \mu_4^* = 4 \qquad \mu_5^* = 13 \qquad \mu_6^* = 19 \qquad \pi_T^* = -2.$$

Using Walker's procedure we obtain:

$$\begin{aligned}
\tilde{c}_{24} &= c_{24} + \mu_2 - \mu_4 - t_{24}\pi_T = 1 + 5 - 4 - 1 \cdot (-2) = 4, \\
\tilde{c}_{34} &= c_{34} + \mu_3 - \mu_4 - t_{34}\pi_T = 5 + 0 - 4 - 7 \cdot (-2) = 15, \\
\tilde{c}_{35} &= c_{35} + \mu_3 - \mu_5 - t_{35}\pi_T = 12 + 0 - 13 - 3 \cdot (-2) = 5.
\end{aligned}$$

Indeed, $\tilde{c}_{CF}(x_{ij}) \geq 0 \ \forall (i,j) \in \widehat{A}$: it means that the current partial compact formulation is optimal and the two-stage column generation scheme terminates with only one iteration of CG2.

Comparing the inner column generation scheme CG1 to standard column generation, we remark a smaller number of iterations (3 vs 5) and a smaller number of generated columns (2 vs 4). Furthermore, the example shows that non-optimal arcs are detected and not added to the problem.

A.1.3 Two-stage CG: contribution of optimal arcs

We start solving the problem on the reduced network $G(N, \bar{A}, c, t)$ where $\bar{A} = \{A \setminus \{(1,3),(2,5)\}$.

In this example, the set $\widehat{A} = \{(1,3),(2,5)\}$ is composed of arcs that are all *optimal* for the linear relaxation of the compact formulation and for the master problem.

The first iteration of CG2 is reported in the following table:

it	**master obj**	π_T	π_0	**pricing obj**	**added path**
1.1	100	0	100	-97	1-2-4-6
1.2	24.5555	-5.389	100	-10.5555	1-2-4-5-6
1.3	14	-6.143	100	0.00004	STOP

The optimal dual solutions associated with the master and pricing problems are:

$$\mu_1^* = -100 \qquad \mu_2^* = -37.57 \qquad \mu_3^* = 0 \qquad \mu_4^* = -30.43 \qquad \mu_5^* = -14.28 \qquad \mu_6^* = 0.$$

Using Walker's procedure we obtain:

$$\tilde{c}_{CF}(x_{13}) = c_{13} + \mu_1 - \mu_3 - t_{13}\pi_T = 10 - 100 - (0) - 3 \cdot (-6.143) = -71.57,$$
$$\tilde{c}_{CF}(X_{25}) = c_{25} + \mu_2 - \mu_5 - t_{25}\pi_T = 2 - 37.57 - (-14.28) - 3 \cdot (-6.143) = -2.857.$$

We add to the partial compact formulation the variable with the most negative reduced cost, i.e., x_{13} and we iterate. In particular, $\bar{A} = \{\bar{A} \cup (1,3)\}$.

The second iteration of CG2 is reported in the following table:

it	**master obj**	π_T	π_0	**pricing obj**	**added path**
2.1	14	-6.14286	100	-26.857	1-3-5-6
2.2	11.4	-2.1	40.8	-0.5	1-3-2-4-6
2.3	11	-2	39	0	STOP

The optimal dual solutions associated with the master and pricing problems are:

$$\mu_1^* = -16 \qquad \mu_2^* = 5 \qquad \mu_3^* = 0 \qquad \mu_4^* = 8 \qquad \mu_5^* = 18 \qquad \mu_6^* = 23 \qquad \pi_T^* = -2.$$

Using Walker's procedure we obtain:

$$\tilde{c}_{CF}(x_{25}) = c_{25} + \mu_2 - \mu_5 - t_{25}\pi_T = 2 + 5 - 18 - 3 \cdot (-2) = -5.$$

As expected, $\tilde{c}_{CF}(x_{25}) < 0$, therefore variable x_{25} is added to the partial compact formulation and $\bar{A} = \{\bar{A} \cup (2,5)\}$. We remark that, since $\bar{A} \equiv A$,

the third CG2 iteration is the final one, since it corresponds to a run of standard column generation; therefore, two-stage column generation has required an additional computational effort in this special case. However, the example shows that optimal arcs are detected correctly. Furthermore, the number of CG2 iterations may be reduced by implementing smarter strategies for adding columns.

A.1.4 Two-stage CG: simultaneous contribution of optimal and non-optimal arcs

We start solving the problem on the reduced network $G(N, \bar{A}, c, t)$ where $\bar{A} = \{(1,2),(1,3),(3,2),(4,5),(4,6),(5,6)\}\}$.

In this example, the set $\widehat{A} = \{(2,4),(2,5),(3,4),(3,5)\}$ is composed of arc (2,5) that is *optimal* for the linear relaxation of the compact formulation and the master problem, plus three arcs that are *non-optimal*.

Furthermore, the initial network is disconnected. Therefore, the first iteration of CG2 stops in one iteration, with the artificial variable y_0 equal to one in the optimal solution.

it	**master obj**	π_T	π_0	**pricing obj**	**added path**
1.1	100	0	100	0	STOP

The optimal dual solutions associated with the master and pricing problems are:

$$\mu_1^* = 0 \qquad \mu_2^* = 0 \qquad \mu_3^* = 10 \qquad \mu_4^* = 99 \qquad \mu_5^* = 98 \qquad \mu_6^* = 100 \qquad \pi_T = 0.$$

Using Walker's procedure we obtain:

$$\begin{aligned}
\tilde{c}_{CF}(x_{24}) &= c_{24} + \mu_2 - \mu_4 - t_{24}\pi_T = 1 + 0 - 99 - 1 \cdot (0) = -98,\\
\tilde{c}_{CF}(x_{25}) &= c_{25} + \mu_2 - \mu_5 - t_{25}\pi_T = 2 + 0 - 98 - 3 \cdot (0) = -96,\\
\tilde{c}_{CF}(x_{34}) &= c_{34} + \mu_3 - \mu_4 - t_{34}\pi_T = 5 + 10 - 99 - 7 \cdot (0) = -84,\\
\tilde{c}_{CF}(x_{35}) &= c_{35} + \mu_3 - \mu_5 - t_{35}\pi_T = 12 + 10 - 98 - 3 \cdot (0) = -76.
\end{aligned}$$

Therefore, variable x_{24} is added to the partial compact formulation and $\bar{A} = \{\bar{A} \cup (2,4)\}$.

The second iteration of CG2 is reported in the following table:

it	master obj	π_T	π_0	pricing obj	added path
2.1	100	0	100	-97	1-2-4-6
2.2	24.55	-5.38	100	-27.5	1-3-2-4-5-6
2.3	12.33	-2.33	45	-1.67	1-3-2-4-6
2.4	11	-2	39	0	STOP

The optimal dual solutions associated with the master and pricing problems are:

$$\mu_1^* = -16 \qquad \mu_2^* = 5 \qquad \mu_3^* = 0 \qquad \mu_4^* = 8 \qquad \mu_5^* = 20 \qquad \mu_6^* = 23 \qquad \pi_T^* = -2.$$

Using Walker's procedure we obtain:

$$\tilde{c}_{CF}(x_{25}) = c_{25} + \mu_2 - \mu_5 - t_{25}\pi_T = 2 + 5 - 20 - 3 \cdot (-2) = -7,$$
$$\tilde{c}_{CF}(x_{34}) = c_{34} + \mu_3 - \mu_4 - t_{34}\pi_T = 5 + 0 - 8 - 7 \cdot (-2) = 11,$$
$$\tilde{c}_{CF}(x_{35}) = c_{35} + \mu_3 - \mu_5 - t_{35}\pi_T = 12 + 0 - 20 - 3 \cdot (-2) = -2.$$

Therefore, variable x_{25} is added to the partial compact formulation and $\bar{A} = \{\bar{A} \cup (2,5)\}$.

The third iteration of CG2 is reported in the following table:

it	master obj	π_T	π_0	pricing obj	added path
3.1	11	-2	39	-4	1-2-5-6
3.2	8.16	-3.16	52.5	-5.9	1-3-2-5-6
3.3	7	-2	35	0	STOP

The optimal dual solutions associated with the master and pricing problems are:

$$\mu_1^* = -16 \qquad \mu_2^* = 5 \qquad \mu_3^* = 0 \qquad \mu_4^* = 4 \qquad \mu_5^* = 13 \qquad \mu_6^* = 19 \qquad \pi_T^* = -2.$$

Using Walker's procedure we obtain:

$$\tilde{c}_{CF}(x_{34}) = c_{34} + \mu_3 - \mu_4 - t_{34}\pi_T = 5 + 0 - 4 - 7 \cdot (-2) = 15,$$
$$\tilde{c}_{CF}(x_{35}) = c_{35} + \mu_3 - \mu_5 - t_{35}\pi_T = 12 + 0 - 13 - 3 \cdot (-2) = 5.$$

We see that $\tilde{c}_{CF}(x_{ij}) \geq 0 \; \forall (i,j) \in \widehat{A}$: it means that the current partial compact formulation is optimal and the two-stage column generation scheme terminates. We remark that 3 iterations of CG2 have been performed: 2 out of 4 arcs have been added, one optimal and one non-optimal.

A.2 Pricing without the integrality property

We consider the problem of finding a shortest path from $s = 1$ to $t = 6$ such that the total traversal time of the path does not exceed $T = 14$ time units and the total capacity of the path does not exceed $Q = 10$. The network is illustrated in Figure 2.

We can easily enumerate all the paths of this small network:

p	path	cost	time	capacity
1	1-2-4-6	3	18	10
2	1-2-5-6	5	15	10
3	1-2-4-5-6	14	14	13
4	1-3-5-6	24	8	6
5	1-3-4-6	16	17	5
6	1-3-4-5-6	27	13	6
7	1-3-2-4-6	13	13	12
8	1-3-2-4-5-6	24	9	15
9	1-3-2-5-6	15	10	12

Some paths are not feasible with respect to the time and/or capacity constraint. The min-cost integer solution for the problem is given by path 1356 with cost $z^*_{IP} = 24$, time consumption 8 and capacity consumption 6.

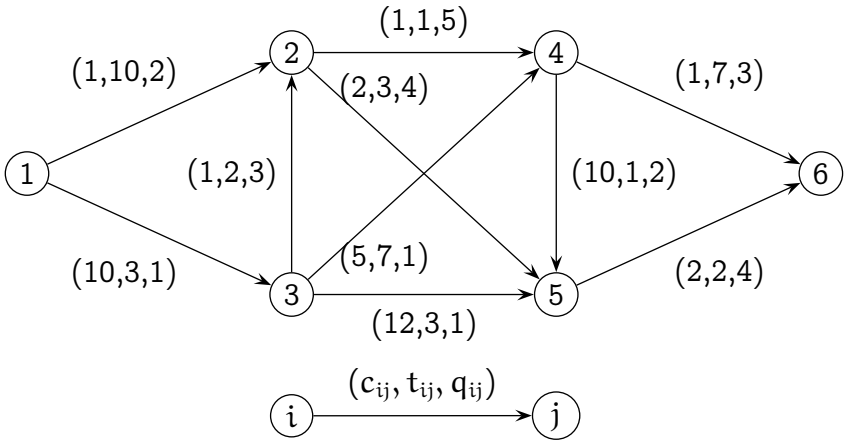

Figure 2: *RCSPP with two resources.*

The optimal value of the linear relaxation of the compact formulation is $z^*_{LP} = 7.2941$ and the optimal fractional solution is:

$$x^*_{12} = 0.82 \qquad x^*_{13} = 0.18 \qquad x^*_{25} = 0.94 \qquad x^*_{32} = 0.12 \qquad x^*_{35} = 0.06 \qquad x^*_{56} = 1,$$

corresponding to paths 1-2-5-6 (flow 0.82), 1-3-2-5-6 (flow 0.12) and 1-3-5-6 (flow 0.06).

A.2.1 Standard column generation

The iterations of standard column generation are reported in the following table:

it	**master obj**	π_0	π_T	**pricing obj**	**added path**
1	100	100	0	-97.00	1-2-4-6
2	24.5	100	-5.39	-32.88	1-3-5-6
3	11.4	40.8	-2.10	-4.3	1-2-5-6
4	7.71	45.7	-2.71	0.0	STOP

The master problem is initialized by an artificial variable y_0 with cost 100. For more details, we refer the reader to Desrosiers and Lübbecke (2005).

The optimal value of the master problem is $z^*_{MP} = 7.71$ and the optimal fractional solution consists of two paths:

$$\lambda^*_{1256} = 0.86 \qquad \lambda^*_{1356} = 0.14.$$

We remark that, since the integrality property does not hold, $z^*_{MP} > z^*_{LP}$.

A.2.2 Two-stage CG: contribution of non-optimal arcs

We start solving the problem on the reduced network $G(N, \bar{A}, c, t)$ where $\bar{A} = \{A \setminus (4,6)\}$.

In this example, arc $(4,6) \in \widehat{A}$ is *non-optimal* both for the linear relaxation of the compact formulation and for the master problem.

We apply column generation to the reduced problem and we solve a (restricted) partial master problem via standard column generation.

The first iteration of CG2 is reported in the following table:

it	master obj	π_0	π_T	pricing obj	added path
1.1	100.00	100.00	0.00	-95.00	1-2-5-6
1.2	11.33	100.00	-6.33	-25.33	1-3-5-6
1.3	7.71	45.71	-2.71	0.00	STOP

The pricing subproblem is solved by enumeration, due to the small size of the network.

At this point, we have a partial master problem that is optimal. Now, we want to determine whether the current partial compact formulation is optimal. We therefore estimate the contribution of arc $(4, 6)$ to the master problem using Irnich et al.'s (2010) method.

All the paths and the associated contributions are reported in the following table:

path	$\tilde{c}_p$	q_p	*note*
1-2-4-6	6.14	10	
1-2-5-6	0.00	10	
1-2-4-5-6	6.29	13	$q_p > Q$
1-3-5-6	0.00	6	
1-3-4-6	16.43	5	
1-3-4-5-6	16.57	6	
1-3-2-4-6	2.57	12	$q_p > Q$
1-3-2-4-5-6	2.71	15	$q_p > Q$
1-3-2-5-6	-3.57	12	$q_p > Q$

We remark that only feasible paths with respect to capacity are considered. According to (65), the extensive reduced cost of x_{46} is given by:

$$\tilde{c}_{EF}(x_{46}) = \min\{\tilde{c}_{12462}, \tilde{c}_{1346}\} = \tilde{c}_{1246} = 6.14. \tag{68}$$

Indeed, $\tilde{c}_{EF}(x_{46}) \geq 0$: it means that the current partial compact formulation is optimal and the two-stage column generation scheme terminates with only one iteration of CG2.

Comparing the inner column generation scheme CG1 to standard column generation, we remark a smaller number of iterations (3 vs 4) and a smaller number of generated columns (2 vs 3). Furthermore, the example shows that non-optimal arcs are detected and not added to the problem.

A.2.3 Two-stage CG: contribution of optimal arcs

We start solving the problem on the reduced network $G(N, \bar{A}, c, t)$ where $\bar{A} = \{A \setminus (5,6)\}$.

In this example, arc $(5,6) \in \widehat{A}$ is *optimal* both for the linear relaxation of the compact formulation and for the master problem.

We apply column generation to the reduced problem and we solve a (restricted) partial master problem via standard column generation.

The first iteration of CG2 is reported in the following table:

it	**master obj**	π_0	π_T	**pricing obj**	**added path**
1.1	100.00	100	0.00	-97.00	1-2-4-6
1.2	24.55	100	-5.39	0.00	STOP

At this point, we have a partial master problem that is optimal. Now, we want to determine whether the current partial compact formulation is optimal. We therefore compute the contribution of arc $(5,6) \in \widehat{A}$ to the master problem using Irnich et al.'s (2010) method.

All the paths and the associated contributions are reported in the following table:

path	$\tilde{c}_p$	q_p	*note*
1-2-4-6	0.00	10	
1-2-5-6	-14.17	10	
1-2-4-5-6	-10.56	13	$q_p > Q$
1-3-5-6	-32.89	6	
1-3-4-6	7.61	5	
1-3-4-5-6	-2.94	6	
1-3-2-4-6	-16.94	12	$q_p > Q$
1-3-2-4-5-6	-27.50	15	$q_p > Q$
1-3-2-5-6	-31.11	12	$q_p > Q$

We remark that only feasible paths with respect to capacity are considered. According to (65), the extensive reduced cost of x_{56} is given by:

$$\tilde{c}_{EF}(x_{56}) = \min\{\tilde{c}_{1256}, \tilde{c}_{1356}, \tilde{c}_{13456}\} = \tilde{c}_{1356} = -32.89. \tag{69}$$

As expected, $\tilde{c}_{EF}(x_{56}) < 0$, therefore variable x_{56} is added to the partial compact formulation and $\bar{A} = \{\bar{A} \cup (5,6)\}$. We remark that, since $\bar{A} \equiv A$, the second CG2 iteration is the final one, since it corresponds to a run of standard column generation; therefore, two-stage column generation has required an additional computational effort in this special case. However, the example shows that optimal arcs are detected correctly.

A.2.4 Two-stage CG: contribution of an arc that is optimal for the relaxed LP and non-optimal for the master problem

We start solving the problem on the reduced network $G(N, \bar{A}, c, t)$ where $\bar{A} = \{A \setminus (3,2)\}$.

In this example, arc $(3,2) \in \widehat{A}$ is *optimal* for the linear relaxation of the compact formulation but is *non-optimal* for the master problem.

We apply column generation to the reduced problem and we solve a (restricted) partial master problem via standard column generation.

The first iteration of CG2 is reported in the following table:

it	**master obj**	π_0	π_T	**pricing obj**	**added path**
1.1	100.00	100.00	0.00	-97.00	1-2-4-6
1.2	24.55	100.00	-5.39	-32.89	1-3-5-6
1.3	11.40	40.80	-2.10	-4.30	1-2-5-6
1.4	7.71	45.71	-2.71	0.00	STOP

At this point, we have a partial master problem that is optimal. Now, we want to determine whether the current partial compact formulation is optimal. We therefore compute the contribution of arc $(3,2) \in \widehat{A}$ to the master problem using Irnich et al.'s (2010) method.

All the paths and the associated contributions are reported in the following table:

path	$\tilde{c}_p$	q_p	*note*
1-2-4-6	51.86	10	
1-2-5-6	45.71	10	
1-2-4-5-6	52.00	13	$q_p > Q$
1-3-5-6	45.71	6	
1-3-4-6	62.14	5	
1-3-4-5-6	62.28	6	
1-3-2-4-6	48.28	12	$q_p > Q$
1-3-2-4-5-6	48.43	15	$q_p > Q$
1-3-2-5-6	42.14	12	$q_p > Q$

No path that make use of arc (3,2) is feasible with respect to the capacity constraint. Therefore, $\mathcal{F}_{32} = \{\emptyset\}$ and the extensive reduced cost is $\tilde{c}_{EF}(x_{32}) = 0.0$.

Since $\tilde{c}_{EF}(x_{32}) \geq 0$, the current partial compact formulation is optimal and the two-stage column generation scheme terminates with only one iteration of CG2.

Remarks The last example is very interesting, since it clearly shows how extensive reduced costs for compact formulation variables are able to guide the optimization towards the optimal solution of the master problem, and not towards the optimal solution of the linear relaxation of (CF).

Remarkably, arc $(3, 2)$ is not added to the partial compact formulation, in spite of the fact that it is optimal in the (CF) linear relaxation. This result is not trivial.

As observed by Desrosiers and Lübbecke (2005), an alternative procedure to obtain reduced costs of compact formulation variables is to directly keep coupling constraints in the formulation (Poggi de Aragão and Uchoa, 2003) and impose $x \geq \epsilon$ for a small $\epsilon > 0$. The shadow prices of these constraints are the reduced costs of x.

Indeed, using this technique for variable x_{32} we obtain a reduced cost of -3.57143: therefore, using this estimation, arc $(3, 2)$ would have been added.

B Detailed computational results

In Sections 6 and 7 we have reported a summary of the computational results that were obtained by standard and two-stage column generation for the DSDVRPTW. In this appendix, we present the corresponding detailed results for each instance that was solved to optimality within the time limit. Tables 6 to 13 are obtained with a time limit of 1 hour, while Tables 14 and 15 are obtained with a time limit of 10 hours.

Tables 6 and 7 provide a comparison between standard column generation and two-stage column generation at the root node for 25 and 50 customers respectively. For standard column generation we report the number of generated columns ($cols$) and the total computational time in seconds (t). For the two-stage framework, two initializations (opt_master and opt_lp) are tested and we report for each one of them the average percentage of active orders ($\%ord$) and active arcs ($\%arc$), the number of iterations of CG2 (it), the total computational time (t_{tot}) and the amount of time spent in the CG2 step (t_{CG2}). The reduction of columns with respect to standard column generation is denoted by $\%_{cols}$. The instances for which the opt_master initialization is not available are denoted by n/a.

Variable elimination is compared to two-stage column generation for instances with 25 and 50 customers in Tables 8 and 9 respectively. For every instance, the total number of orders ($|C|$) and arcs ($|E|$), the number of suboptimal orders ($|C_{sub}|$ and suboptimal arcs ($|E_{sub}|$), the number of inactive orders ($|C_{ina}|$) and inactive arcs ($|E_{ina}|$), as well as the total computational time (t_{tot}) is reported.

Tables 10, 11, 12 and 13 provide a comparison for different strategies for adding CG2 columns.

Finally, computational results for difficult instances of classes R1 and C1 are reported in Tables 14 and 15 respectively. The reduction of number of columns generated the end of the root node is denoted by $\%_{cols}$.

Instance	Stand.CG		Two-stage CG - init:opt_master							Two-stage CG - init:opt_lp						
	cols	t	cols	$\%_{cols}$	it	%ord	%arc	t_{tot}	t_{CG2}	cols	$\%_{cols}$	it	%ord	%arc	t_{tot}	t_{CG2}
R101_25_C_100	990	1	343	-65%	10	48%	6%	1	0	575	-42%	10	50%	7%	1	1
R102_25_C_100	2567	5	653	-75%	12	67%	16%	7	6	1009	-61%	12	78%	19%	8	8
R103_25_C_100	3629	13	1336	-63%	14	85%	20%	39	37	1642	-55%	14	87%	23%	69	66
R104_25_C_100	4021	33	993	-75%	14	87%	23%	100	97	2075	-48%	16	95%	27%	283	276
R105_25_C_100	1475	5	767	-48%	14	83%	17%	4	3	828	-44%	13	84%	17%	4	3
R106_25_C_100	3656	14	785	-79%	14	86%	22%	19	18	1791	-51%	14	85%	21%	39	36
R107_25_C_100	4367	27	923	-79%	18	95%	33%	166	160	2300	-47%	20	94%	34%	296	289
R108_25_C_100	4434	34	1323	-70%	15	92%	24%	317	313	2082	-53%	16	95%	27%	488	482
R109_25_C_100	2916	16	1638	-44%	19	98%	32%	43	37	1859	-36%	21	100%	34%	34	26
R110_25_C_100	2546	24	707	-72%	15	94%	27%	144	140	1433	-44%	15	95%	25%	96	89
R111_25_C_100	3754	23	1401	-63%	19	96%	32%	97	93	2116	-44%	16	99%	28%	151	143
R112_25_C_100	3630	27	778	-79%	14	89%	24%	115	113	1719	-53%	16	95%	26%	211	205
C101_25_C_100	3050	34	939	-69%	11	76%	15%	18	15	1870	-39%	14	90%	22%	100	91
C102_25_C_100	6062	149	997	-84%	14	94%	24%	567	495	x	x	x	x	x	x	x
C103_25_C_100	6787	2412	x	x	x	x	x	x	x	x	x	x	x	x	x	x
C104_25_C_100	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
C105_25_C_100	3420	454	807	-76%	14	91%	19%	85	64	1625	-52%	16	98%	27%	421	396
C106_25_C_100	3222	25	760	-76%	14	84%	16%	30	25	1554	-52%	14	92%	23%	125	116
C107_25_C_100	2979	93	856	-71%	14	90%	22%	192	157	x	x	x	x	x	x	x
C108_25_C_100	3050	142	845	-72%	15	98%	25%	312	252	x	x	x	x	x	x	x
C109_25_C_100	3055	1456	965	-68%	17	99%	31%	1081	736	x	x	x	x	x	x	x

Table 6: *Standard column generation vs two-stage column generation: results for 25 customers.*

Instance	Stand.CG		Two-stage CG - init:opt_master							Two-stage CG - init:opt_lp						
	cols	t	cols	$\%_{cols}$	it	%ord	%arc	t_{tot}	t_{CG2}	cols	$\%_{cols}$	it	%ord	%arc	t_{tot}	t_{CG2}
R101_50_A_50	1035	2	458	-56%	6	85%	5%	1	0	1143	10%	16	100%	7%	2	1
R102_50_A_50	1746	9	607	-65%	9	88%	6%	4	4	1646	-6%	21	100%	8%	17	14
R103_50_A_50	1975	22	655	-67%	22	93%	12%	38	33	1701	-14%	36	100%	15%	93	83
R104_50_A_50	2287	68	793	-65%	22	96%	12%	75	68	2460	8%	33	100%	14%	604	576
R105_50_A_50	1201	6	580	-52%	8	89%	6%	2	1	1181	-2%	18	100%	7%	6	4
R106_50_A_50	1930	16	666	-65%	10	89%	7%	7	6	1442	-25%	21	100%	9%	35	30
R107_50_A_50	1995	33	738	-63%	18	95%	11%	30	26	1867	-6%	32	100%	13%	149	136
R108_50_A_50	2280	83	784	-66%	19	97%	12%	59	51	2393	5%	33	100%	14%	834	801
R109_50_A_50	1392	12	576	-59%	12	92%	7%	6	6	1335	-4%	23	100%	10%	22	18
R110_50_A_50	1475	29	627	-57%	12	94%	8%	15	14	1656	12%	24	100%	10%	87	78
R111_50_A_50	1950	82	810	-58%	24	96%	12%	53	44	1674	-14%	30	100%	12%	102	88
R112_50_A_50	2205	77	704	-68%	17	95%	11%	42	35	1778	-19%	31	100%	13%	356	335
C101_50_A_100	2190	11	929	-58%	10	92%	6%	4	2	1996	-9%	15	100%	6%	10	5
C102_50_A_100	4655	283	909	-80%	19	99%	9%	25	21	3832	-18%	24	100%	10%	150	114
C103_50_A_100	4924	3250	796	-84%	18	93%	10%	55	49	4319	-12%	33	100%	14%	1399	1246
C104_50_A_100	x	x	n/a							x	x	x	x	x	x	x
C105_50_A_100	2232	19	838	-62%	12	96%	7%	8	6	2847	28%	20	100%	9%	28	16
C106_50_A_100	2216	17	824	-63%	12	92%	7%	6	4	2359	6%	18	100%	8%	17	9
C107_50_A_100	2276	26	927	-59%	17	99%	9%	15	12	2065	-9%	21	100%	9%	34	25
C108_50_A_100	2312	84	782	-66%	20	99%	10%	39	29	2599	12%	26	100%	12%	95	80
C109_50_A_100	2475	316	814	-67%	24	97%	12%	85	65	3138	27%	37	100%	16%	416	217

Table 7: *Standard column generation vs two-stage column generation: results for 50 customers.*

Instance	|C|	|E|	Var.Elim. |C_{sub}|	|E_{sub}|	t_{tot}	Two-stage CG - init:opt_master |C_{sub}|	|C_{ina}|	|E_{sub}|	|E_{ina}|	t_{tot}	Two-stage CG - init:opt_lp |C_{sub}|	|C_{ina}|	|E_{sub}|	|E_{ina}|	t_{tot}
R101_25_C_100	175	625	0	349	1	74	17	41	545	1	72	15	50	533	1
R102_25_C_100	175	625	0	190	5	0	57	0	523	7	0	38	4	504	8
R103_25_C_100	175	625	0	100	20	0	26	0	501	39	0	22	2	478	69
R104_25_C_100	175	625	0	53	37	0	22	0	481	100	0	9	1	457	283
R105_25_C_100	175	625	0	275	5	0	29	0	521	4	0	28	6	512	4
R106_25_C_100	175	625	0	150	18	0	25	0	488	19	0	26	3	492	39
R107_25_C_100	175	625	0	77	36	0	9	0	421	166	0	10	1	409	296
R108_25_C_100	175	625	0	37	39	0	14	0	478	317	0	9	0	455	488
R109_25_C_100	175	625	0	173	18	0	4	0	425	43	0	0	2	413	34
R110_25_C_100	175	625	0	64	28	0	11	0	454	144	0	10	1	465	96
R111_25_C_100	175	625	0	80	28	0	7	0	428	97	0	2	0	452	151
R112_25_C_100	175	625	0	0	31	0	19	0	473	115	0	9	0	461	211
C101_25_C_100	175	625	0	236	36	0	42	0	530	18	0	17	4	486	100
C102_25_C_100	175	625	0	119	163	0	11	0	474	567	x	x	x	x	x
C103_25_C_100	175	625	0	52	2470	x	x	x	x	x	x	x	x	x	x
C104_25_C_100	175	625	x	x	x	x	x	x	x	x	x	x	x	x	x
C105_25_C_100	175	625	0	205	461	0	15	0	507	85	0	3	2	455	421
C106_25_C_100	175	625	0	231	29	0	28	0	522	30	0	14	4	479	125
C107_25_C_100	175	625	0	204	166	0	17	0	488	192	x	x	x	x	x
C108_25_C_100	175	625	0	150	163	0	4	0	467	312	x	x	x	x	x
C109_25_C_100	175	625	0	97	1506	0	1	0	433	1081	x	x	x	x	x

Table 8: *Comparison between variable elimination (Irnich et al., 2010) and two-stage column generation: results for 25 customers.*

Instance	|C|	|E|	Var.Elim. $\|C_{sub}\|$	$\|E_{sub}\|$	t_{tot}	Two-stage CG - init:opt_master $\|C_{sub}\|$	$\|C_{ina}\|$	$\|E_{sub}\|$	$\|E_{ina}\|$	t_{tot}	Two-stage CG - init:opt_lp $\|C_{sub}\|$	$\|C_{ina}\|$	$\|E_{sub}\|$	$\|E_{ina}\|$	t_{tot}
R101_50_A_50	150	2500	0	1605	2	0	22	0	2368	1	0	0	17	2319	2
R102_50_A_50	150	2500	0	995	10	0	18	0	2350	4	0	0	13	2276	17
R103_50_A_50	150	2500	0	601	24	0	11	0	2192	38	0	0	5	2121	93
R104_50_A_50	150	2500	0	182	71	0	6	0	2190	75	0	0	1	2146	604
R105_50_A_50	150	2500	0	1326	6	0	16	0	2352	2	0	0	14	2304	6
R106_50_A_50	150	2500	0	794	16	0	17	0	2335	7	0	0	9	2271	35
R107_50_A_50	150	2500	0	462	34	0	8	0	2225	30	0	0	4	2160	149
R108_50_A_50	150	2500	0	128	85	0	4	0	2209	59	0	0	0	2145	834
R109_50_A_50	150	2500	0	851	13	0	12	0	2327	6	0	0	7	2255	22
R110_50_A_50	150	2500	0	399	30	0	9	0	2298	15	0	0	2	2246	87
R111_50_A_50	150	2500	0	415	83	0	6	0	2202	53	0	0	3	2187	102
R112_50_A_50	150	2500	0	1	79	0	7	0	2235	42	0	0	0	2168	356
C101_50_A_100	150	2500	0	1150	12	0	12	0	2348	4	0	0	24	2314	10
C102_50_A_100	150	2500	0	623	284	0	1	0	2267	25	0	0	16	2226	150
C103_50_A_100	150	2500	x	x	x	0	10	0	2254	55	0	0	9	2133	1399
C104_50_A_100	150	2500	x	x	x	n/a					x	x	x	x	x
C105_50_A_100	150	2500	0	1014	19	0	6	0	2333	8	0	0	8	2266	28
C106_50_A_100	150	2500	0	1088	17	0	12	0	2326	6	0	0	19	2286	17
C107_50_A_100	150	2500	0	902	26	0	2	0	2286	15	0	0	7	2260	34
C108_50_A_100	150	2500	0	744	85	0	1	0	2239	39	0	0	2	2209	95
C109_50_A_100	150	2500	0	480	318	0	4	0	2200	85	0	0	0	2095	416

Table 9: *Comparison between variable elimination (Irnich et al., 2010) and two-stage column generation: results for 50 customers.*

	Stand.CG		Two-stage column generation - init:opt_master													
			10ord-10arc			50ord-50arc			10ord-100arc			10ord-150arc			50ord-150arc		
Instance	**cols**	**t**	**cols**	**it**	**t**	**cols**	**it**	**t**	**cols**	**it**	**t**	**cols**	**it**	**t**	**cols**	**it**	**t**
R101_25_C_100	990	1	343	10	1	522	5	1	401	9	1	371	8	0	593	4	1
R102_25_C_100	2567	5	653	12	7	791	4	3	417	8	3	415	8	3	699	4	2
R103_25_C_100	3629	13	1336	14	39	2068	6	22	801	14	26	761	12	29	1560	4	25
R104_25_C_100	4021	33	993	14	100	1741	5	42	731	12	50	775	11	83	1330	5	50
R105_25_C_100	1475	5	767	14	4	1093	5	2	574	11	3	614	10	3	1101	5	2
R106_25_C_100	3656	14	785	14	19	977	5	9	547	13	17	605	14	29	873	5	17
R107_25_C_100	4367	27	923	18	166	1489	5	49	723	15	105	671	14	96	1150	5	44
R108_25_C_100	4434	34	1323	15	317	1575	4	30	844	15	105	930	15	147	1297	4	43
R109_25_C_100	2916	16	1638	19	43	1971	6	17	686	16	36	570	16	23	1214	4	11
R110_25_C_100	2546	24	707	15	144	1332	5	44	604	14	84	651	15	72	1121	4	29
R111_25_C_100	3754	23	1401	19	97	1886	6	34	780	15	92	991	16	92	1346	4	28
R112_25_C_100	3630	27	778	14	115	1373	5	49	664	15	86	913	16	118	1262	4	34
C101_25_C_100	3050	34	939	11	18	1265	5	22	972	12	32	717	10	18	889	4	20
C102_25_C_100	6062	149	997	14	567	1402	7	682	995	14	1579	906	14	847	1438	4	551
C103_25_C_100	6787	2412	x	x	x	x		x	x	x	x	x	x	x	x	x	x
C104_25_C_100	x	x	x	x	x	1441	7	3440	x	x	x	x	x	x	x	x	x
C105_25_C_100	3420	454	807	14	85	1716	5	131	890	14	155	748	14	127	1132	4	57
C106_25_C_100	3222	25	760	14	30	1019	4	17	872	13	35	773	13	44	929	4	15
C107_25_C_100	2979	93	856	14	192	1057	5	142	903	15	884	808	14	245	578	4	289
C108_25_C_100	3050	142	845	15	312	978	6	440	876	15	766	841	15	682	867	4	307
C109_25_C_100	3055	1456	965	17	1081	908	7	2101	836	15	2311	841	15	3076	1149	4	3031

Table 10: *Comparison of strategies for adding columns: 25 customers, initialization with opt master.*

	Stand.CG		Two-stage column generation - init:opt_lp													
			10ord-10arc			50ord-50arc			10ord-100arc			10ord-150arc			50ord-150arc		
Instance	**cols**	**t**	**cols**	**it**	**t**	**cols**	**it**	**t**	**cols**	**it**	**t**	**cols**	**it**	**t**	**cols**	**it**	**t**
R101_25_C_100	990	1	575	10	1	712	4	1	370	7	0	387	10	1	539	5	1
R102_25_C_100	2567	5	1009	12	8	1598	6	5	899	10	5	766	10	4	1576	5	4
R103_25_C_100	3629	13	1642	14	69	3742	6	43	856	13	45	724	13	80	1567	5	38
R104_25_C_100	4021	33	2075	16	283	4495	6	99	1191	14	110	1213	15	84	3883	5	73
R105_25_C_100	1475	5	828	13	4	1333	5	3	681	12	4	648	12	4	1091	5	4
R106_25_C_100	3656	14	1791	14	39	3291	6	26	858	13	19	787	15	31	1701	4	13
R107_25_C_100	4367	27	2300	20	296	3743	6	105	1260	15	218	1090	15	171	2626	4	75
R108_25_C_100	4434	34	2082	16	488	5136	6	319	1317	15	245	1461	15	230	1372	4	185
R109_25_C_100	2916	16	1859	21	34	2584	6	12	707	17	16	706	16	13	1391	5	8
R110_25_C_100	2546	24	1433	15	96	2839	6	56	893	15	55	853	14	81	1513	5	31
R111_25_C_100	3754	23	2116	16	151	3607	6	60	1077	16	139	1134	15	104	2856	5	54
R112_25_C_100	3630	27	1719	16	211	3805	6	130	959	15	151	1007	15	139	2147	4	140
C101_25_C_100	3050	34	1870	14	100	3218	5	65	1195	14	103	1116	13	56	2523	4	46
C102_25_C_100	6062	149	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
C103_25_C_100	6787	2412	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
C104_25_C_100	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
C105_25_C_100	3420	454	1625	16	421	x	x	x	1166	15	201	1133	14	259	2316	4	210
C106_25_C_100	3222	25	1554	14	125	4415	7	226	1316	14	134	1291	13	111	2981	4	102
C107_25_C_100	2979	93	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
C108_25_C_100	3050	142	x	x	x	x	x	x	x	x	x	x	x	x	2762	4	3400
C109_25_C_100	3055	1456	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

Table 11: *Comparison of strategies for adding columns: 25 customers, initialization with opt lp.*

	Stand.CG		Two-stage column generation - init:opt_master													
			10ord-10arc			50ord-50arc			10ord-100arc			10ord-150arc			50ord-150arc		
Instance	**cols**	**t**	**cols**	**it**	**t**	**cols**	**it**	**t**	**cols**	**it**	**t**	**cols**	**it**	**t**	**cols**	**it**	**t**
R101_50_A_50	1035	2	458	6	1	502	4	1	469	5	1	462	5	1	479	4	1
R102_50_A_50	1746	9	607	9	4	693	4	2	699	6	4	716	6	5	663	4	3
R103_50_A_50	1975	22	655	22	38	771	6	10	759	6	14	707	6	12	753	4	9
R104_50_A_50	2287	68	793	22	75	721	6	20	761	6	32	779	6	31	771	5	24
R105_50_A_50	1201	6	580	8	2	565	5	1	561	7	2	592	5	2	599	4	1
R106_50_A_50	1930	16	666	10	7	652	4	3	655	6	6	649	7	7	676	5	6
R107_50_A_50	1995	33	738	18	30	706	6	10	728	6	13	736	6	18	759	3	9
R108_50_A_50	2280	83	784	19	59	717	5	19	829	6	38	865	6	66	874	3	29
R109_50_A_50	1392	12	576	12	6	615	5	3	611	6	5	573	6	5	581	5	4
R110_50_A_50	1475	29	627	12	15	710	5	8	770	6	11	743	7	15	753	4	11
R111_50_A_50	1950	82	810	24	53	844	7	18	923	9	36	870	8	39	976	5	23
R112_50_A_50	2205	77	704	17	42	670	6	17	682	6	28	746	6	28	763	4	23
C101_50_A_100	2190	11	929	10	4	889	4	3	845	8	8	787	8	10	1000	4	5
C102_50_A_100	4655	283	909	19	25	1161	6	14	1055	9	16	1020	10	81	1381	5	19
C103_50_A_100	4924	3250	796	18	55	1288	6	22	1015	10	286	945	10	370	1359	4	21
C104_50_A_100	x	x	n/a			n/a			n/a			n/a			n/a		
C105_50_A_100	2232	19	838	12	8	889	6	5	899	9	23	815	10	20	922	4	7
C106_50_A_100	2216	17	824	12	6	969	6	5	748	9	26	852	9	19	868	4	5
C107_50_A_100	2276	26	927	17	15	1344	8	14	885	10	32	934	10	37	1256	6	15
C108_50_A_100	2312	84	782	20	39	860	6	16	913	9	68	783	9	127	894	4	18
C109_50_A_100	2475	316	814	24	85	870	7	54	779	10	173	839	10	320	977	5	52

Table 12: *Comparison of strategies for adding columns: 50 customers, initialization with opt master.*

	Stand.CG		Two-stage column generation - init:opt_lp													
			10ord-10arc			50ord-50arc			10ord-100arc			10ord-150arc			50ord-150arc		
Instance	cols	t	cols	it	t	cols	it	t	cols	it	t	cols	it	t	cols	it	t
R101_50_A_50	1035	2	1143	16	2	1219	7	2	887	10	2	837	9	2	1059	4	1
R102_50_A_50	1746	9	1646	21	17	1911	9	9	1412	10	12	1491	10	13	1944	5	9
R103_50_A_50	1975	22	1701	36	93	2089	10	51	1801	11	53	1740	10	59	2287	6	39
R104_50_A_50	2287	68	2460	33	604	2657	11	366	1982	10	337	1805	10	506	2457	7	308
R105_50_A_50	1201	6	1181	18	6	1343	7	3	1007	10	4	1002	10	5	1345	5	4
R106_50_A_50	1930	16	1442	21	35	2231	9	16	1703	10	22	1554	10	23	1896	6	16
R107_50_A_50	1995	33	1867	32	149	2057	10	65	1792	10	78	1752	10	86	2373	6	96
R108_50_A_50	2280	83	2393	33	834	2690	11	407	1998	11	423	1939	10	394	2521	7	382
R109_50_A_50	1392	12	1335	23	22	1601	9	11	1339	10	15	1187	9	13	1587	6	11
R110_50_A_50	1475	29	1656	24	87	1865	10	31	1456	11	37	1532	10	43	1775	5	25
R111_50_A_50	1950	82	1674	30	102	2289	10	49	1729	11	70	1702	11	121	2330	7	81
R112_50_A_50	2205	77	1778	31	356	1917	10	83	1826	11	115	1761	10	214	2065	6	99
C101_50_A_100	2190	11	1996	15	10	2337	6	8	1946	11	24	1535	11	18	3270	6	13
C102_50_A_100	4655	283	3832	24	150	5265	10	111	2561	10	124	2776	10	216	4673	7	91
C103_50_A_100	4924	3250	4319	33	1399	5344	11	958	3449	11	1503	3260	10	804	5230	7	383
C104_50_A_100	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
C105_50_A_100	2232	19	2847	20	28	3636	8	24	2033	10	25	2100	10	38	3234	6	23
C106_50_A_100	2216	17	2359	18	17	2775	7	11	1725	10	15	1961	11	26	2651	5	11
C107_50_A_100	2276	26	2065	21	34	2719	8	23	1895	10	32	1583	10	46	3963	6	31
C108_50_A_100	2312	84	2599	26	95	3077	9	64	2424	11	75	2162	10	112	3845	6	47
C109_50_A_100	2475	316	3138	37	416	3725	10	236	3472	11	336	2870	11	441	5059	7	214

Table 13: *Comparison of strategies for adding columns: 50 customers, initialization with opt lp.*

Instance	Stand.CG cols	t	2stage - init:opt_master cols	$\%_{cols}$	it	t	2stage - init:opt_lp cols	$\%_{cols}$	it	t
R101_50_A_50	1035	2	479	-54%	4	1	1059	2%	4	1
R102_50_A_50	1746	9	663	-62%	4	3	1944	11%	5	9
R103_50_A_50	1975	22	753	-62%	4	9	2287	16%	6	39
R104_50_A_50	2287	68	771	-66%	5	24	2457	7%	7	308
R105_50_A_50	1201	6	599	-50%	4	1	1345	12%	5	4
R106_50_A_50	1930	16	676	-65%	5	6	1896	-2%	6	16
R107_50_A_50	1995	33	759	-62%	3	9	2373	19%	6	96
R108_50_A_50	2280	83	874	-62%	3	29	2521	11%	7	382
R109_50_A_50	1392	12	581	-58%	5	4	1587	14%	6	11
R110_50_A_50	1475	29	753	-49%	4	11	1775	20%	5	25
R111_50_A_50	1950	82	976	-50%	5	23	2330	19%	7	81
R112_50_A_50	2205	77	763	-65%	4	23	2065	-6%	6	99
R101_50_B_50	1824	9	793	-57%	3	2	1553	-15%	5	6
R102_50_B_50	3783	55	1140	-70%	4	24	3354	-11%	6	100
R103_50_B_50	3809	182	1356	-64%	5	112	3427	-10%	7	748
R104_50_B_50	4429	1356	1320	-70%	5	1006	4277	-3%	7	5954
R105_50_B_50	2248	24	1022	-55%	5	8	1786	-21%	7	17
R106_50_B_50	3527	102	1290	-63%	5	31	2969	-16%	5	291
R107_50_B_50	3791	290	1360	-64%	5	119	3551	-6%	7	1444
R108_50_B_50	4570	4550	1378	-70%	6	780	3932	-14%	7	7624
R109_50_B_50	2364	73	1175	-50%	5	33	2382	1%	6	138
R110_50_B_50	2858	208	1255	-56%	5	174	3083	8%	8	287
R111_50_B_50	3556	488	1415	-60%	5	430	3553	0%	7	906
R112_50_B_50	4457	425	1455	-67%	5	511	3315	-26%	7	1473
R101_50_C_50	2727	27	1112	-59%	5	8	2127	-22%	7	18
R102_50_C_50	4837	166	1683	-65%	5	181	4007	-17%	7	870
R103_50_C_50	5287	1051	1772	-66%	6	3432	5432	3%	7	13227
R104_50_C_50	6065	5079	2279	-62%	6	7655	x		x	x
R105_50_C_50	3041	72	1291	-58%	6	37	2334	-23%	7	78
R106_50_C_50	4895	453	1773	-64%	5	209	4792	-2%	7	2073
R107_50_C_50	5196	1814	1798	-65%	6	1569	5081	-2%	8	17298
R108_50_C_50	6242	2924	1955	-69%	6	6218	x		x	x
R109_50_C_50	3726	556	1543	-59%	5	151	3513	-6%	8	572
R110_50_C_50	4228	1256	1841	-56%	6	1870	2686	-36%	8	2371
R111_50_C_50	5430	6043	1947	-64%	6	3565	4724	-13%	7	16761
R112_50_C_50	5942	1485	1863	-68%	6	5747	4472	-25%	8	16611

Table 14: *Computational results for class R1, 50 customers, scenarios A, B and C.*

Instance	Stand.CG cols	Stand.CG t	2stage - init:opt_master cols	2stage - init:opt_master $\%_{cols}$	2stage - init:opt_master it	2stage - init:opt_master t	2stage - init:opt_lp cols	2stage - init:opt_lp $\%_{cols}$	2stage - init:opt_lp it	2stage - init:opt_lp t
C101_50_A_100	2190	11	1000	-54%	4	5	3270	49%	6	13
C102_50_A_100	4655	283	1381	-70%	5	19	4673	0%	7	91
C103_50_A_100	4924	3250	1359	-72%	4	21	5230	6%	7	383
C104_50_A_100	x	x	n/a				x		x	x
C105_50_A_100	2232	19	922	-59%	4	7	3234	45%	6	23
C106_50_A_100	2216	17	868	-61%	4	5	2651	20%	5	11
C107_50_A_100	2276	26	1256	-45%	6	15	3963	74%	6	31
C108_50_A_100	2312	84	894	-61%	4	18	3845	66%	6	47
C109_50_A_100	2475	316	977	-61%	5	52	5059	104%	7	214
C101_50_B_100	3883	122	1615	-58%	5	74	4977	28%	6	145
C102_50_B_100	8360	3450	1735	-79%	6	901	7766		7	17375
C103_50_B_100	9256	30367	2610	-72%	6	4928	x		x	x
C104_50_B_100	x	x	n/a				x		x	x
C105_50_B_100	4193	479	1718	-59%	7	609	5653	35%	7	1095
C106_50_B_100	4198	207	1889	-55%	5	106	4305	3%	6	326
C107_50_B_100	4026	615	1936	-52%	6	429	5282	31%	7	1986
C108_50_B_100	4634	1725	1928	-58%	6	2317	4555		6	5873
C109_50_B_100	4269	12638	1546	-64%	6	3553	x		x	x
C101_50_C_100	6298	680	2449	-61%	7	380	7063	12%	7	2479
C102_50_C_100	x	x	n/a				x		x	x
C103_50_C_100	x	x	n/a				x		x	x
C104_50_C_100	x	x	n/a				x		x	x
C105_50_C_100	6324	3023	2480	-61%	7	899	x		x	x
C106_50_C_100	6746	828	2032	-70%	7	355	x		x	x
C107_50_C_100	6402	3215	2237	-65%	8	4905	x		x	x
C108_50_C_100	7423	3931	2163	-71%	7	30714	x		x	x
C109_50_C_100	x	x	n/a				x		x	x

Table 15: *Computational results for class C1, 50 customers, scenarios A, B and C.*