Stabilized Branch-Price-and-Cut for the Commodity-constrained Split Delivery Vehicle Routing Problem

Timo Gschwind*,[a], Nicola Bianchessi[a,b], Stefan Irnich[a]

[a] *Chair of Logistics Management, Gutenberg School of Management and Economics, Johannes Gutenberg University, Jakob-Welder-Weg 9, D-55128 Mainz, Germany.*
[b] *Dipartimento di Informatica, Università degli Studi di Milano, Via Celoria 18, I-20122 Milan, Italy.*

Abstract

In the commodity-constrained split delivery vehicle routing problem (C-SDVRP), customer demands are composed of sets of different commodities. The C-SDVRP asks for a minimum-distance set of routes such that all customer demands are met and vehicle capacities are respected. Moreover, whenever a commodity is delivered by a vehicle to a customer, the entire amount requested by this customer must be provided. Different commodities demanded by one customer, however, can be delivered by different vehicles. Thus, the C-SDVRP is a relaxation of the capacitated vehicle routing problem and a restriction of the split delivery vehicle routing problem. For its exact solution, we propose a branch-price-and-cut algorithm that employs and tailors stabilization techniques that have been successfully applied to several cutting and packing problems. More precisely, we make use of (deep) dual-optimal inequalities which are particularly suited to reduce the negative effects caused by the inherent symmetry of C-SDVRP instances. One main issue here is the interaction between branching and cutting decisions and the different classes of dual inequalities. Extensive computational tests on existing and extended benchmark instances show that all stabilized variants of our branch-price-and-cut are clearly superior to the non-stabilized version. On the existing benchmark, our algorithm is significantly faster than the state-of-the-art algorithm and provides several new optima for instances with up to 60 customers and 180 tasks. Lower bounds are reported for all tested instances with up to 80 customers and 480 tasks, improving the bounds for all unsolved instances and providing first lower bounds for several instances.

Key words: routing, vehicle routing, dual-optimal inequalities, column generation, discrete split delivery

1. Introduction

The *commodity-constrained split delivery vehicle routing problem* (C-SDVRP) was introduced by Archetti *et al.* (2016) to implement one of the possible distribution policies applicable when different commodities have to be distributed to customers. The underlying idea was to study the impact on variable routing costs from adopting different distribution policies, i.e., from using vehicles dedicated to a single commodity compared with using flexible vehicles capable of carrying any set of commodities, and from allowing or forbidding split deliveries of individual commodities. The policy associated with the C-SDVRP allows the vehicles (which are homogeneous) to visit customers several times, but when a commodity is delivered by a vehicle to a customer, the entire amount requested by the customer has to be provided. No compatibility restrictions exist so that vehicles can carry any subset of commodities. Thus, the number of commodies requested by a customer is an upper bound on the number of visits to this customer. Overall, the C-SDVRP is a relaxation of the classical *capacitated vehicle routing problem* (CVRP) and a restriction of the *split delivery vehicle routing problem*, see the book by Toth and Vigo (2014), in particular chapters (Semet *et al.*, 2014; Poggi and Uchoa, 2014; Irnich *et al.*, 2014).

In order to evaluate the proposed distribution policies, Archetti *et al.* (2016) carried out worst-case as well as experimental analyses. In particular, they devised exact and heuristic algorithms. The tests were run on 64 small instances, with 15 customers and up to three commodities, 80 mid-size instances with 20, 40, 60, or 80 customers and up to three commodities, and large instances with 100 customers. One of the main conclusions drawn is that the delivery policy associated with the C-SDVRP is almost the best option w.r.t. to costs and, at the same time, it is likely more acceptable to customers than allowing splitting all the deliveries.

*Corresponding author.
Email address: gschwind@uni-mainz.de (Timo Gschwind)

A straightforward approach for modeling and solving the C-SDVRP is to reduce it to a standard CVRP, as done in (Archetti *et al.*, 2016) to solve the C-SDVRP heuristically. The authors duplicated each customer vertex as many times as the number of commodities requested by the customer, associating with each duplicated vertex the customer's demand for the corresponding commodity. Nevertheless, in order to limit the size of the model and mitigate symmetry issues, the branch-and-cut proposed by the authors is based on a direct model for the C-SDVRP.

Archetti *et al.* (2015) focused on the exact solution of the problem. While the branch-and-cut algorithm proposed in Archetti *et al.* (2016) was able to solve only 25 out of the 64 small instances, the *branch-price-and-cut* (BPC) algorithm proposed in Archetti *et al.* (2015) was able to solve all the small instances within the same time limit, and other instances with up to 40 customers and three commodities.

Under the name *discrete split delivery vehicle routing problem*, the C-SDVRP had already been introduced in the literature by Nakao and Nagamochi (2007) to model a specific practical routing problem as a variant of the SDVRP. The authors proposed a fast heuristic based on dynamic programming to solve real-world instances.

Another related problem is the *discrete split delivery vehicle routing problem with time windows* (DSDVRPTW) proposed by Salani and Vacca (2011), where the demand of a customer consists of several items. The authors assumed that demand can be split in orders, i.e., feasible combinations of items, that each vehicle can serve at most one order per customer and that service time at a customer's location depends on the delivered combination of items. The problem is solved by means of a branch-and-price algorithm. The DSDVRPTW can indirectly model the C-SDVRP by defining the set of orders to associate with a customer as the set of all possibles subsets of commodities to deliver to him.

In this paper, we focus on the exact solution of the C-SDVRP via column-generation techniques (Lübbecke and Desrosiers, 2005; Desaulniers *et al.*, 2005). We enhance the above-mentioned BPC algorithm of Archetti *et al.* (2015) in several aspects. First, we integrate powerful techniques such as implicit bidirectional labeling for solving the column-generation subproblem (Righini and Salani, 2006; Bode and Irnich, 2012) and additional cuts to strengthen the linear relaxation of the master program (Jepsen *et al.*, 2008). Second and more importantly, we tailor recent stabilization techniques originally suggested for bin-packing and vertex-coloring problems to the C-SDVRP (Ben Amor *et al.*, 2006; Gschwind and Irnich, 2016). Here, the stabilization of the column-generation process is achieved by the addition of (deep) dual-optimal inequalities, i.e., inequalities known to hold for (some) optimal dual solutions. We show that stabilization with dual inequalities is particularly suited to reduce the negative effects caused by the inherent symmetry of C-SDVRP instances. Indeed, when the number of tasks, i.e., the number of commodities to deliver to all customers, is kept constant, the most symmetric instances are those that have the largest number of commodities per customer, and these instances become less difficult to solve when dual inequalities are added. From a methodological point of view, the challenge is to correctly handle the impact that branching and cutting have on the solution of the subproblem and the validity of different classes of dual inequalities used for stabilization. The paper puts specific emphasis on these interdependencies.

The remainder of the paper is structured as follows. The C-SDVRP is formally defined in Section 2. Details on the new BPC algorithm including the generation of route variables, stabilization techniques, strengthening of the linear relaxation by adding valid inequalities, and branching are provided in Section 3. The results of our computational studies are provided in Section 4, before the paper closes with conclusions drawn in Section 5.

2. Problem Definition

Let $K = \{1, \dots, \kappa\}$ be the set of *commodities* that has to be distributed from the depot to the customers. Let $N = \{1, \dots, n\}$ be the set of *customers*. The *demand* of commodity $k \in K$ to deliver to customer $i \in N$ is denoted by d_{ik}. The set $K_i = \{k \in K \ : d_{ik} > 0\}$ comprises the commodities to be delivered to customer $i \in N$. A delivery *task* (i, k) is characterized by a customer $i \in N$ and one of its demanded commodities $k \in K_i$. The overall number of delivery tasks is $m = \sum_{i \in N} |K_i|$. Customers are served by means of a *fleet* of F homogeneous vehicles, each of which with a *capacity* of Q. Vehicles are flexible and can deliver any subset of commodities. Each customer may be visited more than once. At each visit to customer $i \in N$, one of the possible non-empty subsets $S_i \subseteq K_i$ of commodities has to be delivered to the customer. When a commodity is delivered by a vehicle to a customer, the entire amount of the commodity requested by the customer has to be provided.

Let $G = (V, A)$ be a directed graph with vertex set $V = N \cup \{0, n+1\}$, and arc set A. Vertices 0 and $n+1$ represent the depot where vehicle routes start and end, respectively. Each arc $(i, j) \in A$, with $i \neq n+1$

and $j \neq 0$, represents the possibility for the vehicles to travel from the location of i to the location of j, and it is associated with nonnegative variable routing costs c_{ij}. In the following, the term *route* is used for the combination of

(i) an elementary 0-$(n+1)$-path representing the sequence in which customers are visited by the vehicle,
(ii) and the selection of commodities S_i delivered to each visited customer $i \in N$.

Let Ω be the set of all *feasible routes*. For each $r \in \Omega$, the *routing costs* are $c^r = \sum_{(i,j)\in A(r)} c_{ij}$, and route r is *feasible* if the total demand delivered does not exceed the vehicle capacity, i.e., $\sum_{i\in N(r)} \sum_{k\in S_i(r)} d_{ik} \leq Q$, where $N(r) \subseteq N$ is the set of customers visited along route r, $A(r) \subseteq A$ is the set of arcs defining the path associated with route r, and $S_i(r)$ is the selection of commodities delivered to visited customer i in route r.

The C-SDVRP is the problem of determining a set of least-cost feasible routes to be associated with the vehicles such that all customers' demands are met.

3. Branch-Price-and-Cut

In order to solve the C-SDVRP, we adopted the same set-partitioning formulation as proposed by Archetti *et al.* (2015). The formulation has, for each route $r \in \Omega$, one binary variable λ^r that assumes a value equal to 1 if route r is performed, and 0 otherwise. Moreover, the non-negative integer variable ϕ models the number of routes performed (i.e., number of vehicles used), and x_{ij} and z_i are non-negative integer variables representing the number of times that an arc $(i,j) \in A$ is traversed and a customer $i \in N$ is visited, respectively. The formulation is as follows:

$$\min \sum_{r\in\Omega} c^r \lambda^r \tag{1a}$$

$$\text{s.t.} \sum_{r\in\Omega} a^r_{ik} \lambda^r = 1 \qquad i \in N, \quad k \in K_i \tag{1b}$$

$$\sum_{r\in\Omega} \lambda^r - \phi = 0 \tag{1c}$$

$$\left\lceil \frac{\sum_{i\in N} \sum_{k\in K_i} d_{ik}}{Q} \right\rceil \leq \phi \leq F \text{ and integer} \tag{1d}$$

$$\lambda^r \in \{0,1\} \qquad r \in \Omega \tag{1e}$$

$$\sum_{r\in\Omega} e^r_i \lambda^r - z_i = 0 \qquad i \in N \tag{1f}$$

$$1 \leq z_i \leq \min\{|K_i|, F\} \text{ and integer} \qquad i \in N \tag{1g}$$

$$\sum_{r\in\Omega} b^r_{ij} \lambda^r - x_{ij} = 0 \qquad (i,j) \in A \tag{1h}$$

$$0 \leq x_{ij} \leq \min\{|K_i|, |K_j|, F\} \text{ and integer} \qquad (i,j) \in A \tag{1i}$$

The objective function (1a) calls for the minimization of the total variable routing costs. Constraints (1b) ensure that all m customers' demands are met (i.e., tasks are fulfilled), where a^r_{ik} is a binary coefficient indicating that task (i,k) is performed, i.e., commodity $k \in K_i$ is delivered to customer $i \in N$, by route r. Constraints (1c) and (1d) limit the number of vehicles to use, and (1e) are domain definition constraints for variables λ^r.

The number of times a customer $i \in N$ is visited and an arc $(i,j) \in A$ is traversed is limited due to constraints (1f)–(1i). For this purpose, the binary coefficients e^r_i and b^r_{ij} are equal to 1 if customer $i \in N$ is visited and arc $(i,j) \in A$ is traversed by r, respectively, and 0 otherwise. Note that constraints (1f)–(1i) are redundant. They may only be added when violated by a fractional solution to the linear relaxation of (1) and for branching, see Section 3.4.

For many VRP variants, BPC is the most successful and leading exact solution approach (Costa *et al.*, 2018). Hence, we solve formulation (1) with BPC. The starting point is a *restricted master program* (RMP) which is the linear relaxation of formulation (1) defined over a small subset $\Omega' \subset \Omega$ of the route variables λ^r. Column generation alternates between the optimization of the RMP and the solution of the column-generation subproblem that generates negative reduced cost route variables λ^r if they exist (Desaulniers *et al.*, 2005). Archetti *et al.* (2015) have shown that this subproblem can be formulated and solved as a variant of the *shortest-path problem with resource constraints* (SPPRC, Irnich and Desaulniers, 2005). If no

negative reduced cost routes exist, a solution to the linear relaxation of (1) is found. The corresponding lower bound can be strengthened by the addition of valid inequalities. Finally, branching is required to ensure integer solutions.

3.1. Column Generation

In this section, we sketch the SPPRC-based solution approach of Archetti *et al.* (2015) for the column-generation subproblem. In addition, we refine parts of this approach by introducing another representation for the knapsack problem associated with each customer and further exploiting the inherent symmetry of the subproblem.

Recall that an instance of the column-generation subproblem is defined by the dual prices π_{ik}, $i \in N, k \in K_i$ associated with constraints (1b), σ associated with (1c), μ_i, $i \in N$ associated with (1f), and ρ_{ij}, $(i,j) \in A$ associated with (1h). The reduced cost of a route r that delivers commodities $S_i(r) \subseteq K_i$ to customers $i \in N(r)$ is given by

$$\tilde{c}^r = \sum_{(i,j)\in A(r)} c_{ij} - \sum_{i\in N(r)} \sum_{k\in S_i(r)} \pi_{ik} - \sigma - \sum_{i\in N(r)} \mu_i - \sum_{(i,j)\in A(r)} \rho_{ij}.$$

Archetti *et al.* (2015) have shown that the combined problem of determining the path and the commodities to deliver can be formulated as an SPPRC over a multi-graph. Slightly differing from their work, we define the multi-graph as an undirected graph (but otherwise identical) as follows: The vertices of the multi-graph comprise two copies $0'$ and $0''$ of the depot as well as two copies i' and i'' for each customer $i \in N$. Moreover, for all original arcs $(i,j) \in A$, there are two edges (i',j'') and (i'',j') in the multi-graph for modeling the movement of the vehicle. Finally, for all non-empty subsets $S_i \subseteq K_i$, $i \in N$, there are parallel edges between i' and i'', denoted as $(i',i'')^{S_i}$, that represent the respective deliveries made to customer i. An example of the multi-graph for an instance with three customers is depicted in Figure 1.

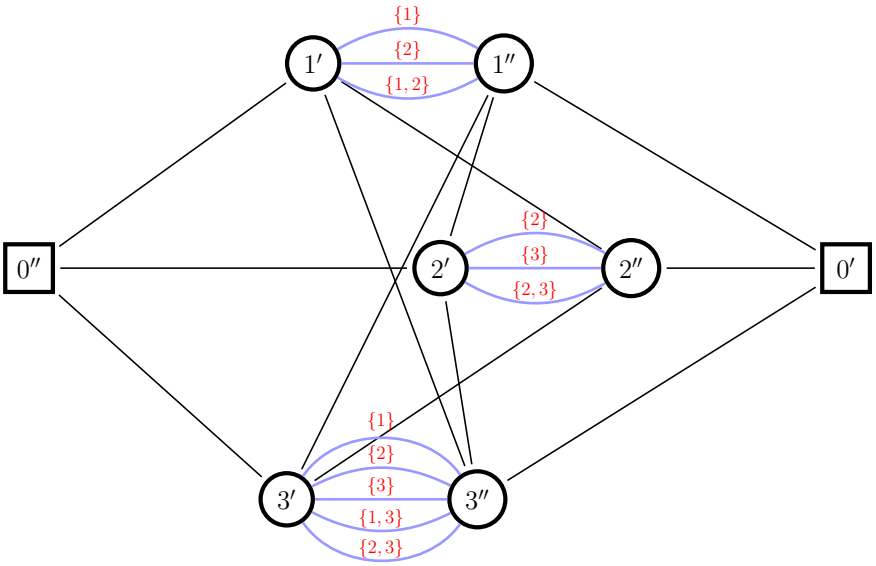

Figure 1: SPPRC pricing network for a C-SDVRP with three customers $\{1,2,3\}$; $K_1 = \{1,2\}, K_2 = \{2,3\}$ and $K_3 = \{1,2,3\}$; for the customer $i = 3$ only the subsets $S_3 = \{1\}, \{2\}, \{3\}, \{1,2\}$, and $\{2,3\}$ are feasible as we assume $d_{3,1} + d_{3,2} + d_{3,3} > Q$.

Any $0''$-$0'$-path in the multi-graph in which edges of the form (i'',j') with $i \neq j$ alternate with edges of the form (j',j'') corresponds with a route, and vice versa. Defining demands of zero on the edges (i',j'') for $i \neq j$ as well as demands

$$d^{S_i}_{i',i''} := \sum_{k\in S_i} d_{ik} \tag{2}$$

on edges $(i',i'')^{S_i}$, such a $0''$-$0'$-path represents a feasible route if it is elementary and the demands accumulated over its edges do not exceed Q. Obviously, edges $(i',i'')^{S_i}$ with a demand exceeding Q can be eliminated from the multi-graph.

With the following definitions, the multi-graph has a completely symmetric reduced-cost structure if the original instance is symmetric ($c_{ij} = c_{ji}$):

$$\tilde{c}_{i',j''} = \tilde{c}_{i'',j'} := c_{ij} - (\mu_i + \mu_j + \rho_{ij} + \rho_{ji})/2 \qquad \forall (i,j) \in A \tag{3a}$$

$$\tilde{c}^{S_i}_{i',i''} := -\sum_{k \in S_i} \pi_{ik} \qquad \forall i \in N, S_i \subseteq K_i \tag{3b}$$

where we additionally define $\mu_0 := \sigma$. Note that the benchmark instances for the C-SDVRP are all symmetric, see Section 4.1.

The solution approach of Archetti *et al.* (2015) proceeds in two phases as follows:

Phase 1: Pre-computation of all Pareto-optimal pairs $(c^{S_i}_{i',i''}, d^{S_i}_{i',i''})$ for each customer $i \in N$;

Phase 2: Solution of an SPPRC defined over a reduced multi-graph containing only Pareto-optimal edges between i' and i'' for all customer $i \in N$.

Pareto-optimal Deliveries. In principle, Pareto-optimal commodity combinations $S_i \subseteq K_i$ could be efficiently determined via enumeration when the number of commodities per customer is small (i.e., less than ten). However, in order to deal with sets K_i of arbitrary size, we present here an SPPRC-based technique that is also beneficial for separating violated dual-optimal inequalities (see Section 3.2) when non-robust cuts are added to (1) (see Section 3.3) and when branching constraints must be taken into account (see Section 3.4).

For each customer $i \in N$, first we arbitrarily order the commodities to deliver at i so that $K_i = \{k_1, k_2, \ldots, k_{|K_i|}\}$ (for the sake of convenience, we omit an extra index i). We define the *customer network* as a digraph (N_i, A_i) with $|K_i| + 1$ vertices given by $N_i = \{0, 1, 2, \ldots, |K_i|\}$. For each index $j \in N_i, j \neq 0$, there are two arcs $(j-1, j)^0$ and $(j-1, j)^1$ between $j-1$ and j. The arc $(j-1, j)^0$ models that the jth commodity k_j is not selected, while arc $(j-1, j)^1$ models the selection of k_j. Accordingly, arc $(j-1, j)^0$ has zero weight and zero profit, and arc $(j-1, j)^1$ has weight d_{ik_j} and profit π_{ik_j}. Figure 2 displays an example of such a customer network.

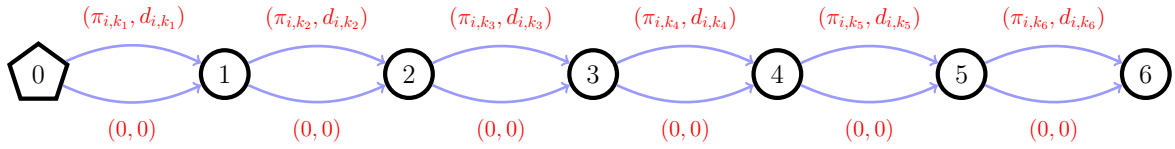

Figure 2: Customer network (N_i, A_i) for a customer $i \in N$ with six commodities $K = \{k_1, k_2, k_3, k_4, k_5, k_6\}$

Each 0-$|K_i|$-path in digraph (N_i, A_i) is in one-to-one correspondence to a (possibly empty) subset S_i of K_i. If the weight of the 0-$|K_i|$-path is positive and does not exceed Q, the corresponding subset S_i represents a feasible delivery. Hence, solving an SPPRC with resources profit (to maximize) and weight (constrained by Q) produces all Pareto-optimal sets S_i. Note that dominance between SPPRC labels must be slightly modified: as $S_i = \varnothing$ is not allowed (at least one commodity must be delivered), labels $(0, 0)$ must not dominate and eliminate other labels.

SPPRC over the Multi-Graph. The reduced multi-graph changes from one column-generation iteration to the next because all edges $(i', i'')^{S_i}$ whose subsets S_i are not Pareto-optimal are temporarily removed. The actual SPPRC is then solved using the following resources: (i) accumulated reduced cost according to (3); (ii) accumulated demand according to (2); and (iii) visit indicators for each customer $i \in N$. Define $V' := \{0'\} \cup \{n' : n \in N\}$ and $V'' := \{0''\} \cup \{n'' : n \in N\}$. In monodirectional forward labeling, labels at a vertex $i' \in V'$ are only propagated towards $i'' \in V''$ (same i), while labels at $i'' \in V''$ are only propagated to vertices $j' \in V'$ (with different $j \neq i$).

All resources are initially set to 0. While reduced costs are unconstrained, the accumulated demand is bounded from above by Q. The visit indicator of customer $i \in N$ is increased when one of the arcs $(i', i'')^{S_i}$ is traversed, and it is bounded from above by 1.

As this elementary SPPRC is NP-hard in the strong sense, relaxations are typically employed. We use the *ng*-path relaxation of Baldacci *et al.* (2011), in which a cycle over a customer $i \in N$ is possible if i is not in the neighborhood of a vertex of the cycle. Neighborhoods must be pre-defined, and in most implementations reported in the literature a global number controls the maximum size of all neighborhoods. The larger the

neighborhoods, the fewer cycles are possible, but on average the computational difficulty increases. Good tradeoffs were often obtained with neighborhoods of size between 5 and 20.

Bidirectional labeling for SPPRCs was coined by Righini and Salani (2006) in order to mitigate the explosion of labels typically observed when partial paths grow longer. In bidirectional labeling, both forward partial paths and backward partial paths are created, but processed only up to a so-called half-way point. In VRP variants with only capacity constraints, the termination criterion is often that the accumulated demand has reached or exceeded half of the vehicle's capacity, i.e., the half-way point is $Q/2$. When forward and backward labeling terminates, suitable forward and backward labels must be merged to obtain a complete feasible route. Several subsequent works have shown that bounded bidirectional labeling algorithms are usually superior to their monodirectional counterparts.

If the SPPRC is completely symmetric, e.g., guaranteed by definitions (3), the computational effort can be further reduced by using an implicit bidirectional approach. Note that forward and backward propagation produce essentially identical partial paths, since forward partial paths start at $0''$ and always traverse edges $(i', i'')^{S_i}$ in the given direction, while backward partial paths start at $0'$ and traverse edges $(i', i'')^{S_i}$ in backward direction (from i'' to i'). Hence, swapping all i' and i'', respectively, allows to produce backward partial paths out of forward partial paths, and vice versa. Thus, label propagation in one direction can be omitted, so that, e.g., only forward partial paths are combined in the merge procedure. This implicit bidirectional techniques has already been successfully implemented and applied in (Bode and Irnich, 2012; Goeke *et al.*, 2018).

3.2. Stabilization and Dual Inequalities

In this section, we describe the key innovation of this research: the use of dual inequalities for the stabilization of the column-generation process. For a general introduction to the topic we refer to (Ben Amor *et al.*, 2006; Gschwind and Irnich, 2016).

Dual Inequalities. We start with some basic definitions. For the C-SDVRP, any linear inequality in the dual variables is a *dual inequality* (DI). The DIs that we will consider in the following refer to the dual variables (π_{ik}) for $i \in N, k \in K_i$ associated with the m partitioning constraints (1b). Recall that these constraints model the fulfillment of the tasks (i, k) for all $i \in N, k \in K_i$. Let D^* be the dual-optimal space, i.e., the set of optimal solutions of the dual model to the linear relaxation of (1). Following Ben Amor *et al.* (2006), a DI $\mathbf{t}^\top \pi \le t$ (with $\mathbf{t} \in \mathbb{Z}^m$ and $t \in \mathbb{Z}$) is a *dual-optimal inequality* (DOI) if $D^* \subseteq \{\pi : \mathbf{t}^\top \pi \le t\}$. A set of DIs comprises *deep dual-optimal inequalities* (DDOIs) if at least one $\pi^* \in D^*$ fulfills all inequalities. Hence, DOIs are always DDOIs, but the reverse is not necessarily true.

A trivial but already effective stabilization happens when the partitioning constraints (1b) are replaced by covering constraints. This is a valid replacement (preserving the linear relaxation bound) if the cost matrix fulfills the triangle inequality. We assume that the triangle inequality is fulfilled for the C-SDVRP instances. Note that this replacement is equivalent to the DIs $\pi_{ik} \ge 0$ for all $i \in N, k \in K_i$.

The constraint matrix of (1) does also possess the following row replacement property. For a fixed customer $i \in N$ and two different commodities $k, k' \in K_i$ with demands fulfilling $d_{ik'} \le d_{ik}$, a route serving customer i and delivering k and not k' can always be feasibly replaced by an otherwise identical route that delivers k' and not k. For the sake of simplicity, we assume that all commodities delivered to customer $i \in N$ are sorted by non-decreasing demand, i.e.,

$$K_i = \{k_{i1}, k_{i2}, \ldots, k_{i|K_i|}\} \quad \text{with} \quad d_{ik_{i1}} \le d_{ik_{i2}} \le \cdots \le d_{ik_{i|K_i|}}$$

holds for all $i \in N$. With this sorting, the DIs

$$\pi_{i,k_p} - \pi_{i,k_q} \le 0 \qquad \text{for } p \le q$$

in the following denoted as *pair inequalities*, are DDOIs, as formally shown in (see Gschwind and Irnich, 2016, Section 3.3, Definition 2, and Property 5). In the primal model, these pair inequalities are extra columns with exactly two entries/non-zero rows, -1 representing the removal of task (i, k_q) and $+1$ representing the addition of task (i, k_p), i.e., the replacement of commodity k_q by commodity k_p for customer i. The pair inequalities result from the subset of $|K_i| - 1$ so-called *ranking inequalities*

$$\pi_{i,k_1} \le \pi_{i,k_2} \le \cdots \le \pi_{i,k_{|K_i|}}, \tag{4}$$

which are as powerful as the pair inequalities. Hence, the $|K_i| - 1$ ranking inequalities can be used instead of all $\binom{|K_i|}{2}$ pair inequalities.

Note also that if two commodities k, k' of a customer $i \in N$ have identical demand, the two pair inequalities result in the equality $\pi_{ik_p} = \pi_{ik_q}$, which is equivalent to replacing the two corresponding covering constraints (1b) by an aggregated ≥ 2 constraint. For more details on the relationship between constraint aggregation and dual equalities we refer to (Gschwind and Irnich, 2016, Property 6). We do not apply aggregation here, because typical C-SDVRP instances have different demands for all commodities delivered to a customer, see Section 4.

As known from bin packing, the pair inequalities can be generalized in the following way: consider a customer $i \in N$, one of its commodities $k' \in K_i$, and a subset $S \subseteq K_i$ with

$$d_{i,k'} \geq \sum_{k \in S} d_{i,k},$$

i.e., the amount to deliver of k' is not smaller than the amount for the commodities $k \in S$. Then,

$$\pi_{i,k'} - \sum_{k \in S} \pi_{i,k} \geq 0 \tag{5}$$

is a so-called *subset inequality* (SI). Pair inequalities are subset inequalities for singleton sets $S = \{k\}$. For $|S| > 1$, SIs are no DDOIs for the C-SDVRP.

Static versus Dynamic Addition of Dual Inequalities. The most general form of DIs we use is the SIs (5), defined by the triple (i, k', S) with $i \in N$, $k' \in K_i$, and $S \subset K_i$. Let Θ be the set of all active DIs (at some point in time). For the primal model (1), each DI for $(i, k', S) \in \Theta$ is implemented as a *DI column* with entry -1 for the row (i, k') and entries $+1$ for all rows (i, k), $k \in S$ of the constraints (1b). With non-negative variables $y_{i,k,S} \geq 0$ for $(i, k, S) \in \Theta$, constraints (1b) are replaced by

$$\sum_{r \in \Omega} a^r_{ik} \lambda^r - \sum_{\substack{(i',k',S') \in \Theta: \\ (i',k') = (i,k)}} y_{i',k',S'} + \sum_{\substack{(i',k',S') \in \Theta: \\ i' = i, k \in S'}} y_{i',k',S'} = 1 \qquad\qquad i \in N, \quad k \in K_i.$$

Moreover, there are several possible strategies to add DIs. We describe three straightforward strategies now:

Static: Right from the initialization of the master program, the DI columns are added to (1).
The advantage is that the stabilization effect happens from the first iteration on. However, more columns in the RMP make the re-optimization slower.

Dynamic: Only DIs that are violated may be added to the master problem (1).
The advantage is a generally smaller RMP with a faster re-optimization, but a less stabilized column-generation process. Moreover, the identification of violated DIs, i.e., their *separation*, may impose additional effort.

Mixed: Obviously, both strategies can be mixed if a subset of DIs is added a priori and other violated DIs are separated and added dynamically.

In all three cases, we do not remove non-binding DI columns even though such a clearance step would be possible. Different strategies for the addition of DIs are presented and analyzed in Section 4.2.

Separation of Violated Dual Inequalities. The separation of violated SIs is a by-product of Phase 1 of the two-phase pricing approach described above. Indeed, forward labeling in the customer network (N_i, A_i) (see Figure 2) produces labels at each vertex $j \in N_i, j \neq 0$. Recall that each such vertex j is associated with the jth commodity k' in K_i. One of the labels at j is $L' = (\pi_{i,k'}, d_{i,k'})$, which results from the path that only includes the jth commodity but no other commodity. Any other non-zero label at j must have included one other or several commodities $S \subseteq K_i$ so that it is identical to $L = (\sum_{k \in S} \pi_{ik}, \sum_{k \in S} d_{ik})$. If L dominates L', in the SPPRC sense of Phase 1, this implies $\sum_{k \in S} \pi_{ik} \geq \pi_{i,k'}$ and $\sum_{k \in S} d_{ik} \leq d_{i,k'}$. If the first inequality is strict, the violated SI for (i, k', S) has been identified. This procedure finds a most violated SI (i, k', S) because commodities are sorted by non-decreasing demand so that any subset $S \subset K_i$ with $\sum_{k \in S} d_{ik} \leq d_{i,k'}$ comprises only commodities with an index smaller than j.

Primal Solutions. If additional variables $y_{i,k',S}$ for SIs are added to the RMP, the solution to the master program may consist not only of columns of routes but of a mixture of these and DI columns. Such a solution can however be transformed into a pure route-columns solution if the DIs are DDOIs. Both statements are in fact equivalent as proven by Gschwind and Irnich (2016, Proposition 1). The authors also describe an iterative transformation procedure. We summarize its basic ideas now: In each iteration, the algorithm

chooses from the current (partly transformed) solution a route column with coefficients $\mathbf{a}^r$ for a route $r \in \Omega$ and a *compatible* SI column (i, k', S), i.e., one that fulfills $a^r_{ik'} = 1$ and $a^r_{ik} = 0$ for all $k \in S$. Let $\bar{\lambda}^r$ and $\bar{y}_{i,k',S}$ be the solution values of the chosen columns, respectively. Then, a new solution is constructed from the current one by decreasing the solution values of the chosen route and SI columns by $\min\{\bar{\lambda}^r, \bar{y}_{i,k',S}\}$ and increasing the solution value of the column corresponding to a new route with coefficient $\mathbf{a} + \mathbf{e}_{iS} - \mathbf{e}_{ik'}$, where the latter are the incidence vectors of $\{i\} \times S$ and (i, k'), respectively. The procedure continues until no more compatible routes and SI columns exist. If all SIs that have been added are DDOIs, it is guaranteed that none of them has positive value in the last transformed solution so that a pure route-columns solution is finally found.

Overstabilization and Recovery Procedure. Recall that, in general, SIs are neither DOIs nor DDOIs for the C-SDVRP. However, they may in fact be DDOIs for many C-SDVRP instances. Even if not, their addition does have a stabilizing effect on the column-generation process. Moreover, this possible *overstabilization* can be remedied typically without significant additional computational effort. Indeed, if the added inequalities are no DDOIs for the given instance, i.e., all dual-optimal solutions are cut-off, a recovery procedure can be used to detect and handle overstabilization. For the C-SDVRP, the recovery procedure first tries to build from the RMP solution a pure route-columns solution. If this is not possible the RMP has been overstabilized. In this case, there exists an SI column with positive value, but no compatible route column. Let (i, k') be the task whose commodity k' is replaced by other commodities in the SI column. The recovery procedure then eliminates all SIs that replace this task (i, k') from the RMP, forbids their re-generation in the separation routine, and restarts the column-generation process to optimize the RMP. The process iterates until a pure route-columns solution is found.

3.3. Valid Inequalities and Cutting Strategy

We use three types of valid inequalities, namely capacity cuts, subset-row inequalities, and strong-degree constraints.

For any subset of customers $C \subset N$, $C \neq \varnothing$, let $\delta^-(C)$ denote the arcs of the digraph $G = (V, A)$ with $i \notin C$ and $j \in C$. The associated *capacity cut* (CC) is

$$\sum_{r\in\Omega} \left(\sum_{(i,j)\in\delta^-(C)} b^r_{ij} \right) \lambda^r \geq \left\lceil \frac{\sum_{i\in C} \sum_{k\in K_i} d_{ik}}{Q} \right\rceil.$$

For any CC added to the RMP of (1), let γ_C be the corresponding dual price. Then, the value $\gamma_C/2$ can be distributed symmetrically on the edges (i', j'') and (i'', j') for all $(i, j) \in \delta^-(C)$ of the undirected SPPRC pricing network. Hence, CCs are robust cuts (Fukasawa *et al.*, 2006).

The second family comprises *subset-row inequalities* (SR inequalities), first introduced for the VRPTW by Jepsen *et al.* (2008). According to the definition of rows for partitioning constraints in (1b), the variant of the SR inequalities for the C-SDVRP is defined for subsets of tasks, i.e., $R \subset \{(i, k) : i \in N, k \in K_i\} \subset N \times K$. As in several other works, we restrict ourselves to subset R of cardinality three. The associated SR inequality for R is

$$\sum_{r\in\Omega} \left\lfloor \frac{\sum_{(i,k)\in R} a^r_{ik}}{2} \right\rfloor \lambda^r \leq 1.$$

For elementary routes, the SR inequality for R ensures that at most one route that fulfills at least two of three tasks is part of a feasible solution. We assume that the active SR inequalities are given by the sets $R \in \mathcal{R}$. For any $R \in \mathcal{R}$, let β_R be the corresponding dual price. To correctly incorporate the dual price β_R, a binary resource has to be added to the labels of both the customer networks and the SPPRC multi-graph. The labeling algorithms are modified as explained by Jepsen *et al.* (2008).

The third family of *strong-degree constraints* (SD constraints) was introduced by Contardo *et al.* (2014). For the C-SDVRP, there is one SD constraint per task (i, k) with $i \in N$ and $k \in K_i$ ensuring that (i, k) is served by at least one route, may this route be elementary or not. Formally, the SD constraint is

$$\sum_{r\in\Omega} \xi^r_{ik} \lambda^r \geq 1,$$

where the coefficient ξ^r_{ik} indicates whether or not route r delivers commodity k to customer i. Hence, $\xi^r_{ik} \leq a^r_{ik}$ with equality for elementary routes. Strictly less occurs, when a non-elementary route r delivers (i, k) twice

or even more often. For any SD constraint added to the RMP of (1), let η_{ik} be the corresponding dual price. As explained by Contardo *et al.* (2014), one binary resource per active SD constraint must be added to the SPPRC multi-graph.

Impact of Cutting on Dual Inequalities. SR inequalities and SD constraints have an impact on the SI. For a specific SI defined by (i, k', S), we define for each set $R \in \mathcal{R}$ the coefficient

$$u^R_{i,k',S} = \left\lceil \frac{\left(|\{(i,k) : k \in S\} \cap R| - \delta_{(i,k'),R}\right)^+}{2} \right\rceil, \tag{6}$$

where $\delta_{(i,k'),R} = 1$ if $(i, k') \in R$ and zero otherwise. This coefficient is only positive if some tasks that define the SR inequality also occur in the SI. More precisely, the inner non-negative part in (6) counts the surplus of tasks added and removed by the SI that occur in R. An even inner part exactly gives the difference in reduced cost of a route when the respective replacement is performed, while for odd values the difference is possibly overestimated. Therefore, the SI for (i, k', S) is now:

$$\pi_{i,k'} - \sum_{k \in S} \pi_{i,k} - \sum_{R \in \mathcal{R}} u^R_{i,k',S} \cdot \beta_R + \eta_{ik'} \geq 0, \tag{7}$$

where the last term on the left-hand side is the correction for the SD constraint for (i, k').

Overall Cutting Strategy. The overall cut-generation strategy is the following: For branch-and-bound nodes up to level 10, we separate violated CCs, SR inequalities, and SD constraints in this order. To avoid the introduction of too many resources, we limit the number of SR inequalities to 30 in total and ten in each round of separation. Moreover, for each task (i, k) we limit the number of SR inequalities defined by R with $(i, k) \in R$ to a maximum of two per round.

For branch-and-bound nodes deeper in the tree, we only separate violated SD constraints as they are required to guarantee the completeness of the branching rule described in the following section.

3.4. Branching

We apply a similar five-level branching strategy as already applied in (Archetti *et al.*, 2015). We briefly summarize this strategy. At the first level, we branch on the number of vehicles. At the second level, we branch on the number of visits to each customer. Here, the branching decision can make some subsets S_i of commodities of customer i infeasible. In this case, we eliminate them from the customer network. For example, a single visit allows only the subset $S_i = K_i$. When forcing at least two visits, the subset $S_i = K_i$ becomes infeasible. At the third level, we branch on the flow on each edge. All these branching rules are implemented by adding inequalities to the RMP. The only exception is the zero-flow decision on edges which is implemented by removing the edge.

Integer flows on all edges are however not sufficient to guarantee integer routes. Therefore, at the fourth and fifth level, we use the Ryan and Foster branching on the tasks. For two tasks (i, k) and (i', k'), let $f_{i,k,i',k'} = \sum_{r \in \Omega} a^r_{ik} a^r_{i'k'} \lambda^r$ be the information if (i, k) and (i', k') are served by the same route. At the fourth level, we branch on pairs (i, k) and (i, k') of tasks of the same customer i for which $f_{i,k,i,k'}$ is fractional. For $f_{i,k,i,k'} = 0$ (separate branch), we modify the pre-computation of the Pareto-optimal deliveries $S_i \subseteq K_i$ by introducing an additional binary resource into the SPPRC. Once that k or k' is reached in the customer network of i the resource is set to one and the visit to the other commodity becomes impossible. The result is that all Pareto-optimal sets S_i do not contain both k and k'. For $f_{i,k,i,k'} = 1$ (together branch), the vertices k and k' are merged together into one vertex of the customer network.

At the fifth level, we consider pairs (i, k) and (i', k') for two different customers i and i'. Both branches (separate and together) are implemented by introducing binary resources into the SPPRC multi-graph. The separate branch uses a single binary resource as described for the separate branch before. The together branch requires two binary resources indicating the service of (i, k) and (i', k'), respectively. The extension to the destination depot $0'$ and the merge of two labels are only allowed if both resources have identical value.

Completeness of Branching Scheme. The validity of the branching scheme relies on the use of the SD constraints that guarantee elementary routes. Otherwise, a minimum example with two customers 1 and 2 and tasks $(1, k_1)$, $(1, k_2)$, and $(2, k)$ uses the following two non-elementary routes $(0, (1, k_1), (2, k), (1, k_1), 0)$ and $(0, (1, k_2), (2, k), (1, k_2), 0)$ exactly 0.5 times each. The result is that overall one vehicle is used, customer 1 is visited twice and customer 2 is visited once, and all edge flows and all f-values are integer. Therefore, none of the branching rules is applicable, but the introduction of the SD constraints for the tasks $(1, k_1)$ and $(1, k_2)$ cuts off this fractional solution.

Impact of Branching on Dual Inequalities. The branching decisions at the levels two, four, and five can have an impact on the validity of the DIs.

Branching decisions of level two can make some subsets S_i of commodities of a customer i infeasible. In this case, we eliminate all SIs that can produce such infeasible S_i. For example, in a branch that requires at least three visits to a customer i who has three commodities, any SI that replaces one commodity by two others is infeasible.

Branching decision using the Ryan and Foster branching at levels four and five impose the elimination of some SIs. In the together branch, the only valid SIs are those that either refer to none of the two tasks or SIs where the set S includes both. In the separate branch, again SIs that do not refer to either of the two tasks are valid. Moreover, SIs that replace one task by the other are also valid. All other SIs must be eliminated.

Tree Exploration and Variable Selection Strategy. We use a best-bound first tree exploration strategy in order to improve the dual bound as fast as possible. On all five levels, the specific branching variable is selected as one where the fractional part is closest to 0.5.

3.5. Acceleration Techniques

To accelerate the column-generation process, we use two types of reduced networks to solve the SPPRC pricing problem heuristically. The first one includes only one Pareto-optimal commodity set S_i for each customer i in the SPPRC multi-graph. More precisely, we use the commodity combination S_i^* with the best reduced cost, i.e., $S_i^* = \arg\min_{S_i \subseteq K_i} c_{i',i''}^{S_i}$. The second one uses only a subset of the inter-customer edges of the SPPRC multi-graph, namely we limit the number of edges adjacent to a customer to a given value D. In our implementation, we use $D \in \{2, 5, 10\}$ to obtain a sequence of pricing heuristics with increasing search space and computational effort. The overall pricing strategy is to first consider the SPPRC multi-graphs with only one Pareto-optimal delivery per customer and $D = 2$, 5, 10, and $|n|$ inter-customer edges, followed by the SPPRC multi-graphs with all Pareto-optimal deliveries and the same values for D. Whenever the labeling algorithm finds negative reduced paths in one of the (reduced) networks they are returned to the RMP and the networks later in the sequence are not solved in this iteration.

The course of our BPC algorithm is summarized in Algorithm 1.

4. Computational Results

We implemented the BPC algorithm in C++ and compiled the code in release mode under MS Visual Studio 2013 (64-bit version). At each column-generation iteration, the RMP was solved by means of CPLEX 12.6.2. The experiments were carried out on a standard PC with an Intel(R) Core(TM) i7-5930k, clocked at 3.5 GHz, and 64 GB of RAM, by allowing a single thread for each run. The time limit TL1 for each run was set to one hour. The main components and important parameters of the algorithm are summarized in Table 1.

4.1. Benchmark Instances

As done in Archetti *et al.* (2015), we considered the small ($n = 15$) and mid-size ($n = 20, 40, 60, 80$) instances introduced by Archetti *et al.* (2016). In each instance, the customer locations are taken from the R101 or the C101 instance of the Solomon (1987) benchmark. Then, as for the small instances, the number κ of commodities is set to 2 or 3, and the probability p with which the single customer requires each specific commodity is set to 1.0 or 0.6. When the customer requires the commodity, the corresponding demand is randomly selected in the interval $[1, 100]$ or $[40, 60]$. The vehicle capacity is finally set to $\beta \cdot \max_{i \in N}\{\sum_{k \in K_i} d_{ik}\}$ with $\beta \in \{1.1, 1.5, 2.0, 2.5\}$. One instance is created for each combination of the defining parameters, for a total of 64 small instances.

Algorithm 1: BPC algorithm

```
   // Initialization
1  Add static DIs                                                    // Section 3.2
   // Main loop
2  while not integer optimal do
3      Select branch-and-bound node                                  // Section 3.4
4      Forbid DIs not compatible with branching decisions            // Section 3.4
       // Solve LP-relaxation
5      do
6          do
7              do
8                  for each customer i ∈ N do
9                      Compute Pareto-optimal pairs (c̃^{S_i}_{i'i''}, d^{S_i}_{i'i''})   // Section 3.1
10                     Separate violated DIs                         // Section 3.2
11                 Build SPPRC multi-graph                           // Section 3.1
12                 Solve pricing problem on multi-graph              // Section 3.1
13             while negative reduced-cost columns found
               // Recovery procedure
14             Try to restore primal solution                        // Section 3.2
15             if overstabilization then
16                 Forbid overstabilizing DIs                        // Section 3.2
17         while overstabilization
18         Separate violated valid inequalities                      // Section 3.3
19     while violated valid inequalities found
20     if LP fractional and no bounding then
21         Perform branching                                         // Section 3.4
```

Column Generation	Two phase approach: (1) Pareto-optimal pairs $(\tilde{c}^{S_i}_{i'i''}, d^{S_i}_{i'i''})$ for each customer (2) SPPRC with implicit bidirectional labeling, *ng*-path	Section 3.1
	Reduced networks: (a) only one Pareto optimal set S_i per customer (b) of sizes $D = 2, 5, 10$	Section 3.5
Cutting	Capacity cuts: all, heuristic separation	Section 3.3
	Subset-row inequalities: only $\|R\| = 3$, exact separation	
	Strong-degree constraints: all, exact separation	
Branching	Best-bound first tree search	Section 3.4
	Five-level hierarchy: (1) #veh, (2) #visits, (3) edge flow, (4) Ryan-Foster on tasks for the same customer, and (5) for different customers	
	Variable selection: fractional part closest to 0.5	

Table 1: Summary of main components and important parameters of the branch-price-and-cut algorithm

The defining parameters for the mid-size instances differ as follows: κ and β are fixed to 3 and 1.5, respectively, and the demand for each commodity required by the customers can be randomly selected only in the interval $[1, 100]$. Five instances are generated for each combination of the parameters, for a total of 80 mid-size instances.

Recall that the C-SDVRP is inherently symmetric. The larger the number of commodities per customer, the larger the number of possible cost equivalent solutions to the problem that differ only in the delivery patterns associated with the same set of elementary 0-$(n+1)$-path. We claimed that the use of DIs should have positive effects especially for such highly symmetric instances. In order to support this statement, we enlarged the set of small and mid-size instances considering additional values for the number κ of commodities, namely 4, 5, and 6. In particular, following the same procedure as described in Archetti *et al.* (2016), we generated 336 additional instances with more commodities (96 small and 240 mid-size).

4.2. Impact of Dual Inequalities

For the analysis of the impact of DIs, we restrict the test set to the 160 mid-size instances with 20 and 40 customers, because these instances pose a challenge due to the large number of up to $40 \cdot 6 = 240$ tasks. Moreover, comparisons on the basis of results obtained for smaller or bigger instances may lead to erroneous conclusions as they are generally either very easy or prohibitively hard to solve.

We define `Plain` as the version of the BPC algorithm in which DIs are not used. We analyze how much the DIs contribute to the performance of the BPC algorithm by comparing `Plain` against the following variants of the BPC algorithm in which:

- `S-Rank`: the DI columns associated with ranking inequalities (4) are statically added to the RMP;
- `S-Rank-S-Subs`: the DI columns associated with ranking inequalities (4) and a selection of the subset inequalities (5) are statically included into the RMP;
- `D-Subs`: the DI columns associated with violated subset inequalities (5) are dynamically added to the RMP;
- `S-Rank-D-Subs`: the DI columns corresponding to ranking inequalities (4) are statically added to the RMP, whereas DI columns associated with subset inequalities (5) are dynamically added when the corresponding inequalities are violated;
- `S-Rank-S/D-Subs`: as `S-Rank-S-Subs` but with the additional possibility to dynamically add to the master problem further DI columns associated with violated subset inequalities;

We analyze the six different BPC algorithms with the help of performance profiles that allow the comparison of a set $\mathcal{A}$ of algorithms that are all applied to the same benchmark set (Dolan and Moré, 2002). The performance profile $\rho_A(\tau)$ of an algorithm $A \in \mathcal{A}$ is the fraction of instances that algorithm A can solve within a factor τ of the fastest algorithm (unsolved instances are taken into account with infinite run time). In particular, $\rho_A(1)$ is the percentage of instances on which A is a fastest algorithm compared to all other algorithms $B \in \mathcal{A} \setminus \{A\}$. The value $1 - \rho_A(\infty)$ is the percentage of instances not solved by A.

We start with the comparison for the solution of the linear relaxation of (1), in the following denoted by *Root*, for $\mathcal{A} = \{$`Plain`, `S-Rank`, `S-Rank-S-Subs`, `D-Subs`, `S-Rank-D-Subs`, `S-Rank-S/D-Subs`$\}$. The performance profiles are depicted in Figure 3. The most striking result is that the `Plain` BPC is clearly inferior to all other variants that use DIs for a stabilization of the RMP. Without doubt, DIs significantly accelerate the column-generation process. The variant `S-Rank` that only uses ranking inequalities and adds them during the initialization of the RMP seems to be the clear winner. The four other variants perform better than `Plain` and worse than `S-Rank`. Among themselves, `S-Rank-S-Subs` seems slightly better than the three others which perform almost identically.

We expected rather similar results for integer solutions produced with the fully-fledged BPC (in the following denoted by *Tree*). Figure 4 depicts the performance profiles of all six BPC algorithms. Also here `Plain` is not at all competitive and the use of DIs can be strongly recommended. Moreover, the simple variant `S-Rank` is no longer superior, instead the variant `S-Rank-D-Subs` seems to be the overall best strategy.

In order to understand the different behavior at the root node and for the branch-and-bound tree, we deeper analyze the pricing iterations. Table 2 compares the number of column-generation iterations for the six variants. The following results are provided for the linear relaxation (Root) and the branch-and-bound tree (Tree):

- #: Number of instances (out of 160) solved by all six algorithms, all following averages are taken only over those instances solved by all six variants;

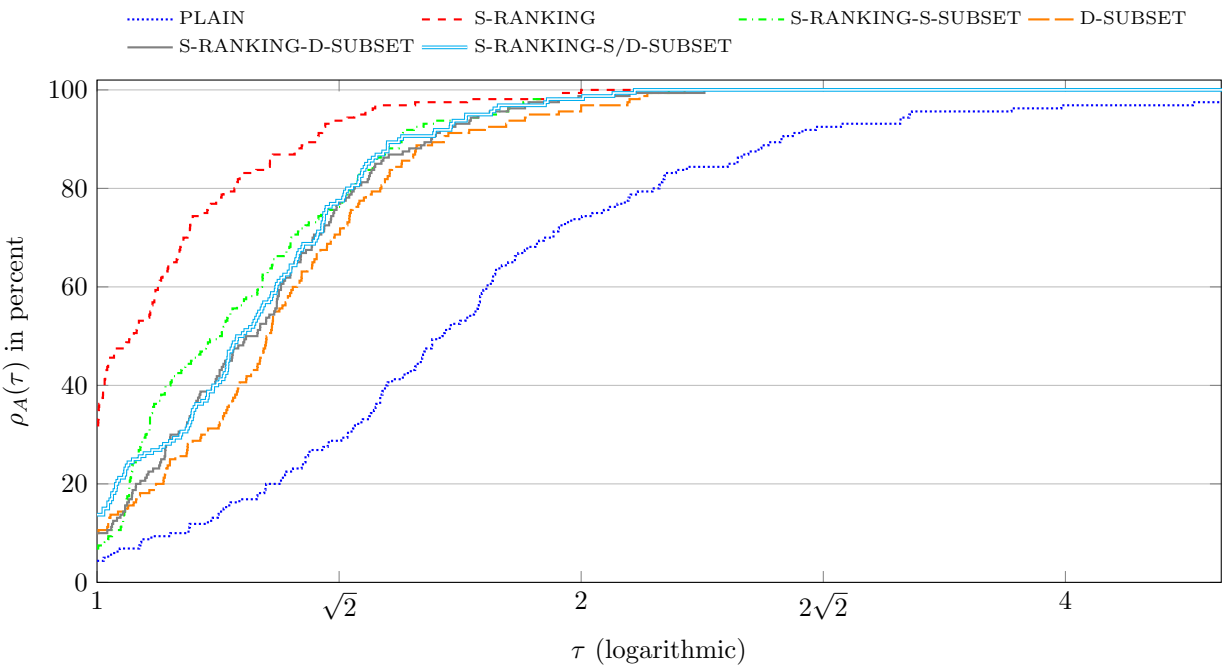

Figure 3: Performance profiles for the solution of the linear relaxation (Root) of the C-SDVRP instances with $n = 20$ and 40 customers.

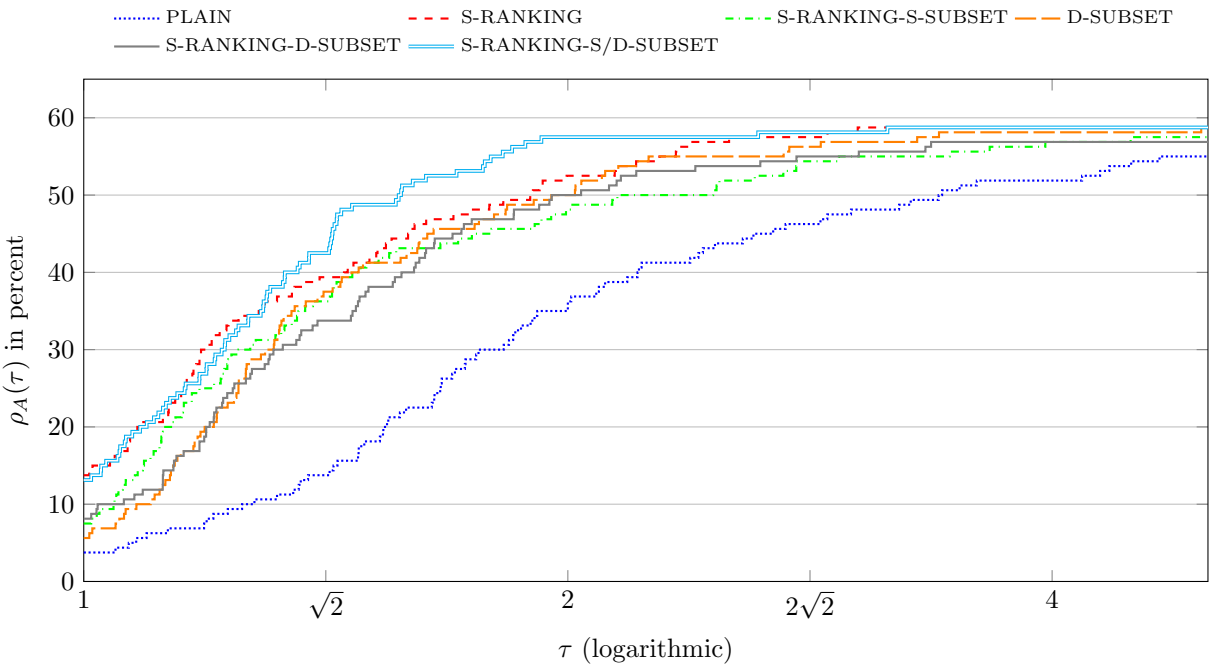

Figure 4: Performance profiles for the integer solution (Tree) of the C-SDVRP instances with $n = 20$ and 40 customers.

			Plain	S-Rank	S-Rank-S-Subs	D-Subs	S-Rank-D-Subs	S-Rank-S/D-Subs
		#	absolute	relative to Plain				
Number of iterations								
total	Root	159	**260.8**	0.47	0.43	0.47	0.44	0.43
	Tree	88	**1862.3**	0.67	0.71	0.75	0.72	0.67
exact (only)	Root	159	**3.2**	0.90	1.41	1.55	1.48	1.55
	Tree	88	**320.2**	0.96	1.07	1.09	1.09	1.05
Number of optima	Tree		**91**	+3	+2	+4	+1	+4
Time	Tree	88	**352.7**	0.68	0.74	0.74	0.74	0.66

Table 2: BPC results (Root and Tree) for the C-SDVRP instances with $n = 20$ and 40 customers: Number of column-generation iterations (total and exact only), number of proven optimal solutions, and computation times.

relative to Plain: The numbers for Plain are given as absolute values. For all other variants, we compute the *geometric mean* of the ratios compared to Plain;
(e.g., S-Rank needs only 0.47 times the number of column-generation iterations for solving the root node compared to Plain)

Number of iterations: The number of column-generation iterations performed (we present the *total* number and the number of times the *exact* pricing problem over the full network was solved);

Number of optima: Number of instances solved to proven optimality;

Time: Overall computation time (in seconds).

All five strategies reduce the total number of pricing iterations. This causes the observed reduction in computation times compared to Plain. On the other hand, all variants except for S-Rank need more exact pricing iterations. This effect is more prominent at the root node (ratios between 1.41 and 1.55) than for the complete branch-and-bound tree (ratios between 1.05 and 1.09). Only S-Rank requires less exact pricing iterations compared to Plain, at the root as well as in the tree. Therefore, the different performance shown in the profiles of Figures 3 and 4 can be explained with two concurring effects. Overall, the use of more than just the ranking inequalities (variant S-Rank) better stabilizes the RMP leading to a smaller total number of iterations, but more exact pricing iterations have to be performed. These additional exact pricing iterations occur more at the root node than in the branch-and-bound tree.

As stated above, on the basis of the performance profiles depicted in Figure 4, strategy S-Rank-S/D-Subs seems to be superior to the other strategies. This is also confirmed by the results shown in Table 2 where S-Rank-D-Subs is best w.r.t. both the number of optima and computation time (95 optima and 0.66 time ratio w.r.t. Plain). Therefore, the following experiments choose S-Rank-S/D-Subs as the stabilized version of the BPC.

Next, we show more detailed results comparing Plain and the winning variant S-Rank-S/D-Subs. Table 3 summarizes the results obtained on all 160 benchmark instances, grouped by the number κ of commodities and the number n of customers. Each group comprises 20 instances with the exception of the group for $n = 40$ and $\kappa = 6$, for which Plain was not able to solve the linear relaxation at the root node in one out of 20 cases. Again, Plain serves as the baseline algorithm and we provide absolute values for it, while values for S-Rank-S/D-Subs, apart from those reported in column *opt*, are given as geometric means of the ratios relative to Plain. The columns with header *Root* refer to the solution of the linear relaxation and those with header *Tree* to the full BPC. The columns of the table have the following meaning:

#: Number of instances for which the linear relaxation is solved by both algorithms, all averages in section "Root" are taken only over those instances;

iter: Number of column-generation iterations;

cols: Number of columns (of routes as well as of DIs) in the RMP at termination;

			Root										Tree			
			`Plain`					`S-Rank-S/D-Subs`								`S-Rank-`
			Number of (absolute)		Time for (absolute)			Number of (ratio)		Time for (ratio)			`Plain`	(abs.)		`S/D-Subs` (rat.)
κ	n	#	iter	cols	*PP*	*RMP*	root	iter	cols	*PP*	*RMP*	root	opt	time	opt	time
3	20	20	58.9	1725.9	0.1	0.1	0.1	0.63	0.63	0.90	0.88	0.81	20	42.4	20	0.73
3	40	20	113.5	5352.5	1.3	1.3	1.5	0.61	0.59	0.91	0.36	0.85	8	510.8	8	0.75
Total/Avg.		40	86.2	3539.2	0.7	0.7	0.8	0.62	0.61	0.90	0.56	0.83	28	176.2	28	0.74
4	20	20	106.2	4107.3	0.8	0.8	0.9	0.46	0.46	0.76	0.40	0.71	18	152.3	19	0.68
4	40	20	184.8	9114.4	6.0	6.0	6.5	0.51	0.52	0.90	0.24	0.85	9	656.2	9	0.98
Total/Avg.		40	145.5	6610.9	3.4	3.4	3.7	0.49	0.49	0.82	0.31	0.78	27	320.2	28	0.77
5	20	20	172.9	6711.1	2.1	2.1	2.4	0.37	0.39	0.81	0.24	0.72	15	216.5	16	0.61
5	40	20	310.3	16328.6	22.7	22.7	24.2	0.42	0.45	0.86	0.18	0.82	4	1773.2	4	0.55
Total/Avg.		40	241.6	11519.8	12.4	12.4	13.3	0.39	0.42	0.83	0.21	0.77	19	544.2	20	0.60
6	20	20	310.6	11638.6	9.8	9.8	10.7	0.30	0.35	0.69	0.14	0.64	15	534.8	16	0.45
6	40	19	859.1	60695.9	458.3	458.3	466.0	0.28	0.31	0.44	0.14	0.43	2	3140.1	3	0.90
Total/Avg.		39	577.8	35538.3	228.3	228.3	232.5	0.29	0.33	0.56	0.14	0.53	17	841.3	19	0.49
Total/Avg.		159	260.8	14168.5	60.2	60.2	61.5	0.43	0.45	0.77	0.27	0.72	91	420.0	95	0.66

Table 3: Comparison of the two BPC algorithms `Plain` and `S-Rank-S/D-Subs` on the benchmark instances with $n = 20$ and 40 customers

PP: Accumulated time (in seconds) for the solution of the pricing subproblems;
RMP: Accumulated time (in seconds) for the reoptimization of the RMP;
root: Total time (in seconds) for solving the linear relaxation;
opt: Number of instances solved to proven optimality;
time: Total time (in seconds) for solving branch-and-bound tree, averages are taken only over those instances solved optimally by both.

We start with observations regarding the solution of the root node: The larger the number n of customer and the number κ of commodities, the more iterations are needed and more columns are generated with the `Plain` BPC. The same is true for the pricing and re-optimization times and, therefore, for the overall root computation times. For a constant number of tasks, e.g., 120 tasks, a comparison of the rows for $\kappa = 3$ and $n = 40$ and for $\kappa = 6$ and $n = 20$ reveals that the latter instances are significantly more difficult for `Plain`. This is due to the higher symmetry in the latter group of instances with $\kappa = 6$ commodities. The columns for the `S-Rank-S/D-Subs` show that the number of iterations and generated columns as well as the computation time for pricing and re-optimization of the RMP are clearly reduced by adding DI columns. Larger instances benefit more from stabilization than smaller instances. Indeed, for the largest instances with $\kappa = 6$ and $n = 40$, i.e., 240 tasks, the average time ratio for the overall root computation goes down to 0.43. Comparing PP and RMP times, the RMP times decrease more. The point is probably that PP times are dominated by the calls to the exact labeling algorithm over the full network. These exact iterations cannot be avoided with stabilization, but the number of iterations and generated columns is reduced so that RMP re-optimization times are reduced by a larger extent.

The comparison of the `Plain` and the `S-Rank-S/D-Subs` regarding the branch-and-bound tree shows the following: In total, computation times of the stabilized BPC are reduced to 66 % on average over all instances. The faster computation times allow the variant `S-Rank-S/D-Subs` to compute four more optimal solutions (95 compared to 91 computed with `Plain`).

4.3. Comparison with Results of Archetti et al. *(2015)*

In this section we compare our chosen variant `S-Rank-S/D-Subs` of the BPC with the implementation of Archetti *et al.* (2015) which does not use any stabilization technique. All instances used in the computational study of Archetti *et al.* (2015) have two or three commodities ($\kappa = 2$ or 3). Since we focus on the stabilization aspect, only the instances with $\kappa = 3$ are interesting for a detailed comparison. In contrast to the previous Section 4.2, however, we use all the available instances with $n = 20, 40, 60$, and 80 customers.

Tables 4 and 5 present instance-by-instance results for the 80 instances, grouped by the original Solomon instance, number of customers n, and commodity probability p. The other columns of the tables have the following meaning:

Instance					(Archetti *et al.*, 2015)			`S-Rank-S/D-Subs`				`S-Rank-S/D-Subs` 4 hours			
	n	p	No.	$\overline{z}^*$	$\underline{z}^*$	gap	time	$\underline{z}^*$	gap	time	nodes	$\underline{z}^*$	gap	time	nodes
C101	20	0.6	1	573.86	573.86	(opt)	1.1	573.86	(opt)	0.1	1	573.86	(opt)	0.1	1
			2	592.07	592.07	(opt)	1.5	592.07	(opt)	0.1	1	592.07	(opt)	0.1	1
			3	595.53	595.53	(opt)	58.2	595.53	(opt)	5.4	54	595.53	(opt)	5.4	54
			4	617.88	617.88	(opt)	5.4	617.88	(opt)	0.7	12	617.88	(opt)	0.7	12
			5	628.28	628.28	(opt)	13.8	628.28	(opt)	2.3	36	628.28	(opt)	2.3	36
		1	1	750.63	750.63	(opt)	13.5	750.63	(opt)	0.4	6	750.63	(opt)	0.4	6
			2	714.65	714.65	(opt)	12.1	714.65	(opt)	0.7	5	714.65	(opt)	0.7	5
			3	626.16	626.16	(opt)	1614.5	626.16	(opt)	119.0	259	626.16	(opt)	119.0	259
			4	747.70	747.70	(opt)	109.2	747.70	(opt)	5.2	26	747.70	(opt)	5.2	26
			5	768.52	768.52	(opt)	21.6	768.52	(opt)	4.2	48	768.52	(opt)	4.2	48
R101	20	0.6	1	457.86	457.86	(opt)	6.0	457.86	(opt)	0.3	8	457.86	(opt)	0.3	8
			2	667.01	667.01	(opt)	2.8	667.01	(opt)	0.7	10	667.01	(opt)	0.7	10
			3	455.05	455.05	(opt)	1.3	455.05	(opt)	0.2	3	455.05	(opt)	0.2	3
			4	589.91	589.91	(opt)	3.7	589.91	(opt)	0.5	9	589.91	(opt)	0.5	9
			5	663.22	663.22	(opt)	1.1	663.22	(opt)	0.1	7	663.22	(opt)	0.1	7
		1	1	599.84	599.84	(opt)	6.6	599.84	(opt)	0.7	6	599.84	(opt)	0.7	6
			2	863.88	863.88	(opt)	30.0	863.88	(opt)	4.3	11	863.88	(opt)	4.3	11
			3	617.12	615.35	0.29	TL2	*617.12*	*(opt)*	*420.1*	694	617.12	(opt)	420.1	694
			4	712.02	712.02	(opt)	24.3	712.02	(opt)	2.6	18	712.02	(opt)	2.6	18
			5	794.41	794.41	(opt)	4120.0	794.41	(opt)	79.4	496	794.41	(opt)	79.4	496
	20	*Avg.*				0.03	318.2		0.00	11.9	86		0.00	11.9	86
C101	40	0.6	1	841.02	838.44	0.31	TL2	840.30	0.09	TL1	932	841.02	(opt)	10707.9	1900
			2	*1006.47	990.26	1.61	TL2	992.06	1.43	TL1	1972	993.23	1.32	TL4	5442
			3	879.26	879.26	(opt)	1538.4	879.26	(opt)	10.8	11	879.26	(opt)	10.8	11
			4	921.06	920.92	0.02	TL2	*921.06*	*(opt)*	*195.9*	335	921.06	(opt)	195.9	335
			5	868.74	868.74	(opt)	2939.9	868.74	(opt)	161.5	163	868.74	(opt)	161.5	163
		1	1	*1370.84	1300.31	5.15	TL2	1301.90	5.03	TL1	3142	1302.75	4.97	TL4	8869
			2	1357.79	1354.52	0.24	TL2	1357.50	0.02	TL1	2149	1357.79	(opt)	6216.7	3228
			3	1299.43	1298.31	0.09	TL2	1299.40	0.00	TL1	1489	1299.43	(opt)	3998.8	1568
			4	1236.16	1231.91	0.34	TL2	1234.36	0.15	TL1	2817	1235.00	0.09	TL4	7854
			5	*1362.22	1263.33	7.26	TL2	1265.71	7.08	TL1	2302	1266.55	7.02	TL4	7014
R101	40	0.6	1	761.78	759.22	0.34	TL2	*761.78*	*(opt)*	*1462.4*	833	761.78	(opt)	1442.9	833
			2	896.01	895.93	0.01	TL2	*896.02*	*(opt)*	*632.4*	981	896.02	(opt)	632.4	981
			3	851.03	851.03	(opt)	2469.4	851.03	(opt)	71.4	143	851.03	(opt)	71.4	143
			4	973.48	970.62	0.29	TL2	971.85	0.17	TL1	4390	972.71	0.08	TL4	11148
			5	854.35	854.35	(opt)	2152.5	854.35	(opt)	149.7	160	854.35	(opt)	149.7	160
		1	1	*1238.25	1232.66	0.45	TL2	1235.80	0.20	TL1	2613	1237.11	0.09	TL4	7241
			2	*1276.11	1234.05	3.30	TL2	1235.80	3.16	TL1	2116	1236.78	3.08	TL4	5442
			3	1056.13	1056.13	(opt)	2411.2	1056.13	(opt)	291.4	310	1056.13	(opt)	291.4	310
			4	1244.14	1242.08	0.17	TL2	1243.06	0.09	TL1	3504	1244.14	(opt)	13141.9	9450
			5	*1130.01	1096.02	3.01	TL2	1097.96	2.84	TL1	1681	1098.87	2.76	TL4	5028
	40	*Avg.*				2.26	2302.3		2.02	137.0	1602		1.94	137.0	3856

Table 4: Comparison with results of Archetti *et al.* (2015), mid-size instances with $\kappa = 3$ (Part 1).

$\underline{z}^*$: Lower bound obtained by BPC at time of termination;
$\overline{z}^*$: The best solution provided by one of the BPC algorithms included in the table, or if marked with an asterisk * by an extra run of `S-Rank-S/D-Subs` with an mixed depth/best first tree exploration strategy;
gap: The percentage gap $100 \cdot (\overline{z}^* - \underline{z}^*)/\overline{z}^*$ if not solved to proven optimality (*opt*), averages are taken over the instances for which all algorithms could provide lower bounds;
time: Total computation time in seconds, averages are taken over the instances that were solved to optimality by all algorithms;
nodes: Number of solved nodes by BPC at time of termination, averages are taken over all instances.

Note that Archetti *et al.* (2015) have set the computation time limit TL2 to two hours (7200 seconds), while we set the time limit TL1 for variant `S-Rank-S/D-Subs` to one hour (3600 seconds) to account for the better computer used in our experiments.

Over the benchmark, our stabilized BPC algorithm solves 32 out of 80 instances optimally, while the unstabilized algorithm of Archetti *et al.* (2015) solves 24. In particular, our algorithm provides 4 out of 20 optima for the 60-customer instances on which the unstabilized algorithm was not able to prove any

Instance					(Archetti *et al.*, 2015)			`S-Rank-S/D-Subs`				`S-Rank-S/D-Subs` 4 hours			
	n	p	No.	$\overline{z}^*$	$\underline{z}^*$	gap	time	$\underline{z}^*$	gap	time	nodes	$\underline{z}^*$	gap	time	nodes
C101	60	0.6	1	1221.36	1216.34	0.41	TL2	1220.10	0.10	TL1	563	1221.36	(opt)	11703.7	1404
			2	1331.77	1330.13	0.12	TL2	*1331.77*	*(opt)*	*3195.5*	599	1331.77	(opt)	3195.5	599
			3	1180.61	1180.29	0.03	TL2	*1180.61*	*(opt)*	*275.6*	64	1180.61	(opt)	275.6	64
			4	1282.72	1281.42	0.10	TL2	*1282.72*	*(opt)*	*1049.2*	264	1282.72	(opt)	1049.2	264
			5	*1339.02	1294.56	3.32	TL2	1299.07	2.98	TL1	957	1299.98	2.92	TL4	2924
		1	1	*2049.20	1966.87	4.02	TL2	1969.31	3.90	TL1	1016	1970.02	3.86	TL4	3626
			2	*1787.11	1657.61	7.25	TL2	1659.11	7.16	TL1	629	1659.90	7.12	TL4	2427
			3	*1897.73	1782.65	6.06	TL2	1784.31	5.98	TL1	1131	1785.08	5.94	TL4	3980
			4	*1936.13	1890.51	2.36	TL2	1896.33	2.06	TL1	1177	1896.99	2.02	TL4	3901
			5	*1773.78	1618.64	8.75	TL2	1625.32	8.37	TL1	563	1626.47	8.30	TL4	2426
R101	60	0.6	1	1284.99	1279.20	0.45	TL2	1282.51	0.19	TL1	1372	1284.44	0.04	TL4	5353
			2	1310.86	1306.21	0.35	TL2	1308.09	0.21	TL1	1601	1309.05	0.14	TL4	3952
			3	1028.52	1028.32	0.02	TL2	*1028.52*	*(opt)*	*331.6*	245	1028.52	(opt)	331.6	245
			4	1221.57	1215.90	0.46	TL2	1219.15	0.20	TL1	1826	1220.17	0.11	TL4	5140
			5	1149.70	1146.77	0.25	TL2	1148.05	0.14	TL1	1246	1149.26	0.04	TL4	3564
		1	1	*2135.42	-	-	TL2	2040.84	4.43	TL1	2398	2042.20	4.37	TL4	8307
			2	*1706.34	1633.53	4.27	TL2	1639.08	3.94	TL1	2436	1640.03	3.89	TL4	7881
			3	*1609.37	1544.33	4.04	TL2	1547.39	3.85	TL1	2280	1548.50	3.78	TL4	7228
			4	*1748.82	1694.81	3.09	TL2	1708.22	2.32	TL1	1346	1709.38	2.26	TL4	4446
			5	*1505.13	1487.25	1.19	TL2	1490.34	0.98	TL1	876	1491.57	0.90	TL4	3303
	60	*Avg.*				2.45	-		2.23	-	1129		2.17	-	3552
C101	80	0.6	1	*1697.63	1629.59	4.01	TL2	1632.21	3.85	TL1	480	1633.24	3.79	TL4	1975
			2	*1637.98	1588.94	2.99	TL2	1595.50	2.59	TL1	668	1596.31	2.54	TL4	2697
			3	*1784.51	1726.68	3.24	TL2	1731.77	2.96	TL1	617	1732.86	2.89	TL4	2301
			4	*1510.45	1430.96	5.26	TL2	1440.28	4.65	TL1	435	1441.19	4.59	TL4	1705
			5	*1783.91	1651.33	7.43	TL2	1662.12	6.83	TL1	460	1663.11	6.77	TL4	1808
		1	1	*2327.33	-	-	TL2	2226.65	4.33	TL1	304	2227.37	4.30	TL4	1321
			2	*2302.58	1992.49	13.47	TL2	2095.53	8.99	TL1	277	2096.99	8.93	TL4	1378
			3	*2750.59	-	-	TL2	2574.99	6.38	TL1	805	2575.56	6.36	TL4	2889
			4	*2509.90	-	-	TL2	2343.46	6.63	TL1	435	2344.22	6.60	TL4	1793
			5	*2507.19	-	-	TL2	2387.96	4.76	TL1	494	2388.47	4.74	TL4	2268
R101	80	0.6	1	*1458.72	1430.71	1.92	TL2	1434.71	1.65	TL1	1048	1436.16	1.55	TL4	4053
			2	*1509.04	1459.24	3.30	TL2	1462.05	3.11	TL1	1197	1463.17	3.04	TL4	4550
			3	*1643.03	1583.16	3.64	TL2	1585.92	3.48	TL1	1980	1587.42	3.38	TL4	6985
			4	*1442.54	1400.02	2.95	TL2	1404.23	2.66	TL1	611	1405.40	2.57	TL4	2440
			5	*1523.49	-	-	TL2	1451.05	4.75	TL1	144	1451.05	4.75	TL4	144
		1	1	*2214.27	-	-	TL2	2084.94	5.84	TL1	934	2085.99	5.79	TL4	3209
			2	*1972.32	-	-	TL2	1913.71	2.97	TL1	907	1915.06	2.90	TL4	3246
			3	*2338.46	-	-	TL2	2261.39	3.30	TL1	1121	2262.44	3.25	TL4	4216
			4	*2191.98	-	-	TL2	2080.61	5.08	TL1	547	2081.89	5.02	TL4	2556
			5	*2183.39	-	-	TL2	2119.67	2.92	TL1	1059	2120.65	2.87	TL4	3896
	80	*Avg.*				4.82	-		4.08	-	726		4.01	-	2772

Table 5: Comparison with results of Archetti *et al.* (2015), mid-size instances with $\kappa = 3$ (Part 2).

optimality. All 80-customer remain unsolved, but we are able to provide lower bounds in all cases. Although there were only modest improvements in terms of the number of optimal solutions found and quality of the bounds computed, the results concerning the 20- and 40-customer instances show how `S-Rank-S/D-Subs` is about one order of magnitude faster than the unstabilized algorithm of Archetti *et al.* (2015).

In order assess the performance of `S-Rank-S/D-Subs` with a larger time limit and to get a clearer picture why the larger instances cannot be solved, we performed an additional run of `S-Rank-S/D-Subs` on the unsolved instances with an extended time limit of four hours (TL4). These results are also shown in Tables 4 and 5. With the longer runtime, five additional instances are solved optimally, while most of the 60-customer and all 80-customer instances remain unsolved. Lower bounds are improved only slightly with the additional computation time, although `S-Rank-S/D-Subs` solves on average around 3.5 times more branch-and-bound nodes. We attribute this behavior to the inherent symmetry of the C-SDVRP that makes the problem hard to solve with an increased number of tasks to perform.

4.4. Results for New Instances with More Commodities

As mentioned at the beginning of the section, we created 96 small ($n = 15$) and 240 mid-size ($n = 20, 40, 60$, and 80) new C-SDVRP instance with four to six commodities ($\kappa \in \{4, 5, 6\}$). The new benchmark

is available at `http://logistik.bwl.uni-mainz.de/benchmarks.php`.

Instances			Results								
κ	n	#	gap	time	opt	gap^{LP}	nodes	CC	SR	SD	rec
4	15	32	0.41	117.0	31	5.10	182.6	26.5	13.4	0.1	2.2
5	15	32	–	39.0	32	5.53	47.8	33.5	17.0	0.1	2.7
6	15	32	–	284.2	31	5.93	35.5	35.5	22.2	0.2	3.6
4	20	20	–	355.6	19	3.93	240.4	47.9	19.4	2.6	6.0
4	40	20	0.11	2326.5	9	4.24	813.0	176.5	30.0	80.1	51.7
4	60	20	–	3486.5	2	4.82	730.5	245.7	30.0	125.3	44.0
4	80	20	–	3600.0	0	–	412.0	266.4	30.0	90.6	24.3
5	20	20	0.33	824.8	16	5.18	372.3	60.5	25.8	19.2	13.3
5	40	20	0.18	3072.2	4	3.81	657.6	184.3	30.0	72.8	69.9
5	60	20	–	3600.0	0	–	324.0	237.7	30.0	67.8	45.0
5	80	20	–	3600.0	0	–	188.2	276.6	30.0	34.1	24.2
6	20	20	0.13	999.0	16	6.12	184.2	73.6	29.5	6.9	13.6
6	40	20	–	3430.9	3	4.47	352.8	180.9	30.0	43.6	52.2
6	60	20	–	3600.0	0	–	198.5	254.8	27.5	36.0	34.3
6	80	20	–	3600.0	0	–	78.4	243.5	22.5	10.2	19.0

Table 6: Aggregated results for new instances with more commodities

Table 6 shows aggregated results whereas the Appendix gives instance-by-instance results, i.e., Tables 7–9 for the small instances and Tables 10–17 for the mid-size instances. The additional columns of the tables have the following meaning:

- $\underline{z}^{\text{LP}}$: Linear-relaxation lower bound of the RMP;
- gap^{LP}: The percentage gap $100 \cdot (\overline{z}^* - \underline{z}^{\text{LP}})/\overline{z}^*$ of the linear-relaxation lower bound;
- CC: Number of generated CC by BPC at time of termination;
- SR: Number of generated SR inequalities by BPC at time of termination;
- SD: Number of generated SD constraints by BPC at time of termination;
- rec: Number of times the recovery procedure found an overstabilized RMP solution in the BPC at time of termination.

Overall, we are able to solve all but two of the 96 small instances with 15 customers and most (51 of 60) of the 20 customer instances. For the larger instances, the number of optima proven decreases significantly: only 16 and two optima for the 40 and 60 customer instances, respectively, while we cannot find any optimal solution for the instances with 80 customers. Lower bounds, however, are provided for all instances.

5. Conclusions

In this paper, we have developed a new BPC algorithm tailored to the commodity-constrained split delivery vehicle routing problem (C-SDVRP). The main novelty is the use of dual-optimal inequalities for the stabilization of the column-generation process. We have focused on the interaction of branching and cutting decisions and the different classes of dual inequalities. For the first time, non-robust VRP cuts such as subset-row inequalities and strong-degree constraints have been used in conjunction with dual subset inequalities that were originally introduced for packing and cutting applications. In order to keep as many dual inequalities as possible in the overall branch-and-bound tree, we have derived precise rules for their validity in the presence of branching and cutting constraints.

In an extensive computational analysis, we compared different strategies for the a-priori addition and dynamic generation of dual inequalities. A first result is that all strategies that employ stabilization based on dual inequalities clearly outperform the unstabilized version of the BPC algorithm. Interestingly, the BPC algorithms with the analyzed strategies perform differently when solving the linear relaxation and within the branch-and-bound tree. A key observation in this context is that certain strategies require less overall column-generation iterations, but more calls to the exact pricing algorithm. This effect is, in the C-SDVRP, more pronounced at the root node resulting in the observed and different behaviors of the strategies.

Our best strategy is one that a priori adds dual ranking inequalities and a few dual subset inequalities, but identifies the most violated dual subset inequalities in each column-generation iteration and adds them to the RMP. Comparing this strategy to the state-of-the-art algorithm on the existing benchmark, we are significantly faster. We have computed several new proven optima for instances with up to 60 customers and 180 tasks. In addition, we have determined lower bounds for all tested instances with up to 80 customers and 480 tasks. This improves the bounds for all unsolved instances and proves lower bounds for the first time for several instances.

Acknowledgement

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

References

Archetti, C., Bianchessi, N., and Speranza, M. (2015). A branch-price-and-cut algorithm for the commodity constrained split delivery vehicle routing problem. *Computers and Operations Research*, **64**, 1–10.

Archetti, C., Campbell, A. M., and Speranza, M. G. (2016). Multicommodity vs. single-commodity routing. *Transportation Science*, **50**(2), 461–472.

Baldacci, R., Mingozzi, A., and Roberti, R. (2011). New route relaxation and pricing strategies for the vehicle routing problem. *Operations Research*, **59**(5), 1269–1283.

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

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

Contardo, C., Cordeau, J.-F., and Gendron, B. (2014). An exact algorithm based on cut-and-column generation for the capacitated location-routing problem. *INFORMS Journal on Computing*, **26**(1), 88–102.

Costa, L., Contardo, C., and Desaulniers, G. (2018). Exact branch-price-and-cut algorithms for vehicle routing. *Transportation Science*. Forthcoming.

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

Dolan, E. D. and Moré, J. J. (2002). Benchmarking optimization software with performance profiles. *Mathematical Programming*, **91**(2), 201–213.

Fukasawa, R., Longo, H., Lysgaard, J., Poggi de Aragão, M., Reis, M., Uchoa, E., and Werneck, R. F. (2006). Robust branch-and-cut-and-price for the capacitated vehicle routing problem. *Mathematical Programming, Series A*, **106**(3), 491–511.

Goeke, D., Gschwind, T., and Schneider, M. (2018). Upper and lower bounds for the vehicle-routing problem with private fleet and common carrier. *Discrete Applied Mathematics*. Forthcoming.

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

Irnich, S. and Desaulniers, G. (2005). Shortest path problems with resource constraints. In G. Desaulniers, J. Desrosiers, and M. Solomon, editors, *Column Generation*, chapter 2, pages 33–65. Springer.

Irnich, S., Schneider, M., and Vigo, D. (2014). *Four Variants of the Vehicle Routing Problem*, chapter 9, pages 241–271. Society for Industrial and Applied Mathematics, Philadelphia, PA.

Jepsen, M., Petersen, B., Spoorendonk, S., and Pisinger, D. (2008). Subset-row inequalities applied to the vehicle-routing problem with time windows. *Operations Research*, **56**(2), 497–511.

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

Nakao, Y. and Nagamochi, H. (2007). A DP-based heuristic algorithm for the discrete split delivery vehicle routing problem. *Journal of Advanced Mechanical Design, Systems, and Manufacturing*, **1**(2), 217–226.

Poggi, M. and Uchoa, E. (2014). *New Exact Algorithms for the Capacitated Vehicle Routing Problem*, chapter 3, pages 59–86. Society for Industrial and Applied Mathematics, Philadelphia, PA.

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.

Salani, M. and Vacca, I. (2011). Branch and price for the vehicle routing problem with discrete split deliveries and time windows. *European Journal of Operational Research*, **213**(3), 470–477.

Semet, F., Toth, P., and Vigo, D. (2014). *Classical Exact Algorithms for the Capacitated Vehicle Routing Problem*, chapter 2, pages 37–57. Society for Industrial and Applied Mathematics, Philadelphia, PA.

Solomon, M. (1987). Algorithms for vehicle routing and scheduling problems with time window constraints. *Operations Research*, **35**, 254–266.

Toth, P. and Vigo, D., editors (2014). *Vehicle Routing*. MOS-SIAM Series on Optimization. Society for Industrial and Applied Mathematics, Philadelphia, PA.

This appendix is supposed to become online supplementary material.

Appendix

I. Detailed Results for the Small Instances ($n = 15$)

Instance				S-Rank-S/D-Subs										
	p	d_i	β	$\underline{z}^*$	$\overline{z}^*$	gap	time	$\underline{z}^{\mathrm{LP}}$	$\mathrm{gap}^{\mathrm{LP}}$	nodes	CC	SR	SD	rec
C101	0.6	[1;100]	1.1	450.69	450.69	(opt)	0.7	421.56	6.46	9	32	11	0	0
	0.6	[40;60]	1.1	410.08	410.08	(opt)	0.1	404.7	1.31	1	13	0	0	0
	1	[1;100]	1.1	490.56	490.56	(opt)	1.5	477.17	2.73	7	57	14	0	2
	1	[40;60]	1.1	624.41	624.41	(opt)	0.2	604.04	3.26	3	32	0	0	0
	0.6	[1;100]	1.5	323.13	323.13	(opt)	0.4	318.31	1.49	7	28	10	0	1
	0.6	[40;60]	1.5	341.26	341.26	(opt)	0.4	329.42	3.47	7	29	26	0	0
	1	[1;100]	1.5	343.63	343.63	(opt)	8.0	306.49	10.81	27	38	30	0	4
	1	[40;60]	1.5	464.62	466.53	0.41	TL1	447.61	4.06	5454	61	30	2	43
	0.6	[1;100]	2.0	264.13	264.13	(opt)	0.6	244.66	7.37	1	15	0	0	0
	0.6	[40;60]	2.0	264.13	264.13	(opt)	0.8	229.27	13.20	6	16	10	0	0
	1	[1;100]	2.0	299.38	299.38	(opt)	17.3	258.44	13.67	36	35	30	0	4
	1	[40;60]	2.0	349.53	349.53	(opt)	1.5	340.78	2.50	5	26	20	0	0
	0.6	[1;100]	2.5	203.33	203.33	(opt)	4.1	183.98	9.52	22	20	30	0	3
	0.6	[40;60]	2.5	231.31	231.31	(opt)	3.9	203.96	11.82	19	26	30	0	0
	1	[1;100]	2.5	263.51	263.51	(opt)	26.3	222.89	15.41	10	29	22	0	3
	1	[40;60]	2.5	311.30	311.30	(opt)	8.5	282.78	9.16	35	36	30	0	2
R101	0.6	[1;100]	1.1	369.77	369.77	(opt)	0.1	364.9	1.32	2	23	0	0	1
	0.6	[40;60]	1.1	626.09	626.09	(opt)	0.3	612.06	2.24	27	27	30	0	0
	1	[1;100]	1.1	549.44	549.44	(opt)	0.1	538.68	1.96	1	25	0	0	1
	1	[40;60]	1.1	672.59	672.59	(opt)	0.1	658.59	2.08	7	39	4	0	0
	0.6	[1;100]	1.5	435.77	435.77	(opt)	5.0	410.34	5.84	14	35	30	0	1
	0.6	[40;60]	1.5	432.91	432.91	(opt)	0.1	425.27	1.76	7	21	4	0	0
	1	[1;100]	1.5	418.08	418.08	(opt)	0.1	401.05	4.07	2	29	0	0	0
	1	[40;60]	1.5	558.96	558.96	(opt)	56.1	538.65	3.63	27	46	30	0	0
	0.6	[1;100]	2.0	316.20	316.20	(opt)	0.1	307.4	2.78	3	6	0	0	0
	0.6	[40;60]	2.0	321.96	321.96	(opt)	0.1	318.31	1.13	1	7	0	0	0
	1	[1;100]	2.0	376.27	376.27	(opt)	0.1	365.95	2.74	1	13	0	0	1
	1	[40;60]	2.0	444.19	444.19	(opt)	0.0	437.64	1.47	1	12	0	0	0
	0.6	[1;100]	2.5	277.82	277.82	(opt)	0.1	264.24	4.89	2	2	4	0	0
	0.6	[40;60]	2.5	348.97	348.97	(opt)	0.1	336.19	3.66	2	18	0	0	0
	1	[1;100]	2.5	344.36	344.36	(opt)	0.2	329.7	4.26	10	23	5	0	3
	1	[40;60]	2.5	402.44	402.44	(opt)	8.4	389.49	3.22	87	30	30	0	1

Table 7: Detailed results for the small instances ($n = 15$) with $\kappa = 4$ commodities.

Instance				S-Rank-S/D-Subs										
	p	d_i	β	$\underline{z}^*$	$\overline{z}^*$	gap	time	$\underline{z}^{\mathrm{LP}}$	$\mathrm{gap}^{\mathrm{LP}}$	nodes	CC	SR	SD	rec
C101	0.6	[1;100]	1.1	393.31	393.31	(opt)	83.3	373.98	4.91	64	37	30	0	1
	0.6	[40;60]	1.1	489.76	489.76	(opt)	0.2	462.39	5.59	7	46	0	0	0
	1	[1;100]	1.1	489.03	489.03	(opt)	49.6	442.21	9.57	64	73	30	0	9
	1	[40;60]	1.1	634.32	634.32	(opt)	140.2	617.5	2.65	424	75	30	2	5
	0.6	[1;100]	1.5	266.16	266.16	(opt)	2.1	244.6	8.10	2	12	10	0	1
	0.6	[40;60]	1.5	376.32	376.32	(opt)	0.3	336.29	10.64	2	22	0	0	0
	1	[1;100]	1.5	382.71	382.71	(opt)	514.3	358.63	6.29	148	24	30	0	13
	1	[40;60]	1.5	457.51	457.51	(opt)	49.4	435.91	4.72	107	70	30	0	2
	0.6	[1;100]	2.0	201.12	201.12	(opt)	6.5	176.82	12.08	1	3	0	0	1
	0.6	[40;60]	2.0	265.02	265.02	(opt)	3.2	223.45	15.69	19	18	30	0	0
	1	[1;100]	2.0	298.43	298.43	(opt)	65.2	262.02	12.20	21	32	30	0	9
	1	[40;60]	2.0	349.40	349.40	(opt)	15.7	343.48	1.69	17	24	30	0	1
	0.6	[1;100]	2.5	169.47	169.47	(opt)	10.4	169.19	0.17	1	1	0	0	1
	0.6	[40;60]	2.5	232.49	232.49	(opt)	1.0	209.28	9.98	10	18	30	0	0
	1	[1;100]	2.5	232.22	232.22	(opt)	152.2	209.79	9.66	5	16	0	0	4
	1	[40;60]	2.5	311.00	311.00	(opt)	38.2	277.07	10.91	24	44	30	0	0
R101	0.6	[1;100]	1.1	480.36	480.36	(opt)	0.2	468.31	2.51	8	48	10	0	2
	0.6	[40;60]	1.1	540.78	540.78	(opt)	0.1	525.59	2.81	5	27	15	0	0
	1	[1;100]	1.1	546.37	546.37	(opt)	22.2	520.15	4.80	41	68	30	0	7
	1	[40;60]	1.1	665.12	665.12	(opt)	11.9	643.58	3.24	142	90	30	0	7
	0.6	[1;100]	1.5	406.66	406.66	(opt)	0.2	386.49	4.96	6	29	0	0	3
	0.6	[40;60]	1.5	393.54	393.54	(opt)	0.1	377.83	3.99	3	41	0	0	0
	1	[1;100]	1.5	454.76	454.76	(opt)	0.2	441.94	2.82	1	22	0	0	2
	1	[40;60]	1.5	566.45	566.45	(opt)	56.2	542.33	4.26	235	78	30	0	7
	0.6	[1;100]	2.0	343.93	343.93	(opt)	0.1	338.18	1.67	1	10	0	0	1
	0.6	[40;60]	2.0	398.16	398.16	(opt)	2.1	381.39	4.21	39	36	30	0	0
	1	[1;100]	2.0	351.19	351.19	(opt)	0.5	338.65	3.57	3	13	0	0	3
	1	[40;60]	2.0	443.56	443.56	(opt)	0.8	430.58	2.93	10	31	30	0	0
	0.6	[1;100]	2.5	324.38	324.38	(opt)	0.1	322.4	0.61	1	9	0	0	1
	0.6	[40;60]	2.5	331.78	331.78	(opt)	0.2	321.96	2.96	5	8	20	0	0
	1	[1;100]	2.5	339.66	339.66	(opt)	0.8	329.29	3.05	7	5	10	0	4
	1	[40;60]	2.5	399.76	399.76	(opt)	21.7	385.2	3.64	108	42	30	0	2

Table 8: Detailed results for the small instances ($n = 15$) with $\kappa = 5$ commodities.

Instance				S-Rank-S/D-Subs										
	p	d_i	β	$\underline{z}^*$	$\overline{z}^*$	gap	time	$\underline{z}^{\mathrm{LP}}$	$\mathrm{gap}^{\mathrm{LP}}$	nodes	CC	SR	SD	rec
C101	0.6	[1;100]	1.1	455.52	455.52	(opt)	40.8	418.69	8.09	124	41	30	0	8
	0.6	[40;60]	1.1	447.61	447.61	(opt)	0.4	416.45	6.96	4	27	10	0	0
	1	[1;100]	1.1	502.64	–	–	TL1	486.62	–	362	68	30	0	14
	1	[40;60]	1.1	628.13	628.13	(opt)	8.2	613.13	2.39	34	32	30	0	2
	0.6	[1;100]	1.5	343.38	343.38	(opt)	2.6	308.43	10.18	3	14	0	0	2
	0.6	[40;60]	1.5	350.85	350.85	(opt)	2.4	315.03	10.21	12	30	30	0	0
	1	[1;100]	1.5	416.85	416.85	(opt)	2463.9	385.12	7.61	46	53	30	5	14
	1	[40;60]	1.5	490.30	490.30	(opt)	43.7	460.17	6.15	59	63	30	2	0
	0.6	[1;100]	2.0	265.56	265.56	(opt)	155.6	218.03	17.90	36	34	30	0	11
	0.6	[40;60]	2.0	307.98	307.98	(opt)	5.7	283.44	7.97	3	29	0	0	0
	1	[1;100]	2.0	308.81	308.81	(opt)	379.0	269.77	12.64	27	40	30	0	9
	1	[40;60]	2.0	377.20	377.20	(opt)	9.0	352.89	6.44	2	14	10	0	0
	0.6	[1;100]	2.5	169.23	169.23	(opt)	92.4	164.03	3.07	2	16	0	0	0
	0.6	[40;60]	2.5	272.20	272.20	(opt)	12.3	237.13	12.88	40	27	30	0	0
	1	[1;100]	2.5	239.32	239.32	(opt)	1534.9	217.71	9.03	35	31	30	0	7
	1	[40;60]	2.5	311.97	311.97	(opt)	57.0	291.83	6.46	11	34	30	0	0
R101	0.6	[1;100]	1.1	525.55	525.55	(opt)	1.0	504.92	3.93	16	39	30	0	4
	0.6	[40;60]	1.1	447.99	447.99	(opt)	1.9	427.8	4.51	23	29	30	0	0
	1	[1;100]	1.1	575.24	575.24	(opt)	34.1	549.55	4.47	57	78	30	0	13
	1	[40;60]	1.1	679.02	679.02	(opt)	1.8	667.67	1.67	19	62	30	0	0
	0.6	[1;100]	1.5	451.37	451.37	(opt)	0.5	440.72	2.36	16	18	30	0	3
	0.6	[40;60]	1.5	450.37	450.37	(opt)	4.0	430.91	4.32	35	47	30	0	0
	1	[1;100]	1.5	488.50	488.50	(opt)	18.2	468.12	4.17	35	85	30	0	9
	1	[40;60]	1.5	551.36	551.36	(opt)	575.9	517.13	6.21	52	71	30	0	0
	0.6	[1;100]	2.0	325.84	325.84	(opt)	0.4	323.57	0.70	2	6	2	0	2
	0.6	[40;60]	2.0	374.28	374.28	(opt)	23.6	360.02	3.81	19	11	30	0	0
	1	[1;100]	2.0	375.32	375.32	(opt)	1.8	364.36	2.92	6	11	10	0	4
	1	[40;60]	2.0	443.77	443.77	(opt)	16.2	430.6	2.97	19	36	30	0	0
	0.6	[1;100]	2.5	334.59	334.59	(opt)	3.5	317.16	5.21	17	28	30	0	6
	0.6	[40;60]	2.5	307.02	307.02	(opt)	0.1	299.72	2.38	2	9	0	0	0
	1	[1;100]	2.5	360.28	360.28	(opt)	3.9	349.24	3.06	14	26	17	0	8
	1	[40;60]	2.5	391.30	391.30	(opt)	0.5	378.98	3.15	3	27	0	0	0

Table 9: Detailed results for the small instances ($n = 15$) with $\kappa = 6$ commodities.

II. Detailed Results for the Mid-Size Instances ($n = 20, 40, 60, 80$)

Instance				S-Rank-S/D-Subs										
	n	p	No.	$\underline{z}^*$	$\overline{z}^*$	gap	time	$\underline{z}^{\mathrm{LP}}$	$\mathrm{gap}^{\mathrm{LP}}$	nodes	CC	SR	SD	rec
C101	20	0.6	1	573.86	573.86	(opt)	0.1	573.28	0.10	1	4	0	0	0
			2	592.07	592.07	(opt)	0.1	566.9	4.25	1	37	0	0	0
			3	595.53	595.53	(opt)	5.4	533.06	10.49	54	54	30	0	0
			4	617.88	617.88	(opt)	0.7	603.3	2.36	12	66	30	0	1
			5	628.28	628.28	(opt)	2.3	604.1	3.85	36	50	30	0	0
		1	1	750.63	750.63	(opt)	0.4	736.56	1.87	6	29	10	0	1
			2	714.65	714.65	(opt)	0.7	686.41	3.95	5	48	10	0	0
			3	626.16	626.16	(opt)	119.0	593.22	5.26	259	47	30	0	2
			4	747.70	747.70	(opt)	5.2	706.87	5.46	26	71	30	0	4
			5	768.52	768.52	(opt)	4.2	745.15	3.04	48	58	30	0	4
R101	20	0.6	1	457.86	457.86	(opt)	0.3	445.58	2.68	8	40	10	0	0
			2	667.01	667.01	(opt)	0.7	655.88	1.67	10	29	27	0	1
			3	455.05	455.05	(opt)	0.2	443.98	2.43	3	36	0	0	0
			4	589.91	589.91	(opt)	0.5	570.97	3.21	9	52	15	0	1
			5	663.22	663.22	(opt)	0.1	648.93	2.15	7	36	8	0	1
		1	1	599.84	599.84	(opt)	0.7	580.87	3.16	6	44	10	0	0
			2	863.88	863.88	(opt)	4.3	838.49	2.94	11	34	30	0	2
			3	617.12	617.12	(opt)	420.1	589.83	4.42	694	57	30	11	24
			4	712.02	712.02	(opt)	2.6	687.47	3.45	18	49	30	0	2
			5	794.41	794.41	(opt)	79.4	751.76	5.37	496	71	30	0	14
C101	40	0.6	1	840.30	–	–	TL1	782.95	–	932	196	30	42	3
			2	992.06	–	–	TL1	948.14	–	1972	147	30	122	14
			3	879.26	879.26	(opt)	10.8	845.46	3.84	11	130	20	0	2
			4	921.06	921.06	(opt)	195.9	883.78	4.05	335	110	30	34	0
			5	868.74	868.74	(opt)	161.5	833.75	4.03	163	77	30	16	6
		1	1	1301.90	–	–	TL1	1210.08	–	3142	182	30	103	72
			2	1357.50	1357.79	0.02	TL1	1275.27	6.08	2149	234	30	110	115
			3	1299.40	1299.49	0.01	TL1	1215.15	6.49	1489	187	30	54	98
			4	1234.36	–	–	TL1	1172.33	–	2817	196	30	135	28
			5	1265.71	–	–	TL1	1153.93	–	2302	162	30	212	11
R101	40	0.6	1	761.78	761.78	(opt)	1462.4	742.65	2.51	833	99	30	37	4
			2	896.02	896.02	(opt)	632.4	874.11	2.45	981	144	30	31	2
			3	851.03	851.03	(opt)	71.4	828.3	2.67	143	117	30	12	1
			4	971.85	973.48	0.17	TL1	938.74	3.57	4390	139	30	465	1
			5	854.35	854.35	(opt)	149.7	814.44	4.67	160	128	30	9	1
		1	1	1235.80	–	–	TL1	1208.52	–	2613	202	30	214	84
			2	1235.80	–	–	TL1	1197.67	–	2116	257	30	214	48
			3	1056.13	1056.13	(opt)	291.4	1038.69	1.65	310	179	30	5	16
			4	1243.06	–	–	TL1	1211.52	–	3504	184	30	422	110
			5	1097.96	–	–	TL1	1064.26	–	1681	153	30	164	50

Table 10: Detailed results for the mid-size instances ($n = 20$ to 40) with $\kappa = 3$ commodities.

Instance				S-Rank-S/D-Subs										
	n	p	No.	$\underline{z}^*$	$\overline{z}^*$	gap	time	$\underline{z}^{\mathrm{LP}}$	$\mathrm{gap}^{\mathrm{LP}}$	nodes	CC	SR	SD	rec
C101	60	0.6	1	1220.10	–	–	TL1	1147.63	–	563	311	30	165	0
			2	1331.77	1331.77	(opt)	3195.5	1277.81	4.05	599	308	30	42	1
			3	1180.61	1180.61	(opt)	275.6	1141.78	3.29	64	182	30	5	1
			4	1282.72	1282.72	(opt)	1049.2	1233.06	3.87	264	215	30	11	2
			5	1299.07	–	–	TL1	1225.12	–	957	315	30	423	31
		1	1	1969.31	–	–	TL1	1889.44	–	1016	358	30	65	40
			2	1659.11	–	–	TL1	1587.69	–	629	300	30	105	49
			3	1784.31	–	–	TL1	1703.73	–	1131	226	30	126	34
			4	1896.33	–	–	TL1	1823.25	–	1177	287	30	155	107
			5	1625.32	–	–	TL1	1569.73	–	563	233	30	35	20
R101	60	0.6	1	1282.51	1284.99	0.19	TL1	1261.02	1.87	1372	189	30	321	11
			2	1308.09	–	–	TL1	1281.35	–	1601	276	30	290	3
			3	1028.52	1028.52	(opt)	331.6	1012.18	1.59	245	150	30	39	2
			4	1219.15	–	–	TL1	1201.34	–	1826	163	30	544	27
			5	1148.05	1151.21	0.27	TL1	1125.66	2.22	1246	287	30	266	33
		1	1	2040.84	–	–	TL1	2013.23	–	2398	158	30	265	186
			2	1639.08	–	–	TL1	1607.80	–	2436	192	30	152	136
			3	1547.39	–	–	TL1	1525.45	–	2280	206	30	673	70
			4	1708.22	–	–	TL1	1681.15	–	1346	284	30	106	74
			5	1490.34	–	–	TL1	1461.98	–	876	233	30	120	37
C101	80	0.6	1	1632.21	–	–	TL1	1550.64	–	480	400	30	95	0
			2	1595.50	–	–	TL1	1526.39	–	668	277	30	118	6
			3	1731.77	–	–	TL1	1660.42	–	617	400	30	170	1
			4	1440.28	–	–	TL1	1375.15	–	435	347	30	247	12
			5	1662.12	–	–	TL1	1562.98	–	460	400	30	207	30
		1	1	2226.65	–	–	TL1	2167.72	–	304	364	30	67	7
			2	2095.53	–	–	TL1	2009.26	–	277	306	30	107	11
			3	2574.99	–	–	TL1	2488.46	–	805	324	30	57	38
			4	2343.46	–	–	TL1	2253.41	–	435	399	30	47	36
			5	2387.96	–	–	TL1	2302.49	–	494	400	30	67	16
R101	80	0.6	1	1434.71	–	–	TL1	1407.84	–	1048	153	30	301	1
			2	1462.05	–	–	TL1	1447.64	–	1197	188	30	231	16
			3	1585.92	–	–	TL1	1570.82	–	1980	190	30	321	17
			4	1404.23	–	–	TL1	1389.55	–	611	159	30	258	3
			5	1451.05	–	–	TL1	1431.73	–	144	159	30	105	2
		1	1	2084.94	–	–	TL1	2058.75	–	934	228	30	60	61
			2	1913.71	–	–	TL1	1897.11	–	907	243	30	206	56
			3	2261.39	–	–	TL1	2242.57	–	1121	181	30	113	43
			4	2080.61	–	–	TL1	2057.73	–	547	137	30	100	17
			5	2119.67	–	–	TL1	2099.55	–	1059	251	30	213	57

Table 11: Detailed results for the mid-size instances ($n = 60$ to 80) with $\kappa = 3$ commodities.

Instance				S-Rank-S/D-Subs										
	n	p	No.	$\underline{z}^*$	$\overline{z}^*$	gap	time	$\underline{z}^{\mathrm{LP}}$	$\mathrm{gap}^{\mathrm{LP}}$	nodes	CC	SR	SD	rec
C101	20	0.6	1	649.08	649.08	(opt)	5.8	620.56	4.39	12	39	30	0	3
			2	681.03	681.03	(opt)	0.2	659.36	3.18	3	39	0	0	1
			3	538.00	538.00	(opt)	1.7	501.7	6.75	15	57	30	0	3
			4	655.95	655.95	(opt)	941.1	623.89	4.89	2612	57	30	38	6
			5	604.50	604.50	(opt)	12.7	581.67	3.78	63	32	30	2	1
		1	1	809.62	809.62	(opt)	179.2	765.3	5.47	215	71	30	10	12
			2	719.47	–	–	TL1	684.53	–	233	82	30	2	18
			3	731.22	731.22	(opt)	2.2	702.74	3.89	4	62	0	0	2
			4	695.00	695.00	(opt)	0.7	670.85	3.47	3	30	0	0	2
			5	798.93	798.93	(opt)	2.5	750.32	6.08	6	49	0	0	2
R101	20	0.6	1	560.61	560.61	(opt)	0.4	540.15	3.65	9	61	17	0	0
			2	724.04	724.04	(opt)	0.3	709.27	2.04	1	13	0	0	0
			3	523.70	523.70	(opt)	8.6	507.42	3.11	16	47	30	0	2
			4	591.95	591.95	(opt)	0.3	565.64	4.44	3	20	0	0	2
			5	651.84	651.84	(opt)	0.4	631.84	3.07	4	36	10	0	0
		1	1	680.43	680.43	(opt)	111.3	665.92	2.13	138	54	30	0	8
			2	864.49	864.49	(opt)	39.5	845.62	2.18	14	28	30	0	3
			3	527.54	527.54	(opt)	15.2	509.3	3.46	29	50	30	0	6
			4	650.29	650.29	(opt)	12.6	636.92	2.06	18	46	30	0	6
			5	807.65	807.65	(opt)	2177.3	753.64	6.69	1409	84	30	0	43
C101	40	0.6	1	948.63	948.63	(opt)	1023.7	895.41	5.61	551	185	30	86	8
			2	990.97	990.97	(opt)	564.4	922.65	6.89	500	134	30	42	9
			3	1158.42	–	–	TL1	1095.86	–	1504	189	30	194	34
			4	976.77	976.77	(opt)	96.8	898.84	7.98	139	145	30	11	9
			5	1153.11	1153.11	(opt)	2212.3	1101.13	4.51	490	186	30	11	6
		1	1	1261.97	–	–	TL1	1151.37	–	1142	221	30	178	54
			2	1239.18	1239.18	(opt)	2085.7	1172.09	5.41	462	200	30	25	57
			3	1269.48	1271.09	0.13	TL1	1203.15	5.35	750	199	30	34	116
			4	1352.52	–	–	TL1	1255.6	–	1400	230	30	72	155
			5	1421.31	–	–	TL1	1332.82	–	1102	239	30	63	47
R101	40	0.6	1	1162.02	1162.02	(opt)	140.9	1143.54	1.59	212	129	30	12	18
			2	881.66	881.66	(opt)	93.4	863.79	2.03	56	132	30	2	3
			3	988.39	988.39	(opt)	362.8	963.12	2.56	408	152	30	2	18
			4	1080.61	1081.68	0.10	TL1	1050.11	2.92	2184	168	30	231	7
			5	800.97	–	–	TL1	768.97	–	363	141	30	36	17
		1	1	1239.62	1239.62	(opt)	349.4	1217.78	1.76	140	138	30	9	35
			2	1117.06	–	–	TL1	1090	–	1078	165	30	52	145
			3	1167.83	–	–	TL1	1133.37	–	1349	221	30	119	192
			4	1244.51	–	–	TL1	1211.98	–	1187	195	30	156	42
			5	1313.80	–	–	TL1	1259.48	–	1243	160	30	266	62

Table 12: Detailed results for the mid-size instances ($n = 20$ to 40) with $\kappa = 4$ commodities.

Instance				S-Rank-S/D-Subs										
	n	p	No.	$\underline{z}^*$	$\overline{z}^*$	gap	time	$\underline{z}^{\mathrm{LP}}$	$\mathrm{gap}^{\mathrm{LP}}$	nodes	CC	SR	SD	rec
C101	60	0.6	1	1636.94	–	–	TL1	1542.06	–	1069	368	30	83	21
			2	1505.91	1505.91	(opt)	1561.6	1427.26	5.22	288	298	30	32	4
			3	1352.17	1352.17	(opt)	3368.1	1292.56	4.41	678	228	30	27	49
			4	1521.40	–	–	TL1	1454.43	–	1076	277	30	239	10
			5	1590.62	–	–	TL1	1504.92	–	757	286	30	32	3
		1	1	1831.44	–	–	TL1	1740.34	–	332	318	30	11	38
			2	1668.72	–	–	TL1	1595.26	–	273	264	30	45	74
			3	2034.77	–	–	TL1	1916.40	–	280	331	30	48	38
			4	1823.32	–	–	TL1	1732.55	–	122	252	30	17	16
			5	1791.78	–	–	TL1	1716.11	–	123	280	30	24	21
R101	60	0.6	1	1552.24	–	–	TL1	1535.52	–	2052	144	30	403	86
			2	1354.81	–	–	TL1	1324.73	–	1598	214	30	188	11
			3	1276.17	–	–	TL1	1255.49	–	1585	206	30	381	14
			4	1432.08	–	–	TL1	1401.70	–	1275	258	30	414	149
			5	1201.86	–	–	TL1	1175.02	–	545	175	30	75	31
		1	1	1802.43	–	–	TL1	1780.56	–	116	128	30	17	15
			2	1648.49	–	–	TL1	1627.88	–	724	224	30	145	104
			3	1699.64	–	–	TL1	1670.63	–	660	262	30	186	49
			4	1601.72	–	–	TL1	1574.18	–	449	189	30	26	74
			5	1726.92	–	–	TL1	1695.05	–	608	211	30	112	72
C101	80	0.6	1	1888.49	–	–	TL1	1822.17	–	415	250	30	69	10
			2	2055.12	–	–	TL1	1976.74	–	604	400	30	149	14
			3	1748.96	–	–	TL1	1670.66	–	482	284	30	49	26
			4	1585.83	–	–	TL1	1526.02	–	299	400	30	91	39
			5	2190.16	–	–	TL1	2102.90	–	637	400	30	119	72
		1	1	2652.92	–	–	TL1	2559.53	–	136	391	30	8	17
			2	2418.94	–	–	TL1	2333.59	–	118	361	30	1	43
			3	2582.24	–	–	TL1	2507.46	–	136	333	30	9	7
			4	2432.52	–	–	TL1	2341.21	–	224	400	30	28	44
			5	2451.31	–	–	TL1	2368.39	–	203	400	30	29	41
R101	80	0.6	1	1628.63	–	–	TL1	1610.91	–	887	161	30	307	9
			2	1661.78	–	–	TL1	1643.11	–	1165	174	30	238	24
			3	1624.20	–	–	TL1	1601.62	–	764	233	30	291	10
			4	1453.66	–	–	TL1	1436.42	–	214	103	30	63	11
			5	1631.95	–	–	TL1	1617.77	–	1063	234	30	206	7
		1	1	2158.99	–	–	TL1	2130.46	–	232	220	30	73	23
			2	2134.54	–	–	TL1	2114.86	–	67	160	30	2	8
			3	2106.72	–	–	TL1	2091.32	–	20	97	30	2	8
			4	2169.42	–	–	TL1	2146.81	–	312	182	30	57	32
			5	2012.88	–	–	TL1	1990.19	–	261	144	30	21	41

Table 13: Detailed results for the mid-size instances ($n = 60$ to 80) with $\kappa = 4$ commodities.

Instance				S-Rank-S/D-Subs										
	n	p	No.	$\underline{z}^*$	$\overline{z}^*$	gap	time	$\underline{z}^{\mathrm{LP}}$	$\mathrm{gap}^{\mathrm{LP}}$	nodes	CC	SR	SD	rec
C101	20	0.6	1	621.01	621.01	(opt)	1.0	596.34	3.97	9	53	20	0	2
			2	756.35	759.85	0.46	TL1	658.41	13.35	3106	69	30	279	7
			3	518.91	518.91	(opt)	1.2	498.77	3.88	7	40	18	0	1
			4	673.26	673.26	(opt)	2.0	620.55	7.83	21	62	30	2	2
			5	665.57	665.57	(opt)	7.5	657.59	1.20	12	26	30	0	2
		1	1	815.67	815.67	(opt)	375.1	786.18	3.62	147	56	30	0	16
			2	743.05	743.05	(opt)	44.3	686.9	7.56	39	57	30	3	5
			3	715.22	717.29	0.29	TL1	681.99	4.92	1149	78	30	8	64
			4	787.38	787.38	(opt)	246.5	724.63	7.97	83	121	30	0	9
			5	733.69	733.69	(opt)	19.6	712.12	2.94	18	52	30	0	7
R101	20	0.6	1	590.23	590.23	(opt)	4.2	563.48	4.53	30	52	30	0	3
			2	733.90	733.90	(opt)	1220.6	679.29	7.44	750	51	30	71	8
			3	482.58	482.58	(opt)	1.7	471.12	2.37	5	27	0	0	3
			4	634.88	634.88	(opt)	2.1	599.14	5.63	16	43	12	0	3
			5	705.36	705.36	(opt)	38.5	680.69	3.50	99	46	30	0	2
		1	1	651.93	651.93	(opt)	16.7	625.81	4.01	18	65	30	0	6
			2	1058.07	–	–	TL1	1019.88	–	355	98	30	16	23
			3	669.64	669.64	(opt)	98.9	642.91	3.99	31	75	30	0	7
			4	687.01	687.01	(opt)	16.5	649.79	5.42	10	77	15	0	5
			5	826.29	828.34	0.25	TL1	792.77	4.29	1541	62	30	5	91
C101	40	0.6	1	925.77	926.38	0.07	TL1	893.8	3.52	468	128	30	58	11
			2	1083.31	1085.86	0.23	TL1	1030.73	5.08	1962	169	30	39	112
			3	893.10	–	–	TL1	881.03	–	230	96	30	33	12
			4	969.26	969.26	(opt)	262.1	933.09	3.73	105	168	30	6	7
			5	1221.69	–	–	TL1	1149.6	–	1313	148	30	165	32
		1	1	1239.04	–	–	TL1	1149.04	–	214	208	30	5	48
			2	1304.84	–	–	TL1	1203.88	–	362	216	30	50	79
			3	1518.51	–	–	TL1	1453.95	–	321	326	30	4	72
			4	1266.96	–	–	TL1	1187.09	–	392	156	30	80	55
			5	1419.43	–	–	TL1	1325.55	–	257	275	30	27	41
R101	40	0.6	1	1011.72	–	–	TL1	991.32	–	1919	135	30	293	72
			2	921.31	–	–	TL1	891.28	–	1544	176	30	125	195
			3	848.73	848.73	(opt)	782.6	819.65	3.43	223	131	30	4	6
			4	925.29	927.62	0.25	TL1	889.94	4.06	851	169	30	307	14
			5	844.19	844.19	(opt)	1552.0	806.22	4.50	455	138	30	33	17
		1	1	1245.76	–	–	TL1	1224.97	–	761	209	30	53	223
			2	1214.52	–	–	TL1	1182.01	–	785	214	30	90	189
			3	1101.84	1101.84	(opt)	1247.6	1076.14	2.33	233	186	30	0	53
			4	1320.11	–	–	TL1	1286.75	–	364	205	30	48	83
			5	1269.16	–	–	TL1	1218.33	–	392	232	30	36	77

Table 14: Detailed results for the mid-size instances ($n = 20$ to 40) with $\kappa = 5$ commodities.

Instance				S-Rank-S/D-Subs										
	n	p	No.	$\underline{z}^*$	$\overline{z}^*$	gap	time	$\underline{z}^{\mathrm{LP}}$	$\mathrm{gap}^{\mathrm{LP}}$	nodes	CC	SR	SD	rec
C101	60	0.6	1	1267.21	–	–	TL1	1174.39	–	247	274	30	40	24
			2	1299.54	–	–	TL1	1233.11	–	413	194	30	97	80
			3	1322.81	–	–	TL1	1255.20	–	284	328	30	70	13
			4	1533.68	–	–	TL1	1437.03	–	380	220	30	41	18
			5	1500.11	–	–	TL1	1417.60	–	465	283	30	40	18
		1	1	1971.97	–	–	TL1	1915.70	–	184	249	30	31	50
			2	2079.26	–	–	TL1	2000.89	–	189	340	30	33	44
			3	2091.39	–	–	TL1	2032.59	–	59	379	30	5	16
			4	1770.93	–	–	TL1	1701.49	–	73	276	30	2	20
			5	1943.15	–	–	TL1	1846.33	–	93	400	30	1	29
R101	60	0.6	1	1578.17	–	–	TL1	1559.40	–	613	129	30	140	27
			2	1214.21	–	–	TL1	1192.54	–	326	208	30	118	62
			3	1234.72	–	–	TL1	1215.45	–	89	134	30	23	6
			4	1283.35	–	–	TL1	1259.06	–	760	200	30	158	179
			5	1391.90	–	–	TL1	1366.53	–	1415	175	30	456	82
		1	1	1989.24	–	–	TL1	1970.33	–	134	163	30	5	31
			2	1713.94	–	–	TL1	1691.10	–	128	264	30	13	45
			3	1680.88	–	–	TL1	1651.51	–	169	199	30	5	45
			4	1775.84	–	–	TL1	1747.53	–	179	188	30	18	48
			5	1653.40	–	–	TL1	1619.96	–	279	150	30	59	62
C101	80	0.6	1	1926.31	–	–	TL1	1843.06	–	393	400	30	75	33
			2	1959.01	–	–	TL1	1874.71	–	219	352	30	55	21
			3	2158.10	–	–	TL1	2069.30	–	491	371	30	88	19
			4	1785.00	–	–	TL1	1711.39	–	190	348	30	41	46
			5	1714.63	–	–	TL1	1616.12	–	149	378	30	67	14
		1	1	2668.17	–	–	TL1	2588.61	–	74	384	30	11	29
			2	2495.55	–	–	TL1	2397.97	–	34	385	30	5	11
			3	2616.50	–	–	TL1	2523.77	–	39	400	30	7	17
			4	2416.57	–	–	TL1	2332.34	–	24	377	30	0	11
			5	2667.28	–	–	TL1	2584.43	–	35	400	30	1	13
R101	80	0.6	1	1876.65	–	–	TL1	1851.68	–	520	189	30	109	30
			2	1722.31	–	–	TL1	1702.87	–	307	216	30	25	61
			3	1814.43	–	–	TL1	1796.54	–	312	156	30	45	13
			4	1703.44	–	–	TL1	1688.64	–	284	130	30	81	15
			5	1739.00	–	–	TL1	1721.68	–	228	192	30	43	31
		1	1	2096.03	–	–	TL1	2064.40	–	97	162	30	4	29
			2	2297.73	–	–	TL1	2273.41	–	130	267	30	9	22
			3	2214.43	–	–	TL1	2201.76	–	88	143	30	3	25
			4	2297.69	–	–	TL1	2280.30	–	80	118	30	6	15
			5	2077.68	–	–	TL1	2060.02	–	70	164	30	6	28

Table 15: Detailed results for the mid-size instances ($n = 60$ to 80) with $\kappa = 5$ commodities.

Instance				S-Rank-S/D-Subs										
	n	p	No.	$\underline{z}^*$	$\overline{z}^*$	gap	time	$\underline{z}^{\mathrm{LP}}$	$\mathrm{gap}^{\mathrm{LP}}$	nodes	CC	SR	SD	rec
C101	20	0.6	1	834.67	834.67	(opt)	9.6	802.77	3.82	26	78	30	0	7
			2	831.18	831.23	0.01	TL1	740.76	10.88	1436	96	30	2	25
			3	637.77	637.77	(opt)	66.8	600.86	5.79	172	81	30	0	10
			4	597.75	597.75	(opt)	84.4	565	5.48	248	45	30	3	9
			5	726.09	726.09	(opt)	41.3	654.53	9.86	25	53	30	0	4
		1	1	830.60	830.60	(opt)	52.2	790.94	4.77	29	85	30	0	12
			2	773.82	773.82	(opt)	95.7	727.34	6.01	32	76	30	0	6
			3	777.16	–	–	TL1	699.31	–	115	93	30	7	12
			4	791.61	791.61	(opt)	81.6	741.28	6.36	30	95	30	0	11
			5	824.57	824.57	(opt)	311.4	737.38	10.57	78	66	30	10	20
R101	20	0.6	1	548.17	548.17	(opt)	2.5	531.01	3.13	7	31	20	0	5
			2	715.71	715.71	(opt)	26.7	685.19	4.26	15	40	30	0	4
			3	598.37	598.37	(opt)	33.6	572.18	4.38	44	70	30	0	10
			4	577.61	579.04	0.25	TL1	527.7	8.87	374	71	30	58	11
			5	766.93	766.93	(opt)	229.2	716.61	6.56	363	59	30	0	16
		1	1	656.77	656.77	(opt)	294.8	636.19	3.13	32	59	30	0	13
			2	968.03	968.03	(opt)	1173.6	931.29	3.80	30	60	30	0	8
			3	704.72	704.72	(opt)	2901.8	656.64	6.82	203	115	30	1	19
			4	804.79	–	–	TL1	749.35	–	386	134	30	57	56
			5	775.02	775.02	(opt)	174.2	730.78	5.71	39	65	30	0	13
C101	40	0.6	1	1024.19	1024.19	(opt)	2745.4	950.47	7.20	125	165	30	3	21
			2	1185.80	–	–	TL1	1128.61	–	1065	238	30	109	76
			3	1150.34	–	–	TL1	1081.63	–	680	197	30	124	110
			4	1101.44	–	–	TL1	1050.21	–	704	187	30	47	92
			5	1191.09	–	–	TL1	1094.6	–	530	191	30	99	36
		1	1	1421.95	–	–	TL1	1322.48	–	68	222	30	9	19
			2	1349.07	–	–	TL1	1265.69	–	114	204	30	12	29
			3	1360.14	–	–	TL1	1273.36	–	40	173	30	4	13
			4	1095.58	–	–	TL1	1061.32	–	39	115	30	6	16
			5	1318.13	–	–	TL1	1242.12	–	39	227	30	3	14
R101	40	0.6	1	1056.06	–	–	TL1	1040.04	–	742	119	30	107	136
			2	1097.22	1097.22	(opt)	2798.7	1069.53	2.52	1023	158	30	127	130
			3	834.82	834.82	(opt)	1873.7	804.08	3.68	248	219	30	56	29
			4	1022.90	–	–	TL1	983.31	–	828	194	30	69	66
			5	906.82	–	–	TL1	876.05	–	337	190	30	50	102
		1	1	1256.90	–	–	TL1	1242.37	–	78	120	30	3	23
			2	1186.80	–	–	TL1	1155.93	–	103	178	30	8	27
			3	1146.18	–	–	TL1	1120.26	–	58	142	30	1	21
			4	1432.91	–	–	TL1	1389.9	–	111	196	30	9	41
			5	1292.08	–	–	TL1	1249.59	–	123	183	30	25	42

Table 16: Detailed results for the mid-size instances ($n = 20$ to 40) with $\kappa = 6$ commodities.

Instance				S-Rank-S/D-Subs										
	n	p	No.	$\underline{z}^*$	$\overline{z}^*$	gap	time	$\underline{z}^{\mathrm{LP}}$	$\mathrm{gap}^{\mathrm{LP}}$	nodes	CC	SR	SD	rec
C101	60	0.6	1	1569.20	–	–	TL1	1485.40	–	251	284	30	47	16
			2	1574.31	–	–	TL1	1489.32	–	300	319	30	114	54
			3	1553.25	–	–	TL1	1493.08	–	283	264	30	40	36
			4	1588.51	–	–	TL1	1510.90	–	285	287	30	71	40
			5	1494.85	–	–	TL1	1409.37	–	246	242	30	31	43
		1	1	2079.05	–	–	TL1	1967.30	–	25	400	30	1	12
			2	2043.49	–	–	TL1	1965.60	–	20	238	30	0	10
			3	1770.63	–	–	TL1	1678.83	–	8	252	0	0	6
			4	1834.04	–	–	TL1	1740.22	–	18	344	10	0	11
			5	2169.39	–	–	TL1	2066.46	–	29	393	30	0	14
R101	60	0.6	1	1687.99	–	–	TL1	1671.66	–	198	144	30	8	35
			2	1453.23	–	–	TL1	1418.17	–	352	207	30	50	55
			3	1386.91	–	–	TL1	1363.01	–	629	273	30	149	93
			4	1431.09	–	–	TL1	1412.11	–	443	125	30	44	50
			5	1504.27	–	–	TL1	1479.86	–	615	251	30	139	117
		1	1	2040.85	–	–	TL1	2021.26	–	61	120	30	7	18
			2	1775.92	–	–	TL1	1743.93	–	48	334	30	3	19
			3	1669.18	–	–	TL1	1645.16	–	63	204	30	5	20
			4	1743.05	–	–	TL1	1715.45	–	63	163	30	5	22
			5	1723.23	–	–	TL1	1700.28	–	32	252	30	5	14
C101	80	0.6	1	1877.47	–	–	TL1	1792.62	–	118	315	30	16	34
			2	1900.44	–	–	TL1	1829.50	–	32	400	30	7	8
			3	2125.22	–	–	TL1	2054.57	–	75	359	30	4	7
			4	1918.60	–	–	TL1	1814.20	–	102	400	30	17	17
			5	2058.94	–	–	TL1	1988.84	–	124	377	30	16	37
		1	1	2310.06	–	–	TL1	2310.06	–	0	234	0	0	7
			2	2565.44	–	–	TL1	2492.06	–	3	230	0	0	4
			3	2320.99	–	–	TL1	2320.99	–	0	205	0	0	2
			4	2537.06	–	–	TL1	2476.59	–	6	378	0	0	5
			5	2705.60	–	–	TL1	2617.11	–	4	302	0	0	5
R101	80	0.6	1	2018.25	–	–	TL1	1989.18	–	262	185	30	42	37
			2	1931.55	–	–	TL1	1907.14	–	252	200	30	48	42
			3	1582.70	–	–	TL1	1566.01	–	227	146	30	16	63
			4	1604.98	–	–	TL1	1587.65	–	65	126	30	15	15
			5	1785.03	–	–	TL1	1771.44	–	171	92	30	22	26
		1	1	2444.64	–	–	TL1	2423.04	–	39	316	30	0	19
			2	2093.90	–	–	TL1	2069.36	–	22	207	30	0	14
			3	2262.04	–	–	TL1	2253.06	–	25	99	30	0	12
			4	2316.47	–	–	TL1	2294.66	–	25	170	30	0	15
			5	2283.79	–	–	TL1	2272.48	–	15	129	30	0	11

Table 17: Detailed results for the mid-size instances ($n = 60$ to 80) with $\kappa = 6$ commodities.