Gutenberg School of Management and Economics

& Research Unit "Interdisciplinary Public Policy"

Discussion Paper Series

Branch-Price-and-Cut for the Soft-Clustered Capacitated Arc-Routing Problem

Timo Hintsch, Stefan Irnich, Lone Kiilerich

October 10, 2019

Discussion paper number 1911

Johannes Gutenberg University Mainz
Gutenberg School of Management and Economics
Jakob-Welder-Weg 9
55128 Mainz
Germany
https://wiwi.uni-mainz.de/

Contact details

Stefan Irnich
Chair of Logistics Management
Gutenberg School of Management and Economics
University of Mainz
Jakob-Welder-Weg 9
55128 Mainz
Germany
irnich@uni-mainz.de

Timo Hintsch
Chair of Logistics Management
Gutenberg School of Management and Economics
Johannes Gutenberg University Mainz,
Jakob-Welder-Weg 9
55128 Mainz
Germany
thintsch@uni-mainz.de

Lone Kiilerich
Department of Economics and Business Economics
School of Business and Social Science
Aarhus University, Denmark
lone.ki.ch@econ.au.dk

Branch-Price-and-Cut for the Soft-Clustered Capacitated Arc-Routing Problem

Timo Hintsch[a], Stefan Irnich[*,a], Lone Kiilerich[b]

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

[b] *Department of Economics and Business Economics, School of Business and Social Sciences, Aarhus University, Fuglesangs Allé 4, 8210 Aarhus V, Denmark.*

Abstract

The soft-clustered capacitated arc-routing problem (SoftCluCARP) is a restricted variant of the classical capacitated arc-routing problem. The only additional constraint is that the set of required edges, i.e., the streets to be serviced, is partitioned into clusters and feasible routes must respect the soft-cluster constraint, that is, all required edges of the same cluster must be served by the same vehicle. In this article, we design an effective branch-price-and-cut algorithm for the exact solution of the SoftCluCARP. Its new components are a metaheuristic and branch-and-cut-based solvers for the solution of the column-generation subproblem, which is a profitable rural clustered postman tour problem. Although postman problems with these characteristics have been studied before, there is one fundamental difference here: clusters are not necessarily vertex-disjoint, which prohibits many preprocessing and modeling approaches for clustered postman problems from the literature. We present an undirected and a windy formulation for the pricing subproblem and develop and computationally compare two corresponding branch-and-cut algorithms. Cutting is also performed at the master-program level using subset-row inequalities for subsets of size up to five. For the first time, these non-robust cuts are incorporated into MIP-based routing subproblem solvers using two different modeling approaches. In several computational studies, we calibrate the individual algorithmic components. The final computational experiments prove that the branch-price-and-cut algorithm equipped with these problem-tailored components is effective: The largest SoftCluCARP instances solved to optimality have more than 150 required edges or more than 50 clusters.

Key words: Arc routing, branch-price-and-cut, branch-and-cut, districting

1. Introduction

The *capacitated arc-routing problem* (CARP, Belenguer *et al.*, 2014) is the basic multiple-vehicle arc-routing problem. For solving the CARP, the task is to determine a set of cost-minimal capacity feasible routes so that a given set of required edges demanding service is covered. Golden and Wong (1981) introduced the CARP into the scientific literature. Postman problems, the CARP, and its various extensions have been discussed and surveyed by Dror (2000); Corberán and Prins (2010); Corberán and Laporte (2014); Mourão and Pinto (2017). Practical applications of these arc-routing problems are, for example, waste collection, postal delivery, winter services (snow plowing, winter gritting, and salt spreading), meter reading, and school bus routing.

In the paper at hand, we focus on an extension of the basic CARP in which the required edges are clustered. Each given cluster can be understood as a *micro district*. The task is now to group together the given micro districts into complete (or final) districts that are served by a single vehicle.

[*]Corresponding author.

Email addresses: thintsch@uni-mainz.de (Timo Hintsch), irnich@uni-mainz.de (Stefan Irnich), lone.ki.ch@econ.au.dk (Lone Kiilerich)

For a comprehensive overview of *districting for arc routing*, we refer to the work of Butsch *et al.* (2014). The authors discuss applications as for the CARP in postal delivery, winter services, municipal solid waste collection, and meter reading. The districting approach of Butsch *et al.* starts from required edges as the *basic units* and builds a given number of districts. Each district comprises a set of basic units that are later on served by a single tour. Each basic unit is exclusively and completely assigned to one district.

In a districting improvement procedure, the initially computed set of districts are then optimized with regard to several criteria: among them, balancedness, connectivity, and compactness are the most important. Balancedness refers to the distribution of workload (the service time) that should be as equally split as possible (this is typically a soft criterion). Compactness refers to the shape of the districts that should be squared or rounded. Finally, connectivity is desirable, probably because connected basic units principally reduce extra deadheading times. The final districts computed are then later served by a vehicle that performs a postman tour over it. This districting-first postman-tour-second approach however does not exploit the full optimization potential that an integrated approach offers: An optimal CARP solution is (by definition) the best solution from a routing point of view, compare Figures 1(a) and (b). However, typical CARP solutions have undesirable resulting districts that are neither compact nor connected. On the positive side, CARP solutions tend to be balanced, in particular when the fleet size and vehicle capacity are chosen accordingly.

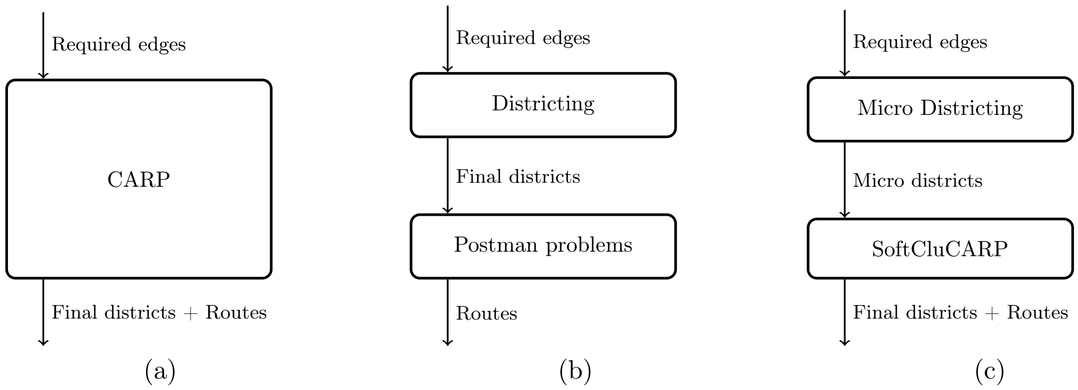

Figure 1: Possible planning steps (a) CARP (fully integrated, lower-quality districts, optimal routes), (b) 2-stage planning with districting first and solving multiple independent postman problems second (optimal districts, lower-quality routes), and (c) 2-stage planning with micro districting first and SoftCluCARP second.

We see the new planning problem, below defined as the *soft-clustered capacitated arc-routing problem* (SoftCluCARP), as a planning problem that allows shifting the traditional 2-stage hierarchical planning approach that follows the districting first-routing second paradigm towards better routing as well as better clustering decisions, see Figure 1(c). Indeed, with not too large micro districts (the input clusters to the SoftCluCARP), one can expect SoftCluCARP solutions that are close to the CARP routing optimum. Similarly, not too small micro districts can be constructed so that they are compact and connected. The expectation is that with such an input, the SoftCluCARP solution comprises "nicer" final districts that are more compact and connected.

For the family of *vehicle-routing problems* (VRPs, Irnich *et al.*, 2014), variants with clusters of customers can be characterized as either hard-clustered or soft-clustered. The former variant, known as the *clustered VRP* (CluVRP, Sevaux and Sörensen, 2008), imposes that all customers belonging to the same cluster are visited consecutively: only if a cluster is completely served, visits to customers of another cluster are allowed. The CluVRP has been approached by exact optimization algorithms (Battarra *et al.*, 2014) as well as metaheuristics (Barthélemy *et al.*, 2010; Expósito Izquierdo *et al.*, 2013; Vidal *et al.*, 2015; Expósito-Izquierdo *et al.*, 2016; Defryn and Sörensen, 2017; Hintsch and Irnich, 2018; Pop *et al.*, 2018). The latter problem is the *soft-clustered VRP* (SoftCluVRP). The SoftCluVRP is a restriction of the *capacitated VRP* (CVRP, Pecin *et al.*, 2017) and a relaxation of the clustered VRP, because visits to customers of the same cluster may or may not be interrupted by visits to other customers. It was recently introduced by Defryn and Sörensen (2017) where it is heuristically solved with a fast two-level variable neighborhood search. Two newer works solve the SoftCluVRP exactly (Hintsch and Irnich, 2019) and heuristically (Hintsch, 2019).

We follow the same taxonomy regarding hard and soft clustering here: The SoftCluCARP is defined on an undirected graph $G = (V, E)$ with vertex set V and edge set E. One of the vertices is the unique depot vertex $0 \in V$ representing the location where a fleet of m homogeneous vehicles, all with capacity Q, is housed. The edges are partitioned into *required edges* E_R and *deadheading edges* $E \setminus E_R$, where the former must be traversed at least once in a feasible solution and the latter can be traversed if convenient. Let $c_e > 0$ be the cost for traversing an edge $e \in E$; note that we do not distinguish between service and deadheading costs, because any possible difference just leads to a fixed overall cost offset. Specific for the SoftCluCARP is that the required edges are again partitioned into clusters with $E_R = \bigcup_{h \in H} E_h$ and $E_h \cap E_{h'} = \varnothing$ for $h \neq h'$ (H is the index set of the clusters). Each cluster E_h for $h \in H$ has a positive demand d_h.

The SoftCluCARP is the problem of finding a least-cost set of feasible routes serving all clusters. Let w be a closed walk in G traversing the depot 0. We define a *route* as a combination of such a walk w and a subset $H' \subset H$ served by the walk, meaning that all edges $\bigcup_{h \in H'} E_h$ are traversed at least once. Clearly, a route (w, H') is *feasible* if $\sum_{h \in H'} d_h \leq Q$, and in this case the walk w also feasibly serves all subsets of H'. Let the (routing) cost of w be c_w, i.e., the sum of the edge costs of the walk (edges traversed more than once are counted according to their frequency). Then, $(w_p, H'_p)_{p=1}^{m'}$ is a feasible solution to the SoftCluCARP, if all walks w_p feasibly serve H'_p, respectively, $m' \leq m$, and $H = \bigcup_{p=1}^{m'} H'_p$ holds. A feasible solution is optimal if it minimizes $\sum_{p=1}^{m'} c_{w_p}$.

The focus of this paper is on the exact solution of the SoftCluCARP by means of a *branch-price-and-cut* (BPC) solution approach. Following the recent survey of Costa *et al.* (2019), BPC is the leading exact methodology for solving many types of VRPs. A BPC algorithm is a branch-and-bound algorithm in which the lower bounds are computed by column generation and cuts are added dynamically to strengthen the linear relaxations. Column generation is iterative and solves, at each iteration, a *restricted master problem* (RMP) and one or several pricing problems. For most VRPs, the pricing problem is an elementary *shortest path problem with resource constraints* (SPPRC), which can be solved by a labeling algorithm (see Irnich and Desaulniers, 2005). When trying to solve the SoftCluVRP with a column-generation algorithm, Hintsch and Irnich (2019) observed that classical labeling-based solution approaches for the SPPRC subproblem work rather poorly, even if the algorithm was featured with otherwise very potent labeling acceleration techniques. Surprisingly, a direct MIP-based approach for the pricing subproblem performed significantly better, solving instances with 400+ customers and 50+ clusters. Since labeling-based approaches for the CARP (Bartolini *et al.*, 2011; Bode and Irnich, 2012, 2014, 2015) are certainly more difficult and less effective compared to those for the CVRP (Pecin *et al.*, 2017), trying a labeling-based approach for the SoftCluCARP subproblem seems very unpromising.

Accordingly, our main contributions are the following:

- We develop new *integer programming* (IP)-based pricing algorithms for SoftCluCARP-tailored BPC algorithms: The first one is based on an undirected formulation inspired by a model of Aráoz *et al.* (2009a) for the *clustered prize-collecting arc routing problem*. The formulation comprises two exponentially-sized families of constraints for ensuring connectivity and even vertex degrees. A major difference to our subproblem is, however, that our clusters are typically not disjoint connected components of the graph spanned by the required edges.
  The second one uses a windy type of formulation as used by Corberán *et al.* (2011) for the *windy clustered prize-collecting arc-routing problem*. Also their work assumes disjoint clusters. A windy model has the advantage of avoiding an exponentially-sized family of constraints ensuring even vertex degrees, but the disadvantage of having double the number of arc-flow variables. We prove that when this type of model is used for symmetric instances, the arc-flow variables can be restricted to binary values.
  For both formulations, we develop *branch-and-cut* (B&C) algorithms to be used for pricing and rigorously compare both types of subproblem algorithms.
- Subset-row inequalities (SRIs, Jepsen *et al.*, 2008) have been identified as essential for strengthening the linear relaxation of the master problem for many types of set-partitioning and set-packing problems.

We show that there are at least two fundamentally different ways to incorporate the dual prices of SRIs in the two IPs used for solving the pricing subproblem. In contrast to many other works, we do not only consider SRIs for three rows but also for four and five rows.

- In comprehensive computational tests, we parameterize the branch-and-cut algorithms as well as a tailored heuristic pricing algorithm for the subproblem. Moreover, we show that the overall BPC algorithms for the SoftCluCARP are highly competitive: some large-sized and almost all medium-sized SoftCluCARP instances can be solved to optimality within relatively short time.

The remainder of this work is structured as follows: In Section 2, we present a two-index formulation and a straightforward set-partitioning formulation for the SoftCluCARP as well as the undirected and windy formulations of the column-generation subproblem. B&C-based solution algorithms for the two latter formulations are developed in Section 3. This section also discusses heuristic pricing techniques used to accelerate the column-generation process. Section 4 focusses on providing integer solutions by incorporating SRIs and by branching. The generation of SoftCluCARP benchmark instances, results of the computational studies analyzing the components of the BPC algorithm separately, and the overall performance of the fine-tuned BPC algorithms are presented and discussed in Section 5. Conclusions close the paper in Section 6.

2. Two-Index, Extensive, and Subproblem Formulations

In this section, the SoftCluCARP is formally defined by a two-index formulation. Moreover, an extended set-partitioning formulation is given and later used as the master program of the BPC algorithm. Finally, the two new subproblem formulations are presented.

In the four different models we use the following standard notation: For a vertex $i \in V$, the set $\delta(i)$ comprises the edges having vertex i as an endpoint. Further, for a subset $S \subseteq V$, the set $\delta(S)$ contains all edges with one endpoint in S and the other one in $V \setminus S$, and the set $E(S)$ contains all edges with both endpoints in S. For all clusters $h \in H$, let V_h be the set of vertices that are endpoints of edges $e \in E_h$. Note that we do *not* assume that the subgraphs (V_h, E_h) for $h \in H$ are connected. Note also that the sets $(V_h)_{h \in H}$ are typically *not* disjoint.

Finally, to simplify formulas, an expression $q(I)$ abbreviates the term $\sum_{i \in I} q_i$ using the implicit assumption that q is a vector with entries for a superset of the indices $i \in I$.

2.1. Two-Index Formulation

In the arc-routing context, two-index formulations refer to models in which the edge/arc-flow variables have one index for the edge/arc and a second index for the vehicle that they refer to. Let the m available vehicles form a fleet $K = \{1, 2, \dots, m\}$. Our two-index formulation for the SoftCluCARP has non-negative integer variables y_e^k indexed by $(e, k) \in E \times K$ indicating the number of times that vehicle k deadheads edge e. In addition, the binary variables z_h^k signal whether (or not) vehicle k serves all required edges of cluster E_h. Auxiliary non-negative integer variables p_i^k, one for each pair $(i, k) \in V \times K$, are used to enforce an even vertex degree at vertex i in the walk performed by vehicle k. Note that the following two-index formulation can be derived from the two-index formulation of Belenguer and Benavent (1998) for the CARP by replacing all of their vehicle-specific service indicator variables x_e^k by our binary indicator z_h^k for all

$e \in E_h$, $h \in H$, and $k \in K$:

$$\min \sum_{k\in K}\sum_{h\in H} c(E_h) z_h^k + \sum_{k\in K}\sum_{e\in E} c_e y_e^k \tag{1a}$$

$$\text{subject to} \sum_{k\in K} z_h^k = 1 \qquad \forall h \in H \tag{1b}$$

$$\sum_{h\in H} |\delta(S)\cap E_h| z_h^k + \sum_{e\in\delta(S)} y_e^k \geq 2 z_h^k \qquad \forall S \subseteq V\setminus\{0\}, h \in H : E(S)\cap E_h \neq \varnothing, k \in K \tag{1c}$$

$$\sum_{h\in H} |\delta(i)\cap E_h| z_h^k + \sum_{e\in\delta(i)} y_e^k = 2 p_i^k \qquad \forall i \in V, k \in K \tag{1d}$$

$$\sum_{h\in H} d_h z_h^k \leq Q \qquad \forall k \in K \tag{1e}$$

$$p_i^k \in \mathbb{Z}_+ \qquad \forall i \in V, k \in K \tag{1f}$$

$$y_e^k \in \mathbb{Z}_+ \qquad \forall e \in E, k \in K \tag{1g}$$

$$z_h^k \in \{0,1\} \qquad \forall h \in H, k \in K \tag{1h}$$

The objective (1a) minimizes the overall traversal cost, where the first term is constant and describes the service cost while the second term describes the deadheading cost. That every cluster is serviced by exactly one of the vehicles is ensured by equations (1b). The connectivity of all walks performed by the vehicles is guaranteed by constraints (1c) and the even vertex degree by constraints (1d). Inequalities (1e) are vehicle capacity constraints. The domains of all decision variables are given by (1f)–(1h).

With this formulation, it is possible to find solutions that use less than m routes/vehicles. Indeed, setting to zero all decision variables p_i^k, y_e^k, and z_h^k for a fixed $k \in K$ is admissible if a bin-packing solution exists to the instance $(Q, (d_h)_{h\in H})$ that uses less than m bins.

The two-index model has two weaknesses. First, the number of variables grows in $|K|$. Second, and more seriously, the inherent symmetry with respect to the numbering of the vehicles makes a branch-and-bound-based approach as used in MIP solvers ineffective (Bode and Irnich, 2012, p. 1169): Note that for a given solution, any permutation of the vehicle indices $k \in K$ leads to one of $|K|!$ equivalent solutions. Even adding symmetry breaking constraints can only very partially eliminate the ineffectiveness in branching (Adulyasak *et al.*, 2014).

2.2. Extensive Route-Based Formulation

The following route-based formulation completely eliminates symmetry with respect to the vehicle indices. Let Ω be the set of all routes that feasibly serve some clusters. Recall that we can represent each element $r \in \Omega$ as a pair $r = (w, H')$ where w is a closed walk traversing the depot and $H' \subset H$ indicates which clusters are served. The following extensive path-based formulation uses binary variables $\lambda_r \in \{0,1\}$ to indicate whether route $r = (w, H') \in \Omega$ is selected.

$$\min \sum_{r=(w,H')\in\Omega} c_w \lambda_r \qquad \text{duals:} \tag{2a}$$

$$\text{subject to} \sum_{\substack{r=(w,H')\in\Omega:\\ h\in H'}} \lambda_r = 1 \qquad \forall h \in H \qquad [\pi_h] \tag{2b}$$

$$\sum_{r\in\Omega} \lambda_r \leq m \qquad [\mu] \tag{2c}$$

$$\lambda_r \in \{0,1\} \qquad \forall r \in \Omega \tag{2d}$$

This model is an extended set-partitioning model. The overall routing cost are minimized by (2a). Constraints (2b) are the partitioning constraints stating that every cluster has to be served exactly once. The fleet-size constraint (2c) requires that exactly m routes are selected. The domain constraints of the binary route variables are stated in (2d).

Note that the partitioning constraints (2b) can be replaced by covering constraints using inequalities with ≥ 1, because for any feasible $r = (w, H') \in \Omega$, all routes $r' = (w, H'')$ serving a subset $H'' \subsetneq H'$ are also feasible and have identical cost. Therefore, we assume covering constraints in the following.

The linear relaxation of the model (2) over a subset $\Omega' \subset \Omega$ of the routes is the RMP of the BPC algorithms that we use to solve the SoftCluCARP. Note that the set of routes Ω can be drastically reduced without sacrificing optimality. For a given subset $H' \subset H$, one can determine a least cost-walk $w = w(H')$. Finding this walk is the well-known *undirected rural postman problem* (URPP, Ghiani and Laporte, 2014) over the graph $G = (V, E)$ with required edges $\bigcup_{h \in H'} E_h$. In Section 3, we discuss in more detail how to exactly solve URPPs to only have routes performing least-cost walks in the RMP.

2.3. Subproblem Formulations

In the iterative column-generation process, the subproblem must identify negative reduced-cost variables (=routes) or prove that there exists none. Let $(\pi_h)_{h \in H}$ be the dual prices of the covering constraints (2b) and let μ be the dual price of the fleet-size constraint (2c). The reduced cost of a route $r = (w, H') \in \Omega$ is then

$$\tilde{c}_r = c_w - \sum_{h \in H'} \pi_h - \mu, \tag{3}$$

with the feasibility condition that $\sum_{h \in H'} d_h \leq Q$ must hold.

We can analyze the structure of the subproblem now: First, following the taxonomy introduced by Feillet *et al.* (2005), the subproblem can be characterized as a *profitable* postman tour problem: Reduced-cost minimization requires routing cost minimization in combination with profit maximization in the objective. Due to the valid replacement of partitioning by covering constraints dual values π_h are non-negative for all $h \in H$. Undirected and windy profitable postman problems are covered by works of Aráoz *et al.* (2006); Ávila *et al.* (2016); the more general class of postman problems with profits is an active research field and is comprehensively surveyed in Archetti and Speranza (2014); Mourão and Pinto (2017). Second, the selected clusters described by H' do not necessarily form a connected graph, i.e., $(\bigcup_{h \in H'} V_h, \bigcup_{h \in H'} E_h)$ may be disconnected. Therefore, the subproblem is clearly a *rural* postman problem (see, Eiselt *et al.*, 1995a; Ghiani and Laporte, 2014). Third, the clustering aspect makes the subproblem a *clustered* postman problem as described and analyzed in Franquesa (2008); Aráoz *et al.* (2009a); Corberán *et al.* (2011); Aráoz *et al.* (2013). Recall that the VRP literature would characterize these problems as soft-cluster constrained.

Even if earlier works cover the individual aspects, none of these works covers exactly the subproblem to solve for the SoftCluCARP. One major difference is that the earlier works on clustered postman problems assume disjoint clusters, i.e., the sets V_h for $h \in H$ have pairwise empty intersection. This is certainly not fulfilled in the SoftCluCARP context.

2.3.1. Undirected Formulation

Our first formulation of the subproblem is undirected using the graph $G = (V, E)$ directly and exploiting the fact that a least-cost walk in G traverses each edge at most twice. Therefore, binary variables x_e and y_e indicate the first and second traversal for all edges $e \in E$, respectively. Note that the first traversal can either be a service or deadheading, while the second traversal is always deadheading. In order to select clusters to be serviced, a third set of binary variables z_h with $h \in H$ is needed. The model reads as follows:

$$\tilde{c}(\pi_h, \mu) = \min \sum_{e \in E} c_e x_e + \sum_{e \in E} c_e y_e - \sum_{h \in H} \pi_h z_h - \mu \tag{4a}$$

$$\begin{aligned}
\text{subject to} \quad & x_e \geq y_e && \forall e \in E && \text{(4b)}\\
& z_h \leq x_e && \forall e \in E_h, h \in H && \text{(4c)}\\
& x(\delta(S) \setminus F) + y(F \setminus L) \geq x(F) + y(L) + 1 - |F| - |L| && \forall S \subseteq V \setminus \{0\}, && \text{(4d)}\\
& && \varnothing \subseteq L \subseteq F \subseteq \delta(S) \text{ with } |L| + |F| \text{ odd} &&\\
& x(\delta(S)) + y(\delta(S)) \geq 2x_e && \forall S \subseteq V \setminus \{0\}, e \in E_R(S) && \text{(4e)}\\
& \sum_{h \in H} d_h z_h \leq Q && && \text{(4f)}\\
& x_e \in \{0, 1\} && \forall e \in E && \text{(4g)}\\
& y_e \in \{0, 1\} && \forall e \in E && \text{(4h)}\\
& z_h \in \{0, 1\} && \forall h \in H && \text{(4i)}
\end{aligned}$$

The profitable tour objective (4a) minimizes the difference between the cost of the walk (first two terms) and the profit resulting from the clusters that are served (third term). The last constant term μ is added to correctly describe the reduced cost $\tilde{c}(\pi_h, \mu)$ for the route $r = (w, H')$, where the walk w results from selecting each edge $x_e + y_e$ times and the subset is $H' = \{h \in H : z_h = 1\}$. The coupling constraints (4b) state that a second traversal is only possible after a first traversal. The second class of coupling constraints (4c) guarantees that a profit for cluster E_h is only collected if all edges are traversed. The generalized *cocircuit inequalities* (4d) (a.k.a. odd cut inequalities) are inspired by the models of Aráoz *et al.* (2009a,b). They ensure an even vertex degree in the graph imposed by $x + y$: If the number of traversals over the cut set $\delta(S)$ is odd, one can define F as the set of edges traversed at least once and L as the set of edges traversed a second time. Then $|F| + |L|$ is odd and the inequality imposes that at least one more edge of the cut set needs to be chosen. The connectivity of the imposed walk results from inequalities (4e). The capacity constraint is (4f) and the domains of all decision variables are given by (4g)–(4i).

The cocircuit inequalities (4d) and connectivity constraints (4e) are two classes of mandatory inequalities of exponential size. Hence, the formulation (4) is typically not applicable out-of-the-box. Instead, cutting-plane procedures to identify violated inequalities are used to add them dynamically to the respective relaxed formulation. We describe the B&C algorithms including details of the separation algorithms in Section 3.2.

2.3.2. Windy Formulation

Our motivation to develop an alternative formulation for the subproblem is threefold. First, windy formulations can be stated without using cocircuit inequalities so that the only exponentially sized class of constraints are connectivity constraints. This makes the formulation somewhat more elegant. Second, we suspect that modern MIP solvers can exploit the network-flow nature of windy models so that they can be solved faster than undirected models (like model (4)) which do not comprise any flow-conservation constraints. Third, we found a property of optimal solutions to undirected postman problems that can be exploited when a windy formulation is used for its solution. We present this property in the following:

Proposition 1. *Let P be an instance of an undirected postman problem that can be solved by determining a cost-minimal Eulerian extension. We assume that all edge costs are non-negative. Then, there exists an optimal postman tour (a walk) w for P such that no edge is traversed in the same direction more than once.*

Proof. Every optimal solution to P imposes a Eulerian extension (i.e., a multi-graph) denoted by $G^{ext} = (V, E^{ext})$. For an optimal solution, we can assume that no edge is traversed more than two times (there are not more than two parallel edges in G^{ext}), because otherwise the removal of two parallel copies of such an edge from the Eulerian extension would create another Eulerian extension covering the same set of edges but with smaller or equal cost.

We can now build a mixed graph G^{mix} from G^{ext} in which all edges traversed twice are replaced by two anti-parallel arcs, i.e., two parallel edges $\{i, j\}$ are replaced by arcs (i, j) and (j, i). All edges traversed only once remain undirected. This graph G^{mix} is a Eulerian mixed graph, because it fulfills the balanced-set conditions (see Eiselt *et al.*, 1995a, p. 232). Hence, a walk through G^{mix} provides another solution to the original problem with the required property. □

As a consequence of Proposition 1, our new windy formulation of the subproblem contains one binary variable x_{ij} and one binary variable x_{ji} for each $e = \{i, j\} \in E$ to show whether the edge is traversed in the indicated direction (from i to j and/or from j to i). As before, the set of binary variables z_h with $h \in H$ indicates service to the respective cluster.

$$\tilde{c}(\pi_h, \mu) = \min \sum_{\{i,j\}\in E} (c_{ij}x_{ij} + c_{ji}x_{ji}) - \sum_{h\in H} \pi_h z_h - \mu \tag{5a}$$

$$\text{subject to} \quad x_{ij} + x_{ji} \geq z_h \qquad \forall \{i, j\} \in E_h, h \in H \tag{5b}$$

$$\sum_{\{i,j\}\in\delta(i)} (x_{ij} - x_{ji}) = 0 \qquad \forall i \in V \tag{5c}$$

$$x(\delta_A(S)) \geq 2z_h \qquad \forall h \in H, S \subseteq V \setminus \{0\} \text{ with } E_h \cap E(S) \neq \varnothing \tag{5d}$$

$$\sum_{h\in H} d_h z_h \leq Q \tag{5e}$$

$$x_{ij}, x_{ji} \in \{0, 1\} \qquad \forall \{i, j\} \in E \tag{5f}$$

$$z_h \in \{0, 1\} \qquad \forall h \in H \tag{5g}$$

The profitable tour objective (5a) minimizes the reduced cost of the resulting route, with the first term for the routing cost, the second for the collected profit, and the last term with the constant μ. The coupling constraints (5b) ensure that selected clusters are completely traversed. Equations (5c) are the flow-conservation constraints which actually ensure an even vertex degree at all vertices. The connectivity of the imposed postman tour results from inequalities (5d), where

$$\delta_A(S) = \{(i, j), (j, i) : \{i, j\} \in E \text{ with } i \in S, j \notin S \text{ or } i \notin S, j \in S\}.$$

Inequality (5e) is the capacity constraint. The domains of the variables are stated in (5f) and (5g).

The model (5) is an adaptation of the model presented by Corberán *et al.* (2011). However, Corberán *et al.* (2011) systematically exploited that their clusters are vertex-disjoint, which is not fulfilled in our case.

3. Solution of the Subproblem

In many BPC algorithms for routing applications, more than 99 percent of the time is spent with solving the pricing subproblems and separating violated valid inequalities for the master program. This is also true for our SoftCluCARP-tailored BPC algorithm. We now focus on the fast heuristic and exact solution of the subproblem (Sections 3.1 and 3.2), while subset-row inequalities are discussed in the next Section 4.

3.1. Primal Heuristics

The main idea of the primal heuristics is to start from a basic solution of the RMP with columns and associated routes of reduced cost zero. For such a route $r = (w, H') \in \Omega$ with walk w and served subset $H' \subset H$, we systematically alter the subset H' into H'', compute a new cost-minimal walk w' traversing H'' and the depot 0, and compute the reduced cost of the new route $r' = (w', H'')$. An important observation is that the reduced cost $\tilde{c}_{r'}$ decomposes into two parts $c_{w'}$ and $-\sum_{h\in H''} \pi_h - \mu$, where the first part is the routing cost $c_{w'}$ of the walk w' independent of the actual dual solution, while the second is fully determined by H'' and independent of the walk.

Regarding the modification of H', we use *add* and *drop operators*, where the add operator adds one element $h \in H \setminus H'$ to H' resulting in the new subset $H'' = H' \cup \{h\}$. We only allow feasible additions, i.e., require $d(H'') \leq Q$. For the drop, any $h \in H'$ can be removed resulting in $H'' = H' \setminus \{h\}$. Both neighborhoods are of linear size $\mathcal{O}(|H|)$.

The computation of a cost-minimal walk w' for the subset H'', denoted by $w(H'')$ in the following, requires the solution of an URPP on a modified graph. In order to ensure that feasible routes traverse the depot 0, we introduce the additional edge $e_0 = \{0, 0\}$ (this is a loop) and the additional depot cluster E_0 containing only the edge e_0. The cost of e_0 is defined as $c_{e_0} = 0$ and the demand of cluster E_0 is defined as $d_0 = 0$. Moreover, let $H_0 = H \cup \{0\}$, $E_{00} = E \cup \{\{0, 0\}\}$, and $G_{00} = (V, E_{00})$. We define a new set of required edges as $R = \{e_0\} \cup \bigcup_{h \in H''} E_h$. Now, the solution of an URPP on $G_{00} = (V, E_{00})$ with required edges R provides the walk w' and its routing cost $c_{w'}$.

We now discuss the three basic components of the primal heuristics which are the exact solution algorithm for URPPs, the use of a hash table, and the metaheuristic that controls how add and drop operators are applied.

3.1.1. Solution of URPPs

Although the URPP is an $\mathcal{NP}$-hard problem, rather large instances of the URPP can nowadays be routinely solved with the approach proposed by Ghiani and Laporte (2014). In a first step, the instance given by $G_{00} = (V, E_{00})$ with required edges R can be preprocessed and reduced so that all remaining vertices of the equivalent transformed graph are incident to at least one edge of R. Let $G(R) = (V(R), E(R))$ be this transformed graph (depending on the set of required edges). Note that all edges R remain unchanged so that $R \subset E(R)$ holds true.

In a second step, a *minimum spanning tree* (MST) is computed on the component graph, i.e., the graph resulting from contracting all edges R in $G(R) = (V(R), E(R))$. Ghiani and Laporte (2014) have shown that there always exists an optimal URPP solution in $G(R)$ where all edges are deadheaded at most once except for those edges that belong to the MST solution. These edges must be allowed to be traversed (=deadheaded) twice. It should be noted that in our application, the number of components is typically very small, because the clusters often overlap in some vertices.

In the last step, a binary formulation for the URPP on $G(R) = (V(R), E(R))$ is constructed and solved with B&C. The binary variables x_e of this formulation indicate deadheadings. For those edges that may be deadheaded twice, two binary variables are present. The formulation has only two types of constraints, one set to ensure connectivity of the components and a second set of cocircuit constraints to guarantee that all vertices have an even degree in the solution (for further details we refer to Ghiani and Laporte, 2014):

$$\sum_{e \in \delta_{G(R)}(S)} x_e \geq 2 \qquad \forall S \subset \text{non-empty union of components of } G(R) = (V(R), R) \tag{6a}$$

$$\sum_{e \in \delta_{G(R)}(S) \setminus F} x_e - \sum_{e \in F} x_e \geq 1 - |F| \qquad \forall S \subset V(R), F \subseteq \delta(S) \text{ with } |F| + |R \cap \delta_{G(R)}(S)| \text{ is odd} \tag{6b}$$

where $\delta_{G(R)}(S)$ is the cut set of S in the transformed graph $G(R)$. Since (6a) and (6b) are simpler versions of the connectivity constraints (4e) and (5d) and cocircuit inequalities (4d), respectively, we do not discuss their separation in length but refer to Section 3.2 where we present the B&C algorithms for the subproblems (4) and (5). We only mention here that compared to the work of Ghiani and Laporte (2014), we use more efficient algorithms of Letchford *et al.* (2004, 2008) for the exact separation of violated cocircuit inequalities.

3.1.2. Hash Table of URPP Results

Note that the solution of the URPP only depends on the required edges R that are in turn determined by the given cluster subset H''. After solving the URPP for the subset H'', we store the corresponding routing cost $c_{w(H'')}$ of the optimal walk $w(H'')$ in a *hash table* (Cormen *et al.*, 2009, chapter 11). The hash table is exploited in two ways:

(i) If the URPP for a given subset H'' has already been solved, there exists an entry in the hash table and we simply use the already computed cost $c_{w(H'')}$ instead of solving the URPP again.

(ii) Before starting the add-drop-based metaheuristic (Section 3.1.3), we search for negative reduced-cost routes by iterating over the hash table. As the reduced cost $\tilde{c}_{r'}$ of a route $r' = (w(H''), H'')$ decomposes into $c_{w(H'')}$ and $-\sum_{h \in H''} \pi_h - \mu$, each hash table entry provides the first term while the second term can be quickly computed in $\mathcal{O}(|H''|)$ time. All routes r' with negative reduced-cost $\tilde{c}_{r'} < 0$ are added to the RMP, which is then re-optimized. We refer to this pricing strategy as *hash-table inspection*. Overall, pricing is then performed in a three-level hierarchy with hash-table inspection first, add-drop-based metaheuristic second, and B&C third.

3.1.3. Add-Drop-based Metaheuristic

If no negative reduced-cost route was found by searching the hash table (Section 3.1.2), we apply an add-drop-based metaheuristic. Starting from the primal solution $(\bar{\lambda}_r)_{r \in \Omega'}$ of the RMP, we loop over all routes $r \in \Omega'$ with $\bar{\lambda}_r > 0$. For each of these routes, we apply the primal heuristic `Add-Drop-based Metaheuristic`(r_{init}) given by Algorithm 1 and described in the following.

The main loop of the primal heuristic (Steps 2–16) runs for *MaxIter* iterations. Steps 3–8 comprise a variable neighborhood descent (VND, Hansen and Mladenović, 2001) including a drop and an add operator: First, we search for the best cluster $h \in H'$ to drop from the current route $r = (w, H')$. If the dropping results in an improvement in reduced cost $\tilde{c}_r$, cluster h is removed from r and the procedure is repeated. Second, if no improvement was found, we search for the best cluster $h \in H \setminus H'$ that is currently not served by r but can be added as it respects the capacity constraint. If this results in an improvement, we repeat the procedure starting with the drop operator. Otherwise, the VND is terminated. Afterwards, in Steps 9–14 the best derived route r^* is updated or the current route is reset to r^*. Possibly, r^* is returned as a negative reduced-cost route, if $\tilde{c}_{r^*}$ is negative. Otherwise, a random cluster is dropped from the current route (Steps 15–16), resulting in the starting solution for the next iteration.

Algorithm 1: `Add-Drop-based Metaheuristic`(r_{init})

Input: A feasible route $r_{init} = (w, H')$
Output: A negative reduced-cost route r^* or `FAILED` if none is found

1 $r^* := r := r_{init} = (w, H')$
2 **for** *Iter* $= 1, 2 \ldots,$ *MaxIter* **do**
3   **do**
4   **do**
5   `BestImprovementMove`(r, `DropCluster`, $h \in H'$)
6   **while** *improvement found*
7   `BestImprovementMove`(r, `AddCluster`, $h \in H \setminus H'$ with $d_h + d(H') < Q$)
8   **while** *improvement found*
9   **if** $\tilde{c}_r < \tilde{c}_{r^*}$ **then**
10         $r^* := r$
11   **if** $\tilde{c}_{r^*} < 0$ **then**
12   **return** r^*
13   **else**
14         $r := r^*$
15   Randomly choose $h \in H'$
16   `Move`(r, `DropCluster`, h)
17 **return** `FAILED`

3.2. Branch-and-Cut

To solve the pricing subproblem exactly, we use a B&C algorithm for either of the two formulations (4) and (5) presented in Section 2.3. Both models include connectivity constraints in the form of (4e) and (5d), respectively. While cocircuit constraints (4d) are needed for the validity of the first formulation, they are not mandatory for the second. However, it is straightforward to show that the following cocircuit inequalities are valid for windy formulations like (5) in which all routing variables are binary. For any $S \subset V$ and $F \subseteq \delta_A(S)$ with $|F|$ odd, the cocircuit inequalities are

$$x(\delta_A(S) \setminus F) + x(F) \geq 1 + |F|. \tag{7}$$

3.2.1. Separation of violated Connectivity Constraints

The algorithms used for separating violated connectivity constraints (4e) and (5d) are based on procedures described in several works on rural postman problems (see Ghiani and Laporte, 2014, and the various references given there). Let $(\bar{x}, \bar{y}, \bar{z})$ (or $(\bar{x}, \bar{z})$) be the possibly fractional solution of a relaxation of (4) (or (5)), i.e., we want to separate the respective vector from the feasible integer solutions. Separation is done by constructing an undirected weighted graph and solving min-cut problems in it: For each $e \in E$, define the weight $\mathbf{w}_e = \bar{x}_e + \bar{y}_e$ in the undirected case and $\mathbf{w}_e = \bar{x}_{ij} + \bar{x}_{ji}$ for $e = \{i, j\}$ in the windy case. Let the weighted graph $G_{\mathbf{w}} = (V_{\mathbf{w}}, E_{\mathbf{w}})$ be the edge-induced subgraph of G induced by the edges with positive weight, i.e., by $E_{\mathbf{w}} = \{e \in E : \mathbf{w}_e > 0\}$ (note that in general only a proper subset $V_{\mathbf{w}} \subseteq V$ of the vertices is present).

We compute the connected components of $G_{\mathbf{w}}$ using a *union-find algorithm* (Cormen *et al.*, 2009, chapter 21). Any component $S \subset V_{\mathbf{w}}$ of $G_{\mathbf{w}}$ not containing the depot 0 provides a potential set S for a violated connectivity constraint. In the undirected case, we next determine an edge $e \in E_R(S)$ having maximum value $\bar{x}_e > 0$. In the windy case, we determine a cluster $h \in H$ with $E_h \cap E(S) \neq \varnothing$ and maximum value $\bar{z}_h > 0$. Then, (4e) is violated for (S, e) (or (5d) for (S, h)). We refer to this componentwise test as the *level-1 separation*.

If the graph $G_{\mathbf{w}}$ is connected, we calculate a minimum-cut tree for it (Gomory and Hu, 1961). For an edge of the cut tree, let S be the cut set that separates the two end-vertices of the edge. In the undirected case, for each such set S, we first find an edge $e \in E_R(S)$ with maximum weight $\bar{x}_e$. If $\mathbf{w}(\delta(S)) < 2\bar{x}_e$, the connectivity constraint (4e) for the pair (S, e) is violated. The windy case works analogously considering clusters $h \in H$ with $E_h \cap E(S) \neq \varnothing$ and their values $\bar{z}_h > 0$. We refer to this procedure as the *level-2 separation*.

3.2.2. Separation of violated Cocircuit Constraints

We separate violated cocircuit constraints again with a 2-level algorithm. For the cocircuit constraints of the form (7), the algorithm of Letchford *et al.* (2004, 2008) is directly applicable. The algorithm constructs another weighted multi-graph in which the flow values $\bar{x}$ produce weights $\min\{\bar{x}, 1 - \bar{x}\}$.

We first sketch the algorithm for the windy model (5): Each pair $\bar{x}_{ij}$ and $\bar{x}_{ji}$ produces two parallel edges $e = \{i, j\}$ and $e' = \{i, j\}$ with weights $\tilde{\mathbf{w}}_e = \min\{\bar{x}_{ij}, 1 - \bar{x}_{ij}\}$ and $\tilde{\mathbf{w}}_{e'} = \min\{\bar{x}_{ji}, 1 - \bar{x}_{ji}\}$, respectively. Let the undirected multi-graph $G_{\tilde{\mathbf{w}}} = (V_{\tilde{\mathbf{w}}}, E_{\tilde{\mathbf{w}}})$ be the edge-induced subgraph of G induced by the edges with positive weight. The level-1 separation checks whether $G_{\tilde{\mathbf{w}}}$ is disconnected, and if so, it considers the connected components. For each connected component $S \subseteq V$, the arc set $F = \{(i, j) \in \delta_A(S) : 1 - \bar{x}_{ij} < \bar{x}_{ij}\} \cup \{(j, i) \in \delta_A(S) : 1 - \bar{x}_{ji} < \bar{x}_{ji}\}$ is determined. If $|F|$ is even, then either one arc is removed from F or one arc from $\delta(S) \setminus F$ is added to F, in order to make F odd. The arc with smallest value $|1 - 2\bar{x}_{ij}|$ (or $|1 - 2\bar{x}_{ji}|$) is chosen. If $\bar{x}(\delta(S) \setminus F) + \bar{x}(F) < 1 + |F|$ the cocircuit constraint (7) for this pair (S, F) is violated.

The level-2 separation continues the algorithm of Letchford *et al.* (2004, 2008) by computing a cut tree for each component of $G_{\tilde{\mathbf{w}}}$. An edge in the cut tree further decomposes the component S into $S = S' \cup \bar{S}'$ with $S' \neq \varnothing$ and $\bar{S}' = S \setminus S' \neq \varnothing$. The above computation of the set F (now a subset of $\delta_A(S')$) including the parity check and the subsequent check of the violation is done analogously as described above. As proven by Letchford *et al.*, the level-1 and level-2 procedures together yield an exact cocircuit-separation algorithm.

For the separation of violated cocircuit constraints (6b) in the URPP model (see Section 3.1.1), the exactly same two-level separation is applicable.

Finally, Aráoz *et al.* (2009b) have shown how the above procedure has to be modified in order to separate violated cocircuit inequalities of the form (4d), where $L \subseteq F \subseteq \delta(S)$ and $|L| + |F|$ needs to be odd. Similar to the original procedure, one first defines tentative sets $L = \{e \in \delta(S) : 1 - \bar{y}_e < \bar{y}_e\}$ and $F = \{e \in \delta(S) : 1 - \bar{x}_e < \bar{x}_e\}$. Note that constraints (4b), i.e., $x_e \geq y_e$ for all $e \in E$, ensure $L \subseteq F$. If $|L| + |F|$ is even, the consideration of four different cases, described in (Aráoz *et al.*, 2009b, Remark 5.3), adds one edge to or removes one edge from one of the two sets so that finally $|L| + |F|$ becomes odd.

4. Branch-Price-and-Cut

The remaining components of the BPC algorithm are presented in this section. We first elaborate on the cutting strategies and afterwards the branching strategies.

4.1. Cutting

Subset-row inequalities (SRIs, Jepsen *et al.*, 2008) are valid inequalities for set-packing formulations. As these inequalities are directly formulated on the master-program variables and cannot be directly formulated on an original compact model (a model from which the master program can be derived via Dantzig-Wolfe decomposition, see Lübbecke and Desrosiers, 2005), SRIs are considered *non-robust*. The consequence is that additional attributes need to be integrated in the subproblems. Despite the resulting additional effort, later works building on the results of Jepsen *et al.* (2008) have confirmed that the success of many BPC approaches can be attributed to the use of SRIs.

A SRI can be described by a subset $S \subset H$ and weights $u_h > 0$ for all $h \in S$. As separation of violated SRIs is hard, practical approaches typically rely on enumeration and heuristics for sets S of restricted size. Table 1 shows the non-dominated combinations of weights for all SRIs defined over sets S of size $|S| \in \{3, 4, 5\}$, taken from Pecin *et al.* (2017). In all cases, the SRI associated with $(S, (u_h)_{h \in H})$ is of the form

$$\sum_{r=(w,H')\in\Omega} \left\lfloor \sum_{h\in S\cap H'} u_h \right\rfloor \lambda_r \leq \left\lfloor \sum_{h\in S} u_h \right\rfloor . \qquad [\sigma_{S,u}] \tag{8}$$

Let the dual price of the SRI defined by (S, u) be $\sigma_{S,u}$. The consequence is that the reduced-cost formula (3) of a route $r = (w, H')$ must be extended and becomes

$$\tilde{c}_r = c_w - \sum_{h\in H'} \pi_h - \mu - \sum_{(S,u)} \left\lfloor \sum_{h\in S\cap H'} u_h \right\rfloor \sigma_{S,u}, \tag{9}$$

where the last sum is taken over all active SRIs defined by (S, u).

We next show how to handle the dual prices $\sigma_{S,u}$ in the subproblem: For each active SRI defined by (S, u), a non-negative integer variable $s_{S,u} \in \mathbb{Z}_+$ must be added to formulation (4) or (5), respectively. The variable $s_{S,u}$ describes the coefficient $\lfloor \sum_{h\in S\cap H'} u_h \rfloor$ of the route (w, H') computed by the subproblem, see equation (8). Hence, this variable is added with the coefficient $-\sigma_{S,u}$ to the objectives (4a) and (5a).

Moreover, there are at least two possibilities to couple the new variable $s_{S,u}$ with the decisions z_h for $h \in H$. The first possibility is a single constraint of the form

$$\sum_{h\in S} p_h z_h - q\, s_{S,u} \leq q - 1 \tag{10}$$

where the weights u_h are written as fractions $u_h = p_h/q$ with nominators $p_h \in \mathbb{Z}_{>0}$ and unique denominator $q \in \mathbb{Z}_{>0}$. For example, $|S| = 3$ and $(u_{h_1}, u_{h_2}, u_{h_3}) = (1/2, 1/2, 1/2)$ produces the inequality $z_{h_1} + z_{h_2} + z_{h_3} - 2s_{S,u} \leq 2-1 = 1$ (forcing $s_{S,u}$ to become one when two or three of the z-variables are one). Another example is $|S| = 5$ and $(u_{h_1}, u_{h_2}, u_{h_3}, u_{h_4}, u_{h_5}) = (2/3, 2/3, 2/3, 1/3, 1/3)$ for which the inequality $2z_{h_1} + 2z_{h_2} + 2z_{h_3} + z_{h_4} + z_{h_5} - 3s_{S,u} \leq 3 - 1 = 2$ results (here $s_{S,u}$ can be forced to become one or two). It is straightforward to prove the validity of (10) by simple term manipulations. We refer to subproblem formulations supplemented with constraints of type (10) as *single SRI-enforcing formulations.*

Size $\lvert S\rvert$	Weights $u = (u_{h_1}, \dots, u_{h_{\lvert S\rvert}})$	Minimal subsets of S $M \in \mathcal{M}(S,u)$
3	$(\frac{1}{2}, \frac{1}{2}, \frac{1}{2})$	$\{h_1,h_2\}, \{h_1,h_3\}, \{h_2,h_3\}$
4	$(\frac{2}{3}, \frac{1}{3}, \frac{1}{3}, \frac{1}{3})$	$\{h_1,h_2\}, \{h_1,h_3\}, \{h_1,h_4\}, \{h_2,h_3,h_4\}$
5	$(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}, \frac{1}{3}, \frac{1}{3})$	$\{h_1,h_2,h_3\}, \{h_1,h_2,h_4\}, \{h_1,h_2,h_5\}, \{h_1,h_3,h_4\}, \{h_1,h_3,h_5\},$ $\{h_1,h_4,h_5\}, \{h_2,h_3,h_4\}, \{h_2,h_3,h_5\}, \{h_2,h_4,h_5\}, \{h_3,h_4,h_5\}$
	$(\frac{2}{4}, \frac{2}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4})$	$\{h_1,h_2\}, \{h_1,h_3,h_4\}, \{h_1,h_3,h_5\}, \{h_1,h_4,h_5\}, \{h_2,h_3,h_4\}, \{h_2,h_3,h_5\}, \{h_2,h_4,h_5\}$
	$(\frac{3}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4})$	$\{h_1,h_2\}, \{h_1,h_3\}, \{h_1,h_4\}, \{h_1,h_5\}, \{h_2,h_3,h_4,h_5\}$
	$(\frac{3}{5}, \frac{2}{5}, \frac{2}{5}, \frac{1}{5}, \frac{1}{5})$	$\{h_1,h_2\}, \{h_1,h_3\}, \{h_1,h_4,h_5\}, \{h_2,h_3,h_4\}, \{h_2,h_3,h_5\}$
	$(\frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2})$	$\{h_1,h_2\}, \{h_1,h_3\}, \{h_1,h_4\}, \{h_1,h_5\}, \{h_2,h_3\}, \{h_2,h_4\}, \{h_2,h_5\},$ $\{h_3,h_4\}, \{h_3,h_5\}, \{h_4,h_5\},$ (with $\sum u_h \geq 1$) $\{h_1,h_2,h_3,h_4\}, \{h_1,h_3,h_4,h_5\}, \{h_1,h_2,h_4,h_5\},$ $\{h_1,h_2,h_3,h_5\}, \{h_2,h_3,h_4,h_5\}$ (with $\sum u_h \geq 2$)
	$(\frac{2}{3}, \frac{2}{3}, \frac{2}{3}, \frac{1}{3}, \frac{1}{3})$	$\{h_1,h_2\}, \{h_1,h_3\}, \{h_1,h_4\}, \{h_1,h_5\}, \{h_2,h_3\}, \{h_2,h_4\},$ $\{h_2,h_5\}, \{h_3,h_4\}, \{h_3,h_5\},$ (with $\sum u_h \geq 1$) $\{h_1,h_2,h_3\}, \{h_1,h_2,h_4,h_5\}, \{h_1,h_3,h_4,h_5\}, \{h_2,h_3,h_4,h_5\}$ (with $\sum u_h \geq 2$)
	$(\frac{3}{4}, \frac{3}{4}, \frac{2}{4}, \frac{2}{4}, \frac{1}{4})$	$\{h_1,h_2\}, \{h_1,h_3\}, \{h_1,h_4\}, \{h_1,h_5\}, \{h_2,h_3\},$ $\{h_2,h_4\}, \{h_2,h_5\}, \{h_3,h_4\},$ (with $\sum u_h \geq 1$) $\{h_1,h_2,h_3\}, \{h_1,h_2,h_4\}, \{h_1,h_3,h_4,h_5\}, \{h_2,h_3,h_4,h_5\}$ (with $\sum u_h \geq 2$)

Table 1: Sets S, non-dominated weights u, and minimal subsets for SRIs associated with S and u.

The second possibility is to add several inequalities per SRI to the model of the subproblems, where each inequality refers to a so-called minimal subset, i.e., a subset of S where the coefficient $\lfloor \sum_{h \in S \cap H'} u_h \rfloor$ in the SRI (8) increases. We define that a subset $M \subseteq S$ is a *minimal subset* for S and weights u if there exists an integer $m \geq 1$ with

$$\sum_{h \in M} u_h \geq m \qquad \text{and} \qquad \sum_{h \in M'} u_h < m \quad \forall M' \subsetneq M.$$

Let $\mathcal{M}(S,u)$ be the set of all minimal subsets of S and u. The following system of inequalities, one for each $M \in \mathcal{M}(S,u)$ is added to formulation (4) or (5):

$$\sum_{h \in M} z_h - s_{S,u} \leq |M| - \left\lfloor \sum_{h \in M} u_h \right\rfloor \qquad \forall M \in \mathcal{M}(S,u) \tag{11}$$

For the same example as above, i.e., $|S| = 3$ and $(u_{h_1}, u_{h_2}, u_{h_3}) = (1/2, 1/2, 1/2)$, the result is three inequalities $z_{h_1} + z_{h_2} - s_{S,u} \leq 1$, $z_{h_1} + z_{h_3} - s_{S,u} \leq 1$, and $z_{h_2} + z_{h_3} - s_{S,u} \leq 1$. For $|S| = 5$ and $(u_{h_1}, u_{h_2}, u_{h_3}, u_{h_4}, u_{h_5}) = (2/3, 2/3, 2/3, 1/3, 1/3)$ there are 13 inequalities, where the first is $z_{h_1} + z_{h_2} - s_{S,u} \leq 1$ and the last is $z_{h_2} + z_{h_3} + z_{h_4} + z_{h_5} - s_{S,u} \leq 4 - 2 = 2$. We refer to subproblem formulations supplemented with constraints of type (11) as *multiple SRI-enforcing formulations.*

The following proposition highlights that there is no "better" subproblem formulation comparing the two.

Proposition 2. *Single SRI-enforcing formulations do not dominate multiple SRI-enforcing formulations, nor vice versa.*

Proof. We consider $S = \{h_1, h_2, h_3\}$ and $(u_{h_1}, u_{h_1}, u_{h_1}) = (1/2, 1/2, 1/2)$ again to show that there is no dominance between the two possibilities.

On the one hand, consider the fractional point $(z_{h_1}, z_{h_2}, z_{h_3}, s_{S,u}) = (1, 1, 0, 1/2)$. This point is feasible for $z_{h_1} + z_{h_2} + z_{h_3} - 2s_{S,u} \leq 1$ but cut off by $z_{h_1} + z_{h_2} - s_{S,u} \leq 1$. Hence, the single SRI-enforcing formulation does not dominate the multiple SRI-enforcing formulation.

On the other hand, consider the fractional point $(z_{h_1}, z_{h_2}, z_{h_3}, s_{S,u}) = (2/3, 2/3, 2/3, 1/3)$. This point is cut off by $z_{h_1} + z_{h_2} + z_{h_3} - 2s_{S,u} \leq 1$ but fulfills all three inequalities $z_{h_1} + z_{h_2} - s_{S,u} \leq 1$, $z_{h_1} + z_{h_3} - s_{S,u} \leq 1$, and $z_{h_2} + z_{h_3} - s_{S,u} \leq 1$. Hence, the multiple SRI-enforcing formulation does not dominate the single SRI-enforcing formulation, which completes the proof. □

The consequence is that three computational setups should be tested: using the single SRI-enforcing formulation, the multiple SRI-enforcing formulation, or a combination of the two. Section 5.5 provides empirical evidence that on average the combination works best.

Since the number of clusters (=rows) is relatively small in the SoftCluCARP instances that we consider in the computational study (see Section 5.1), we use an exact enumeration procedure to detect the most violated SRIs with $|S| = 3$. For larger subsets with $|S| > 3$, we use a straightforward heuristic separation algorithm comparable to the one presented by Pecin *et al.* (2017). Also, the general strategy for selecting violated SRIs is adopted from the work of Pecin *et al.*. Only SRIs violated by a minimum violation value $\varepsilon_{SRI} = 0.1$ are considered. Moreover, in each round of separation, a maximum of 30 SRIs can be added (the most violated ones), but not more than three SRIs that refer to the same cluster.

Impact of SRIs on Primal Heuristics. Note that the additional terms for the dual prices $\sigma_{S,u}$ of the active SRIs (S, u) must also be considered in the primal heuristics of Section 3.1 to correctly compute the reduced cost (9). This is however straightforward because the coefficients $\lfloor \sum_{h \in S \cap H'} u_h \rfloor$ directly depend on the chosen subset H'. Add- and drop-steps that modify the subset H' can directly compute the resulting difference in the SRI-specific terms.

4.2. Branching

Let $(\bar{\lambda}_r)_{r\in\Omega}$ be a fractional solution of the master program (2). As in the benchmark problems the number of vehicles is always restricted to the minimum number needed (found by solving a bin-packing problem), the branching for the SoftCluCARP is based solely on Ryan-Foster branching for pairs of clusters. Formally, the values

$$B_{h,h'} = \sum_{\substack{r=(w,H')\in\Omega: \\ \{h,h'\}\subseteq H'}} \bar{\lambda}_r$$

are computed first for all pairs $h, h' \in H$ with $h \neq h'$. If several branching values $B_{h,h'}$ are fractional, one where the fractional value is closest to 0.5 is selected. Then, two branches are created.

The first one is the *separate branch* in which all routes $(w, H') \in \Omega$ with $\{h, h'\} \subseteq H'$ are fixed to zero in the RMP. Moreover, in the subproblems the additional constraint

$$z_h + z_{h'} \leq 1$$

must be added.

The second one is the *together branch* in which all routes (w, H') with $h \in H', h' \notin H'$ or $h \notin H', h' \in H'$ are fixed to zero. In addition, the two clusters E_h and $E_{h'}$ must be merged into one new cluster. For the sake of simplicity, in formulations (4) and (5) we implement this merge with the additional constraint

$$z_h = z_{h'}$$

but use the merged cluster in the metaheuristic. Ryan-Foster branching guarantees that branching finally produces integer solutions.

Globally, in the BPC algorithm, we explore the branch-and-bound search tree with a mixture of a best bound-first and a depth-first node-selection strategy: If a branch-and-bound node is bounded, a next node is chosen with the best-bound first rule, while otherwise the tree search is continued with depth-first search (ties are broken choosing the together branch first). The intention of this mixed strategy is to find integer solutions quickly while keeping the search trees small.

Impact of Branching on Primal Heuristics. Branching affects the primal heuristic in two ways: (i) during the *hash-table inspection* (Section 3.1.2), we only consider entries in the hash-table that fulfill all active branching decisions; (ii) for our *add-drop-based metaheuristic* (Section 3.1.3), we only need to modify the add operator for the case of separate constraints. If a separate constraint is active for clusters h and h' and we add cluster h to a route $r = (w, H')$ that serves cluster $h' \in H'$, then we have to remove h' from H' so that the new subset becomes $(H' \cup \{h\}) \setminus \{h'\}$.

5. Computational Results

We implemented the BPC algorithm in `C++` and compiled the code in release mode under MS Visual Studio 2015 (64-bit version). CPLEX 12.8.0 was used to re-optimize the RMP, to solve the pricing subproblems as well as the URPPs via B&C. The experiments were carried out on a standard PC with an Intel(R) Core(TM) i7-5930k CPU, clocked at 3.5 GHz, and 64 GB of RAM, by allowing a single thread for each run. The time limit for each run was set to one hour.

5.1. Instances

In all previous works on clustered arc routing or postman problems, the clusters have been defined such that they are the connected components of the graph induced by required edges (Franquesa, 2008; Aráoz *et al.*, 2009a; Aráoz *et al.*, 2013; Corberán *et al.*, 2011). In real-world applications, however, clusters may be small city districts so that their induced graphs are not necessarily vertex-disjoint. As no such benchmark instances for the SoftCluCARP are available, we generated new instances starting from the widely-used traditional CARP benchmarks `KSHS` (Kiuchi *et al.*, 1995), `GDB` (Golden *et al.*, 1983), `VAL` (Benavent *et al.*, 1992), `BMCV` (Beullens *et al.*, 2003), and `EGL` (Li and Eglese, 1996). The only necessary information to add is the clustering information for the required edges E_R.

We applied a hierarchical agglomerative approach (Ward Jr., 1963) that works as follows: Initially, each required edge $e \in E_R$ forms a separate cluster leading to the singleton set $E_e = \{e\}$, i.e., $H = E_R$. Then, iteratively, two clusters are selected and merged into one, following the idea that two clusters that are the "most similar" should be merged first. Therefore, a *similarity measure* (to be maximized) or *distance measure* (to be minimized) for pairs of clusters must be defined. For $h, h' \in H$ with $h \neq h'$, we use:

(i) *Vertices in intersection:* $|V_h \cap V_{h'}|$;
(ii) *Total demand:* $d(E_h) + d(E_{h'})$;
(iii) *Required edges in union:* $|E_h| + |E_{h'}|$;
(iv) *Minimum distance:* $\min_{(i,j) \in V_h \times V_{h'}} D_{ij}$,
where D_{ij} denotes the shortest-path distance in G between vertices i and j;
(v) *Average distance:* $\sum_{(i,j) \in V_h \times V_{h'}} D_{ij} / (|V_h| \cdot |V_{h'}|)$;

The first is a similarity measure and the latter four are distance measures. The purpose of the two measures (ii) and (iii) is to generate clusters that are equally sized. We combine these five measures using weighted sums. For the measures that are to be minimized the reciprocal number related to the measure is used.

In order to create feasible SoftCluCARP instances, the total demand of a newly built cluster must not exceed a given value M, where we use $M = {}^4\!/_5\, Q$. Hence, in each iteration, two clusters for E_h and $E_{h'}$ maximizing the weighted sum and not violating the total demand constraint are selected and merged into the new cluster $E_h \cup E_{h'}$. The iterative merging continues until either no more cluster can be merged or a wanted number $H^{\max}$ of clusters is obtained.

In order to create a diverse set of instances, four different sets of weights were used. The weights were chosen such that a reasonable balance between the measures was obtained. The priorities $({}^1\!/\underline{c}, 0, 0, 1, 2)$, $({}^{\underline{c}}\!/2, 0, 0, 1, 3)$, $({}^1\!/\underline{c}, 2\max_{e \in E_R} {}^{d_e}\!/\underline{c}, 0, 1, 2)$, and $({}^1\!/\underline{c}, 0, {}^7\!/\underline{c}, 1, 2)$ were used in the four sets, where $\underline{c} = \min_{e \in E_R} c_e$ is the minimum cost of a required edge. In the first two sets of weights, only the closeness of clusters is considered. In the remaining two sets, clusters of smaller size measured by total demand or the number of edges are favored compared to clusters that are larger. For each original CARP instance, several clustered versions were created, where the number of wanted clusters and the set of weights were chosen differently. The resulting benchmark comprises 8 `KSHS`, 54 `GDB`, 119 `VAL`, 348 `BMCV`, and 82 `EGL` instances available at https://logistik.bwl.uni-mainz.de/forschung/benchmarks/. The interested reader finds a characterization of each instance in the Appendix.

5.2. Parameter Study for B&C

We use the following general setup and acceleration strategies in the three B&C algorithms (to solve URPPs and pricing subproblems (4) and (5)). In order to keep the setup reproducible and simple, the parameters are chosen identically in the three B&C algorithms: First, we set the threshold for the minimum cut violation to $\varepsilon_{cut} = 0.01$.

Second, when solving pricing problems (4) and (5), we set the upper bound for the reduced-cost objective to zero, which cuts off feasible but not improving integer solutions.

Third, we allow heuristic (a.k.a. partial) pricing and let the B&C terminate with a negative reduced-cost integer solution when at least 100,000 simplex iterations have been performed. If such a feasible integer solution is found after 100,000 simplex iterations, B&C is terminated immediately (the value of 100,000 iterations has been found in pretests). Moreover, we exploit the solution pool of CPLEX and add all negative reduced-cost routes stored there. In particular, every non-optimal route $r = (w, H')$ of the solution pool is first checked using the hash-table with the key H' to find the cost-minimal walk $w(H')$. If no entry is found, we run the exact URPP algorithm to compute the walk with minimal cost $c_{w(H')}$.

Fourth, pretests have also revealed that the more time consuming level-2 separation for connectivity and cocircuit constraints is only effective at the beginning of the B&C. We tested multiple different criteria and found that a reasonable strategy is to switch off level-2 separation when 50 branch-and-bound nodes have been solved.

Fifth, the sequence of separation procedures is level-1 separation for connectivity constraints, level-1 separation for cocircuit constraints, level-2 separation for connectivity constraints, and level-2 separation for cocircuit constraints. If one of the four procedures finds at least one violated constraint, separation is immediately terminated and the LP is re-optimized.

Finally, for all other B&C strategies, like branching-variable selection, tree search strategy, use of primal LP-based heuristics etc., we rely on the default settings of the callable library of CPLEX.

In the following experiment, we analyze for both the undirected and the windy subproblems, whether level-1 and level-2 separation for connectivity and cocircuit constraints is effective for cutting off fractional solutions. Note that checking connectivity constraints (4e) and (5d) and cocircuit constraints (4d) is indispensable for integer solutions. For the comparison, we restricted the test to the solution of the linear relaxation of the master problem (2). Moreover, we have selected a subset of 113 SoftCluCARP instances for this parameter study in order to keep the computational effort lower. These 113 instances are the result of running a preliminary column-generation implementation and selecting those instances with a run time between 10 seconds and 1 minute for the linear relaxation. Some less time-consuming but also more time-consuming instances were additionally selected so that all five benchmarks (`KSHS`, `GDB`, `VAL`, `BMCV`, and `EGL`) contribute with at least some instances.

The results of experiments comparing nine different *cut strategies* are summarized in Table 2. These cut strategies include no separation on fractional solutions (S_{00}), separating either only connectivity constraints (S_{10} or S_{20})) or only cocircuit constraint (S_{01} or S_{02}), using the level-1 separation only (S_{11}), and the use of all available separation algorithms (S_{22}). The mixed strategies S_{12} and S_{21} use different levels for connectivity and cocircuit constraints. The table entries are average computation times (arithmetic mean *Avg.* T and geometric mean *Geo.* T in seconds) over the 113 instances, and how often the linear relaxation was solved within the time limit of $TL = 3600$ seconds (*#Solved*).

For the undirected formulation (4), the two cut strategies S_{21} and S_{22} outperform all others (they are Pareto-optimal regarding the average times and solved instances). For the windy formulation (5), the strategy S_{21} is Pareto-optimal. Thus, all subsequent computational experiments are performed with cut strategy S_{21}. The strategy S_{21} is also used when solving URPPs with B&C (see Section 3.1.1).

	Cut Strategies	S_{00}	S_{10}	S_{20}	S_{01}	S_{02}	S_{11}	S_{21}	S_{12}	S_{22}
Connectivity:	level-1 separation		×	×			×	×	×	×
	level-2 separation			×				×		×
Cocircuit:	level-1 separation				×	×	×	×	×	×
	level-2 separation					×			×	×
Undirected formulation (4)										
Avg. T		757.1	395.8	25.2	698.3	699.9	207.4	**14.6**	209.6	**14.6**
Geo. T		144.9	96.9	15.7	98.7	98.7	46.6	**11.6**	46.6	**11.6**
#Solved (of 113)		96	106	**113**	96	96	111	**113**	111	**113**
Windy formulation (5)										
Avg. T		726.5	21.5	14.2	732.2	729.9	22.2	**13.9**	22.1	14.4
Geo. T		96.4	14.4	11.8	102.0	99.4	14.7	**11.5**	14.6	12.0
#Solved (of 113)		95	**113**	**113**	95	95	**113**	**113**	**113**	**113**

Table 2: Comparison of separation strategies for the undirected and windy formulations tested on 113 selected SoftCluCARP instances.

5.3. Impact of Heuristic Pricing

In this second experiment, we analyze the performance of the heuristic pricing components, i.e., the hash-table inspection on the very first level and the use of the add-drop-based metaheuristic at the second level, before the exact pricing is done with the B&C algorithm (cut strategy S_{21} based on either formulation (4) or (5)). Regarding the hash-table inspection, we either skip it (*w/o*) or use it (*with*). Regarding the add-drop-based metaheuristic, we vary the number of iterations (*MaxIter*) of the main loop. The tested values for *MaxIter* are 0 (do not use the metaheuristic), 5, 20, and 50.

Pricing Strategies	P_0	P_5	P_{20}	P_{50}	P_5^H	P_{20}^H	P_{50}^H
		Use add-drop-based metaheuristic					
		w/o hash-table inspection			with hash-table inspection		
Iterations *MaxIter*	0	5	20	50	5	20	50
Avg. T	14.2	26.1	9.9	10.0	**9.2**	9.5	20.5
Geo. T	11.5	7.7	7.3	7.3	**7.0**	7.1	7.4
#Solved (of $2 \times 113 = 226$)	**226**	225	**226**	**226**	**226**	**226**	**226**

Table 3: Comparison of heuristic pricing strategies using 113 selected SoftCluCARP instances, solved with both formulations (4) and (5).

Table 3 shows aggregated linear-relaxation results over the 226 runs for each of the seven *pricing strategies* (two runs for each of the 113 instances using either the undirected or windy formulation in the B&C). The table entries have the same meaning as in Table 2.

Also in this experiment, there is a winner among the seven strategies: it is the strategy P_5^H using hash-table inspection (superscript H) in combination with only $MaxIter = 5$ iterations of the add-drop-based metaheuristic. Even if P_5^H is Pareto-optimal, also some other setups like P_{20}, P_{50}, and P_{20}^H that also use the metaheuristic are competitive. The results also show that arithmetic and geometric means provide different recommendations (the reader may compare P_0 with P_{50}^H). It should be noted that the run times of different instances vary significantly so that arithmetic means are dominated by the run times of difficult instances. Indeed, the rather bad *Avg.* T-value of 26.1 seconds for pricing strategy P_5 largely results from reaching the time limit in one of the 226 runs. Similarly, there is one very time-consuming instance for P_{50}^H leading to a comparably large *Avg.* T-value of 20.5 seconds.

For the remaining experiments, all column-generation iterations are done with pricing strategy P_5^H, i.e., with hast-table inspection and $\mathit{MaxIter} = 5$ iterations of the add-drop-based metaheuristic.

5.4. Comparison of B&C Algorithms using the Undirected and Windy Formulations

The next experiments were conducted with the goal to identify the better suited formulation for finally solving the pricing subproblems to optimality. We consider the two B&C algorithms described in Section 3.2 for the undirected formulation (4) and the windy formulation (5). Since branching and cutting on the master-program level may lead to very different trajectories of the overall BPC algorithms, we restrict the analysis to the solution of the linear relaxation of (2). However, we use the complete new benchmark set with 611 SoftCluCARP instances.

The outcome of the computational comparison is summarized in Table 4, grouped by the five classes of instances. The first three columns show the class with the number of instances (in brackets), the range of the number $|E_R|$ of required edges, and the range of the number $|H|$ of clusters. The two blocks with three columns each show for both formulations the number of solved linear relaxations as well as arithmetic and geometric means of the computation times (in seconds).

Benchmark set			Undirected formulation (4)			Windy formulation (5)		
				Time			Time	
	$\|E_R\|$	$\|H\|$	#Solved	*Avg. T*	*Geo. T*	#Solved	*Avg. T*	*Geo. T*
`KSHS` (8)	15	5–7	8	0.1	0.1	8	0.1	0.1
`GDB` (54)	11–55	4–24	54	0.3	0.2	54	0.3	0.2
`VAL` (119)	34–97	4–41	114	211.6	**5.9**	**118**	**122.8**	6.9
`BMCV` (348)	28–121	2–53	342	106.2	8.6	**348**	**55.4**	**8.2**
`EGL` (82)	51–190	12–84	29	2360.5	710.3	**50**	**1500.9**	**265.4**
Total (611)	11–190	2–84	547	418.5	9.9	**578**	**257.0**	**8.7**

Table 4: Comparison of B&C algorithms using either the undirected formulation (4) or the windy formulation (5) grouped by benchmark sets.

Overall, the column-generation algorithm using the windy formulation (5) outperforms the one using the undirected formulation (4). The `KSHS` and `GDB` instances require only very small computation times making a comparison redundant. The comparison on the classes `VAL` and `BMCV` reveals that the windy formulation allows the column-generation algorithm to solve all but one linear relaxation (instance `10A_clustered38` of the `VAL` benchmark), while the column-generation algorithm with the undirected formulation fails in 11 of the 467 cases. For the 82 `EGL` instances, the linear relaxation is also solved more often by the windy formulation (50 versus 29 instances). The only value in Table 4 that speaks for the undirected formulation is the geometric mean time of 5.9 seconds spent for the `VAL` benchmark. The advantage over the geometric mean time of 6.9 seconds for the windy formulation is however not striking.

These findings regrading the superiority of the windy formulation are also supported by the performance profiles depicted in Figure 2. The performance profiles are computed as follows: For any set $\mathcal{A}$ of algorithms applied to the same set of instances (here we have $\mathcal{A} = \{$column generation using (4), column generation using (5)$\}$), the function $\rho_A(\tau)$ of algorithm $A \in \mathcal{A}$ is the fraction of instances that algorithm A can solve within a factor τ of the fastest algorithm, where unsolved instances are taken into account with infinite run time. In particular, the value $\rho_A(1)$ is the percentage of instances on which A is a fastest algorithm and the value $1 - \rho_A(\infty)$ is the percentage of instances not solved by A. Note that τ in Figure 2 is displayed in logarithmic scale and the percent-axis starts at 40 % (cutting off the uninteresting part between 0 % and 40 %).

The two profiles show that the column-generation algorithm with the undirected formulation is the faster variant in 49.6 % of the cases, while the windy one is the fastest in 45.2 % of the cases (the remaining cases are unsolved instances). However, already for $\tau \geq 1.1$, i.e., accepting an up to 10 % slower algorithm, the

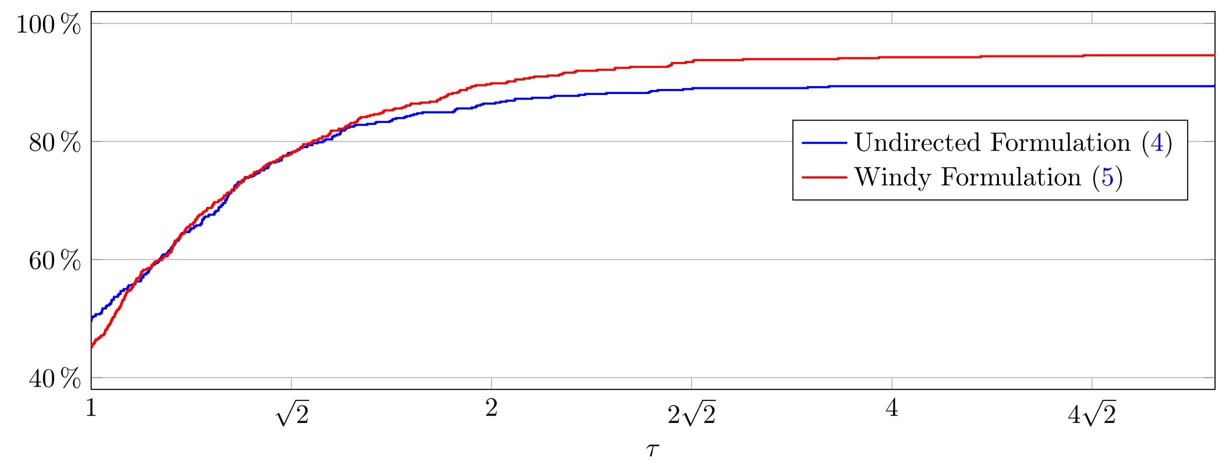

Figure 2: Performance profiles $\rho_A(\tau)$ for $A \in \mathcal{A} = \{\text{column generation using (4)}, \text{column generation using (5)}\}$ comparing the two resulting BPC algorithms using the undirected and windy formulations for the final pricing steps.

two curves overlap (until $\tau \approx 1.6$) and at the end the windy formulation enables solving significantly more instances.

In all following experiments, we use the windy formulation (5) in the final pricing steps.

5.5. Parameter Study for Subset-Row Inequalities

The purpose of the following experiments is to properly calibrate the SRI strategy. We use the complete benchmark of 611 SoftCluCARP instances again but now try to solve them to proven integer optimality.

We compare ten different separation strategies: no SRIs at all (denoted by SR_{-}), only SRIs for subsets S with $|S| = 3$ (SR_3), with $|S| \in \{3,4\}$ (SR_{34}), and with $|S| \in \{3,4,5\}$ (SR_{345}). For the three latter strategies, we further distinguish between implementing the SRIs via single SRI-enforcing formulations (indicated by the superscript "^{s}"), multiple SRI-enforcing formulations (superscript "^{m}"), and the combination of both (superscript "^{sm}").

Subset-row strategies	SR_{-}	SR_3^s	SR_3^m	SR_3^{sm}	SR_{34}^s	SR_{34}^m	SR_{34}^{sm}	SR_{345}^s	SR_{345}^m	SR_{345}^{sm}	Overall
		$\|S\| = 3$			$\|S\| \in \{3,4\}$			$\|S\| \in \{3,4,5\}$			
single SRI-enforce.		×		×	×		×	×		×	
multiple SRI-enforce.			×	×		×	×		×	×	
Avg. T	711.6	656.6	616.7	623.2	675.6	625.7	**613.7**	695.2	658.2	653.3	
Geo. T	24.0	20.3	19.5	**19.4**	20.5	19.6	**19.4**	21.2	20.3	20.2	
#Int	**565**	537	548	550	528	540	544	522	531	530	570
#Opt	519	525	535	**538**	519	532	537	518	526	527	547
exclusive	0	0	0	1	0	0	0	0	0	1	
exclusive per group	0			2			0			2	
best LB_{tree} (of 64 unsolved)	3	10	15	18	8	9	14	6	10	11	
exclusive per group	0			4			2			9	

Table 5: Comparison of subset-row separation strategies for all 611 instances.

Table 5 presents the aggregated results with arithmetic and geometric mean computation times. Moreover, the next two rows ("#Int" and "#Opt") provide the number of instances for which an integer solution and a proven optimal integer solution could be computed, respectively. The additional column ("Overall") shows the same numbers counting whether at least one of the ten SRI strategies was able to provide the respective result.

We can summarize that the results are not as clear cut as in the previous experiments. Overall, 547 of the 611 instances are solved to optimality and integer results are available for 570 instances. No SRI-separation strategy outperforms all others. As we use a mixed node-selection strategy for the branch-and-bound, it could be expected that SR_{-} provides by far the most integer solutions (565 of 611), because nodes are processed faster compared to the other SRI-separation strategies. Regarding the number of optimally solved instances, the strategy SR_3^{sm} is slightly better than SR_{34}^{sm} (538 versus 537 optima), while the other strategies perform worse. In all three blocks (for SR_3, SR_{34}, and SR_{345}), the combination of single-SRI and multiple-SRI enforcing constraints outperforms the solo strategies (the only exceptions are the *Avg. T*-value for SR_3^{sm} and the #Int-value for SR_{345}^{sm}).

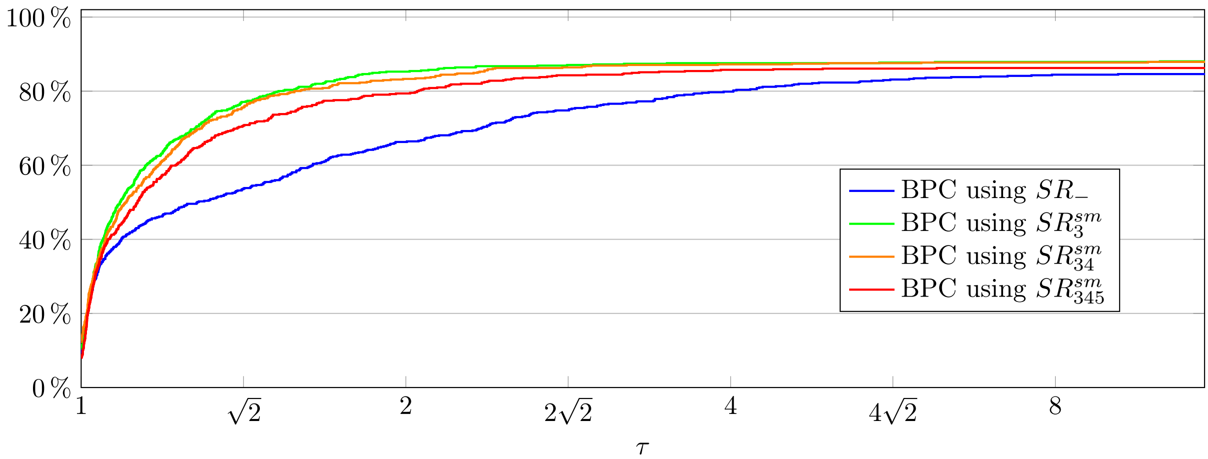

Figure 3: Performance profiles of four selected BPC algorithms using the SRI-separation strategies $SR_{-}, SR_3^{sm}, SR_{34}^{sm}$, and SR_{345}^{sm} comparing among $\mathcal{A} = \{$BPC using one of the ten different SR strategies$\}$.

The additional rows directly below "#Opt" show how often an optimal solution was determined by exactly one of ten strategies only ("exclusive"). The same information is also displayed per group of strategies ("exclusive per group"). Here, we find that SR_3 as well as SR_{345} exclusively prove two optima each. A similar information is provided in the last two rows where, for the 64 instances that remain open, the quality of the tree lower bound $\mathrm{LB}_{\mathrm{tree}}$ is compared. All strategies provide several tightest tree lower bounds. The group SR_{345} of the strategies exclusively contributes the most (9 compared to only 4 and 2 for the groups SR_3 and SR_{34}, respectively).

Regarding computation times in Table 5, both strategies SR_3^{sm} and SR_{34}^{sm} seem to be very good, but all geometric means are close to each other. We therefore compare the BPC algorithms also on the basis of performance profiles depicted in Figure 3. Note that the performance profiles are computed comparing all ten SRI-separation strategies. For the sake of clarity, however, we only show the three best strategies with combined SRI-enforcing constraints and the strategy SR_{-} in order to show the positive impact that the SRIs have on the BPC performance. One can clearly see that the BPC algorithms with SR_3^{sm} and SR_{34}^{sm} lead to very similar results, while the BPC algorithm with SR_{-} is inferior.

In summary, SR_3^{sm} and SR_{34}^{sm} are best regarding the overall number of optima as well as regarding computation times. However, strategy SR_{345}^{sm} is complementary and provides several best tree lower bounds for several unsolved instances.

5.6. Overall Integer Results

For the experiment with complete benchmark, we have chosen two BPC algorithms. Both algorithms share the separation strategy S_{21} (2-level separation for connectivity constraints and 1-level separation for cocircuit constraints), the pricing strategy P_5^H (five iterations of the add-drop-based metaheuristic including hash-table inspection), and use the windy formulation (5) for the final pricing steps. The two BPC algorithms only differ in their SRI strategy using either SR_3^{sm} or SR_{34}^{sm}.

Benchmark set			SR_3^{sm}				SR_{34}^{sm}			
					Time				Time	
	$\lvert E_R \rvert$	$\lvert H \rvert$	#Int	#Opt	*Avg.*	*Geo.*	#Int	#Opt	*Avg.*	*Geo.*
`KSHS` (8)	15	5–7	8	8	0.1	0.1	8	8	0.1	0.1
`GDB` (54)	11–55	4–24	54	54	0.5	0.3	54	54	**0.4**	0.3
`VAL` (119)	34–97	4–41	**106**	**102**	**715.0**	**19.7**	103	100	723.2	20.7
`BMCV`										
`C (84)`	32–121	7–53	79	76	492.5	27.4	79	**78**	**424.6**	**26.4**
`D (86)`	32–121	2–42	**86**	84	276.1	10.8	85	**85**	**269.6**	**10.7**
`E (84)`	28–107	7–41	78	75	650.2	27.6	78	**76**	**580.4**	**26.6**
`F (94)`	28–107	4–45	**91**	**91**	**365.7**	**21.2**	89	89	413.1	21.6
`EGL`										
`E (39)`	51–98	12–44	**39**	**39**	**351.3**	**73.9**	38	38	373.5	75.7
`S (43)`	75–190	13–84	9	9	**2973.6**	**2220.8**	**10**	9	2973.7	2222.4
Total (611)	11–190	2–84	**550**	**538**	623.2	19.4	544	537	**613.7**	19.4

Table 6: Overall integer results using the windy formulation (5) and subset-row strategies SR_3^{sm} and SR_{34}^{sm}, grouped by benchmark sets.

The results are summarized in Table 6, grouped by benchmark sets. For the large `BMCV` and `EGL` benchmarks, results for the subsets `C`, `D`, `E`, and `F` and the subsets `E` and `S` are provided also. Over the different benchmark sets, the two BPC algorithms with strategies SR_3^{sm} and SR_{34}^{sm} perform equally. There is no clear pattern observable, neither in the number of integer and optimal solutions nor in computation times.

The Appendix contains further detailed per instance results (Tables 7–16). For these results, we have selected the BPC algorithm with the SRI-separation strategy SR_3^{sm}.

5.7. Systematic Agglomeration of the Clusters

We briefly analyze now the impact of the hierarchical agglomerative clustering approach that has been used to create SoftCluCARP instances (see Section 5.1). Recall first that every SoftCluCARP instance is a restriction of the corresponding CARP. We denote by $\mathcal{I}_N$ the SoftCluCARP instance that has a predefined number N of clusters. Using the same clustering algorithm, instances $\mathcal{I}_N$ and $\mathcal{I}_{N+1}$ are restriction and relaxation of another, respectively. This statement holds only true if the fleet size m is not constrained. In the CARP and also in the previous experiments, the fleet size was always set to the minimum (resulting from solving the bin-packing problem). In this case, instance $\mathcal{I}_N$ is a restriction of $\mathcal{I}_{N+1}$ only if the fleet-size limit m is identical.

As an example, we consider the CARP instance `C12` from the `BMCV` benchmark. It has $|E_R| = 72$ required edges and its optimal solution value $z_{CARP} = 4{,}240$ provides a valid lower bound for the restricted fleet-size case. For the following experiment, we have generated 33 SoftCluCARP instances with between $N = 16$ and 48 clusters using the hierarchical agglomerative clustering approach. Each instance is then solved two times, once with the minimum fleet-size limit and once with unrestricted fleet. The results are displayed in Figure 4.

There are several things that stand out: On average, the larger number of clusters, the longer the computation times. Instances with more than 36 clusters become very difficult, probably because the average size of a cluster falls below two edges per cluster, so that the resulting problem is rather close to the original CARP. For these instances, a labeling-based solution approach may become more appropriate than the MIP-based solution approach used here.

Regarding routing costs, the curve for instances with unlimited fleet, i.e., $m = \infty$, is non-increasing. In contrast, for instances with limited fleet, i.e., $m = \min$, the cost curve is non-monotone. For N between 16 and 19, the minimum fleet size is 10 vehicles, while for N between 20 and 48 the minimum fleet size is 9 vehicles. This explains the jump discontinuity.

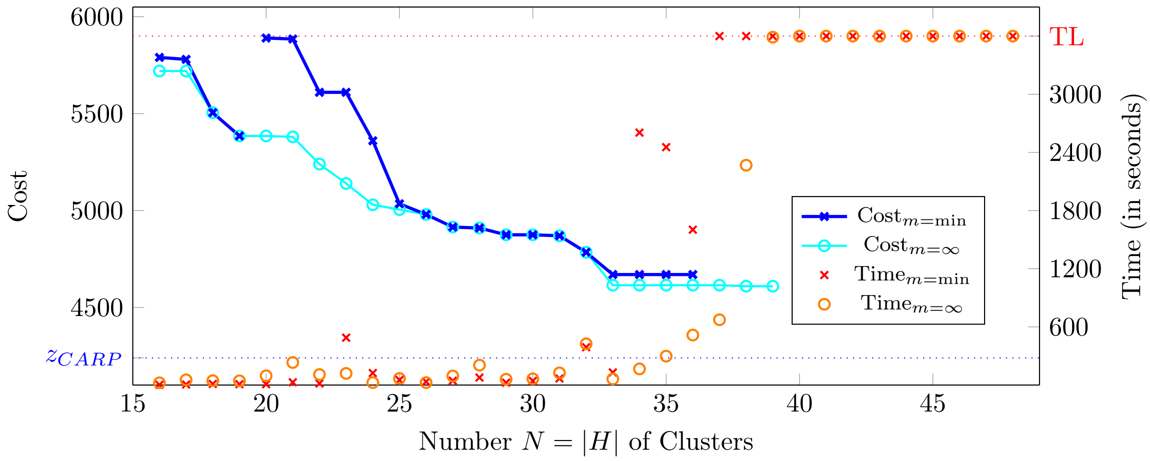

Figure 4: Impact of the hierarchical agglomerative clustering on costs and computation times, using either a minimum ($m =$ min) or an unrestricted fleet of vehicles ($m = \infty$).

6. Conclusions

In this paper, we have introduced the SoftCluCARP as a planning problem that sits in the middle between districting for arc routing (Butsch *et al.*, 2014) and the CARP-based route planning (Eiselt *et al.*, 1995b; Belenguer *et al.*, 2014). We suggest solving moderately-sized instances of the SoftCluCARP via branch-price-and-cut (BPC). For this purpose, we have developed a problem-tailored BPC algorithm with some innovative components. Routing subproblems are not solved as shortest-path problems with resource constraints via labeling algorithm but by using a MIP-based approach. Important insights from the computational analysis are the following: a windy formulation of the pricing subproblem is slightly better compared to an undirected formulation when used as the underlying MIP model for a branch-and-cut. A favorable separation strategy in the branch-and-cut algorithm applies a two-level separation algorithm to find violated connectivity constraints, but only a less careful one-level separation algorithm for finding violated cocircuit constraints. Results comparing subset-row separation strategies on the master-program level in the BPC are not clear cut, but show that, depending on the individual SoftCluCARP instance, strategies are complementary. For some hard instances, the use of subset-row inequalities referring to more than three rows can be beneficial. Future research may try to automatically identify a good subset-row separation strategy in the course of the column-generation process.

For future research, we think that the use of MIP-based approaches for clustered versions of the CARP is helpful to directly integrate the additional requirements that play a key role in districting: balancedness, connectivity, and compactness of the final districts covered by a vehicle. These requirements are very hard to incorporate into shortest-path problems with resource constraints and we doubt that an effective solution of such subproblems is possible with a labeling algorithm. Enforcing balancedness, connectivity, and compactness is somewhat simpler in a MIP-based approach.

Acknowledgement

This research was supported by the Deutsche Forschungsgemeinschaft (DFG) under grant IR 122/10-1. This support is gratefully acknowledged.

References

Adulyasak, Y., Cordeau, J.-F., and Jans, R. (2014). Formulations and branch-and-cut algorithms for multivehicle production and inventory routing problems. *INFORMS Journal on Computing*, **26**(1), 103–120.

Aráoz, J., Fernández, E., and Zoltan, C. (2006). Privatized rural postman problems. *Computers & Operations Research*, **33**(12), 3432–3449.

Aráoz, J., Fernández, E., and Franquesa, C. (2009a). The clustered prize-collecting arc routing problem. *Transportation Science*, **43**(3), 287–300.

Aráoz, J., Fernández, E., and Meza, O. (2009b). Solving the prize-collecting rural postman problem. *European Journal of Operational Research*, **196**(3), 886–896.

Aráoz, J., Fernández, E., and Franquesa, C. (2013). GRASP and path relinking for the clustered prize-collecting arc routing problem. *Journal of Heuristics*, **19**(2), 343–371.

Archetti, C. and Speranza, M. G. (2014). Chapter 12: Arc routing problems with profits. In *Arc Routing*, pages 281–299. Society for Industrial and Applied Mathematics, Philadelphia, PA.

Ávila, T., Corberán, Á., Plana, I., and Sanchis, J. M. (2016). A branch-and-cut algorithm for the profitable windy rural postman problem. *European Journal of Operational Research*, **249**(3), 1092–1101.

Barthélemy, T., Rossi, A., Sevaux, M., and Sörensen, K. (2010). Metaheuristic approach for the clustered VRP. In *EU/ME 2010 – 10th anniversary of the metaheuristic community*, Lorient, France.

Bartolini, E., Cordeau, J.-F., and Laporte, G. (2011). Improved lower bounds and exact algorithm for the capacitated arc routing problem. *Mathematical Programming*, **137**(1-2), 409–452.

Battarra, M., Erdoğan, G., and Vigo, D. (2014). Exact algorithms for the clustered vehicle routing problem. *Operations Research*, **62**(1), 58–71.

Belenguer, J. and Benavent, E. (1998). The capacitated arc routing problem: Valid inequalities and facets. *Computational Optimization and Applications*, **10**(2), 165–187.

Belenguer, J. M., Benavent, E., and Irnich, S. (2014). The capacitated arc routing problem: Exact algorithms. In *Arc Routing*, chapter 9, pages 183–221. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA.

Benavent, E., Campos, V., Corberán, A., and Mota, E. (1992). The capacitated Chinese postman problem: Lower bounds. *Networks*, **22**(7), 669–690.

Beullens, P., Muyldermans, L., Cattrysse, D., and Oudheusden, D. V. (2003). A guided local search heuristic for the capacitated arc routing problem. *European Journal of Operational Research*, **147**(3), 629–643.

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

Bode, C. and Irnich, S. (2014). The shortest-path problem with resource constraints with $(k, 2)$-loop elimination and its application to the capacitated arc-routing problem. *European Journal of Operational Research*, **238**(2), 415–426.

Bode, C. and Irnich, S. (2015). In-depth analysis of pricing problem relaxations for the capacitated arc-routing problem. *Transportation Science*, **49**(2), 369–383.

Butsch, A., Kalcsics, J., and Laporte, G. (2014). Districting for arc routing. *INFORMS Journal on Computing*, **26**(4), 809–824.

Corberán, Á. and Laporte, G., editors (2014). *Arc Routing*. Society for Industrial and Applied Mathematics, Philadelphia, PA.

Corberán, A., Fernández, E., Franquesa, C., and Sanchis, J. (2011). The windy clustered prize-collecting arc-routing problem. *Transportation Science*, **45**(3), 317–334.

Corberán, A. and Prins, C. (2010). Recent results on arc routing problems: An annotated bibliography. *Networks*, **56**(1), 50–69.

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

Costa, L., Contardo, C., and Desaulniers, G. (2019). Exact branch-price-and-cut algorithms for vehicle routing. *Transportation Science*, **53**(4), 946–985.

Defryn, C. and Sörensen, K. (2017). A fast two-level variable neighborhood search for the clustered vehicle routing problem. *Computers & Operations Research*, **83**, 78–94.

Dror, M., editor (2000). *Arc Routing: Theory, Solutions and Applications*. Springer US, New York, NY.

Eiselt, H. A., Gendreau, M., and Laporte, G. (1995a). Arc routing problems, Part I: The Chinese postman problem. *Operations Research*, **43**(2), 231–242.

Eiselt, H. A., Gendreau, M., and Laporte, G. (1995b). Arc routing problems, Part II: The rural postman problem. *Operations Research*, **43**(3), 399–414.

Expósito Izquierdo, C., Rossi, A., and Sevaux, M. (2013). Modeling and Solving the Clustered Capacitated Vehicle Routing Problem. In A. Fink and M.-J. Geiger, editors, *Proceedings of the 14th EU/ME workshop, EU/ME 2013*, pages 110–115, Hamburg, Germany.

Expósito-Izquierdo, C., Rossi, A., and Sevaux, M. (2016). A two-level solution approach to solve the clustered capacitated vehicle routing problem. *Computers & Industrial Engineering*, **91**, 274–289.

Feillet, D., Dejax, P., and Gendreau, M. (2005). Traveling salesman problems with profits. *Transportation Science*, **39**(2), 188–205.

Franquesa, C. (2008). *The clustered prize-collecting arc routing problem*. Ph.d. dissertation, Technical University of Catalonia, Barcelona, Spain.

Ghiani, G. and Laporte, G. (2014). The undirected rural postman problem. In *Arc Routing*, chapter 5, pages 85–99. Society for Industrial and Applied Mathematics, Philadelphia, PA.

Golden, B., Dearmon, J., and Baker, E. (1983). Computational experiments with algorithms for a class of routing problems. *Computers & Operations Research*, **10**(1), 47–59.

Golden, B. L. and Wong, R. T. (1981). Capacitated arc routing problems. *Networks*, **11**(3), 305–315.

Gomory, R. and Hu, T. (1961). Multi-terminal network flows. *Journal of the Society for Industrial and Applied Mathematics*, **9**(4), 551–570.

Hansen, P. and Mladenović, N. (2001). Variable neighborhood search: Principles and applications. *European Journal of Operational Research*, **130**(3), 449–467.

Hintsch, T. (2019). Large multiple neighborhood search for the soft-clustered vehicle-routing problem. Technical Report LM-2019-01, Chair of Logistics Management, Gutenberg School of Management and Economics, Johannes Gutenberg University Mainz, Mainz, Germany.

Hintsch, T. and Irnich, S. (2018). Large multiple neighborhood search for the clustered vehicle-routing problem. *European Journal of Operational Research*, **270**(1), 118–131.

Hintsch, T. and Irnich, S. (2019). Exact solution of the soft-clustered vehicle-routing problem. *European Journal of Operational Research*, **280**, 164–178.

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

Irnich, S., Toth, P., and Vigo, D. (2014). The family of vehicle routing problems. In D. Vigo and P. Toth, editors, *Vehicle Routing*, chapter 1, pages 1–33. 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.

Kiuchi, M., Shinano, Y., Hirabayashi, R., and Saruwatari, Y. (1995). An exact algorithm for the capacitated arc routing problem using parallel branch and bound method. In *Spring national conference of the operational research society of Japan*, pages 28–29.

Letchford, A. N., Reinelt, G., and Theis, D. O. (2004). A faster exact separation algorithm for blossom inequalities. In *Integer Programming and Combinatorial Optimization*, pages 196–205. Springer Berlin Heidelberg.

Letchford, A. N., Reinelt, G., and Theis, D. O. (2008). Odd minimum cut sets and b-matchings revisited. *SIAM Journal on Discrete Mathematics*, **22**(4), 1480–1487.

Li, L. Y. O. and Eglese, R. W. (1996). An interactive algorithm for vehicle routeing for winter-gritting. *Journal of the Operational Research Society*, **47**(2), 217–228.

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

Mourão, M. C. and Pinto, L. S. (2017). An updated annotated bibliography on arc routing problems. *Networks*, **70**(3), 144–194.

Pecin, D., Pessoa, A., Poggi, M., and Uchoa, E. (2017). Improved branch-cut-and-price for capacitated vehicle routing. *Mathematical Programming Computation*, **9**(1), 61–100.

Pop, P. C., Fuksz, L., Marc, A. H., and Sabo, C. (2018). A novel two-level optimization approach for clustered vehicle routing problem. *Computers & Industrial Engineering*, **115**(Supplement C), 304–318.

Sevaux, M. and Sörensen, K. (2008). Hamiltonian paths in large clustered routing problems. In *Proceedings of the EU/MEeting 2008 workshop on Metaheuristics for Logistics and Vehicle Routing, EU/ME'08*, pages 4:1–4:7, Troyes, France.

Vidal, T., Battarra, M., Subramanian, A., and Erdoğan, G. (2015). Hybrid metaheuristics for the clustered vehicle routing problem. *Computers & Operations Research*, **58**, 87–99.

Ward Jr., J. (1963). Hierarchical grouping to optimize an objective function. *Journal of the American Statistical Association*, **58**(301), 236–244.

This appendix is supposed to become online supplementary material.

Detailed Results

Tables 7–16 provide, on an instance basis, detailed results for the BPC algorithm with the following settings:

1. Separation strategy S_{21}, i.e., 2-level separation for connectivity constraints and 1-level separation for cocircuit constraints, see Section 5.2;
2. Pricing strategy P_5^H, i.e., with hast-table inspection and $MaxIter = 5$ iterations of the add-drop-based metaheuristic, see Section 5.3;
3. Windy formulation (5) in the final pricing steps, see Section 5.4;
4. Subset-row strategy SR_3^{sm}, i.e., using SRIs for subsets S with $|S| = 3$ and combined single and multiple SRI-enforcing formulations, see Section 5.5.

The columns of the tables have the following meaning:

Name: name of the instance
BKS: best known solution, bold if proven optimal (marked with * if solution or proof of optimality is derived by another than the default setting during computational studies)
Time: computation time in seconds ("TL" when prematurely terminated after 3600 seconds)
LB_{LP}: linear relaxation lower bound
LB_{SRI}: linear relaxation lower bound after adding SRIs
LB_{tree}: lower bound at termination
UB: upper bound at termination
% Gap: percentage optimality gap when reaching the time limit of 1 hour ($100 \cdot (\text{UB-LB}_{\text{tree}})/\text{LB}_{\text{tree}}$)
#SRIs: number of subset-row inequalities added
#B&B: number of solved branch-and-bound nodes

Instance							BPC Statistics							
								Bounds					Cuts/Tree	
Name	$\vert V\vert$	$\vert E\vert$	$\vert E_R\vert$	$\vert H\vert$	m	BKS	Time	$\mathrm{LB}_{\mathrm{LP}}$	$\mathrm{LB}_{\mathrm{SRI}}$	$\mathrm{LB}_{\mathrm{tree}}$	UB	% Gap	#SRIs	#B&B
kshs1_7	8	15	15	7	4	**16171**	0.1	16171	16171	16171	16171		0	1
kshs2_6	10	15	15	6	4	**12121**	0.1	12121	12121	12121	12121		0	1
kshs3_7	6	15	15	7	5	**11424**	0.1	11424	11424	11424	11424		0	1
kshs4_7	8	15	15	7	5	**13090**	0.1	13090	13090	13090	13090		0	1
kshs5_5	8	15	15	5	5	**14461**	0.1	14461	14461	14461	14461		0	1
kshs5_6	8	15	15	6	4	**12473**	0.1	12473	12473	12473	12473		0	1
kshs6_5	9	15	15	5	3	**14762**	0.1	14762	14762	14762	14762		0	1
kshs6_6	9	15	15	6	3	**11977**	0.1	11977	11977	11977	11977		0	1

Table 7: Detailed results for the `KSHS` instances.

							BPC Statistics							
Instance								Bounds					Cuts/Tree	
Name	$\|V\|$	$\|E\|$	$\|E_R\|$	$\|H\|$	m	BKS	Time	LB_{LP}	LB_{SRI}	LB_{tree}	UB	% Gap	#SRIs	#B&B
gdb1_8	12	22	22	8	6	**406**	0.1	406	406	406	406		0	1
gdb1_9	12	22	22	9	5	**364**	0.1	364	364	364	364		0	1
gdb2_10	12	26	26	10	6	**396**	0.1	396	396	396	396		0	1
gdb2_11	12	26	26	11	6	**396**	0.3	395	396	396	396		1	2
gdb3_8	12	22	22	8	7	**430**	0.2	430	430	430	430		0	1
gdb3_9	12	22	22	9	5	**352**	0.2	352	352	352	352		0	1
gdb4_8	11	19	19	8	5	**384**	0.2	382	384	384	384		1	2
gdb5_10	13	26	26	10	8	**530**	0.2	530	530	530	530		0	1
gdb5_11	13	26	26	11	7	**502**	0.2	502	502	502	502		0	1
gdb6_8	12	22	22	8	6	**393**	0.1	393	393	393	393		0	1
gdb6_9	12	22	22	9	6	**387**	0.1	387	387	387	387		0	1
gdb6_10	12	22	22	10	5	**337**	0.1	337	337	337	337		0	1
gdb7_8	12	22	22	8	6	**419**	0.1	419	419	419	419		0	1
gdb7_9	12	22	22	9	5	**376**	0.2	376	376	376	376		0	1
gdb8_19	27	46	46	19	10	**464**	0.6	464	464	464	464		0	1
gdb8_20	27	46	46	20	10	**415**	0.9	415	415	415	415		0	1
gdb9_18	27	51	51	18	12	**429**	0.6	429	429	429	429		0	1
gdb9_19	27	51	51	19	11	**374**	1.1	374	374	374	374		0	1
gdb9_22	27	51	51	22	10	**373**	1.0	373	373	373	373		0	1
gdb10_7	12	25	25	7	5	**353**	0.2	353	353	353	353		0	1
gdb10_9	12	25	25	9	4	**314**	0.2	314	314	314	314		0	1
gdb10_11	12	25	25	11	4	**315**	0.3	315	315	315	315		0	1
gdb11_8	22	45	45	8	6	**511**	0.2	511	511	511	511		0	1
gdb11_9	22	45	45	9	6	**506**	0.3	506	506	506	506		0	1
gdb11_12	22	45	45	12	5	**476**	1.4	476	476	476	476		0	1
gdb11_13	22	45	45	13	5	**473**	1.4	473	473	473	473		0	1
gdb12_11	13	23	23	11	8	**574**	0.3	574	574	574	574		0	1
gdb13_10	10	28	28	10	8	**619**	0.1	619	619	619	619		0	1
gdb13_11	10	28	28	11	8	**619**	0.2	616	619	619	619		2	2
gdb13_12	10	28	28	12	7	**589**	0.4	589	589	589	589		0	1
gdb14_8	7	21	21	8	5	**118**	0.1	118	118	118	118		0	1
gdb14_9	7	21	21	9	5	**120**	0.2	119	120	120	120		1	3
gdb15_6	7	21	21	6	4	**68**	0.1	68	68	68	68		0	1
gdb15_8	7	21	21	8	4	**66**	0.1	66	66	66	66		0	1
gdb16_9	8	28	28	9	6	**143**	0.2	142	143	143	143		1	2
gdb16_11	8	28	28	11	5	**145**	0.3	145	145	145	145		0	1
gdb16_12	8	28	28	12	5	**137**	0.3	137	137	137	137		0	1
gdb17_10	8	28	28	10	5	**95**	0.2	95	95	95	95		0	1
gdb17_12	8	28	28	12	5	**95**	0.5	95	95	95	95		4	4
gdb18_10	9	36	36	10	5	**176**	0.2	176	176	176	176		0	1
gdb18_12	9	36	36	12	5	**176**	0.4	176	176	176	176		0	1
gdb19_4	8	11	11	4	3	**75**	0.0	75	75	75	75		0	1
gdb20_6	11	22	22	6	5	**149**	0.1	149	149	149	149		0	1
gdb20_8	11	22	22	8	5	**142**	0.1	142	142	142	142		0	1
gdb20_9	11	22	22	9	5	**148**	0.5	147	147	148	148		1	4
gdb21_13	11	33	33	13	7	**185**	0.4	184	185	185	185		1	2
gdb21_14	11	33	33	14	6	**192**	0.5	192	192	192	192		0	1
gdb22_14	11	44	44	14	10	**228**	0.6	227	228	228	228		1	3
gdb22_17	11	44	44	17	9	**220**	1.9	219	220	220	220		4	5
gdb22_18	11	44	44	18	9	**216**	2.9	216	216	216	216		14	10
gdb23_17	11	55	55	17	12	**264**	0.3	264	264	264	264		0	1
gdb23_19	11	55	55	19	12	**260**	0.8	260	260	260	260		0	1
gdb23_20	11	55	55	20	11	**258**	0.5	258	258	258	258		0	1
gdb23_24	11	55	55	24	11	**252**	2.0	252	252	252	252		0	1

Table 8: Detailed results for the `GDB` instances.

Instance							BPC Statistics							
								Bounds					Cuts/Tree	
Name	$\|V\|$	$\|E\|$	$\|E_R\|$	$\|H\|$	m	BKS	Time	$\mathrm{LB}_{\mathrm{LP}}$	$\mathrm{LB}_{\mathrm{SRI}}$	$\mathrm{LB}_{\mathrm{tree}}$	UB	% Gap	#SRIs	#B&B
1A_5	24	39	39	5	2	**181**	0.1	181	181	181	181		0	1
1A_8	24	39	39	8	2	**186**	0.2	186	186	186	186		0	1
1A_12	24	39	39	12	2	**181**	2.5	175	181	181	181		15	4
1A_13	24	39	39	13	2	**181**	2.0	175	181	181	181		9	3
1B_7	24	39	39	7	4	**210**	0.1	210	210	210	210		0	1
1B_13	24	39	39	13	3	**221**	2.6	221	221	221	221		0	1
1B_16	24	39	39	16	3	**204**	7.6	204	204	204	204		0	1
1C_16	24	39	39	16	11	**298**	0.4	298	298	298	298		0	1
1C_17	24	39	39	17	10	**286**	0.4	286	286	286	286		0	1
2A_4	24	34	34	4	2	**248**	0.1	248	248	248	248		0	1
2A_6	24	34	34	6	2	**247**	0.2	247	247	247	247		0	1
2A_9	24	34	34	9	2	**243**	0.9	243	243	243	243		0	1
2A_11	24	34	34	11	2	**247**	0.7	247	247	247	247		0	1
2B_5	24	34	34	5	3	**322**	0.1	322	322	322	322		0	1
2B_10	24	34	34	10	3	**296**	0.8	296	296	296	296		0	1
2B_12	24	34	34	12	3	**296**	8.2	292	294	296	296		29	12
2C_15	24	34	34	15	10	**581**	0.3	581	581	581	581		0	1
3A_6	24	35	35	6	2	**88**	0.1	88	88	88	88		0	1
3A_11	24	35	35	11	2	**86**	2.7	85	86	86	86		8	2
3A_12	24	35	35	12	2	**86**	3.5	85	86	86	86		8	2
3B_6	24	35	35	6	3	**122**	0.2	122	122	122	122		0	1
3B_9	24	35	35	9	3	**100**	0.4	100	100	100	100		0	1
3B_11	24	35	35	11	3	**99**	3.6	96	98	99	99		24	8
3B_12	24	35	35	12	3	**99**	3.8	96	98	99	99		24	7
3C_11	24	35	35	11	11	**203**	0.3	203	203	203	203		0	1
3C_12	24	35	35	12	9	**184**	0.3	184	184	184	184		0	1
3C_13	24	35	35	13	8	**165**	0.2	165	165	165	165		0	1
4A_14	41	69	69	14	3	**441**	2.8	441	441	441	441		0	1
4A_21	41	69	69	21	3	**434**	41.3	431	434	434	434		15	2
4A_22	41	69	69	22	3	**436**	372.7	422	435	436	436		96	22
4A_28	41	69	69	28	3	**430**	3577.6	410	425	430	430		262	37
4B_19	41	69	69	19	4	**456**	61.9	452	456	456	456		11	2
4B_20	41	69	69	20	4	**457**	65.3	451	457	457	457		26	3
4B_24	41	69	69	24	4	**445**	735.7	436	444	445	445		92	11
4B_27	41	69	69	27	4	*455	TL	433	447	447	456	2.01	143	29
4C_14	41	69	69	14	5	**497**	2.8	496	497	497	497		3	2
4C_19	41	69	69	19	5	**491**	12.7	491	491	491	491		0	1
4C_24	41	69	69	24	5	**493**	30.1	493	493	493	493		0	1
4D_19	41	69	69	19	9	**659**	7.0	657	659	659	659		4	2
4D_20	41	69	69	20	9	**656**	9.4	653	656	656	656		4	3
4D_25	41	69	69	25	9	**665**	114.9	660	665	665	665		26	8
4D_26	41	69	69	26	9	**627**	49.4	625	627	627	627		13	3
5A_23	34	65	65	23	3	**453**	350.5	443	453	453	453		74	8
5A_25	34	65	65	25	3	**453**	3513.7	442	452	453	453		143	34
5B_9	34	65	65	9	4	**524**	0.9	524	524	524	524		0	1
5B_12	34	65	65	12	4	**518**	2.1	516	518	518	518		4	2
5B_13	34	65	65	13	4	**518**	8.4	516	518	518	518		10	3
5B_28	34	65	65	28	4	*469	TL	457	467	467	475	1.71	126	30
5C_16	34	65	65	16	5	**543**	7.0	540	543	543	543		2	2
5C_17	34	65	65	17	5	**536**	29.5	533	536	536	536		17	4
5C_21	34	65	65	21	5	**531**	134.5	523	528	531	531		48	16
5C_22	34	65	65	22	5	**531**	232.5	522	528	531	531		44	14
5D_16	34	65	65	16	10	**753**	10.4	745	748	753	753		3	7
5D_17	34	65	65	17	9	**729**	1.2	729	729	729	729		0	1
5D_18	34	65	65	18	9	**725**	3.4	725	725	725	725		0	1
5D_20	34	65	65	20	9	**709**	10.8	709	709	709	709		5	2

Table 9: Detailed results for the `VAL` instances (`1A`–`5D`).

							BPC Statistics							
Instance								Bounds					Cuts/Tree	
Name	$\|V\|$	$\|E\|$	$\|E_R\|$	$\|H\|$	m	BKS	Time	LB_{LP}	LB_{SRI}	LB_{tree}	UB	% Gap	#SRIs	#B&B
6A_5	31	50	50	5	4	**269**	0.2	269	269	269	269		0	1
6A_11	31	50	50	11	3	**241**	1.2	238	241	241	241		2	2
6A_18	31	50	50	18	3	**253**	89.9	248	252	253	253		42	12
6A_22	31	50	50	22	3	**245**	174.5	241	244	245	245		83	17
6B_19	31	50	50	19	4	**259**	5.2	259	259	259	259		13	2
6B_20	31	50	50	20	4	**253**	8.4	253	253	253	253		6	2
6B_21	31	50	50	21	4	**253**	12.0	253	253	253	253		17	2
6C_18	31	50	50	18	11	**397**	0.6	397	397	397	397		0	1
6C_20	31	50	50	20	10	**385**	2.4	385	385	385	385		1	2
6C_21	31	50	50	21	10	**384**	31.0	379	380	384	384		28	46
6C_22	31	50	50	22	10	**384**	31.2	379	380	384	384		20	43
7A_10	40	66	66	10	3	**347**	0.9	347	347	347	347		0	1
7A_12	40	66	66	12	3	**347**	3.7	341	347	347	347		8	2
7A_21	40	66	66	21	3	**337**	137.7	327	337	337	337		61	9
7A_29	40	66	66	29	3	*337	TL	321	336	336	—	n.a.	164	20
7B_7	40	66	66	7	5	**352**	0.3	352	352	352	352		0	1
7B_8	40	66	66	8	4	**332**	0.3	332	332	332	332		0	1
7B_14	40	66	66	14	4	**332**	1.4	332	332	332	332		0	1
7B_28	40	66	66	28	4	**319**	53.5	319	319	319	319		12	2
7C_16	40	66	66	16	10	**456**	0.6	456	456	456	456		0	1
7C_22	40	66	66	22	9	**431**	7.2	428	431	431	431		6	3
7C_25	40	66	66	25	9	**422**	4.8	422	422	422	422		0	1
7C_28	40	66	66	28	9	**401**	14.3	396	401	401	401		16	4
8A_8	30	63	63	8	3	**444**	0.6	444	444	444	444		0	1
8A_10	30	63	63	10	3	**448**	1.4	448	448	448	448		3	2
8A_19	30	63	63	19	3	**429**	425.0	424	428	429	429		75	19
8A_21	30	63	63	21	3	**425**	576.1	417	424	425	425		104	13
8B_6	30	63	63	6	5	**508**	0.5	508	508	508	508		0	1
8B_24	30	63	63	24	4	**427**	184.7	424	427	427	427		42	6
8C_16	30	63	63	16	11	**678**	0.9	678	678	678	678		0	1
8C_20	30	63	63	20	9	**642**	3.9	642	642	642	642		0	1
8C_22	30	63	63	22	9	**619**	7.0	619	619	619	619		0	1
8C_28	30	63	63	28	9	**589**	34.2	587	588	589	589		14	6
9A_26	50	92	92	26	3	**346**	1225.6	339	346	346	346		74	5
9A_28	50	92	92	28	3	*355	TL	345	353	353	—	n.a.	99	9
9A_38	50	92	92	38	3	—	TL	328	339	339	—	n.a.	67	4
9A_39	50	92	92	39	3	—	TL	321	329	329	—	n.a.	29	1
9B_11	50	92	92	11	4	**368**	1.1	368	368	368	368		0	1
9B_28	50	92	92	28	4	369	TL	359	368	368	369	0.27	88	13
9B_31	50	92	92	31	4	**353**	2593.7	347	353	353	353		73	7
9B_36	50	92	92	36	4	*366	TL	340	346	346	—	n.a.	74	3
9C_17	50	92	92	17	5	**382**	5.3	382	382	382	382		0	1
9C_24	50	92	92	24	5	**379**	74.0	377	379	379	379		16	4
9C_28	50	92	92	28	5	**368**	2064.7	363	368	368	368		57	8
9C_37	50	92	92	37	5	*358	TL	351	357	357	—	n.a.	83	4
9D_25	50	92	92	25	10	**471**	10.3	471	471	471	471		0	1
9D_32	50	92	92	32	10	**455**	108.0	451	455	455	455		21	3
9D_39	50	92	92	39	10	**437**	1818.3	432	436	437	437		72	16
9D_41	50	92	92	41	10	***444**	TL	434	442	443	444	0.23	88	23
10A_22	50	97	97	22	3	**451**	206.4	449	451	451	451		15	2
10A_25	50	97	97	25	3	**452**	3390.0	445	452	452	452		60	5
10A_32	50	97	97	32	3	—	TL	433	440	440	—	n.a.	55	2
10A_38	50	97	97	38	3	—	TL	386	386	386	—	n.a.	0	0
10B_36	50	97	97	36	4	*476	TL	446	451	451	—	n.a.	59	2
10B_37	50	97	97	37	4	—	TL	446	450	450	—	n.a.	56	2
10B_41	50	97	97	41	4	—	TL	441	441	441	—	n.a.	29	1
10C_11	50	97	97	11	5	**512**	3.3	512	512	512	512		0	1
10C_14	50	97	97	14	5	**523**	25.0	522	523	523	523		2	4
10C_31	50	97	97	31	5	*490	TL	480	487	487	—	n.a.	54	3
10C_32	50	97	97	32	5	*485	TL	462	477	477	—	n.a.	76	4
10D_24	50	97	97	24	10	**641**	37.1	641	641	641	641		0	1
10D_28	50	97	97	28	10	**591**	216.7	586	591	591	591		16	9
10D_36	50	97	97	36	10	**597**	855.5	594	597	597	597		28	3

Table 10: Detailed results for the `VAL` instances (`6A`–`10D`).

Instance							BPC Statistics							
								Bounds					Cuts/Tree	
Name	$\lvert V\rvert$	$\lvert E\rvert$	$\lvert E_R\rvert$	$\lvert H\rvert$	m	BKS	Time	LB_{LP}	LB_{SRI}	LB_{tree}	UB	% Gap	#SRIs	#B&B
C01_18	69	98	79	18	9	**5305**	4.5	5305	5305	5305	5305		0	1
C01_24	69	98	79	24	9	***4840**	TL	4771	4834	4834	—	n.a.	8	4
C01_29	69	98	79	29	9	**4685**	405.4	4563	4657	4685	4685		43	18
C02_18	48	66	53	18	7	**3685**	4.6	3685	3685	3685	3685		0	1
C02_20	48	66	53	20	7	**3375**	5.0	3375	3375	3375	3375		0	1
C02_22	48	66	53	22	7	**3375**	6.3	3375	3375	3375	3375		0	1
C03_15	46	64	51	15	6	**3030**	5.5	3007	3030	3030	3030		5	2
C03_20	46	64	51	20	6	**2735**	7.0	2702	2735	2735	2735		16	3
C03_21	46	64	51	21	6	**2675**	14.4	2597	2675	2675	2675		24	4
C04_16	60	84	72	16	8	**4745**	2.8	4745	4745	4745	4745		0	1
C04_17	60	84	72	17	8	**3920**	1.2	3920	3920	3920	3920		0	1
C04_26	60	84	72	26	8	**4195**	19.8	4185	4195	4195	4195		5	2
C05_20	56	79	65	20	11	**7105**	3.2	7105	7105	7105	7105		0	1
C05_22	56	79	65	22	10	**6540**	5.4	6540	6540	6540	6540		0	1
C05_23	56	79	65	23	10	**6290**	3.3	6290	6290	6290	6290		0	1
C05_25	56	79	65	25	10	**6830**	65.2	6770	6830	6830	6830		10	5
C06_12	38	55	51	12	6	**3210**	0.8	3210	3210	3210	3210		0	1
C06_17	38	55	51	17	6	**3170**	59.5	3085	3130	3170	3170		36	44
C06_18	38	55	51	18	6	**3000**	4.4	2985	3000	3000	3000		10	3
C06_20	38	55	51	20	6	**2980**	8.3	2937	2980	2980	2980		21	3
C07_16	54	70	52	16	9	**4560**	0.9	4560	4560	4560	4560		0	1
C07_17	54	70	52	17	9	**4560**	0.8	4560	4560	4560	4560		0	1
C07_18	54	70	52	18	8	**5725**	16.2	5597	5705	5725	5725		11	9
C08_15	66	88	63	15	10	**4915**	1.0	4915	4915	4915	4915		0	1
C08_17	66	88	63	17	10	**5005**	2.7	5005	5005	5005	5005		0	1
C08_22	66	88	63	22	9	**5015**	131.4	4943	4972	5015	5015		18	25
C08_24	66	88	63	24	8	**4960**	53.7	4929	4960	4960	4960		10	2
C09_22	76	117	97	22	15	**6560**	12.4	6560	6560	6560	6560		0	1
C09_37	76	117	97	37	12	**6270**	582.4	6202	6270	6270	6270		19	3
C09_39	76	117	97	39	12	***5990**	TL	5908	5943	5972	6005	0.55	75	28
C10_17	60	82	55	17	9	**5445**	1.9	5445	5445	5445	5445		0	1
C10_22	60	82	55	22	9	**5055**	9.4	4981	5055	5055	5055		10	2
C11_23	83	118	94	23	10	**5670**	37.3	5668	5670	5670	5670		3	2
C11_24	83	118	94	24	10	**5710**	34.7	5710	5710	5710	5710		0	1
C11_30	83	118	94	30	10	**5225**	241.9	5202	5225	5225	5225		17	4
C11_35	83	118	94	35	10	**5240**	2824.0	5160	5218	5240	5240		69	43
C12_22	62	88	72	22	9	**5695**	45.4	5687	5695	5695	5695		5	2
C12_28	62	88	72	28	9	**4975**	59.8	4975	4975	4975	4975		0	1
C12_31	62	88	72	31	9	*5150	TL	5015	5071	5105	5175	1.37	129	86
C13_18	40	60	52	18	7	**3520**	6.2	3515	3520	3520	3520		3	2
C13_21	40	60	52	21	7	**3290**	6.9	3250	3275	3290	3290		5	5
C13_22	40	60	52	22	7	**3285**	14.9	3248	3273	3285	3285		8	6
C14_15	58	79	57	15	10	**5195**	2.1	5195	5195	5195	5195		0	1
C14_18	58	79	57	18	9	**5100**	2.7	5068	5100	5100	5100		1	2
C14_19	58	79	57	19	8	**5495**	3.2	5495	5495	5495	5495		0	1
C14_21	58	79	57	21	8	**4565**	7.0	4565	4565	4565	4565		0	1
C15_24	97	140	107	24	11	**6185**	87.1	6143	6185	6185	6185		2	2
C15_35	97	140	107	35	11	**5625**	497.1	5432	5625	5625	5625		28	4
C15_43	97	140	107	43	11	*5475	TL	5399	5458	5458	—	n.a.	37	7
C15_45	97	140	107	45	11	5410	TL	5337	5383	5408	5410	0.04	69	13
C16_7	32	42	32	7	3	**1935**	0.3	1935	1935	1935	1935		0	1
C16_13	32	42	32	13	3	**1520**	2.6	1520	1520	1520	1520		0	1
C17_15	43	56	42	15	7	**4135**	3.8	4064	4124	4135	4135		3	4
C17_16	43	56	42	16	7	**4140**	2.4	4140	4140	4140	4140		0	1
C18_31	93	133	121	31	11	**7130**	1156.1	7033	7055	7130	7130		16	9
C18_36	93	133	121	36	11	***6480**	TL	6284	6431	6431	—	n.a.	38	7
C18_39	93	133	121	39	11	*6450	TL	6278	6395	6395	—	n.a.	53	10
C18_53	93	133	121	53	11	*6465	TL	5876	5996	5996	—	n.a.	92	6
C19_24	62	84	61	24	6	**3470**	237.2	3368	3456	3470	3470		60	13
C19_25	62	84	61	25	6	**3400**	325.0	3280	3385	3400	3400		72	13
C19_26	62	84	61	26	6	**3340**	321.9	3235	3339	3340	3340		70	11
C19_27	62	84	61	27	6	**3340**	379.8	3235	3335	3340	3340		64	11
C20_11	45	64	53	11	5	**2660**	1.1	2660	2660	2660	2660		0	1
C20_12	45	64	53	12	5	**2600**	0.9	2600	2600	2600	2600		0	1
C20_21	45	64	53	21	5	**2415**	21.1	2355	2415	2415	2415		24	3
C21_23	60	84	76	23	8	**4535**	26.7	4528	4535	4535	4535		3	2
C21_27	60	84	76	27	8	**4270**	35.8	4236	4255	4270	4270		9	5
C21_30	60	84	76	30	8	**4260**	251.2	4215	4239	4260	4260		48	25
C21_33	60	84	76	33	8	**4225**	904.1	4145	4190	4225	4225		92	45
C22_8	56	76	43	8	4	**2935**	0.8	2935	2935	2935	2935		0	1
C22_10	56	76	43	10	5	**2945**	22.5	2828	2850	2945	2945		3	32
C22_16	56	76	43	16	4	**2665**	17.7	2648	2665	2665	2665		9	2
C22_17	56	76	43	17	4	**2425**	9.6	2425	2425	2425	2425		10	2
C23_27	78	109	92	27	8	**5030**	669.9	4899	5009	5030	5030		34	8
C23_31	78	109	92	31	8	**5190**	2118.8	5123	5188	5190	5190		29	8
C23_38	78	109	92	38	8	**4465**	443.6	4415	4465	4465	4465		15	2
C24_14	77	115	84	14	8	**4370**	2.9	4370	4370	4370	4370		0	1
C24_18	77	115	84	18	7	**4750**	22.6	4750	4750	4750	4750		0	1
C24_22	77	115	84	22	7	**4435**	170.1	4373	4429	4435	4435		23	8
C24_31	77	115	84	31	7	**3695**	100.4	3695	3695	3695	3695		0	1
C25_11	37	50	38	11	6	**2945**	1.0	2933	2945	2945	2945		1	2
C25_13	37	50	38	13	5	**2710**	1.1	2710	2710	2710	2710		0	1
C25_15	37	50	38	15	5	**2805**	4.2	2738	2805	2805	2805		8	2
C25_16	37	50	38	16	5	**2640**	4.2	2600	2640	2640	2640		8	3

Table 11: Detailed results for the `BMCV` instances, subset `C`.

Instance							BPC Statistics							
								Bounds					Cuts/Tree	
Name	$\|V\|$	$\|E\|$	$\|E_R\|$	$\|H\|$	m	BKS	Time	LB_{LP}	LB_{SRI}	LB_{tree}	UB	% Gap	#SRIs	#B&B
D01_14	69	98	79	14	5	**4045**	1.9	4045	4045	4045	4045		0	1
D01_17	69	98	79	17	5	**3985**	12.5	3965	3985	3985	3985		5	3
D01_26	69	98	79	26	5	**3740**	183.5	3684	3740	3740	3740		42	6
D01_27	69	98	79	27	5	**3490**	104.7	3463	3490	3490	3490		9	2
D02_6	48	66	53	6	4	**2960**	0.3	2960	2960	2960	2960		0	1
D02_12	48	66	53	12	4	**2885**	2.8	2885	2885	2885	2885		0	1
D02_16	48	66	53	16	4	**2745**	3.0	2745	2745	2745	2745		0	1
D02_23	48	66	53	23	4	**2645**	40.2	2620	2633	2645	2645		12	6
D03_8	46	64	51	8	3	**2540**	0.7	2540	2540	2540	2540		0	1
D03_12	46	64	51	12	3	**2370**	5.9	2368	2370	2370	2370		4	2
D03_16	46	64	51	16	3	**2500**	20.4	2462	2500	2500	2500		17	3
D04_7	60	84	72	7	5	**3315**	0.8	3315	3315	3315	3315		0	1
D04_12	60	84	72	12	4	**3375**	12.9	3305	3353	3375	3375		13	6
D04_14	60	84	72	14	4	**3375**	36.7	3301	3323	3375	3375		12	8
D05_11	56	79	65	11	6	**4715**	0.9	4715	4715	4715	4715		0	1
D05_16	56	79	65	16	5	**4605**	8.7	4605	4605	4605	4605		0	1
D05_19	56	79	65	19	5	**4605**	182.8	4412	4589	4605	4605		28	9
D05_22	56	79	65	22	5	**5165**	225.0	5085	5150	5165	5165		38	9
D06_5	38	55	51	5	4	**2570**	0.3	2570	2570	2570	2570		0	1
D06_8	38	55	51	8	3	**2450**	0.6	2450	2450	2450	2450		0	1
D07_7	54	70	52	7	5	**4495**	0.6	4495	4495	4495	4495		0	1
D07_10	54	70	52	10	4	**4075**	0.8	4075	4075	4075	4075		0	1
D07_11	54	70	52	11	4	**3815**	1.9	3815	3815	3815	3815		0	1
D07_22	54	70	52	22	4	**3575**	61.5	3500	3575	3575	3575		26	3
D08_21	66	88	63	21	4	**3615**	52.3	3610	3615	3615	3615		7	2
D08_24	66	88	63	24	4	**3615**	52.9	3593	3615	3615	3615		13	2
D08_25	66	88	63	25	4	**3575**	735.7	3430	3575	3575	3575		53	5
D08_26	66	88	63	26	4	**3615**	570.2	3536	3615	3615	3615		62	7
D09_11	76	117	97	11	7	**5095**	5.2	5095	5095	5095	5095		0	1
D09_14	76	117	97	14	6	**5090**	10.5	5090	5090	5090	5090		0	1
D09_37	76	117	97	37	6	**4275**	278.1	4268	4275	4275	4275		9	2
D09_42	76	117	97	42	6	4270	TL	4207	4263	4266	4270	0.09	93	12
D10_15	60	82	55	15	5	**3650**	0.9	3650	3650	3650	3650		0	1
D10_16	60	82	55	16	5	**3815**	5.7	3810	3815	3815	3815		1	2
D10_17	60	82	55	17	5	**3550**	2.4	3550	3550	3550	3550		0	1
D11_10	83	118	94	10	6	**4775**	2.6	4775	4775	4775	4775		0	1
D11_34	83	118	94	34	5	**4075**	386.5	4064	4075	4075	4075		12	2
D11_35	83	118	94	35	5	**3935**	793.2	3863	3935	3935	3935		41	4
D11_41	83	118	94	41	5	**3900**	2599.2	3859	3900	3900	3900		60	6
D12_9	62	88	72	9	5	**4345**	2.1	4345	4345	4345	4345		0	1
D12_14	62	88	72	14	5	**4100**	5.3	4100	4100	4100	4100		0	1
D12_18	62	88	72	18	5	**3660**	18.9	3655	3660	3660	3660		2	2
D12_32	62	88	72	32	5	**3740**	364.0	3605	3740	3740	3740		98	7
D13_6	40	60	52	6	4	**2785**	0.3	2785	2785	2785	2785		0	1
D13_11	40	60	52	11	4	**2710**	0.9	2710	2710	2710	2710		0	1
D13_15	40	60	52	15	4	**2755**	2.3	2755	2755	2755	2755		0	1
D14_8	58	79	57	8	5	**3875**	0.4	3875	3875	3875	3875		0	1
D14_12	58	79	57	12	4	**4480**	1.6	4480	4480	4480	4480		0	1
D14_13	58	79	57	13	4	**4080**	5.3	4025	4080	4080	4080		5	2
D14_24	58	79	57	24	4	**3665**	1051.8	3599	3641	3665	3665		109	25
D15_26	97	140	107	26	6	**4395**	151.5	4345	4395	4395	4395		6	2
D15_39	97	140	107	39	6	**4270**	1120.4	4221	4270	4270	4270		35	4
D15_40	97	140	107	40	6	**4850**	3515.4	4662	4850	4850	4850		72	7
D16_2	32	42	32	2	2	**1600**	0.1	1600	1600	1600	1600		0	1
D16_5	32	42	32	5	2	**1520**	0.2	1465	1520	1520	1520		1	2
D16_9	32	42	32	9	2	**1470**	1.7	1358	1470	1470	1470		12	3
D17_10	43	56	42	10	4	**2965**	1.5	2942	2965	2965	2965		4	2
D17_17	43	56	42	17	4	**2750**	6.6	2627	2750	2750	2750		10	2
D18_13	93	133	121	13	6	**5525**	13.6	5525	5525	5525	5525		0	1
D18_23	93	133	121	23	6	**4770**	98.3	4757	4770	4770	4770		4	2
D18_34	93	133	121	34	6	***4625**	TL	4563	4601	4623	4625	0.04	63	19
D19_11	62	84	61	11	3	**2920**	2.7	2920	2920	2920	2920		0	1
D19_12	62	84	61	12	3	**2580**	1.8	2580	2580	2580	2580		0	1
D19_17	62	84	61	17	3	**2580**	4.9	2580	2580	2580	2580		0	1
D20_4	45	64	53	4	3	**2035**	0.2	2035	2035	2035	2035		0	1
D20_6	45	64	53	6	3	**1935**	0.2	1935	1935	1935	1935		0	1
D20_10	45	64	53	10	3	**2035**	0.7	2035	2035	2035	2035		0	1
D20_18	45	64	53	18	3	**1960**	38.3	1905	1960	1960	1960		25	4
D21_10	60	84	76	10	4	**3580**	2.1	3575	3580	3580	3580		1	2
D21_12	60	84	76	12	4	**3505**	4.8	3493	3505	3505	3505		4	2
D21_13	60	84	76	13	4	**3450**	5.7	3425	3450	3450	3450		3	2
D21_28	60	84	76	28	4	**3145**	2734.3	3055	3116	3145	3145		166	72
D22_4	56	76	43	4	3	**2285**	0.4	2285	2285	2285	2285		0	1
D22_9	56	76	43	9	2	**2115**	2.0	2115	2115	2115	2115		0	1
D22_15	56	76	43	15	2	**1915**	6.3	1915	1915	1915	1915		0	1
D23_7	78	109	92	7	5	**4400**	2.1	4400	4400	4400	4400		0	1
D23_19	78	109	92	19	4	**3810**	107.6	3810	3810	3810	3810		0	1
D23_20	78	109	92	20	4	**3635**	31.2	3635	3635	3635	3635		0	1
D23_31	78	109	92	31	4	**3285**	349.4	3269	3285	3285	3285		18	2
D24_12	77	115	84	12	4	**3480**	5.9	3480	3480	3480	3480		0	1
D24_14	77	115	84	14	4	**3235**	11.0	3235	3235	3235	3235		0	1
D24_24	77	115	84	24	4	**3265**	253.0	3160	3265	3265	3265		33	5
D24_32	77	115	84	32	4	**2885**	197.5	2860	2885	2885	2885		9	3
D25_4	37	50	38	4	3	**2280**	0.2	2280	2280	2280	2280		0	1
D25_5	37	50	38	5	3	**2155**	0.3	2155	2155	2155	2155		0	1
D25_16	37	50	38	16	3	**1915**	14.2	1860	1910	1915	1915		20	8

Table 12: Detailed results for the `BMCV` instances, subset `D`.

Instance							BPC Statistics							
								Bounds					Cuts/Tree	
Name	$\lvert V\rvert$	$\lvert E\rvert$	$\lvert E_R\rvert$	$\lvert H\rvert$	m	BKS	Time	LB_{LP}	LB_{SRI}	LB_{tree}	UB	% Gap	#SRIs	#B&B
E01_24	73	105	85	24	11	**6165**	33.3	6165	6165	6165	6165		0	1
E01_26	73	105	85	26	11	**5775**	19.9	5775	5775	5775	5775		0	1
E01_37	73	105	85	37	10	**5580**	1421.6	5486	5575	5580	5580		33	6
E02_17	58	81	58	17	9	**4730**	3.3	4730	4730	4730	4730		0	1
E02_20	58	81	58	20	8	**5305**	7.9	5305	5305	5305	5305		0	1
E02_25	58	81	58	25	8	**4715**	60.9	4715	4715	4715	4715		0	1
E02_26	58	81	58	26	8	**4635**	396.4	4541	4599	4635	4635		26	17
E03_8	46	61	47	8	6	**2475**	0.3	2475	2475	2475	2475		0	1
E03_18	46	61	47	18	5	**2110**	3.9	2083	2110	2110	2110		13	3
E04_18	70	99	77	18	10	**5225**	2.0	5225	5225	5225	5225		0	1
E04_25	70	99	77	25	9	**4930**	113.1	4890	4913	4930	4930		14	10
E05_15	68	94	61	15	10	**5725**	4.6	5635	5725	5725	5725		1	2
E05_16	68	94	61	16	9	**5830**	3.2	5830	5830	5830	5830		0	1
E05_18	68	94	61	18	9	**5715**	4.2	5715	5715	5715	5715		0	1
E05_20	68	94	61	20	9	**5395**	6.1	5395	5395	5395	5395		2	2
E06_9	49	66	43	9	6	**2720**	0.4	2720	2720	2720	2720		0	1
E06_11	49	66	43	11	5	**2815**	2.4	2815	2815	2815	2815		0	1
E06_12	49	66	43	12	5	**2205**	0.8	2205	2205	2205	2205		0	1
E06_14	49	66	43	14	5	**2595**	5.2	2595	2595	2595	2595		0	1
E07_15	73	94	50	15	9	**5045**	2.9	5045	5045	5045	5045		0	1
E07_18	73	94	50	18	8	**5085**	11.7	5085	5085	5085	5085		0	1
E08_17	74	98	59	17	9	**6350**	5.3	6350	6350	6350	6350		0	1
E08_19	74	98	59	19	9	**6220**	6.1	6220	6220	6220	6220		0	1
E08_23	74	98	59	23	9	**5550**	37.0	5550	5550	5550	5550		0	1
E09_32	93	141	103	32	12	**8120**	889.9	8089	8120	8120	8120		7	2
E09_36	93	141	103	36	12	***6945**	TL	6839	6931	6931	—	n.a.	34	6
E09_37	93	141	103	37	12	**7205**	1090.8	7180	7205	7205	7205		12	2
E09_38	93	141	103	38	12	—	TL	7025	7070	7070	—	n.a.	39	6
E10_14	56	76	49	14	7	**4190**	2.0	4190	4190	4190	4190		0	1
E10_15	56	76	49	15	7	**4100**	1.7	4100	4100	4100	4100		0	1
E10_17	56	76	49	17	7	**4040**	2.7	4040	4040	4040	4040		0	1
E10_19	56	76	49	19	7	**4155**	5.7	4115	4155	4155	4155		3	2
E11_29	80	113	94	29	10	**5160**	161.4	5102	5151	5160	5160		24	6
E11_39	80	113	94	39	10	**4960**	3197.2	4904	4927	4960	4960		73	38
E11_41	80	113	94	41	10	**5220**	3117.7	5107	5199	5220	5220		73	25
E12_19	74	103	67	19	9	**5410**	6.0	5410	5410	5410	5410		0	1
E12_21	74	103	67	21	9	**5080**	24.9	5065	5080	5080	5080		3	2
E12_24	74	103	67	24	9	**4745**	14.5	4745	4745	4745	4745		0	1
E13_13	49	73	52	13	8	**4065**	1.0	4065	4065	4065	4065		0	1
E13_21	49	73	52	21	7	**3840**	16.0	3840	3840	3840	3840		3	2
E14_18	53	72	55	18	8	**4680**	2.6	4680	4680	4680	4680		0	1
E14_21	53	72	55	21	8	**4990**	4.1	4990	4990	4990	4990		0	1
E14_24	53	72	55	24	8	**4500**	13.1	4478	4500	4500	4500		10	2
E15_19	85	126	107	19	9	**6000**	7.3	6000	6000	6000	6000		0	1
E15_28	85	126	107	28	9	**4940**	227.9	4842	4940	4940	4940		14	3
E15_35	85	126	107	35	9	**4830**	572.6	4668	4825	4830	4830		47	10
E15_36	85	126	107	36	9	**4815**	2992.9	4650	4809	4815	4815		73	17
E16_15	60	80	54	15	7	**4610**	1.9	4610	4610	4610	4610		0	1
E16_20	60	80	54	20	7	**4170**	10.0	4155	4170	4170	4170		4	2
E16_22	60	80	54	22	7	**4120**	15.8	4114	4120	4120	4120		8	2
E16_24	60	80	54	24	7	**3955**	18.0	3915	3955	3955	3955		7	2
E17_9	38	50	36	9	6	**3080**	0.3	3080	3080	3080	3080		0	1
E17_11	38	50	36	11	6	**3045**	0.8	3045	3045	3045	3045		0	1
E17_14	38	50	36	14	5	**3215**	2.0	3210	3215	3215	3215		8	2
E17_16	38	50	36	16	5	**3135**	19.2	3078	3109	3135	3135		25	16
E18_16	78	110	88	16	8	**4930**	13.3	4930	4930	4930	4930		0	1
E18_26	78	110	88	26	8	**4020**	2236.2	3915	3988	4020	4020		79	54
E18_38	78	110	88	38	8	*4150	TL	3991	4054	4054	—	n.a.	69	10
E19_17	77	103	66	17	6	**4520**	9.3	4520	4520	4520	4520		0	1
E19_20	77	103	66	20	6	**4500**	21.8	4500	4500	4500	4500		0	1
E19_22	77	103	66	22	6	**3920**	115.3	3920	3920	3920	3920		0	1
E19_29	77	103	66	29	6	*3920	TL	3698	3774	3796	3995	5.24	102	30
E20_12	56	80	63	12	7	**3510**	0.9	3510	3510	3510	3510		0	1
E20_14	56	80	63	14	7	**3495**	6.0	3493	3493	3495	3495		0	3
E20_17	56	80	63	17	7	**3385**	3.7	3385	3385	3385	3385		0	1
E20_28	56	80	63	28	7	***3205**	TL	3099	3172	3194	3210	0.50	123	88
E21_16	57	82	72	16	7	**4455**	2.0	4455	4455	4455	4455		0	1
E21_26	57	82	72	26	7	**4090**	292.5	4015	4071	4090	4090		40	15
E21_27	57	82	72	27	7	**3995**	87.2	3958	3995	3995	3995		33	7
E22_12	54	73	44	12	5	**2825**	38.3	2740	2773	2825	2825		18	31
E22_14	54	73	44	14	5	**2695**	23.3	2653	2680	2695	2695		24	13
E22_16	54	73	44	16	5	**2585**	56.1	2534	2567	2585	2585		26	20
E22_17	54	73	44	17	5	**2650**	2564.4	2484	2514	2650	2650		227	490
E23_16	93	130	89	16	9	**4545**	3.7	4545	4545	4545	4545		0	1
E23_28	93	130	89	28	8	***4260**	TL	4243	4243	4243	—	n.a.	15	1
E23_35	93	130	89	35	8	**4110**	386.3	4099	4110	4110	4110		25	2
E23_40	93	130	89	40	8	**3840**	1769.0	3731	3833	3840	3840		74	12
E24_15	97	142	86	15	9	**4795**	14.4	4795	4795	4795	4795		0	1
E24_23	97	142	86	23	8	***4645**	TL	4597	4642	4645	4650	0.11	14	9
E24_31	97	142	86	31	8	***4450**	TL	4360	4360	4360	—	n.a.	14	1
E24_37	97	142	86	37	8	*4530	TL	4208	4314	4314	—	n.a.	74	10
E25_7	26	35	28	7	4	**2045**	0.3	2033	2045	2045	2045		1	2
E25_10	26	35	28	10	4	**1725**	0.5	1725	1725	1725	1725		0	1
E25_11	26	35	28	11	4	**1685**	0.6	1685	1685	1685	1685		0	1

Table 13: Detailed results for the `BMCV` instances, subset `E`.

Instance							BPC Statistics							
								Bounds					Cuts/Tree	
Name	$\|V\|$	$\|E\|$	$\|E_R\|$	$\|H\|$	m	BKS	Time	LB_{LP}	LB_{SRI}	LB_{tree}	UB	% Gap	#SRIs	#B&B
F01_11	73	105	85	11	6	**4785**	2.0	4785	4785	4785	4785		0	1
F01_15	73	105	85	15	5	**5210**	35.5	5190	5210	5210	5210		11	3
F01_18	73	105	85	18	5	**4790**	69.4	4790	4790	4790	4790		0	1
F02_13	58	81	58	13	4	**4265**	4.2	4265	4265	4265	4265		0	1
F02_18	58	81	58	18	4	**3740**	59.3	3740	3740	3740	3740		0	1
F02_19	58	81	58	19	4	**3740**	102.0	3740	3740	3740	3740		0	1
F02_23	58	81	58	23	4	**3750**	253.0	3708	3750	3750	3750		25	3
F03_9	46	61	47	9	3	**1915**	1.2	1890	1915	1915	1915		6	3
F03_11	46	61	47	11	3	**1845**	0.9	1845	1845	1845	1845		0	1
F03_16	46	61	47	16	3	**1685**	2.7	1685	1685	1685	1685		0	1
F03_21	46	61	47	21	3	**1685**	7.6	1685	1685	1685	1685		0	1
F04_14	70	99	77	14	5	**3805**	7.2	3765	3805	3805	3805		5	3
F04_16	70	99	77	16	5	**3925**	2.6	3925	3925	3925	3925		0	1
F04_17	70	99	77	17	5	**3675**	3.4	3675	3675	3675	3675		0	1
F04_28	70	99	77	28	5	**3890**	580.6	3792	3884	3890	3890		75	16
F05_13	68	94	61	13	5	**4100**	5.5	4084	4100	4100	4100		5	2
F05_24	68	94	61	24	5	**3750**	649.2	3707	3735	3750	3750		97	24
F05_26	68	94	61	26	5	**3725**	566.3	3684	3712	3725	3725		71	11
F06_8	49	66	43	8	3	**1990**	1.6	1977	1990	1990	1990		10	3
F06_9	49	66	43	9	3	**2075**	0.5	2075	2075	2075	2075		0	1
F06_10	49	66	43	10	3	**2120**	1.7	2118	2120	2120	2120		3	2
F06_12	49	66	43	12	3	**2050**	1.8	2050	2050	2050	2050		0	1
F07_11	73	94	50	11	4	**3780**	1.1	3780	3780	3780	3780		0	1
F07_15	73	94	50	15	4	**3780**	3.6	3780	3780	3780	3780		0	1
F07_21	73	94	50	21	4	**3610**	221.6	3511	3610	3610	3610		45	5
F07_22	73	94	50	22	4	**3750**	77.0	3632	3750	3750	3750		26	3
F08_12	74	98	59	12	5	**4250**	2.3	4250	4250	4250	4250		0	1
F08_14	74	98	59	14	5	**4250**	8.2	4238	4250	4250	4250		4	2
F08_15	74	98	59	15	5	**3995**	5.7	3995	3995	3995	3995		0	1
F08_22	74	98	59	22	5	**3995**	39.4	3965	3995	3995	3995		11	2
F09_15	93	141	103	15	7	**5865**	82.1	5800	5865	5865	5865		5	3
F09_16	93	141	103	16	7	**5625**	30.6	5613	5625	5625	5625		1	2
F09_18	93	141	103	18	6	**6605**	50.2	6605	6605	6605	6605		0	1
F09_42	93	141	103	42	6	—	TL	5021	5021	5021	—	n.a.	30	1
F10_13	56	76	49	13	4	**3325**	4.0	3269	3325	3325	3325		10	3
F10_15	56	76	49	15	4	**3230**	30.8	3152	3230	3230	3230		36	8
F10_16	56	76	49	16	4	**3125**	2.6	3125	3125	3125	3125		0	1
F10_18	56	76	49	18	4	**3145**	9.1	3089	3145	3145	3145		17	3
F11_15	80	113	94	15	5	**4160**	7.6	4160	4160	4160	4160		0	1
F11_20	80	113	94	20	5	**4365**	14.4	4365	4365	4365	4365		0	1
F11_29	80	113	94	29	5	**4180**	1241.0	4105	4170	4180	4180		68	14
F11_42	80	113	94	42	5	**4070**	1466.4	3917	4070	4070	4070		93	5
F12_10	74	103	67	10	5	**4125**	4.4	4093	4125	4125	4125		1	2
F12_14	74	103	67	14	5	**4070**	15.7	4070	4070	4070	4070		0	1
F12_28	74	103	67	28	5	**3780**	329.2	3607	3780	3780	3780		44	5
F13_11	49	73	52	11	4	**3315**	3.8	3305	3315	3315	3315		5	3
F13_15	49	73	52	15	4	**3140**	5.0	3118	3140	3140	3140		11	2
F13_17	49	73	52	17	4	**3140**	8.8	3118	3140	3140	3140		12	2
F13_23	49	73	52	23	4	**2990**	42.2	2960	2990	2990	2990		24	3
F14_7	53	72	55	7	5	**3850**	0.4	3850	3850	3850	3850		0	1
F14_17	53	72	55	17	4	**3745**	22.6	3740	3745	3745	3745		5	2
F14_19	53	72	55	19	4	**3670**	11.9	3670	3670	3670	3670		0	1
F14_22	53	72	55	22	4	**3590**	51.6	3568	3590	3590	3590		23	4
F15_21	85	126	107	21	5	**4145**	36.0	4138	4145	4145	4145		6	2
F15_36	85	126	107	36	5	**3985**	2634.5	3819	3985	3985	3985		133	9
F15_37	85	126	107	37	5	**3985**	2380.3	3819	3985	3985	3985		126	9
F15_45	85	126	107	45	5	**3925**	1351.0	3829	3925	3925	3925		40	3
F16_7	60	80	54	7	4	**3935**	0.8	3935	3935	3935	3935		0	1
F16_15	60	80	54	15	4	**3345**	5.5	3345	3345	3345	3345		0	1
F16_18	60	80	54	18	4	**3345**	5.0	3345	3345	3345	3345		0	1
F17_4	38	50	36	4	3	**2680**	0.2	2680	2680	2680	2680		0	1
F17_8	38	50	36	8	3	**2500**	1.0	2475	2500	2500	2500		3	2
F17_11	38	50	36	11	3	**2295**	0.8	2295	2295	2295	2295		0	1
F17_13	38	50	36	13	3	**2115**	1.4	2115	2115	2115	2115		0	1
F18_19	78	110	88	19	4	**3290**	79.4	3289	3290	3290	3290		9	2
F18_20	78	110	88	20	4	**3300**	63.2	3295	3300	3300	3300		3	2
F18_27	78	110	88	27	4	**3240**	277.1	3234	3240	3240	3240		15	2
F18_37	78	110	88	37	4	*3275	TL	3172	3247	3247	—	n.a.	71	4
F19_12	77	103	66	12	4	**2880**	8.3	2870	2880	2880	2880		1	2
F19_26	77	103	66	26	3	**2575**	304.3	2575	2575	2575	2575		0	1
F19_28	77	103	66	28	3	**2830**	1823.5	2771	2830	2830	2830		40	3
F19_29	77	103	66	29	3	**2830**	3300.3	2765	2830	2830	2830		56	4
F20_9	56	80	63	9	4	**2620**	0.7	2620	2620	2620	2620		0	1
F20_23	56	80	63	23	4	**2570**	201.0	2498	2538	2570	2570		60	15
F20_24	56	80	63	24	4	**2510**	267.3	2456	2484	2510	2510		78	18
F20_28	56	80	63	28	4	**2500**	1900.4	2423	2455	2500	2500		154	59
F21_8	57	82	72	8	5	**3485**	0.9	3485	3485	3485	3485		0	1
F21_19	57	82	72	19	4	**3130**	54.0	3083	3130	3130	3130		21	3
F21_28	57	82	72	28	4	**3045**	1159.8	2954	3045	3045	3045		95	7
F22_7	54	73	44	7	3	**2310**	2.6	2252	2310	2310	2310		6	3
F22_9	54	73	44	9	3	**2305**	12.3	2218	2258	2305	2305		26	13
F22_14	54	73	44	14	3	**2130**	14.2	2099	2130	2130	2130		23	3
F23_11	93	130	89	11	4	**3520**	10.8	3450	3520	3520	3520		2	2
F23_14	93	130	89	14	4	**3585**	31.7	3510	3585	3585	3585		5	2
F23_28	93	130	89	28	4	**3395**	1275.3	3308	3388	3395	3395		65	12
F23_31	93	130	89	31	4	**3245**	160.6	3245	3245	3245	3245		0	1
F24_7	97	142	86	7	5	**4040**	4.0	4040	4040	4040	4040		0	1
F24_9	97	142	86	9	4	**3895**	15.3	3895	3895	3895	3895		0	1
F24_18	97	142	86	18	4	**3585**	42.2	3585	3585	3585	3585		0	1
F24_28	97	142	86	28	4	*3745	TL	3605	3727	3727	—	n.a.	72	10
F25_4	26	35	28	4	2	**1535**	0.1	1535	1535	1535	1535		0	1
F25_7	26	35	28	7	2	**1410**	0.2	1410	1410	1410	1410		0	1
F25_10	26	35	28	10	2	**1410**	1.1	1410	1410	1410	1410		0	1
F25_11	26	35	28	11	2	**1390**	1.7	1390	1390	1390	1390		0	1

Table 14: Detailed results for the `BMCV` instances, subset `F`.

Instance							BPC Statistics							
								Bounds					Cuts/Tree	
Name	$\lvert V\rvert$	$\lvert E\rvert$	$\lvert E_R\rvert$	$\lvert H\rvert$	m	BKS	Time	LB_{LP}	LB_{SRI}	LB_{tree}	UB	% Gap	#SRIs	#B&B
egl-e1-A_12	77	98	51	12	6	**4197**	3.5	4197	4197	4197	4197		0	1
egl-e1-A_14	77	98	51	14	5	**3786**	12.7	3786	3786	3786	3786		0	1
egl-e1-A_20	77	98	51	20	5	**3954**	307.7	3904	3941	3954	3954		30	11
egl-e1-B_12	77	98	51	12	8	**5481**	3.4	5481	5481	5481	5481		0	1
egl-e1-B_20	77	98	51	20	7	**4905**	148.4	4805	4901	4905	4905		22	7
egl-e1-B_22	77	98	51	22	7	**4831**	24.1	4786	4831	4831	4831		8	2
egl-e1-C_17	77	98	51	17	11	**6727**	6.0	6727	6727	6727	6727		0	1
egl-e1-C_18	77	98	51	18	12	**6898**	10.6	6898	6898	6898	6898		0	1
egl-e1-C_20	77	98	51	20	11	**6259**	5.3	6259	6259	6259	6259		0	1
egl-e1-C_22	77	98	51	22	10	**6324**	41.1	6305	6324	6324	6324		9	3
egl-e2-A_20	77	98	72	20	7	**5554**	12.2	5554	5554	5554	5554		0	1
egl-e2-A_26	77	98	72	26	7	**5813**	163.0	5765	5813	5813	5813		17	3
egl-e2-A_31	77	98	72	31	7	**5349**	894.3	5245	5334	5349	5349		94	22
egl-e2-B_18	77	98	72	18	11	**7461**	7.6	7461	7461	7461	7461		0	1
egl-e2-B_22	77	98	72	22	11	**7220**	50.7	7195	7220	7220	7220		3	2
egl-e2-B_23	77	98	72	23	10	**7770**	35.5	7770	7770	7770	7770		0	1
egl-e2-B_25	77	98	72	25	10	**7037**	36.2	6962	7037	7037	7037		1	2
egl-e2-C_29	77	98	72	29	15	**9430**	11.1	9430	9430	9430	9430		0	1
egl-e2-C_32	77	98	72	32	14	**9292**	156.7	9290	9290	9292	9292		0	3
egl-e3-A_24	77	98	87	24	8	**6597**	531.9	6516	6568	6597	6597		40	20
egl-e3-A_31	77	98	87	31	8	**6775**	308.5	6764	6775	6775	6775		10	2
egl-e3-A_37	77	98	87	37	8	**6207**	556.0	6174	6204	6207	6207		60	8
egl-e3-B_22	77	98	87	22	14	**9183**	29.3	9111	9183	9183	9183		1	2
egl-e3-B_23	77	98	87	23	13	**9898**	14.1	9898	9898	9898	9898		0	1
egl-e3-B_32	77	98	87	32	12	**8299**	181.4	8286	8299	8299	8299		6	2
egl-e3-B_37	77	98	87	37	12	**8256**	3404.6	8147	8210	8256	8256		99	58
egl-e3-C_32	77	98	87	32	20	**12206**	9.4	12206	12206	12206	12206		0	1
egl-e3-C_36	77	98	87	36	17	**11380**	615.7	11310	11364	11380	11380		13	15
egl-e3-C_38	77	98	87	38	17	**11318**	137.0	11260	11318	11318	11318		7	3
egl-e4-A_22	77	98	98	22	9	**7298**	29.8	7268	7298	7298	7298		2	2
egl-e4-A_28	77	98	98	28	9	**6892**	59.2	6892	6892	6892	6892		0	1
egl-e4-A_34	77	98	98	34	9	**6892**	3471.9	6832	6855	6892	6892		113	61
egl-e4-B_30	77	98	98	30	14	**10800**	20.6	10800	10800	10800	10800		0	1
egl-e4-B_38	77	98	98	38	14	**10043**	473.6	10019	10043	10043	10043		10	3
egl-e4-B_43	77	98	98	43	14	**9524**	335.0	9504	9524	9524	9524		13	2
egl-e4-B_44	77	98	98	44	14	**9470**	407.6	9442	9470	9470	9470		18	3
egl-e4-C_41	77	98	98	41	20	**13518**	938.7	13437	13445	13518	13518		8	20
egl-e4-C_42	77	98	98	42	20	**12624**	131.4	12624	12624	12624	12624		0	1
egl-e4-C_43	77	98	98	43	20	**12590**	115.6	12590	12590	12590	12590		0	1

Table 15: Detailed results for the EGL instances, subset E.

Instance							BPC Statistics							
								Bounds					Cuts/Tree	
Name	$\|V\|$	$\|E\|$	$\|E_R\|$	$\|H\|$	m	BKS	Time	LB_{LP}	LB_{SRI}	LB_{tree}	UB	% Gap	#SRIs	#B&B
egl-s1-A_13	140	190	75	13	8	**6253**	33.9	6253	6253	6253	6253		0	1
egl-s1-A_17	140	190	75	17	7	**6224**	378.1	6224	6224	6224	6224		0	1
egl-s1-B_22	140	190	75	22	10	**7005**	135.6	7005	7005	7005	7005		0	1
egl-s1-B_23	140	190	75	23	10	*6994	TL	6966	6966	6966	—	n.a.	2	1
egl-s1-B_24	140	190	75	24	10	—	TL	6953	6953	6953	—	n.a.	0	1
egl-s1-B_26	140	190	75	26	10	**6930**	295.9	6930	6930	6930	6930		0	1
egl-s1-C_26	140	190	75	26	16	***9591**	TL	9591	9591	9591	—	n.a.	0	0
egl-s1-C_27	140	190	75	27	15	**9881**	871.7	9783	9881	9881	9881		2	2
egl-s1-C_29	140	190	75	29	14	—	TL	9486	9486	9486	—	n.a.	0	0
egl-s2-A_42	140	190	147	42	14	—	TL	11020	11020	11020	—	n.a.	0	0
egl-s2-A_44	140	190	147	44	14	—	TL	10750	10750	10750	—	n.a.	0	0
egl-s2-A_48	140	190	147	48	14	—	TL	10715	10715	10715	—	n.a.	0	0
egl-s2-A_50	140	190	147	50	14	—	TL	11000	11000	11000	—	n.a.	0	0
egl-s2-B_39	140	190	147	39	23	**14903**	228.7	14903	14903	14903	14903		0	1
egl-s2-B_53	140	190	147	53	21	—	TL	14506	14506	14506	—	n.a.	0	0
egl-s2-B_56	140	190	147	56	20	—	TL	14701	14701	14701	—	n.a.	0	0
egl-s2-B_60	140	190	147	60	20	—	TL	14935	14935	14935	—	n.a.	0	0
egl-s2-C_57	140	190	147	57	28	**18292**	2197.5	18292	18292	18292	18292		0	1
egl-s2-C_61	140	190	147	61	27	—	TL	18475	18475	18475	—	n.a.	0	0
egl-s3-A_42	140	190	159	42	15	**11420**	759.1	11420	11420	11420	11420		0	1
egl-s3-A_45	140	190	159	45	15	—	TL	10923	10923	10923	—	n.a.	0	0
egl-s3-A_62	140	190	159	62	15	—	TL	10822	10822	10822	—	n.a.	0	0
egl-s3-A_64	140	190	159	64	15	—	TL	10813	10813	10813	—	n.a.	0	0
egl-s3-B_41	140	190	159	41	23	**16593**	565.6	16593	16593	16593	16593		0	1
egl-s3-B_57	140	190	159	57	22	—	TL	14610	14610	14610	—	n.a.	0	0
egl-s3-B_58	140	190	159	58	22	—	TL	14829	14829	14829	—	n.a.	0	0
egl-s3-B_70	140	190	159	70	22	—	TL	14367	14367	14367	—	n.a.	0	0
egl-s3-C_61	140	190	159	61	29	—	TL	20135	20135	20135	—	n.a.	0	0
egl-s3-C_65	140	190	159	65	29	—	TL	19450	19450	19450	—	n.a.	0	0
egl-s3-C_69	140	190	159	69	29	—	TL	18995	18995	18995	—	n.a.	0	0
egl-s3-C_71	140	190	159	71	29	—	TL	19905	19905	19905	—	n.a.	0	0
egl-s4-A_48	140	190	190	48	19	—	TL	13901	13901	13901	—	n.a.	0	0
egl-s4-A_51	140	190	190	51	19	—	TL	13732	13732	13732	—	n.a.	0	0
egl-s4-A_68	140	190	190	68	19	—	TL	13057	13057	13057	—	n.a.	0	0
egl-s4-A_74	140	190	190	74	19	—	TL	13044	13044	13044	—	n.a.	0	0
egl-s4-B_55	140	190	190	55	28	—	TL	19829	19829	19829	—	n.a.	0	0
egl-s4-B_69	140	190	190	69	27	—	TL	17928	17928	17928	—	n.a.	0	0
egl-s4-B_70	140	190	190	70	27	—	TL	18552	18552	18552	—	n.a.	0	0
egl-s4-B_72	140	190	190	72	27	—	TL	17573	17573	17573	—	n.a.	0	0
egl-s4-C_70	140	190	190	70	38	—	TL	24687	24687	24687	—	n.a.	0	0
egl-s4-C_73	140	190	190	73	37	—	TL	23052	23052	23052	—	n.a.	0	0
egl-s4-C_78	140	190	190	78	36	—	TL	23012	23012	23012	—	n.a.	0	0
egl-s4-C_84	140	190	190	84	35	—	TL	54933	54933	54933	—	n.a.	0	0

Table 16: Detailed results for the `EGL` instances, subset `S`.