Exact Solution of the Soft-Clustered Vehicle-Routing Problem

Timo Hintsch, Stefan Irnich

September 24, 2018

Discussion paper number 1813

Contact details

Timo Hintsch
Gutenberg School of Management and Economics
Chair of Logistics Management
University of Mainz
Jakob-Welder-Weg 9
55128 Mainz
Germany

thintsch@uni-mainz.de

Stefan Irnich
Gutenberg School of Management and Economics
Chair of Logistics Management
University of Mainz
Jakob-Welder-Weg 9
55128 Mainz
Germany

irnich@uni-mainz.de

Exact Solution of the Soft-Clustered Vehicle-Routing Problem

Timo Hintsch[*,a], Stefan Irnich[a]

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

Abstract

The soft-clustered vehicle-routing problem (SoftCluVRP) extends the classical capacitated vehicle-routing problem by one additional constraint: The customers are partitioned into clusters and feasible routes must respect the soft-cluster constraint, that is, all customers of the same clusters must be served by the same vehicle. In this article, we design and analyze different branch-and-price algorithms for the exact solution of the SoftCluVRP. The algorithms differ in the way the column-generation subproblem, a variant of the shortest-path problem with resource constraints (SPPRC), is solved. The standard approach for SPPRCs is based on dynamic-programming labeling algorithms. We show that even with all the recent acceleration techniques and tricks (e.g., partial pricing, bidirectional labeling, decremental state space relaxation) available for SPPRC labeling algorithms, the solution of the subproblem remains extremely difficult. The main contribution is the modeling and solution of the subproblem using a branch-and-cut algorithm. The conducted computational experiments prove that branch-and-price equipped with this integer programming-based approach outperforms sophisticated labeling-based algorithms by one order of magnitude. The largest SoftCluVRP instances solved to optimality have more than 400 customers or more than 50 clusters.

Key words: Vehicle Routing, branch-and-price, shortest-path problem with resource constraints, dynamic-programming labeling, branch-and-cut

1. Introduction

The *clustered vehicle-routing problem* (CluVRP, Sevaux and Sörensen, 2008) is a variant of the classical *capacitated vehicle-routing problem* (CVRP, Vigo and Toth, 2014) in which the customers are grouped into disjoint clusters. A feasible CluVRP route must serve each cluster integrally, that is, all customers of a cluster must be served by the same vehicle and in consecutive visits. This problem 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). We consider a relaxation of the CluVRP, the *soft-clustered vehicle-routing problem* (SoftCluVRP), in which the only additional constraint compared to the CVRP is that all customers of a cluster must be served by the same vehicle. In contrast to the CluVRP, visits to customers of the same cluster may or may not be interrupted by visits to other customers. The SoftCluVRP has been introduced by Defryn and Sörensen (2017).

CluVRP and SoftCluVRP arise in applications where the routing decision must take into account already taken clustering decisions. For example, Sevaux and Sörensen (2008) mention parcel/small-package delivery in courier companies as an application field: In a first step, parcels are sorted according to given districting (see Butsch *et al.*, 2014, for districting). Typically, the sorting policy, i.e., the sorting of parcels e.g. by regional zones and/or ZIP codes, is altered only once in a while. Note that the sorting policy must always

[*]Corresponding author.
Email address: `thintsch@uni-mainz.de` (Timo Hintsch)

be fixed before the actual demand distribution over the zones is known. In a second step, the batches of parcels, as they result from the sorting, are delivered to the recipients.

An instance of the SoftCluVRP is defined over a complete undirected graph $G = (V, E)$ with the vertex set $V = \{0, \ldots, n\}$ and the edge set E. The vertex set comprises the depot vertex 0 and the customer vertices $V \setminus \{0\} = \{1, \ldots, n\}$. The vertices are partitioned into $N + 1$ clusters $V_0, V_1, V_2, \ldots, V_N$, where we define the depot cluster $V_0 = \{0\}$ for convenience. The *customer clusters* are indexed by $h \in H = \{1, 2, \ldots, N\}$ and a positive demand $d_h > 0$ is associated with cluster V_h, while the depot cluster V_0 has zero demand $d_0 = 0$. For $h \in H \cup \{0\}$, the cardinality of a cluster is $n_h = |V_h|$ and $h(i) \in H \cup \{0\}$ refers to the index of the cluster to which vertex $i \in V$ belongs. A homogenous fleet of m vehicles with capacity Q is hosted at the depot 0. Non-negative routing costs c_{ij} are associated with every edge $\{i, j\} \in E$.

A *route* $r = (i_0, i_1, \ldots, i_r, i_{r+1})$ is a cycle in G passing through the depot 0, i.e., a cycle with $i_0 = i_{r+1} = 0$. The customer clusters *touched* by the route r, i.e., with $V_h \cap \{i_1, \ldots, i_r\} \neq \varnothing$, are denoted by $H(r) \subseteq H$. The route r is feasible for the SoftCluVRP if

(i) $i_1, \ldots, i_r$ are all different, (elementarity constraint)
(ii) for all $h \in H(r)$, $V_h \subseteq \{i_1, \ldots, i_r\}$, (soft-cluster constraints)
(iii) and $\sum_{h \in H(r)} d_h \leq Q$. (capacity constraint)

Note that for the CluVRP the constraints (ii) must be replaced by the following conditions:

(ii') for all $h \in H(r)$, there exists an index $k \in \{1, 2, \ldots, r - n_h + 1\}$
such that $V_h = \{i_k, i_{k+1}, \ldots, i_{k+n_h-1}\}$. (hard-cluster constraints)

The contribution of this paper is the design and computational analysis of different branch-and-price algorithms for the exact solution of the SoftCluVRP. Branch-and-price is the leading solution approach for many variants of the VRP (recently surveyed by Irnich *et al.*, 2014) and can be summarized as follows: A route/path-based extended formulation of the VRP variant, the so-called *master program*, is solved via *column generation* (Desaulniers *et al.*, 2005). The starting point is always a *restricted master program* (RMP) that comprises a (small) subset of routes and relaxes the integrality constraints on the route variables (integrality is later enforced via branching). Missing routes for the solution of the linear relaxation of the master program are dynamically and iteratively generated with the help of a pricing subproblem. This pricing problem can be formulated as a *shortest-path problem with resource constraints* (SPPRC, Irnich and Desaulniers, 2005). For almost all VRP variants, the associated SPPRCs are best solved with dynamic-programming labeling algorithms (e.g. Feillet *et al.*, 2004; Irnich and Villeneuve, 2006; Righini and Salani, 2008; Baldacci *et al.*, 2011). To our surprise, the SPPRCs that result from the SoftCluVRP are extremely difficult to solve even for instances of rather moderate size. The paper at hand will show that even with all the recent acceleration techniques and tricks available for the SPPRC dynamic-programming labeling algorithms, the situation hardly improves. For example, bidirectional labeling has proven very effective for many VRP variants (Righini and Salani, 2006; Tilk *et al.*, 2017) but the soft-cluster constraints are so loose constraints that the combinatorial explosion is often not effectively suppressed (Gschwind *et al.* (2017) report a similar phenomenon for some loosely-constrained VRPs with pickup-and-delivery structure). Our most important finding is, therefore, that a relatively simple *integer programming* (IP) formulation solved with a standard IP-solver is highly effective for the SoftCluVRP pricing problems. This result is rather remarkable because IP-based approaches such as branch-and-cut have almost never reached the performance of labeling-based pricing algorithms (we refer to the discussion in Drexl and Irnich, 2012).

The remainder of this paper is structured as follows. Section 2 provides a compact three-index formulation for the SoftCluVRP that we use to derive the path-based reformulation and the pricing subproblem. Section 3 presents the two exact solution approaches for the pricing subproblem that are based on dynamic-programming labeling and branch-and-cut. Moreover, a primal heuristic pricer tailored to the SoftCluVRP is presented. The overall pricing strategies as well as branching and its impact on the subproblem is discussed in Section 4. Section 5 summarizes the comprehensive computational studies conducted on the new branch-and-price algorithms. Final conclusions are drawn in Section 6.

2. Three-Index, Extensive, and Subproblem Formulation

A three-index formulation for the asymmetric version of the SoftCluVRP was presented by Defryn and Sörensen (2017). In Section 2.1, we present another three-index formulation for the symmetric version of the SoftCluVRP, because the available benchmark instances are all symmetric. From this three-index model, we derive an extensive route-based formulation via IP Dantzig-Wolfe decomposition (Lübbecke and Desrosiers, 2005) in Section 2.2. Moreover, the associated SPPRC subproblem is formulated as an IP in Section 2.3.

For the models and any subset $S \subseteq V$, we use the notation $\delta(S)$ for the set of edges $e = \{i, j\} \in E$ with exactly one endpoint in S, i.e., either $i \in S$ and $j \in V \setminus S$ or $i \in V \setminus S$ and $j \in S$. For singleton sets $S = \{i\}$ with $i \in V$, we write $\delta(i)$ instead of $\delta(\{i\})$. Finally, we define $r(S)$ as the minimum number of vehicles needed to serve the customers $S \subseteq V \setminus \{0\}$. For the models presented subsequently, the lower bound $r(S) = \lceil (\sum_{h \in H: V_h \cap S \neq \varnothing} d_h)/Q \rceil$ is sufficient. As for other purely capacitated VRP variants, exact bounds can be determined by solving the corresponding bin-packing problem.

2.1. Three-Index Formulation

The following three-index formulation explicitly models the routes performed by each of the m vehicles. We therefore define the *fleet* as $K = \{1, 2, \ldots, m\}$. The three-index formulation uses two types of integer decision variables both indexed by $k \in K$. First, for each edge $e \in E$ and vehicle $k \in K$ there is a routing variable x_e^k indicating how often vehicle k traverses edge e (leading to three indices i, j, k for $e = \{i, j\} \in E$ and $k \in K$). Recall that edges adjacent to the depot 0 may be traversed twice. The cluster-assignment variable z_h^k, one for each cluster index $h \in H$ and vehicle $k \in K$, indicates that vehicle k serves cluster V_h. The model is:

$$\min \sum_{k \in K} \sum_{e \in E} c_e x_e^k \tag{1a}$$

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

$$\sum_{e \in \delta(0)} x_e^k = 2 \qquad \forall k \in K \tag{1c}$$

$$\sum_{e \in \delta(i)} x_e^k = 2 z_{h(i)}^k \qquad \forall i \in V \setminus \{0\}, \forall k \in K \tag{1d}$$

$$\sum_{k \in K} \sum_{e \in \delta(S)} x_e^k \geq 2 r(S) \qquad \forall S \subseteq V \setminus \{0\}, S \neq \varnothing \tag{SEC}$$

$$x_e^k \in \{0, 1, 2\} \qquad \forall e \in \delta(0), \forall k \in K \tag{1e}$$

$$x_e^k \in \{0, 1\} \qquad \forall e \in E \setminus \delta(0), \forall k \in K \tag{1f}$$

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

The objective (1a) minimizes the routing costs over all vehicles. Constraints (1b) ensure that each cluster is served exactly once. The fleet size is set to m by condition (1c). The routing and cluster-assignment variables are coupled via (1d) making sure that each customer $i \in V_h$ (for each $h \in H$) is served by vehicle k, i.e., $\sum_{e \in \delta(i)} x_e^k = 2$, if and only if the cluster V_h is assigned to that vehicle k. The exponentially-sized family of *subtour-elimination constraints* is given by (SEC). Note that these *aggregated* subtour-elimination constraints (SEC) can also be written in *disaggregated* form as

$$\sum_{e \in \delta(S)} x_e^k \geq 2 z_{h(i)}^k \qquad \forall S \subseteq V \setminus \{0\}, S \neq \varnothing, \forall i \in S, \forall k \in K \tag{1h}$$

together with explicit *capacity constraints*

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

With this replacement, i.e., (SEC) replaced by (1h) and (1i), the only constraints of model (1) that are not vehicle-specific are the constraints (1b).

2.2. Extensive Route-Based Formulation

An IP Dantzig-Wolfe decomposition on the non-vehicle specific constraints (1b) and a subsequent aggregation over the vehicles $k \in K$ (see Lübbecke and Desrosiers, 2005) leads to an extensive formulation with one binary decision variable for each feasible route. Let Ω be the set of all feasible SoftCluVRP routes fulfilling conditions (i)–(iii) of Section 1 (feasible w.r.t. elementarity, soft-cluster, and capacity constraints). Let y_r be the binary decision variable indicating whether route $r \in \Omega$ is chosen in the solution or not. Moreover, let c_r be the cost of route r defined as the sum of the routing costs for all edges traversed by r. Finally, we define a_{hr} as the binary indicator whether route r serves cluster V_h or not.

The extensive path-based formulation is the following extended set-partitioning model:

$$\min \sum_{r\in\Omega} c_r y_r \tag{2a}$$

$$\text{subject to} \sum_{r\in\Omega} a_{hr} y_r = 1 \qquad \forall h \in H \tag{2b}$$

$$\sum_{r\in\Omega} y_r = m \tag{2c}$$

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

The objective (2a) minimizes the total routing costs. The partitioning constraints (2b) ensure that each cluster is served exactly once. Condition (2c) is the aggregated convexity constraint of the Dantzig-Wolfe decomposition saying that the number of routes to choose in the solution is m.

For a subset $\bar{\Omega} \subset \Omega$ of the routes, the linear relaxation of (2) defined over $\bar{\Omega}$ is the corresponding RMP. Missing routes are determined by solving the following pricing subproblem.

2.3. Subproblem Formulation

The task of the subproblem is to identify negative reduced-cost variables (=routes) or to prove that there exist none. Let π_h for $h \in H$ be the dual prices of the partitioning constraints (2b) and let μ be the dual price of the fleet-size constraint (2c). Since we assume the triangle inequality to hold for the costs (c_{ij}), the partitioning constraints (2b) can always be replaced by covering constraints. Therefore, $\pi_h \geq 0$ can be assumed for all $h \in H$.

Recall that for any feasible route $r \in \Omega$, the clusters served by route r are given by the indices $h \in H(r)$. The set $H(r)$ can be written as $\{h \in H : a_{hr} = 1\}$. Hence, the reduced cost of a route r denoted by $\tilde{c}_r$ is given by $c_r - \sum_{h\in H} a_{hr}\pi_h - \mu$.

Note first that the optimal routing cost c_r can be computed by solving a *traveling salesman problem* (TSP, Gutin and Punnen, 2002) over the vertex set $V(r) = V_0 \cup \bigcup_{h\in H(r)} V_h$. Note second that $\sum_{h\in H} a_{hr}\pi_h = \sum_{h\in H(r)} \pi_h$ and that μ is independent of the route r.

The formulation of the subproblem can be formally derived from the original three-index formulation (1) by (i) dropping the vehicle index $k \in K$ from the routing variables x and cluster-assignment variables z, (ii) incorporating the dual prices into the objective, and (iii) leaving out the non-vehicle-specific constraints (1b). The resulting model has integer routing variables x_e describing how often edges $e \in E$ are traversed and binary variables z_h for the selection of the served clusters $h \in H$:

$$\tilde{c}(\pi_h, \mu) = \min \sum_{e \in E} c_e x_e - \sum_{h \in H} \pi_h z_h - \mu \tag{3a}$$

$$\text{subject to} \sum_{e \in \delta(0)} x_e = 2 \tag{3b}$$

$$\sum_{e \in \delta(i)} x_e = 2 z_{h(i)} \qquad \forall i \in V \tag{3c}$$

$$\sum_{e \in \delta(S)} x_e \geq 2 z_{h(i)} \qquad \forall S \subseteq V \setminus \{0\}, S \neq \varnothing, i \in S \tag{3d}$$

$$\sum_{h \in H} d_h z_h \leq Q \tag{3e}$$

$$x_e \in \{0, 1, 2\} \qquad \forall e \in \delta(0) \tag{3f}$$

$$x_e \in \{0, 1\} \qquad \forall e \in E \setminus \delta(0) \tag{3g}$$

$$z_h \in \{0, 1\} \qquad \forall h \in H \tag{3h}$$

The objective (3a) is the minimization of the reduced cost. Constraint (3b) ensures that the route starts and ends at the depot, and constraints (3c) couple the routing decision with the cluster selection, i.e., customer $i \in V \setminus \{0\}$ is visited if and only if its cluster $V_{h(i)}$ is selected. Subtour-elimination constraints are given by (3d). Constraint (3e) is the capacity constraint. Note that these constraints (3b)–(3e) are derived from constraints (1c), (1d), (1h), and (1i), respectively.

The subproblem (3) generalizes the well known *two-matching model* of the TSP (see, e.g., Gutin and Punnen, 2002). Indeed, without the capacity constraint (3e) and for large positive dual prices $\pi_h \gg 0$, an optimal solution would select all clusters (i.e., $z_h = 1$ for all $h \in H$) so that (3b) and (3c) become the two-matching constraints. With the capacity constraints, the subproblem is a *TSP with profits* (Feillet *et al.*, 2005), even if profits π_h are associated to clusters and not individual customers/vertices. According to the classification by Feillet *et al.* (2005), on the one hand, the subproblem is a *profitable tour problem* because it combines the routing cost minimization and the maximization of the cluster profits π_h in its objective. On the other hand, the capacity constraint imposes a lower bound on the not-chosen clusters. Considering the not-selected clusters introduces a penalty for not choosing clusters into the objective function so that the subproblem can also be characterized as a *prize-collecting TSP* (in the sense of Balas, 1989).

3. Solution of the Subproblem

In this section, we describe the two competing solution methods for the solution of the pricing subproblem (3). We start with the traditional and established solution approach based on dynamic-programming labeling in Section 3.1. Our goal is to concisely define the basic components of a SoftCluVRP-tailored labeling algorithm (definition of attributes, initial label, label extension, and feasibility conditions as well as the dominance principle). Moreover, we refine the basic labeling algorithm by integrating the most recent algorithmic enhancements including bidirectional labeling, several acceleration techniques, and the adaptations for the *ng*-path relaxation. The most important novelty of this paper is however the presentation of an IP-based solution approach in Section 3.2. Our branch-and-cut algorithm bases directly on formulation (3) for which we present the initial model and separation procedures for cutting off infeasible integer solutions as well as for finding violated inequalities in fractional solutions. Finally, we developed a direct primal heuristic solver for SoftCluVRP subproblems that searches for negative reduced-cost routes by applying cluster-based add- and drop-exchanges. The primal heuristic solver is used as a heuristic acceleration technique in conjuction with the labeling or branch-and-cut algorithms. It is summarized in Section 3.3.

3.1. Labeling Algorithms

In this subsection, we define the SPPRC on the corresponding directed graph $G' = (V', A)$. The vertex set V' comprises all customer vertices $V \setminus \{0\}$ and two copies of the depot denoted by 0 and $0'$, where the origin depot 0 has no ingoing arcs and the destination depot $0'$ has no outgoing arcs. Hence, the arc set is $A = \{(i,j) : i,j \in V \setminus \{0\}, i \neq j\} \cup \{(0,i) : i \in V \setminus \{0\}\} \cup \{(i,0') : i \in V \setminus \{0\}\}$. Recall that a route r in G was defined as a cycle through 0. Any (feasible) route r in G imposes a 0-$0'$-path $r' = (i_0, i_1, \dots, i_r, i_{r+1})$ in G' (fulfilling conditions (i)–(iii) of Section 1), and vice versa.

3.1.1. Monodirectional Labeling

A forward monodirectional labeling algorithm starts with an initial label at the origin depot 0 that represents the partial path (0). It propagates labels over arcs toward the destination depot $0'$ with the help of so-called *resource extension functions* (REFs, Desaulniers *et al.*, 1998). Each label stores the resource consumption of the corresponding partial path $(0, ..., i)$ that starts at 0 and ends at some vertex $i \in V'$. To avoid the enumeration of all feasible partial paths, provably redundant labels are eliminated through a dominance criterion.

In the SoftCluVRP, a partial path $p = (0, \dots, i)$ is represented by a label L that has the following attributes a.k.a. resources:

L^{cost}:	the accumulated reduced cost of the partial path p;
L^{load}:	the total demand of clusters touched by p;
$(L^{rem_h})_{h \in H}$:	for each $h \in H$, the number of remaining customers $i \in V_h$ that must be visited by a feasible completion of path p; the initial value is set to -1 to indicate that cluster V_h has not yet been visited;
$(L^{visit_v})_{v \in V \setminus \{0\}}$:	the binary visit attributes for customers indicating whether or not customer $v \in V \setminus \{0\}$ has been visited along path p.

The attribute $L_i^{visit_v}$ can also be set to 1 and the attribute $L_i^{rem_h}$ set to 0, if cluster $V_h = V_{h(v)}$ has not yet been visited but it is *unreachable* for all completions of p. The latter condition is fulfilled if the demand d_h exceeds the residual capacity $Q - L_i^{load}$ of p (for details see Feillet *et al.*, 2004).

The initial label for $p = (0)$ is defined as $L_0 = (L_0^{cost}, L_0^{load}, L_0^{rem}, L_0^{visit}) = (0, 0, (\mathbf{-1}), (\mathbf{0}))$, where $\mathbf{-1}$ and $\mathbf{0}$ are vectors with all entries equal to -1 and 0, respectively, of appropriate size.

Next we describe the extension of a label L_i belonging to a feasible partial path $(0, \dots, i)$ along an arc $(i,j) \in A$ to vertex j. The REFs create a new label $L_j = (L_j^{cost}, L_j^{load}, L_j^{rem}, L_j^{visit})$ with the following attributes:

$$L_j^{cost} = L_i^{cost} + c_{ij} - \begin{cases} \pi_{h(j)}, & \text{if } L_i^{rem_{h(j)}} = -1 \\ 0, & \text{otherwise} \end{cases} - \begin{cases} \mu/2, & \text{if } i = 0 \text{ or } j = 0' \\ 0, & \text{otherwise} \end{cases} \tag{4a}$$

$$L_j^{load} = L_i^{load} + \begin{cases} d_{h(j)}, & \text{if } L_i^{rem_{h(j)}} = -1 \\ 0, & \text{otherwise} \end{cases} \tag{4b}$$

$$L_j^{rem_h} = \begin{cases} n_h - 1, & \text{if } h = h(j) \text{ and } L_i^{rem_{h(j)}} = -1 \\ L_i^{rem_h} - 1, & \text{if } h = h(j) \text{ and } L_i^{rem_{h(j)}} > 0 \\ L_i^{rem_h}, & \text{otherwise} \end{cases} \qquad \forall h \in H \tag{4c}$$

$$L_j^{visit_v} = \begin{cases} L_i^{visit_v} + 1, & \text{if } v = j \\ L_i^{visit_v}, & \text{otherwise} \end{cases} \qquad \forall v \in V \setminus \{0\} \tag{4d}$$

The condition $L_i^{rem_{h(j)}} = -1$ in (4a), (4b), and (4c) tests whether the next visit to vertex j is one to a non-visited cluster $V_{h(j)}$. Accordingly, the dual prices $\pi_{h(j)}$ and demands $d_{h(j)}$ are incorporated only if $V_{h(j)}$ has not been visited yet.

The new label L_j for $j \in V \setminus \{0, 0'\}$ and the associated partial path $(p, j) = (0, \dots, i, j)$ is feasible if and only if

$$L_j^{load} \leq Q \qquad \text{and} \qquad L_j^{visit_j} \leq 1, \forall j \in V \setminus \{0\}. \tag{5a}$$

For an extension to the destination depot $j = 0'$, the additional feasibility conditions

$$L_j^{rem_h} \leq 0 \qquad \forall h \in H \tag{5b}$$

are needed to guarantee that no cluster is served incompletely.

Weak Dominance. Let L and L' be two labels of different partial paths p and p', respectively, that end at the same vertex i. Domination of labels generally uses the following *auxiliary criterion*: If for each feasible completion q' of p' into a feasible 0-0$'$-path $r' = (p', q')$ there exists a feasible completion q of p into a feasible 0-0$'$-path $r = (p, q)$ and the reduced cost of r is not greater than the reduced cost of r', then label L' is *dominated* by L. A dominated label can be discarded (if the dominating label is kept).

For many VRP variants, a simplified criterion is applied by choosing the completion q identical to the completion q'. Using this simplified criterion is unnecessarily restrictive for the SoftCluVRP as it is also too restrictive in VRPs with pickup-and-delivery structure (see discussion in Section 3.1.5). However, the following weak dominance rule can be directly derived from the simplified criterion:

Rule 1. *(Weak Dominance Rule) Let L and L' be two labels of different partial paths that end at the same vertex i. Label $L = (L^{cost}, L^{load}, L^{rem}, L^{visit})$ dominates label $L' = (L'^{cost}, L'^{load}, L'^{rem}, L'^{visit})$ if all of the following conditions are fulfilled:*

$$L^{cost} \leq L'^{cost} \tag{6a}$$

$$L^{load} \leq L'^{load} \tag{6b}$$

$$\left\{ \begin{array}{lll} (L^{rem_h} = L'^{rem_h} & \text{and} & L_i^{visit_v} = L_i'^{visit_v}, \forall v \in V_h) \\ \text{or } (L^{rem_h} = -1 & \text{and} & L'^{rem_h} = 0) \end{array} \right\} \qquad \forall h \in H \tag{6c}$$

The conditions (6c) ensure that L_i dominates $L_i{}'$ only if either (i) both partial paths have visited exactly the same vertices $v \in V_h$ of a cluster or (ii) path p has not visited a cluster V_h that is already completely served by p'.

Example 1. *We consider a SoftCluVRP instance and two routes that visit only one cluster $V_1 = \{1, 2, 3\}$, but customers are visited in different sequences. The relevant part of the (symmetric) cost matrix (c_{ij}) is:*

c_{ij}	$0'$	1	2	3
0	–	4	3	4
1	4	–	4	3
2	3	4	–	3
3	4	3	3	–

Let the two routes be $r = (0, 1, 3, 2, 0')$ and $r' = (0, 1, 2, 3, 0')$. Moreover, we assume that the dual prices π_1 and μ are zero because they are irrelevant for the exposition. Figure 1 depicts the propagation of the attributes L^{cost}, L^{rem_h}, and L^{visit_v} along the routes. The attribute L^{load} is only altered along arc $(0, 1)$ and not presented in order to keep the example as small as possible.

Note first that both routes share the same partial path $(0, 1)$, for which only one label is created, so that there is no dominance between subpaths of r and r' at vertex 1. In contrast, there are two labels at vertex 2, say L_2 and L_2' but with the weak dominance rule neither label dominates the other one. The same holds at vertex 3 for the corresponding labels. However, at the destination vertex $0'$, the label of r dominates the label of r' due to its better cost but otherwise identical attributes.

Strong Dominance. A stronger dominance can be achieved if the distance matrix (c_{ij}) respects the *triangle inequality* (TI), i.e., for any three different vertices $i, j, k \in V$ the inequality $c_{ik} \leq c_{ij} + c_{jk}$ holds.

The idea is now that if the TI holds, no path completion can benefit from visiting an additional customer. More precisely, let

$$R_h = \{i \in V_h : L^{rem_h} > 0, L^{visit_i} = 0\} \tag{7}$$

be the vertices in V_h that *must* be visited by any completion of path p corresponding with L (due to conditions (5b)). Similarly, we define $R_h' = \{i \in V_h : L'^{rem_h} > 0, L'^{visit_i} = 0\}$ as the vertices that *must* be

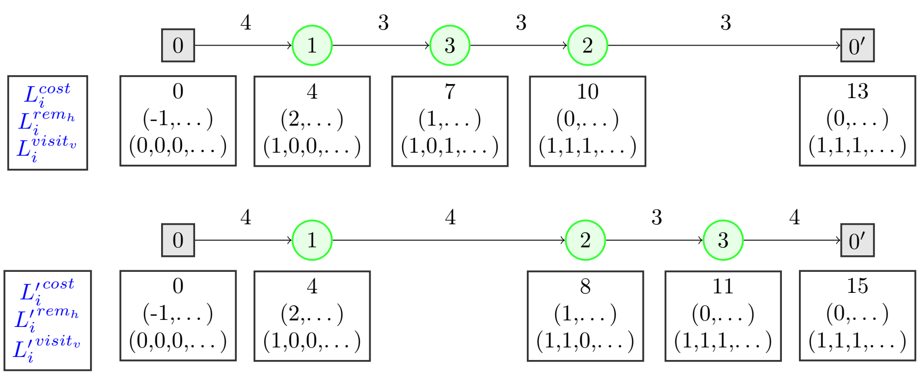

Figure 1: Propagation of the attributes for two routes $r = (0, 1, 3, 2, 0')$ and $r' = (0, 1, 2, 3, 0')$ serving the same cluster $V_1 = \{1, 2, 3\}$

visited by any completion of path p' corresponding with L'. If $R_h \subseteq R'_h$, then every feasible extension of p' must visit additional customers compared to a corresponding feasible extension of p. Therefore, the strong dominance rules is:

Rule 2. *(Strong Dominance Rule) Let L and L' be two labels of different partial paths that end at the same vertex i. Label $L = (L^{cost}, L^{load}, L^{rem}, L^{visit})$ dominates label $L' = (L'^{cost}, L'^{load}, L'^{rem}, L'^{visit})$ if all of the following conditions are fulfilled:*

$$\text{(6a) } \textit{and } \text{(6b)} \tag{8a}$$

$$\left\{\begin{array}{lll} & (L^{rem_h} \le L'^{rem_h} & \textit{and} \quad L^{visit_v} \ge L'^{visit_v}, \forall v \in V_h) \\ \textit{or} & & L^{rem_h} = -1 \end{array}\right\} \qquad \forall h \in H \tag{8b}$$

Proof. The proof relies on the above mentioned auxiliary criterion. The partial path associated with L (L') is denoted by p (p'). Let q' be an arbitrary feasible extension of p' into a feasible route $r' = (p', q')$. We have to show that there exists a feasible extension q of p so that the route $r = (p, q)$ is feasible and fulfills $\tilde{c}_r \le \tilde{c}_{r'}$.

Let q be the path that results from the removal of all vertices $\{v \in V : L^{visit_v} = 1, L'^{visit_v} = 0\} \cup \{v \in V : L^{rem_{h(v)}} = -1, L'^{rem_{h(v)}} > 0, L'^{visit_v} = 0\}$ from q'. The REFs (4) and feasibility conditions (5) guarantee that (p, q) is feasible. Indeed, every cluster that is only partly served with p is at the end served completely with (p, q).

Moreover, the TI ensures that the routing cost of q is not greater than the routing cost of q'. Finally, all clusters completely served with q' are also completely served with q. Consequently, both extensions q and q' collect exactly the same dual prices $\pi_{h(j)}$ in (4a). Therefore, the reduced cost of $r = (p, q)$ is not greater than the reduced cost of $r' = (p', q')$, i.e., $\tilde{c}_r \le \tilde{c}_{r'}$. □

Weak dominance (6) allows domination only for labels with $R_h = R'_h$ for all $h \in H$. In contrast, strong dominance (8) allows domination also if $R_h \subseteq R'_h$ holds.

Example 2. (cont'd from Example 1) *The stronger dominance Rule 2 is not imposing additional dominance relations in the situation of Example 1. Indeed, while L_2 and L'_2 fulfill conditions* (8b) *and* (6b)*, they fail on the reduced cost condition* (6a)*.*

However, the situation can change with another cost matrix. Assume that instead of (c_{ij}) the cost matrix were $(\dot{c}_{ij})$ defined by:

$\dot{c}_{ij}$	$0'$	1	2	3
0	–	3	2	3
1	3	–	2	1
2	2	2	–	1
3	3	1	1	–

The result of this re-definition is that now both labels L_2 and L'_2 have identical reduced costs $L_2^{cost} = {L'}_2^{cost} = 5$ so that L_2 dominates L'_2 by Rule 2. We discuss in Section 3.1.4 how a systematic manipulation of the cost matrix can be exploited as an acceleration technique.

3.1.2. Bidirectional Labeling

Righini and Salani (2006) coined *bounded bidirectional labeling* for SPPRCs. Numerous subsequent works have shown that bidirectional labeling algorithms are usually superior to their monodirectional counterparts. Accordingly, bidirectional labeling has become a quasi-standard for solving SPPRCs. In addition, the two more recent works (Tilk *et al.*, 2017; Gschwind *et al.*, 2017) have shown that bidirectional labeling can significantly benefit from a dynamically chosen half-way point which exploits the inherent asymmetry on many SPPRC instances. We briefly explain the ideas of bidirectional labeling in the following.

Bidirectional labeling requires the definition of a monotone resource, i.e., an attribute L^{res} to be used for bounding the propagation of labels in both directions. More precisely, for labels L_{fw} propagated in forward direction, the respective attribute L^{res}_{fw} is increasing, and for labels L_{bw} propagated in backward direction, the attribute L^{res}_{bw} is decreasing along the path. Moreover, for any feasible 0-0′-path (p, q) with $p = (0, \dots, i)$ and associated forward label L_{fw} and $q = (i, \dots, 0')$ with associated backward label L_{bw}, the attributes must fulfill $L^{res}_{fw} \leq L^{res}_{bw}$. The bounding component of the bidirectional labeling algorithm uses a so-called *half-way point* HWP und only propagates forward labels with $L^{res}_{fw} \leq HWP$ and backward labels with $L^{res}_{bw} > HWP$. Examples of such monotone forward and backward resources are earliest and latest service times for VRPs with time windows, or the accumulated demand and the residual capacity for capacitated VRPs.

In order to not generate multiple copies of the same route (a non-trivial route r has multiple representations $r = (p, q) = (p', q')$ where p is a subpath of p' and q' a subpath of q, or vice versa), the *merge criterion* also relies on the half-way point: For the merge of two labels L_{fw} and L_{bw}, either L_{fw} must represent a full 0-0′-path with $L^{res}_{fw} \leq HWP$ (so that L_{bw} represents the trivial path $(0')$) or $L^{res}_{fw} > HWP$ is required.

Furthermore, if there exists a strictly monotone resource, the sequence of label propagation can be fully controlled with the help of buckets (for details we refer to Tilk *et al.*, 2017). However, none of our attributes propagated via (4) is strictly monotone. Hence, we consider the *divided demand* calculated by $d^{div}_j = d_{h(j)}/n_{h(j)}$ and introduce a new attribute L^{div}. For a route r that respects the soft-cluster constraints, the conditions $\sum_{h \in H(r)} d_h \leq Q$ is equivalent to $\sum_{i \in V(r)} d^{div}_i \leq Q$. Thus, the corresponding REF to propagate the divided demand from a label L_i ending at vertex i along arc (i, j) is $L^{div}_j = L^{div}_i + d^{div}_j$. Note that this attribute is strictly increasing and can therefore be used as the monotone resource.

The SoftCluVRP has a subproblem that is completely symmetric, i.e., any path is feasible if its reversed path is feasible, and vice versa. The consequence for the bidirectional labeling is therefore that only forward partial paths need to be generated and considered. The half-way point is set to $HWP = Q/2$ and we propagate forward labels L_i only if $L^{div}_i \leq HWP$. Any feasible backward partial path $p_{bw} = (i, j, \dots, 0')$ exists as a feasible forward partial path $p_{fw} = (0, \dots, j, i)$. For convenience, we define a *reversal operator* rev with $\text{rev}(p_{fw}) = p_{bw}$ and $\text{rev}(p_{bw}) = p_{fw}$, i.e., it reverses the partial path and exchanges 0 and 0′. In the final merge step, forward labels are then merged with other forward labels. Such an implicit bidirectional labeling has already been successfully implemented and applied by Bode and Irnich (2012) and Goeke *et al.* (2017).

Merge Procedure. We now present the merge condition that tests whether two labels L and L' representing the forward paths p and p' from 0 to a customer vertex $i \in V \setminus \{0\}$ can be combined and produce a feasible 0-0′-path $r = (p, \text{rev}(p'))$. It very much simplifies the merge condition to consider the predecessor label of one of the two label. Thus, let L_- be the predecessor label of L, i.e., L results from the extension of L_- along an arc (j, i). The route r is feasible if

$$L^{load}_- + {L'}^{load} - \sum_{h \in H: L^{rem_h}_- \geq 0, {L'}^{rem_h} \geq 0} d_h \leq Q \tag{9a}$$

(which is equivalent to $L_-^{div} + L'^{div} \leq Q$)

$$\left\{\begin{array}{rcll} L_-^{rem_h} & \leq & 0, & \text{if } L'^{rem_h} = -1 \\ L_-^{rem_h} & = & -1, & \text{if } L'^{rem_h} = 0 \\ L_-^{rem_h} + L'^{rem_h} & = & |V_h|, & \text{otherwise} \end{array}\right\} \qquad \forall h \in H \tag{9b}$$

$$L_-^{visit_v} + L'^{visit_v} \leq 1, \qquad \forall v \in V \tag{9c}$$

where (9a) checks the capacity constraint, (9b) that no cluster is served incompletely, and (9c) that no customer is visited more than once. (When storing unreachable instead of visited customers, see Section 3.1.1, one direction must however rely on the actually visited customers.) The cost of the resulting 0-0′-path $r = (p, \text{rev}(p'))$ is given by

$$\tilde{c}_r = L^{cost} + L'^{cost} + \sum_{h \in H: L^{rem_h} \geq 0, L'^{rem_h} \geq 0} \pi_h. \tag{10}$$

Note that the rightmost sum in (9a) and (10) considers the clusters that are incompletely served in both forward and backward direction. For these clusters, the demands d_h must be subtracted in (9a) and dual prices π_h must be added in (10) to prevent the incorrect double incorporation.

Dominance. The following proposition shows that a bidirectional labeling algorithm is correct in two cases: (i) the weak dominance rule is applied or (ii) the strong dominance rule is applied in combination with a cost matrix that fulfills the *strict triangle inequality* (STI), i.e., for any three different vertices $i, j, k \in V$ the strict inequality $c_{ik} < c_{ij} + c_{jk}$ holds.

Proposition 1. *A bidirectional labeling algorithm that*
(i) either uses the weak dominance (Rule 1)
(ii) or the strong dominance (Rule 2) on a cost matrix that respects the STI
and the described merge procedure with conditions (9) *finds an optimal solution to the pricing subproblem of the SoftCluVRP.*

Proof. Let r be an optimal path, i.e., an elementary, feasible 0-0′-path with minimum reduced cost. We have to show that the bidirectional labeling algorithm either finds this path or one with the same reduced cost.

The forward labeling algorithm is certainly correct with either dominance, weak or strong, see Section 3.1.1. This has two consequences: First, all Pareto-optimal labels that arrive at the destination depot 0′ with divided demand $\leq HWP$ are found in the forward part of the bidirectional labeling. Hence, we can restrict the remainder of the proof to cases where all optimal routes r result from a merge. With these assumptions, r can be represented as $r = (p, \text{rev}(q))$, where p and q both end at a merge vertex i.

Second, it can be assumed that $r = (p, \text{rev}(q))$ is generated by the complete forward labeling algorithm (otherwise there would exist another route fulfilling this). Therefore, the path p and its label L are considered in the merge of the bidirectional labeling algorithm. If the label L' of q would also be available in the merge, nothing remains to show.

Hence, we can now assume that L' is not generated, or it was generated and dominated. Then, there must exist a label L'^* that dominates L'. If $L'^* = L'$ (identical attributes), the path q^* associated with L'^* would produce the route $(p, \text{rev}(q^*))$ with identical reduced cost as r because of (10). A route equivalent to r is constructed.

Therefore, we can assume $L'^* \neq L'$. The proof now considers the two preconditions separately:

Case (i): We assume that the weak dominance (Rule 1) is applied. The weak dominance implies either $L'^{*cost} < L'^{cost}$ or that some clusters completely served with q are not touched by q^* (or both). Then, the pair L and L'^* also qualifies for the merge. The resulting 0-0′-path $(p, \text{rev}(q^*))$ has a reduced cost not greater than r because $L'^{*cost} \leq L'^{cost}$ and the (reverse) completion by p of both labels L' and L'^* produce the same sum of dual prices π_h in (4a). Hence, another route equivalent to r is constructed.

Case (ii): We assume that the strong dominance (Rule 2) is applied and that the cost matrix (c_{ij}) respects the STI. The two labels cannot differ only in cost, i.e., $L'^{*cost} < L'^{cost}$, because then the pair L and L'^{*} also qualifies for the merge and produces a route with smaller reduced cost than r, a contradiction!

Now, the only possibility left is that the two labels must also differ in the visit and remaining attributes. Recall that q^* and q are the partial paths associated with L'^* and L', respectively. As a backward path, p is a feasible (reverse) completion of q. As in the proof of Rule 2, there also exists a feasible completion p^* of q^* that results from path p by eliminating some vertices. (In our case, the different visit attributes ensure that at least one vertex is actually eliminated.) The STI now ensures that the (reduced) cost of this alternative completion p^* is strictly smaller than the (reduced) cost L^{cost} of p. As a consequence, the feasible route (p^*, q^*) has a strictly smaller reduced cost compared to $r = (p, q)$, again a contradiction! □

The use of the strong dominance (Rule 2) on a cost matrix that does not respect the STI but the TI may lead to incorrect results of the bidirectional labeling algorithm. This is shown in following example.

Example 3. *We consider a SoftCluVRP instance with only one cluster $V_1 = \{1, 2\}$ with demand $d_1 = 6$ and vehicle capacity $Q = 10$. The divided demand of customers 1 and 2 is $d_1^{div} = d_2^{div} = 3$ and the half-way point is $HWP = 5$. The following cost matrix*

c_{ij}	$0'$	1	2
0	–	1	2
1	1	–	1
2	2	1	–

respects the TI but not the STI due to $c_{02} = c_{01} + c_{12}$. Furthermore, we assume the dual prices to be $\pi_1 = 2$ and $\mu = 4$.

The bidirectional labeling algorithm creates labels for the partial paths (0), $(0, 1)$, $p = (0, 1, 2)$, and $p' = (0, 2)$. However, no complete route is created with forward labeling due to the half-way point condition. Hence, routes can only result from a merge.

The only dominance between labels occurs for p and p' and the associated labels $L = (L^{cost}, L^{load}, L^{rem_h}, L^{visit_v}) = (-2, 6, 0, (1, 1))$ and $L' = (L'^{cost}, L'^{load}, L'^{rem_h}, L'^{visit_v}) = (-2, 6, 1, (0, 1))$, where L dominates L' using Rule 2. Thus, after applying a dominance algorithm, L' is discarded. The consequence is that no extension of L' is created either, in particular no label for the partial path $(0, 2, 1)$. Finally, the merge procedure does not find any feasible combination of labels.

However, the route $r = (0, 1, 2, 0') = (p, rev(p'))$ has cost 4 and reduced cost $\tilde{c}_r = 4 - \pi_1 - \mu = -2 < 0$. This shows that the bidirectional labeling with strong dominance may fail to produce optimal or even feasible solutions if the STI does not hold.

3.1.3. Manipulation of the Cost Matrix

Some of the SoftCluVRP instances that we use in the later computational analysis do not respect the STI or even the TI. In order to apply bidirectional labeling algorithms, we transform these instances into equivalent new instances only differing in the cost matrix (c_{ij}) but not in optimal solutions. As every customer $k \in V \setminus \{0\}$ must be visited exactly once, it is possible to add any value $x = x(k) \in \mathbb{R}$ to all edges $\delta(k)$. The cost of all feasible solutions then increases by $2 \cdot x(k)$. This procedure is summarized as Procedure `Manipulate` that also records the modification in the customer-indexed array `mod`$[k]$.

If a SoftCluVRP instance does not respect the STI, we preprocess the instance with Algorithm 1. The first loop (Steps 1 to 5) resets the accumulated cost modification `mod`$[j]$ for each customer $j \in V \setminus \{0\}$ to zero, computes the largest violation of the TI when j is the middle vertex, and adds half of the maximum violation vio to the jth column and the jth row of the cost matrix. As we use integer arithmetics for the routing costs $\lceil vio/2 \rceil + 1$ guarantees that the new matrix also comprises only integer values. After Step 5 the resulting cost matrix (c_{ij}) already respects the STI.

It can however happen that, for some customers $j \in V \setminus \{0\}$, the STI is fulfilled with a rather large slack. The strong dominance (see Rule 2) however benefits from a tightly fulfilled STI. Indeed, the chance to have

Procedure `Manipulate`

```
Input: A customer $k \in V \setminus \{0\}$, a value $x \in \mathbb{R}$
Output: Modified cost matrix $(c_{ij})$ and accumulated cost modification mod[$k$]
1 for $j \in V, j \neq k$ do
2     $c_{jk} := c_{jk} + x$
3     $c_{kj} := c_{kj} + x$
4 mod[$k$] := mod[$k$] $+ 2x$
```

a smaller reduced cost while having more customers visited increases with smaller routing costs. Therefore, the optional loop in Steps 7 to 13 iteratively decreases entries of the cost matrix as long as possible. We analyze the impact of this reduction (`reduce` = `true`) on the overall performance in the computational results section.

Algorithm 1: `Preprocessing( reduce )`

```
Input: Flag reduce
Output: Modified cost matrix $(c_{ij})$ and accumulated cost modification mod[$k$]
1 for $j \in V \setminus \{0\}$ do
2     mod[$j$] := 0
3     $vio := \max_{i,k \in V, i \neq j \neq k, i \neq k}(c_{ik} - c_{ij} - c_{jk})$
4     if $vio \geq 0$ then
5         Manipulate($j$, $\lceil vio/2 \rceil + 1$)
6 update = reduce
7 while $update$ do
8     $update$ := FALSE
9     for $j \in V \setminus \{0\}$ do
10        $slack := \min_{i \in V, i \neq j}\{c_{ij}; \min_{k \in V, i \neq k \neq j}(c_{ij} + c_{jk} - c_{ik})\}$
11        if $slack > 2$ then
12            Manipulate($j$, $-\lceil slack/2 \rceil + 1$)
13            $update$ := TRUE
```

The final cost matrix (c_{ij}) computed by Algorithm 1 still respects the STI, the values `mod`$[j]$ give the overall modification for each customer j, and the cost of all feasible solutions increases by the constant $C := \sum_{j \in V \setminus \{0\}} \texttt{mod}[j]$.

Example 4. (cont'd from Examples 1 and 2) *Recall that Example 1 gave an example where the strong dominance rule (Rule 2) was not applicable at vertex 2, because label L had higher reduced cost than the otherwise better label L'. Example 2 replaced the cost matrix (c_{ij}) of Example 1 by another cost matrix $(\dot{c}_{ij})$ so that the two labels' cost became identical and the strong dominance rule became applicable.*

The point is that matrix $(\dot{c}_{ij})$ results from matrix (c_{ij}) by subtracting -1 for each customer $j \in V \setminus \{0\}$. As a result, all edges $\{0, j\} \in \delta(0)$ have $\dot{c}_{0j} = c_{0j} - 1$, while the edges $\{i, j\} \in E \setminus \delta(0)$ connecting two customers i and j have $\dot{c}_{0j} = c_{0j} - 2$.

3.1.4. Heuristic Pricing and Acceleration Techniques

We now discuss three techniques that can be used to speed up the labeling algorithm: (1) the systematic violation of the triangle inequality, (2) decremental state space relaxation and *ng*-path relaxation, and (3) the use of reduced networks.

Systematic Violation of the Triangle Inequality. The previous Section 3.1.3 has shown that a systematic manipulation (reduction) of the cost matrix can be used to create equivalent SoftCluVRP instances that may have a higher chance to exploit the strong dominance rule (Rule 2). This chance can be increased even more, if we do not consider the STI to be fulfilled after reducing the cost matrix. We determine the minimal entry $c_{min} = \min_{i,j \in V, i \neq j} c_{ij}$ in the cost matrix (c_{ij}) derived by Section 3.1.3 and calculate $offset = \lceil c_{min}/2 \rceil - 1$. Then, `Manipulate`$(j, -offset)$ is performed for each customer $j \in V \setminus \{0\}$. The resulting cost matrix $(\tilde{c}_{ij})$ possibly violates the (S)TI, but can be used together with strong dominance (Rule 2) as heuristic pricing.

Decremental State Space Relaxation and ng-Path Relaxation. General elementary SPPRCs are NP-hard in the strong sense (Dror, 1994). The idea of a *decremental state space relaxation* (DSSR, Righini and Salani, 2008) is therefore to solve less difficult relaxations (less difficult in practice or in theory such as pseudo-polynomial relaxations). The relaxations must therefore be parametrizable so that a strongest relaxation guarantees elementary paths. One starts however with a weaker but relatively well-solvable SPPRC relaxation. If the solution of this relaxation contains a cycle, a stronger relaxation must be chosen and solved instead. The iterative process ends if an optimal solution to the relaxation is elementary (or just one with negative reduced costs).

We adopt the *ng*-path relaxation of Baldacci *et al.* (2011) to the SoftCluVRP. Let the subset $N_i \subseteq V \setminus \{0\}$ be the vertex-specific *ng*-neighborhood of each vertex $i \in V$. The *ng*-path relaxation for $(N_i)_{i \in V}$ results from altering the REF (4d) of the visit attributes in the following way:

$$L_j^{visit_v} = \begin{cases} 0 & \text{if } v \notin N_j \\ L_i^{visit_v} + 1, & \text{if } v = j \text{ and } v = N_j \\ L_i^{visit_v}, & \text{if } v \neq j \text{ and } v = N_j \end{cases} \qquad \forall v \in V \setminus \{0\} \tag{11}$$

As a result, a feasible route may visit a customer more than once. However, the REF (4c) and conditions (5b) ensure that there are either exactly n_h visits to a cluster $V_h, h \in H$, or no visit at all.

There are several issues however related to the dominance rules introduced before. Already in monodirectional labeling (Section 3.1.1), the use of the strong dominance rule with a proper *ng*-relaxation can lead to undesirable results: A dominated label L' (Rule 2) can have feasible extensions that are however infeasible for the dominating label L.

Example 5. *We consider an example with two clusters* $V_1 = \{1, 2, 3\}$ *and* $V_2 = \{4, 5\}$*. Let* L_2 *be the label for partial path* $p = (0, 1, 3, 4, 2)$ *and* L'_2 *for the partial path* $p' = (0, 4, 1, 2)$*. Since* p *has visited a superset of the customers compared to* p'*, domination with Rule 2 is possible.*

Now consider the extension $q' = (2, 3, 4, 0')$ *of* p'*. It produces a non-elementary path* $r' = (p', q') = (0, 4, 1, 2, 3, 4, 0')$*. If* $N_1 = N_2 = N_4 = N_5 = V \setminus \{0\}$ *and* $N_3 = \{1, 2, 3, 5\}$*, then* r' *is a feasible ng-route in the sense of REFs* (11) *and conditions* (5)*.*

However, the associated extension $q = (2, 4, 0')$ *of* p*, that result from the removal of the already visited customer* 3 *from* q'*, produces the route* $r = (0, 1, 3, 4, 2, 4, 0')$ *that is infeasible w.r.t. the above ng-neighborhoods. In summary,* L'_2 *is dominated but one of its extensions cannot be used to extend the dominating label* L_2*. If the route* r' *were optimal for the ng-path relaxation, the labeling algorithm with this strong dominance would not find* r'*.*

Such a behavior was first observed for VRPs with pickup-and-delivery (P&D) structure by Cherkesly *et al.* (2015). In essence, non-elementary paths can be incorrectly dominated and lower bounds computed with these relaxations are not unique. The bounds depend on the sequence of label extensions and dominance test. However, these bounds are valid as exploited in the selective pricing paradigm of Desaulniers *et al.* (2017).

As we want to use the *ng*-path relaxations in DSSR and bidirectional labeling, we will not use the strong dominance Rule 2. Note that intentionally the work of Gschwind *et al.* (2017) does not present an *ng*-path relaxation for a P&D-tailored bidirectional labeling algorithm with strong dominance, because the validity of this combination is to date unclear. In line with these remarks, we now present another dominance rule, tailored for *ng*-path relaxations, for which correct domination can be shown for both monodirectional and bidirectional labeling.

Rule 3. *(ng-Dominance Rule) Let L and L' be two labels of different partial paths that end at the same vertex i. Label $L = (L^{cost}, L^{load}, L^{rem}, L^{visit})$ dominates label $L' = (L'^{cost}, L'^{load}, L'^{rem}, L'^{visit})$ if all of the following conditions are fulfilled:*

$$\text{(6a) } \textit{and } \text{(6b)} \tag{12a}$$

$$\left\{ \begin{array}{lll} (L^{rem_h} = L'^{rem_h} & \textit{and} & L_i^{visit_v} = L_i'^{visit_v}, \forall v \in V_h) \\ \textit{or } (L^{rem_h} \leq 0 & \textit{and} & L'^{rem_h} \geq 0) \end{array} \right\} \qquad \forall h \in H \tag{12b}$$

Proof. Similar to Proof of Rule 2 and Proof of Proposition 1. □

Note that the *ng*-specific Rule 3 applies a stronger criterion than Rule 1 but a weaker criterion than Rule 2.

Heuristic/Partial Pricing using Reduced Networks. Using reduced networks is a standard technique to speed-up the labeling algorithm (see, e.g., Dumas *et al.*, 1991). The elementary SPPRC (ESPPRC) is solved on an incomplete subgraph $\tilde{G} = (V', \tilde{A})$ with $\tilde{A} \subset A$. For the SoftCluVRP we characterize the subgraph by the non-negative integer parameter σ and define $\tilde{A}$ by

(i) all arcs connecting the origin or destination depot with customer nodes: $(i, j) \in A$ with $i = 0$ or $j = 0'$,
(ii) all intra-cluster arcs: $(i, j) \in A$ with $h(i) = h(j)$,
(iii) and σ inter-cluster arcs for every node $i \in V \setminus \{0\}$, connecting i to its σ nearest neighbors $j \notin V_{h(i)}$: $(i, j) \in A$ with $h(i) \neq h(j)$ and minimal routing costs c_{ij}.

A hierarchy of pricing heuristics can be used to iteratively solve the ESPPRC on subgraphs build by increasing parameter σ, until a route with negative reduced cost is found. Note that the complete graph G' must be used in the last iteration in order to find the exact solution (and possibly proof that no route with negative reduced cost exists).

3.1.5. Comparison with Pickup-and-Delivery Problems

SoftCluVRP	vs.	VRPs with P&D structure
Similarities:		
cluster V_h	$\sim$	request $\{i^+, i^-\}$
remaining customers	$\sim$	open requests
strong dominance based on subsets of remaining customers	$\sim$	strong dominance based on subsets of open requests
strong dominance and *ng*-path: incorrectly dominated non-elementary paths	$\sim$	strong dominance and *ng*-path: incorrectly dominated non-elementary paths (Cherkesly *et al.*, 2015)
Differences:		
no precedences	$\neq$	i^+ precedes i^-
dual prices for covering clusters are managed via REFs	$\neq$	dual prices for covering requests are incorporated into reduced cost matrix ($\tilde{c}_{ij}$)
strong dominance requires triangle inequality (TI) on original cost matrix (c_{ij})	$\neq$	strong dominance requires *delivery triangle inequality* (DTI) on reduced cost matrix ($\tilde{c}_{ij}$) (Ropke and Cordeau, 2009)
bidirectional labeling requires strict triangle inequality (STI)	$\neq$	bidirectional labeling requires two matrices (fw/bw with DTI/pickup TI) (Gschwind *et al.*, 2017)

Table 1: Similarities and differences between SoftCluVRP and VRPs with P&D structure, in particular regarding labeling algorithms for the SPPRC subproblem.

We would like to point out that the SoftCluVRP shares some similarities with VRPs that have a pickup-and-delivery (P&D) structure. In the latter problems, the task is to fulfill a set of transportation requests, where each request i consists of the collection of some item(s) from a given pickup point i^+, the possibly shared transportation with other items, and the delivery of the item(s) to a given delivery point i^-. The basic observation is that clusters V_h of customers in the SoftCluVRP correspond to requests $i = \{i^+, i^-\}$ in P&D VRPs. More similarities but also the most important differences in these VRP variants and their SPPRC labeling subproblems (arising from their exact solution via column-generation approaches) are summarized in Table 1.

3.2. Branch-and-Cut

Our branch-and-cut algorithm for the SoftCluVRP subproblem is based on the formulation (3) and uses the callable library of CPLEX 12.8.1.0. for so-called lazy cuts and user cuts. We have kept all default settings of CPLEX except for enforcing CPLEX to run in single-thread mode. The initial linear program (LP) comprises the objective (3a), the depot degree constraints (3b), the coupling constraints (3c), and the capacity constraint (3e).

In the following we assume that all direct routes $r = (0, i, 0')$ for singleton clusters $V_h = \{i\}$ are already in the RMP. Therefore, these routes do not have to be priced out. The consequence is that we do not have to distinguish between binary and integer routing variables (see (3f) and (3g)), but all routing variables are binary.

For the detection of violated *subtour-elimination constraints* (SECs) we added callbacks to CPLEX. Let $(\bar{x}_e, \bar{z}_h)$ be a solution to the LP. We define the *support graph* to $(\bar{x}_e, \bar{z}_h)$ as the graph $\bar{G} = (V, \bar{E})$ where the edge set is defined as $\bar{E} = \{e \in E : \bar{x}_e > 0\}$. The separation algorithm distinguishes between integer and fractional solutions.

For an integer solution $(\bar{x}_e, \bar{z}_h)$, we determine the connected components of $\bar{G}$. Our implementation uses an efficient implementation of a union-find algorithm (Tarjan, 1979). If a connected component induced by $S \subset V$ fulfills $0 \notin S$ and $|S| > 1$, violated SECs are found. Indeed, for all $i \in S$, the SEC to the pair (i, S) is violated (LHS is zero, RHS is two). We add only one SEC per subset S choosing i arbitrarily.

For fractional solutions $(\bar{x}_e, \bar{z}_h)$, we first also compute the connected components of $\bar{G}$. For a connected component induced by $S \subset V$ that fulfills $0 \notin S$ and $|S| > 1$, we determine $\bar{\imath} = \arg\max_{\bar{\imath} \in S} \bar{z}_{h(\bar{\imath})}$. If the SEC for the pair $(\bar{\imath}, S)$ is violated by more than a given threshold $\epsilon := 0.01$, we add the violated SEC. If none of the connected components with $0 \notin S$ gives a violated SEC, we analyze the connected component that contains the depot 0. Let $S_0 \subset V$ be the subset that induces this component. We next compute the maximum flow between the depot 0 and every vertex $i \in S_0, i \neq 0$ in the induced graph $G[S_0]$. These max-flow problems are solved with the algorithm of Boykov and Kolmogorov (2004) (available in the BOOST C++ library, `https://www.boost.org/doc/libs/1_67_0/libs/graph/doc/boykov_kolmogorov_max_flow.html`). Let the maximum-flow value be $\bar{f}_{0i}$. The degree of violation of the SEC for the (i, S_0) is $2\bar{z}_{h(i)} - \bar{f}_{0i}$. If several violations greater than ϵ exist, we chose $\bar{\imath}$ as one vertex that maximizes the violation and add the violated SEC for the pair $(\bar{\imath}, S_0)$ to the LP.

Note that we do not add separation procedures for other classes of valid inequalities such as, e.g., cover inequalities/cuts (Wolsey, 1998, p. 147f) induced by the knapsack-like constraints (3e) because these cuts are already implemented in CPLEX.

3.3. Primal Heuristic

We now present a metaheuristic for heuristic/partial pricing that systematically manipulates a given feasible route r using edge-exchange procedures for the TSP and additional operators that can drop the vertices of a cluster V_h with $h \in H(r)$ or add the vertices of a cluster V_h with $h \in H \setminus H(r)$ to the route r. The different operators are combined in variable neighborhood decent (VND) procedures (a variation of variable neighborhood search, see Mladenović and Hansen, 1997). We use the following basic operators:

`DoBest2Opt`(r):	Search for a best-improving 2-OPT TSP move in r and perform this move if it is improving.
`DoBest3Opt`(r):	The same, but with 3-OPT TSP moves.
`VND.TSP`(r):	Perform a VND with the operators `DoBest2Opt` and `DoBest3Opt` on r.
`DropCluster`(r, h):	Remove all vertices $i \in V_h$ from r.

`AddCluster`(r,h):	Check whether the addition of the cluster V_h to r is feasible. If so, loop over all $i \in V_h$ and insert i into r at a position with smallest insertion cost. Otherwise leave r unchanged.
`DoBestDropCluster`(r):	Loop over all $h \in H(r)$, make a copy r' of r, apply `DropCluster`(r',h) and `VND.TSP`(r'), and compute the reduced cost of the resulting route r''. Finally, set r to the route r'' with minimum reduced cost if $\tilde{c}_{r''} < \tilde{c}_r$, i.e., if r'' is improving. Otherwise leave r unchanged.
`DoBestAddCluster`(r):	The same, but with a loop over all $h \in H \setminus H(r)$ and with `AddCluster` moves.
`VND`(r):	Perform a VND with the operators `DoBestDropCluster` and `DoBestAddCluster` on r.

Our implementation of the best-improvement 2-OPT and 3-OPT local search procedures uses a so-called *radius* or *sequential search* mechanism to reduce the computational effort of local search for the quadratic and cubic neighborhoods, see (Bentley, 1992), (Hoos and Stützle, 2004, p. 373) and (Irnich *et al.*, 2006).

The staring point of the primal heuristic is the primal solution $\bar{y}_r$, $r \in \bar{\Omega}$ of the RMP (2). Note that all routes with $\bar{y}_r > 0$ have reduced cost $\tilde{c}_r = 0$, so that they are promising starting solutions. We loop over all these routes and apply the primal heuristic `PrimalHeuristic`(r), detailed in Algorithm 2, to each of them. In the primal heuristic, the loop (Steps 3 to 12) first tries to improve the current route r' by applying the `VND` with all operators, i.e., 2-OPT, 3-OPT, removal of a cluster, and addition of a cluster (in this order of increasing search effort). Then, it randomly removes up to three clusters from the resulting route r' (Steps 10 to 12). The loop is repeated up to 140 times, but a premature termination happens if the best found route r^* has negative reduced cost. All negative reduced-cost routes that are found are added to the RMP.

Algorithm 2: `PrimalHeuristic`(r) for the SoftCluVRP pricing subproblem

```
   Input: A feasible route $r$
   Output: A negative reduced-cost route $r^*$ or FAILED if none is found
1  $r^* := r$
2  $r' := r$
3  for $Iter = 1, 2 \ldots, MaxIter$ do
4      VND.TSP($r'$)
5      VND($r'$)
6      if $\tilde{c}_{r'} < \tilde{c}_{r*}$ then
7          $r^* := r'$
8      if $\tilde{c}_{r*} < 0$ then
9          return $r^*$
10     for up to 3 times, as long as $H(r') \neq \varnothing$ do
11         Randomly chose $h \in H(r')$
12         DropCluster($r', h$)
13 return FAILED
```

4. Branch-and-Price

Two important aspects of the branch-and-price algorithms for the SoftCluVRP are clarified now. In Section 4.1, we describe possible strategies for combining the heuristic pricing algorithms with a final exact labeling-based pricing algorithm. In Section 4.2, we discuss branching rules and their impact on pricing algorithms.

4.1. Pricing Strategies

The different acceleration and heuristic pricing methods offer a plethora of pricing strategies. The question is how to combine the different pricing algorithms into a hierarchy of pricers. If a pricer on a lower level of the hierarchy fails to produce (sufficiently good) negative reduced-cost routes, the pricer on the next level is called. In extensive pretest we tried numerous combinations. We summarize the most important findings of the pretest:

1. The use of the Preprocessing is beneficial (Algorithm 1 called with `reduce = TRUE`).
2. The primal heuristic pricer should be applied before the labeling pricers (Algorithm 2 of Section 3.3).
3. Bidirectional labeling (Section 3.1.2) very often outperforms monodirectional labeling (Section 3.1.1).
4. DSSR with *ng*-path relaxation is most of the time faster than directly solving the elementary SPPRC (Section 3.1.4).
5. Heuristic pricers that use the strong dominance rule should also systematically violate the TI to further accelerate computations (also Section 3.1.4).

Therefore, we create a hierarchy of nine pricers, where the first one is the primal heuristic pricer, the next seven are heuristic labeling algorithms, and the last one is an exact labeling-based pricer. All labeling pricers use bidirectional labeling and DSSR. They differ in the size of the reduced networks (we use networks with 2, 5, 10 nearest neighbors, and the full network). For each network size, a first heuristic pricer applies the strong dominance Rule 2 to speed up computations even though this rule does not guarantee optimal solutions together with bidirectional labeling. With the same reasoning, the distance matrix is systematically violated. The second pricer per network size is one that is an exact pricer for this network, i.e., it does apply the *ng*-specific dominance Rule 3 and does not further manipulate the distance after preprocessing.

Default Strategy				Alternative Strategies				
Level	Size of Reduced Network	Dominance Rule	Violate STI	w/o violation of STI	w/o reduction (`reduce = false`)	w/o DSSR + *ng*-path relax.	w/o primal heuristic	only mono-directional
0	full	—use primal heuristic—					×	
1	2	strong: Rule 2	yes	STI		×		mono
2	2	*ng*: Rule 3	no		×	×		mono, TI
3	5	strong: Rule 2	yes	STI		×		mono
4	5	*ng*: Rule 3	no		×	×		mono, TI
5	10	strong: Rule 2	yes	STI		×		mono
6	10	*ng*: Rule 3	no		×	×		mono, TI
7	full	strong: Rule 2	yes	STI		×		mono
8	full	*ng*: Rule 3	no		×	×		mono, TI

Table 2: Default pricing strategy and five alternative pricing strategies.

This default pricing strategy is depicted in the four leftmost columns of Table 2. In addition, we define five alternative pricing strategies that one-by-one vary one of the fundamental components of the default strategy. These alternative strategies are summarized in the five rightmost columns of Table 2.

In detail: (1) strategy 'w/o violation of STI' does not systematically manipulate the cost matrix so that the cost matrix fulfills the STI in every pricer, (2) strategy 'w/o reduction' calls the Preprocessing Algorithm 2 with `reduce = FALSE` so that the preprocessed cost matrix fulfills the STI with a larger slack, (3) strategy 'w/o DSSR + *ng*-path relax.' directly solves elementary subproblems instead of using DSSR, (4) strategy 'w/o primal heuristic' drops the pricer at level 0, and (5) strategy 'only monodirectional' replaces the bidirectional labeling by monodirectional labeling where the respective exact pricers reduce the cost matrix further so that only the TI and not the STI needs to hold.

In Section 5.2, we will compare the six strategies (default and five alternative) on SoftCluVRP benchmark instances. Moreover, by eliminating some pricers including level 8, i.e., the exact labeling on the full network, the pricing hierarchy can be complemented with the branch-and-cut pricer from Section 3.2. Results with the respective pricing strategies are presented in Sections 5.3 and 5.4.

4.2. Branching

Note that in our model and the later analyzed benchmark instances the number of routes/vehicles is always given and fixed. Therefore, no branching on the number of vehicles is applicable here.

To finally ensure integrality of solutions, we apply the Ryan-Foster branching rule (Ryan and Foster, 1981) on the partitioning constraints (2b). These constraints ensure service for every cluster. Given an RMP solution $(\bar{y}_r)$, we determine for each pair $(h, h') \in H \times H, h < h'$ the number $f_{h,h'} = \sum_{r\in\Omega:a_{hr}=a_{h'r}=1} \bar{y}_r$. If the RMP solution is fractional, then there exists a value $f_{h,h'}$ strictly between 0 and 1 for some h and h'. We choose a pair with $f_{h,h'}$ closest to 0.5 and create the following two branches defined by the constraints

$$\sum_{r\in\Omega:a_{hr}=a_{h'r}=1} y_r = 0 \qquad \text{and} \qquad \sum_{r\in\Omega:a_{hr}=a_{h'r}=1} y_r = 1.$$

In the first branch, the *separate branch*, routes that serve both clusters V_h and $V_{h'}$ are not allowed. The second branch, the *together branch*, requires that the two clusters V_h and $V_{h'}$ are served by the same vehicle. Both types of branching decisions can be enforced without explicitly adding the above constraints to the RMP: Eliminate all routes $r \in \Omega$ that violate the branching condition from the current RMP. Moreover, ensure that forbidden routes are not priced out. This can be done as follows:

Separate Branch. Separate branching decisions have an impact on the structure of the subproblem and likewise on the subproblem algorithms. Assume that two clusters V_h and $V_{h'}$ must be served separately. In the case of labeling (Section 3.1), we modify the propagation rule. Once that a first customer of V_h is visited, we do not extend to vertices of cluster $V_{h'}$. This new behavior can be achieved using binary attributes for all clusters. For the dominance, we do as if the attributes had values $L_j^{visit_v} = 1$ for all $v \in V_{h'}$ and $L_j^{rem_{h'}} = 0$, in order to mimic that cluster $V_{h'}$ were already served completely. For merge conditions, however, these attributes remain at their correct values $L_j^{visit_v} = 0$ for all $v \in V_{h'}$ and $L_j^{rem_{h'}} = -1$. The same is done with exchanged roles of h and h'.

In the case of branch-and-cut (Section 3.2), we add the constraint $z_h + z_{h'} \leq 1$. Nothing else has to be done.

In the case of the primal heuristic (Section 3.3), the operator `AddCluster`(r, h) must be modified. Recall that it adds cluster V_h to a given route r. If a separate constraint for h and h' is active, the operator adds V_h to route r but removes all vertices of cluster $V_{h'}$.

Together Branch. A together branching decision for clusters V_h and $V_{h'}$ is trivial to impose for all types of subproblem algorithms. Instead of the original SoftCluVRP instance one just has to consider a new one in which the two clusters V_h and $V_{h'}$ are replaced by one bigger cluster $V_h \cup V_{h'}$ with demand $d_h + d_{h'}$.

5. Computational Results

Our algorithm is coded in C++ and compiled with MS Visual Studio 2015 in release mode. The callable library of CPLEX 12.8.1.0 is used to reoptimize the RMPs and to solve the subproblems with the branch-and-cut algorithm. We run all computations on a standard PC equipped with MS Windows 7 and an Intel(R) Core(TM) i7-5930K CPU clocked at 3.5 GHz and with 64 GB RAM of main memory.

5.1. SoftCluVRP Benchmark Instances

We test our algorithm on three benchmark sets. The first and second benchmark sets were derived from the CVRP benchmarks called `A`, `B`, `P`, `G`, and `C` by Bektaş *et al.* (2011). They defined θ as the desired average number of customers per customer cluster and, accordingly, $N = \lceil (n + 1)/\theta \rceil$ customer clusters are built (for details, we refer to Fischetti *et al.*, 1997; Bektaş *et al.*, 2011). Choosing $\theta \in \{2, 3\}$, the `GVRP-2` and `GVRP-3` benchmarks comprise 79 instances each, resulting in 158 small- and medium-sized instances with 16 to 262 vertices and 6 to 131 clusters. These benchmarks are available online at `http://www.personal.soton.ac.uk/tb12v07/gvrp.html`. The third set `Golden-Bat` was proposed by Battarra *et al.* (2014) for the CluVRP and comprises 220 large-scale instances with 201 to 484 vertices and 14 to

97 clusters. They are based on the well-known CVRP instances by Golden *et al.* (1998). For each of the 20 original instances `Golden1` to `Golden20`, different clusterings are generated by choosing $\theta = \{5, \ldots, 15\}$, resulting in eleven groups with 20 instances each.

For all experiments reported in the following, we run our branch-and-price algorithms with a time limit of 1 hour (3600 seconds) per instance. All computation times are displayed in seconds.

5.2. Comparison of Labeling Strategies

In a first series of experiments, we analyze the performance of the six pricing strategies that we discussed in Section 4.1. To keep the computational effort limited, we restrict ourselves to solving the linear relaxation of the master program, i.e., the root node of the branch-and-bound tree. Moreover, we do not use all 158 benchmark instances but only those 32 instances that we were able to solve with some labeling-based pricing strategy during pretests.

Table 3 briefly summarizes the results obtained for the 32 instances. Detailed instance-by-instance results can be found in the respective Table 8 of the Online Supplement. In both tables, T refers to the computation time for solving the linear relaxation of (2), *Avg. T* is the arithmetic mean, and *Geo. T* the geometric mean of the computation times over the 32 instances. In Table 3, #Solved is the number of instances for which the linear relaxation could be solved within the time limit (of 1 hour).

	Time for solving the linear relaxation					
	default	w/o violation of STI	w/o reduction (`reduce = false`)	w/o DSSR + *ng*-path relax.	w/o primal heuristic	only mono-directional
Avg. T	903.9	1226.9	1569.7	2595.5	1518.9	1661.2
Geo. T	60.0	79.7	95.9	321.3	94.8	73.3
#Solved	30	27	22	10	23	21

Table 3: Comparison of labeling-based pricing strategies using 32 selected SoftCluVRP instances.

The results for labeling-based pricing are very clear. From all acceleration techniques, the use of DSSR with the *ng*-path relaxation has the most positive impact: If replaced by directly solving subproblems as elementary SPPRC (strategy *w/o DSSR + ng-path relax.*), less than one third of the 32 linear relaxations can be solved. The impact of cost matrix reduction, primal heuristic, and bidirectional labeling is comparable, as only approximately 2/3 of the linear relaxations are solved when these techniques are not applied (see strategies *w/o reduction*, *w/o primal heuristic*, and *only monodirectional*, respectively). To not use the systematic manipulation of the cost matrix (strategy *w/o violation of STI*) leads to the smallest but still significant average speedups. The best strategy is clearly the *default* strategy in which all acceleration techniques switched on: 30 of the 32 linear relaxations are solved within the time limit, and average (arithmetic and geometric) computation times are considerably smaller compared to all other strategies.

5.3. Comparison of Pricing Strategies including Branch-and-Cut

In the second series of experiments, we want to find a best strategy for the use of the branch-and-cut-based subproblem algorithm introduced in Section 3.2. On the one hand, the primal heuristic can be called and in case of success it is not necessary to invoke the branch-and-cut algorithm. This is partial pricing with the primal heuristic. On the other hand, also the labeling-based pricing algorithms can be used for partial pricing. In this case, we use the default strategy from Section 4.1 but only the heuristic levels 1 and 3 (*truncated default strategy*). Moreover, the primal heuristic (level 0) may or may not be used. This leads to four possible strategies:

`Pure`:	Pure branch-and-cut.
`+Prim`:	Primal heuristic first.
`+Label`:	Truncated default labeling strategy w/o primal heuristic first.
`+Prim+Label`:	Truncated default labeling strategy with primal heuristic first.

For the comparison, we use the identical subset of 32 instances as in the previous section. The results are summarized in Table 4 and the corresponding instance-by-instance results are given in Table 8 of the Online Supplement.

	Time for solving the linear relaxation			
	`Pure:`	`+Prim:`	`+Label:`	`+Prim+Label:`
primal heuristic:	no	yes	no	yes
truncated labeling:	no	no	yes	yes
Avg. T	7.9	2.6	152.7	9.9
Geo. T	2.5	0.9	5.7	1.9
#Solved	32	32	31	32

Table 4: Comparison of branch-and-cut-based pricing strategies using 32 selected SoftCluVRP instances.

The outcome of the experiments is that all four branch-and-cut-based pricing strategies outperform the labeling-based strategies by at least one order of magnitude of computation time. The worst strategy is `+Label` confirming the primal heuristic is essential and labeling-based pricing inferior. Compared to the other branch-and-cut strategies, this strategy fails in solving one linear relaxation (recall that the best labeling strategy failed in two cases). The strategies `Pure` and `+Prim+Label` are incomparable w.r.t. arithmetic and geometric means of computation time. The best strategy (undominated) is `+Prim` where primal heuristic and branch-and-cut are combined. The speedup factor compared to the default labeling is >340 (ratio of arithmetic means) and >66 (ratio of geometric means).

For the remainder of the paper, we use `+Prim` as the *default branch-and-cut strategy.*

5.4. Comparison of Labeling-based and Branch-and-Cut-based Pricing

Even if the previous sections clearly indicate the superiority of branch-and-cut over labeling for solving the subproblem, we want to test the two respective default strategies against each other on the full benchmark set and within the fully-fledged branch-and-price algorithm. Table 5 gives aggregated results for all 158 `GVRP` instances, grouped by the subclasses `A`, `B`, `P`, and `GC` of `GVRP-2` and `GVRP-3`. For the solution of the root, the table shows the number of successfully solved linear relaxation of (2) (#Solved) and average computation times T (*Avg.* and *Geo.*). Similarly, for the branch-and-price, *#Int* and *#Opt* is the number of instances where an integer solution was found and proven optimal, respectively.

Branch-and-price with the default labeling strategy cannot solve any group of instances completely, neither the linear relaxation nor the full branch-and-bound tree. It seems however that instances from the group `GVRP-2` are slightly easier for this approach than from the group `GVRP-3`. In comparison, branch-and-price with the default branch-and-cut strategy performs much better than the default labeling strategy: 34 vs. 150 solved linear relaxations and 23 vs. 142 exactly solved instances, respectively. The observed aggregated computation times underline the impressive predominance of branch-and-cut for the SoftCluVRP subproblems. We would like to add that the labeling-based approach is almost always inferior in an per instance comparison (however, six instances all with computation times below 2 seconds are solved faster with labeling).

Detailed instance-by-instance results of the branch-and-price with the default branch-and-cut strategy are given in Tables 9 to 12 in the Online Supplement.

5.5. Results for the `Golden-Bat` Instances

Since the `Golden-Bat` instances are much larger regarding the number of customers and the number of clusters, they are not at all accessible for branch-and-price algorithms that use the labeling techniques of Section 3.1. In this section, we therefore restrict the presentation of results to those obtained with the branch-and-price that uses the default branch-and-cut strategy.

Tables 6a and 6b summarize the results, where the first table groups the instances by the number $n+1$ of vertices and the second by the average cluster size θ. On the one hand, the difficulty of instances increases

	Default Labeling Strategy							Default Branch-and-Cut Strategy						
	linear relaxation			integer				linear relaxation			integer			
	#	Time T		#		Time T		#	Time T		#		Time T	
Set (#inst.)	Solved	*Avg.*	*Geo.*	Int	Opt	*Avg.*	*Geo.*	Solved	*Avg.*	*Geo.*	Int	Opt	*Avg.*	*Geo.*
`GVRP-2`														
`A (27)`	7	3000	2573	5	4	3353	3219	27	33	9	27	26	713	114
`B (23)`	7	2825	2229	3	2	3368	3198	23	24	10	23	17	1105	144
`P (24)`	11	2107	162	11	9	2393	250	24	155	4	23	23	474	17
`GC (5)`	0	3600	3600	0	0	3600	3600	1	2912	1929	1	1	2994	2489
`GVRP-3`														
`A (27)`	0	3600	3600	0	0	3600	3600	27	17	5	27	27	83	17
`B (23)`	1	3522	3494	0	0	3600	3600	23	7	4	23	23	160	18
`P (24)`	8	2411	316	8	8	2431	367	24	32	2	24	24	107	4
`GC (5)`	0	3600	3600	0	0	3600	3600	1	2899	1747	1	1	3221	3101
Total (158)	34	2966	1362	27	23	3163	1637	150	225	7	149	142	605	36

Table 5: Results for the 158 `GVRP` instances.

with the number of vertices, which seems natural. The three largest instances solved have 420 customers. On the other hand, instances with a larger average cluster size θ become better solvable. Also this behaviour is as expected because with larger clusters the number of partitioning constraints (2b) in the master program decreases and also the model (3) of the subproblem has less cluster variables z_h making it less difficult to solve. The smallest average cluster size of the above solved 420-customer instances has $\theta = 13$. Detailed instance-by-instance results are reported in Tables 13 to 16 in the Online Supplement.

5.6. Comparison of CluVRP and SoftCluVRP Solutions

For those instances, for which optimal SoftCluVRP and CluVRP solutions are known, we compute the *gap to CluVRP* (in percent) as $100 \cdot (Z_{CluVRP} - Z_{SoftCluVRP})/Z_{SoftCluVRP}$ where the minimal routing costs are $Z_{SoftCluVRP}$ and Z_{CluVRP}, respectively. Table 7 shows aggregated results for the `GVRP-3` and `Golden-Bat` instances. Note that detailed results for the benchmark `GVRP-2` have not been published so that for them Z_{CluVRP} is not known. (We took the detailed results for the benchmark `GVRP-3` from (Defryn and Sörensen, 2017).)

The results for the `GVRP-3` instances in Table 7a show for how many instances optimal solutions of both SoftCluVRP and CluVRP are known (#both opt). Over these 76 instances, the average gap varies over the subsets `A`, `B`, `P`, and `GC`. Gaps are larger in subset `P` compared to `A`, and these larger compared to `B`. We attribute this difference to the different ways in which the subsets were constructed.

The results for the `Golden-Bat` instances are shown in Table 7b: Average gaps strongly depend on the average number θ of customers per cluster. The larger $\theta \geq 5$, the smaller the average gaps. This behaviour is intuitive, since a SoftCluVRP route tends to change clusters less frequently when clusters are larger. Thus, serving the same clusters but respecting hard-cluster constraints requires only fewer modifications on the SoftCluVRP route. If θ grows even further so that routes serve single clusters, the gap vanishes.

The described dependency of the gaps on θ is definitely not the same for smaller values $\theta \leq 5$: It is impossible that gaps constantly increase when θ decreases, because the extreme case of $\theta = 1$ means that clusters are singleton sets, for which optimal SoftCluVRP and CluVRP coincide again. Hence, gaps are zero for $\theta = 1$. This is in line with what the comparison of the `GVRP-3` and `Golden-Bat` benchmarks shows: The former benchmark has $\theta = 3$ and smaller average gaps compared to the larger gaps for $\theta \geq 5$ for the latter benchmark.

Table 6: Results for the 220 large-scale `Golden-Bat` instances.

	linear relaxation			integer			
	#	Time T		#		Time T	
$n+1$	Solved	*Avg.*	*Geo.*	Int	Opt	*Avg.*	*Geo.*
201 (11)	10	832	487	10	10	832	487
241 (22)	17	1553	963	11	9	2307	1426
253 (11)	9	938	394	9	9	1119	475
256 (11)	10	943	494	8	8	1446	747
281 (11)	5	2702	2392	5	5	2702	2392
301 (11)	6	2338	1918	6	6	2338	1918
321 (22)	7	2875	2485	5	5	3196	2950
324 (11)	6	2179	1483	6	6	2453	2140
361 (22)	5	3171	2936	5	5	3171	2936
397 (11)	2	3256	3142	1	1	3561	3559
400 (11)	3	3112	2945	1	1	3515	3502
401 (11)	0	3600	3600	0	0	3600	3600
421 (11)	3	3033	2790	3	3	3033	2790
441 (11)	0	3600	3600	0	0	3600	3600
481 (22)	0	3600	3600	0	0	3600	3600
484 (11)	0	3600	3600	0	0	3600	3600
Total (220)	83	2626	1927	70	68	2817	2172

(a) Grouped by number of vertices $n + 1$.

	linear relaxation			integer			
	#	Time T		#		Time T	
θ	Solved	*Avg.*	*Geo.*	Int	Opt	*Avg.*	*Geo.*
5 (20)	1	3521	3497	1	1	3521	3497
6 (20)	1	3523	3501	0	0	3600	3600
7 (20)	4	3136	2891	2	2	3394	3261
8 (20)	5	3036	2568	4	4	3042	2572
9 (20)	6	2866	2308	5	5	2988	2582
10 (20)	9	2603	1939	7	7	2816	2169
11 (20)	9	2496	1726	7	7	2744	1937
12 (20)	9	2326	1562	8	8	2617	1898
13 (20)	13	1889	1139	13	12	2140	1340
14 (20)	14	1705	1085	12	11	2097	1382
15 (20)	12	1790	1003	11	11	2031	1258
Total (220)	83	2626	1927	70	68	2817	2172

(b) Grouped by average cluster size θ.

6. Conclusions

In this article, we have designed and analyzed different branch-and-price algorithms for the exact solution of the SoftCluVRP. The research has mainly focussed on the solution of the column-generation subproblem, a variant of the SPPRC. It has turned out that, for this variant of the vehicle-routing problem, dynamic programming-based labeling algorithms are strongly outperformed by a branch-and-cut algorithm that works directly on the SoftCluVRP subproblem formulation. The latter integer programming-based solution approach contributes with computation times that are by one order of magnitude shorter than those of sophisticated dynamic-programming labeling algorithms. The largest SoftCluVRP instances that we solved to optimality have more than 400 customers or more than 50 clusters.

We attribute the success of branch-and-cut for the solution of SoftCluVRP subproblems to the following facts: The subproblem is very close to a TSP, more precisely, it is a TSP with profits that also shares characteristics with the prize-collecting TSP (Feillet *et al.*, 2005; Balas, 1989). For these types of TSPs, branch-and-cut is the leading state-of-the-art solution approach (Gutin and Punnen, 2002). Previous attempts of using IP-based methods for SPPRCs, like (Jepsen *et al.*, 2008), concentrated on the solution of CVRP subproblems via branch-and-cut. Results were competitive with monodirectional labeling algorithms without DSSR of that time. However, powerful techniques such as bidirectional labeling and the *ng*-path relaxation to be used separately or within DSSR were not yet available when Jepsen *et al.* (2008) conducted their experiments. When including these newer techniques, labeling algorithms outperform the branch-and-cut algorithm on CVRP subproblems.

In comparison, the SoftCluVRP subproblem is even closer to a TSP than the CVRP subproblem. The point is that in the CVRP subproblem visits are decided on a customer level, while in the SoftCluVRP subproblem on a cluster level, and nothing is to decide regarding visits in a pure TSP. This may explain why branch-and-cut is superior to fully-fledged labeling algorithms in SoftCluVRP subproblems but not in CVRP subproblems.

Future research may focus on further enhancing the branch-and-cut with new classes of valid inequalities as well as effective and fast separation heuristics. We think that soft-cluster constraints could also play an important role in districting and capacitated arc-routing applications (Butsch *et al.*, 2014; Belenguer *et al.*,

Table 7: Comparison of optimal SoftCluVRP and CluVRP solutions.

θ	Set (#inst.)	#both opt	Avg. gap to CluVRP
3			
	A (27)	27	2.66
	B (23)	23	1.16
	P (24)	24	4.73
	GC (5)	2	3.73
Total	(79)	76	2.89

(a) `GVRP-3` instances.

θ (#inst.)	#both opt	Avg. gap to CluVRP
5 (20)	1	9.45
6 (20)	3	7.21
7 (20)	4	7.46
8 (20)	6	6.79
9 (20)	7	7.46
10 (20)	8	6.90
11 (20)	9	6.16
12 (20)	11	5.94
13 (20)	12	5.62
14 (20)	13	5.51
15 (20)	13	5.46
Total (220)	87	6.21

(b) `Golden-Bat` instances.

2014).

References

Balas, E. (1989). The prize collecting traveling salesman problem. *Networks*, **19**(6), 621–636.

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

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.

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

Bektaş, T., Erdoğan, G., and Ropke, S. (2011). Formulations and branch-and-cut algorithms for the generalized vehicle routing problem. *Transportation Science*, **45**(3), 299–316.

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 & Applied Mathematics (SIAM).

Bentley, J. J. (1992). Fast algorithms for geometric traveling salesman problems. *ORSA Journal on Computing*, **4**(4), 387–411.

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

Boykov, Y. and Kolmogorov, V. (2004). An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **26**, 1124–1137.

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

Cherkesly, M., Desaulniers, G., and Laporte, G. (2015). Branch-price-and-cut algorithms for the pickup and delivery problem with time windows and last-in-first-out loading. *Transportation Science*, **49**(4), 752–766.

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.

Desaulniers, G., Desrosiers, J., Ioachim, I., Solomon, M., Soumis, F., and Villeneuve, D. (1998). A unified framework for deterministic time constrained vehicle routing and crew scheduling problems. In T. Crainic and G. Laporte, editors, *Fleet Management and Logistics*, chapter 3, pages 57–93. Kluwer Academic Publisher, Boston, Dordrecht, London.

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

Desaulniers, G., Pecin, D., and Contardo, C. (2017). Selective pricing in branch-price-and-cut algorithms for vehicle routing. *EURO Journal on Transportation and Logistics*. `http://dx.doi.org/10.1007/s13676-017-0112-9`.

Drexl, M. and Irnich, S. (2012). Solving elementary shortest-path problems as mixed-integer programs. *OR Spectrum*, **36**(2), 281–296.

Dror, M. (1994). Note on the complexity of the shortest path models for column generation in VRPTW. *Operations Reasearch*, **42**(5), 977–978.

Dumas, Y., Desrosiers, J., and Soumis, F. (1991). The pick-up and delivery problem with time windows. *European Journal of Operational Research*, **54**, 7–22.

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., Gendreau, M., and Guéguen, C. (2004). An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems. *Networks*, **44**(3), 216–229.

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

Fischetti, M., González, J. J. S., and Toth, P. (1997). A branch-and-cut algorithm for the symmetric generalized traveling salesman problem. *Operations Research*, **45**(3), 378–394.

Goeke, D., Gschwind, T., and Schneider, M. (2017). Upper and lower bounds for the vehicle-routing problem with private fleet and common carrier. Working Paper DPO-2017-08, Deutsche Post Chair – Optimization of Distribution Networks, RWTH Aachen University, Aachen, Germany.

Golden, B. L., Wasil, E. A., Kelly, J. P., and Chao, I.-M. (1998). The impact of metaheuristics on solving the vehicle routing problem: Algorithms, problem sets, and computational results. In T. G. Crainic and G. Laporte, editors, *Fleet Management and Logistics*, pages 33–56. Springer US, Boston, MA.

Gschwind, T., Irnich, S., Rothenbächer, A.-K., and Tilk, C. (2017). Bidirectional labeling in column-generation algorithms for pickup-and-delivery problems. *European Journal of Operational Research*, **266**, 521–530.

Gutin, G. and Punnen, A., editors (2002). *The Traveling Salesman Problem and Its Variations*, volume 12 of *Combinatorial Optimization*. Kluwer, Dordrecht.

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.

Hoos, H. H. and Stützle, T. (2004). *Stochastic Local Search: Foundations & Applications*. Morgan Kaufmann.

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

Irnich, S. and Villeneuve, D. (2006). The shortest path problem with resource constraints and k-cycle elimination for $k \geq 3$. *INFORMS Journal on Computing*, **18**(3), 391–406.

Irnich, S., Funke, B., and Grünert, T. (2006). Sequential search and its application to vehicle-routing problems. *Computers & Operations Research*, **33**(8), 2405–2429.

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. K., Petersen, B., and Spoorendonk, S. (2008). A branch-and-cut algorithm for the elementary shortest path problem with a capacity constraint. Technical report 08-01, Department of Computer Science, University of Copenhagen, Kopenhagen, Denmark.

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

Mladenović, N. and Hansen, P. (1997). Variable neighborhood search. *Computers & Operations Research*, **24**(11), 1097–1100.

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.

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

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

Ropke, S. and Cordeau, J.-F. (2009). Branch and cut and price for the pickup and delivery problem with time windows. *Transportation Science*, **43**(3), 267–286.

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

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.

Tarjan, R. E. (1979). A class of algorithms which require nonlinear time to maintain disjoint sets. *Journal of Computer and System Sciences*, **18**(2), 110–127.

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

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

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

Wolsey, L. A. (1998). *Integer Programming*. Wiley-Interscience, New York, NY.

This appendix is supposed to become online supplementary material.

Appendix

I. Detailed Results

I.1. Linear-Relaxation Results for the 32 Filtered `GVRP` Instances

Table 8 shows detailed instance-by-instance results for the 32 filtered `GVRP` instances per labeling- and branch-and-cut-strategy as described in Section 5.2.

Instance						Labeling-Strategies						Branch-&-Cut-Strategies			
θ	Set	n	k	N	m	default	w/o violation of STI	w/o reduction (`reduce = false`)	w/o DSSR + ng-path relax.	w/o primal heuristic	only mono-directional	`Pure`	`+Prim`	`+Label`	`+Prim` `+Label`
2	A	32	5	17	3	193.3	311.2	163.4	3600.0	116.8	127.0	3.6	1.1	2.1	1.5
	A	32	6	17	3	1268.5	3164.2	3394.2	3600.0	2220.2	2368.4	3.8	2.9	9.6	4.5
	A	33	5	17	3	984.5	2302.1	1728.0	3600.0	2672.3	3600.0	4.3	1.2	14.9	5.3
	A	36	6	19	3	3596.3	3600.0	1709.4	3600.0	3600.0	3600.0	8.2	4.0	93.1	32.1
	A	37	5	19	3	942.4	3600.0	1195.6	3600.0	1382.5	3600.0	6.7	1.4	10.5	4.4
	A	44	6	23	4	1159.4	3370.9	3600.0	3600.0	3440.2	3567.0	7.8	4.1	8.3	4.4
	A	54	9	28	5	1053.0	1100.1	3600.0	3600.0	3600.0	3600.0	28.2	3.8	232.3	45.1
	B	30	5	16	3	132.3	145.5	92.3	3600.0	210.1	129.3	3.0	1.1	6.0	5.0
	B	34	5	18	3	1599.7	1671.4	3600.0	3600.0	3600.0	3600.0	7.3	2.8	5.1	2.1
	B	37	6	19	3	531.1	576.7	1587.0	3600.0	557.6	3600.0	7.8	1.6	5.3	1.4
	B	43	7	22	4	2639.6	2827.9	3464.0	3600.0	3600.0	3600.0	12.7	3.3	15.1	5.5
	B	44	5	23	3	957.7	1022.2	3600.0	3600.0	3600.0	3600.0	14.9	4.8	33.6	5.2
	B	49	7	25	4	756.7	815.8	3600.0	3600.0	3600.0	176.1	12.3	1.3	38.3	3.8
	B	50	7	26	4	463.8	504.9	3600.0	3600.0	1652.5	3600.0	29.8	9.7	161.3	51.5
	P	15	8	8	5	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	0.1	0.1	0.1	0.1
	P	18	2	10	2	0.2	0.8	1.3	3.9	0.7	0.5	0.2	0.1	0.1	<0.1
	P	19	2	10	2	4.8	10.5	14.9	679.3	6.7	4.7	0.2	<0.1	0.1	0.1
	P	20	2	11	2	9.4	11.2	27.6	982.7	7.4	1.6	0.2	0.1	0.1	0.1
	P	21	2	11	2	15.6	26.0	42.9	3600.0	42.4	17.3	0.1	0.1	0.5	0.1
	P	21	8	11	5	0.1	<0.1	<0.1	<0.1	0.1	<0.1	0.1	0.1	0.2	0.2
	P	22	8	12	5	0.1	<0.1	<0.1	0.1	<0.1	<0.1	0.6	0.3	0.3	0.2
	P	49	10	25	5	3600.0	3600.0	3600.0	3600.0	3600.0	3257.1	19.2	11.2	64.7	58.2
	P	50	10	26	6	369.4	1225.1	488.6	3600.0	1117.6	328.3	10.1	2.9	26.0	15.4
	P	54	10	28	5	3117.1	3600.0	3600.0	3600.0	3600.0	3452.9	26.2	4.7	136.9	11.1
	P	54	15	28	8	47.9	58.9	42.8	2188.7	59.1	16.0	20.2	12.9	19.9	7.8
	P	59	15	30	8	105.5	192.7	240.1	3600.0	239.4	74.6	14.0	6.3	16.1	10.3
3	B	30	5	11	2	1708.0	1841.8	3601.1	3600.0	2336.2	3600.0	2.0	0.5	370.2	8.9
	B	38	5	13	2	3600.0	3600.0	3600.0	3600.0	3600.0	3600.0	2.5	0.2	3600.0	23.9
	P	15	8	6	4	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1
	P	21	8	8	4	<0.1	0.1	0.1	0.1	<0.1	<0.1	0.1	0.2	0.2	0.2
	P	22	8	8	3	0.1	0.2	0.1	0.2	0.3	0.1	0.5	0.1	0.6	0.2
	P	54	15	19	6	68.1	80.8	38.9	3600.0	141.6	37.1	4.7	1.6	13.6	7.2
Avg. T						903.9	1226.9	1569.7	2595.5	1518.9	1661.2	7.9	2.6	152.7	9.9
Geo. T						60.0	79.7	95.9	321.3	94.8	73.3	2.5	0.9	5.7	1.9

Table 8: Detailed results for 32 filtered `GVRP` instances: Time T (in seconds) per labeling- and branch-and-cut-strategy for solving the linear relaxation.

I.2. Detailed Results for the `GVRP` *Instances*

Detailed instance-by-instance results of the branch-and-price with the default branch-and-cut strategy are given in Tables 9 to 12. We describe the instance (number of customers n, number of vehicles k in the original CVRP instance, number of customer clusters N, number of vehicles m, and average cluster size θ) and additionally provide the following information:

BKS:	Best known solution, bold if proven optimal;
Gap to CluVRP:	Gap in percent between optimal CluVRP and SoftCluVRP solutions, see Section 5.6;
First found by:	Article that first computed the BKS (Bat14=Battarra *et al.* (2014), DS17=Defryn and Sörensen (2017), Vidal15=Vidal *et al.* (2015), B&P=this work);
LB root:	Lower bound provided by the linear relaxation;
LB tree:	Lower bound provided by branch-and-price;
UB:	Upper bound provided by branch-and-price;
#B&B nodes:	Number of branch-and-bound nodes explored;
Time T:	Computation time (in seconds).

Furthermore, for the column *First found by* and our branch-and-price-algorithm we give the following information:

B&P:	this work with the default branch-and-cut pricing (`+Prim`);
B&P*:	found during computational studies with another variant ($\neq$ `+Prim`);
B&P$^{\diamondsuit}$:	as B&P*, but with an extended time limit (7200 seconds);
B&P†:	as B&P, but with an extended time limit (up to 36,000 seconds).

Regarding BKS and optimal solutions, recall that the SoftCluVRP is a relaxation of the CluVRP. Any lower bound for the SoftCluVRP is also a lower bound for the respective CluVRP. The same holds for upper bounds of the CluVRP that are upper bounds for the respective SoftCluVRP. Therefore, the heuristics of Defryn and Sörensen (2017) and Vidal *et al.* (2015) as well as the exact approach for the CluVRP by Battarra *et al.* (2014) provide upper bounds that may be BKS or may even prove optimality of our solutions when the branch-and-price is stopped at the time limit. Note that here, in all cases the optimal solution is known, the proof of optimality is given by our branch-and-price algorithm, also if this solution was found by Defryn and Sörensen (2017), Vidal *et al.* (2015), or Battarra *et al.* (2014) before.

Instance								Branch-and-Price results				
	n	k	N	m	BKS	*Gap to CluVRP*	*First found by*	*LB* root	*LB* tree	*UB*	#B&B nodes	Time T
A	31	5	16	2	**595**	-	B&P	595	595	595	1	7
A	32	5	17	3	**528**	-	B&P	517	528	528	17	12
A	32	6	17	3	**561**	-	B&P	560	561	561	4	5
A	33	5	17	3	**568**	-	B&P	562	568	568	14	13
A	35	5	18	2	**596**	-	B&P	589	596	596	12	65
A	36	5	19	3	**573**	-	B&P	571	573	573	5	6
A	36	6	19	3	**660**	-	B&P	660	660	660	1	4
A	37	5	19	3	**547**	-	B&P	547	547	547	1	1
A	38	5	20	3	**659**	-	B&P	639	659	659	37	78
A	38	6	20	3	**676**	-	B&P	658	676	676	41	78
A	43	6	22	3	**723**	-	B&P	722	723	723	3	23
A	44	6	23	4	**679**	-	B&P	679	679	679	1	4
A	44	7	23	4	**774**	-	B&P	761	774	774	60	242
A	45	7	23	4	**708**	-	B&P	685	708	708	143	209
A	47	7	24	4	**784**	-	B&P	760	784	784	356	1431
A	52	7	27	4	**732**	-	B&P	707	732	732	68	285
A	53	7	27	4	**806**	-	B&P	797	806	806	26	265
A	54	9	28	5	**778**	-	B&P	765	778	778	25	84
A	59	9	30	5	**877**	-	B&P	860	877	877	133	2010
A	60	9	31	5	**749**	-	B&P	744	749	749	14	142
A	61	8	31	4	**849**	-	B&P	838	849	849	34	839
A	62	9	32	5	**1043**	-	B&P	1030	1043	1043	123	3159
A	62	10	32	5	**895**	-	B&P	882	895	895	55	512
A	63	9	32	5	**895**	-	B&P	886	895	895	41	1132
A	64	9	33	5	**825**	-	B&P	800	825	825	313	2544
A	68	9	35	5	**857**	-	B&P	838	857	857	337	2506
A	79	10	40	5	**1115**	-	B&P	1110	1113	1115	30	3600
B	30	5	16	3	**451**	-	B&P	449	451	451	7	7
B	33	5	17	3	**495**	-	B&P	484	495	495	17	26
B	34	5	18	3	**654**	-	B&P	613	654	654	23	27
B	37	6	19	3	**479**	-	B&P	479	479	479	2	3
B	38	5	20	3	**378**	-	B&P	372	378	378	3	5
B	40	6	21	3	**514**	-	B&P	509	514	514	3	13
B	42	6	22	3	**522**	-	B&P	498	522	522	320	897
B	43	7	22	4	**562**	-	B&P	531	562	562	183	363
B	44	5	23	3	**542**	-	B&P	542	542	542	2	7
B	44	6	23	4	**506**	-	B&P	491	506	506	73	141
B	49	7	25	4	**495**	-	B&P	495	495	495	1	1
B	49	8	25	5	954	-	B&P$^\diamond$	904	938	958	981	3600
B	50	7	26	4	**672**	-	B&P	661	672	672	37	123
B	51	7	26	4	**485**	-	B&P	471	485	485	57	224
B	55	7	28	4	520	-	B&P	457	476	520	972	3600
B	56	7	29	4	777	-	B&P*	716	744	781	317	3600
B	56	9	29	5	**983**	-	B&P	980	983	983	33	251
B	62	10	32	5	**865**	-	B&P	850	865	865	50	1402
B	63	9	32	5	**550**	-	B&P	550	550	550	1	21
B	65	9	33	5	850	-	B&P$^\diamond$	827	836	869	186	3600
B	66	10	34	5	725	-	B&P$^\diamond$	682	699	736	587	3600
B	67	9	34	5	**745**	-	B&P	744	745	745	8	293
B	77	10	39	5	**842**	-	B&P$^\diamond$	815	832	849	130	3600

Table 9: Detailed results for the `GVRP-2` instances, subsets `A` and `B`.

Instance								Branch-and-Price results				
	n	k	N	m	BKS	*Gap to CluVRP*	*First found by*	*LB* root	*LB* tree	*UB*	#B&B nodes	Time T
P	15	8	8	5	**299**	-	B&P	299	299	299	1	<1
P	18	2	10	2	**195**	-	B&P	195	195	195	1	<1
P	19	2	10	2	**208**	-	B&P	208	208	208	2	<1
P	20	2	11	2	**208**	-	B&P	208	208	208	1	<1
P	21	2	11	2	**209**	-	B&P	209	209	209	1	<1
P	21	8	11	5	**397**	-	B&P	397	397	397	1	<1
P	22	8	12	5	**369**	-	B&P	366	369	369	8	2
P	39	5	20	3	**401**	-	B&P	400	401	401	7	10
P	44	5	23	3	**443**	-	B&P	443	443	443	1	5
P	49	7	25	4	**464**	-	B&P	458	464	464	46	119
P	49	8	25	4	**501**	-	B&P	490	501	501	24	230
P	49	10	25	5	**512**	-	B&P	509	512	512	20	56
P	50	10	26	6	**548**	-	B&P	545	548	548	13	21
P	54	7	28	4	**477**	-	B&P	477	477	477	1	8
P	54	8	28	4	**484**	-	B&P	480	484	484	8	40
P	54	10	28	5	**514**	-	B&P	514	514	514	1	5
P	54	15	28	8	**684**	-	B&P	681	684	684	18	70
P	59	10	30	5	**575**	-	B&P	564	575	575	84	725
P	59	15	30	8	**700**	-	B&P	692	700	700	118	216
P	64	10	33	5	**616**	-	B&P	611	616	616	28	291
P	69	10	35	5	**643**	-	B&P	635	643	643	71	934
P	75	4	38	2	**557**	-	B&P	555	557	557	34	2745
P	75	5	38	3	**571**	-	B&P	567	571	571	49	2311
P	100	4	51	2	646	-	B&P$^{\diamond}$	645	645	-	1	3600
G	261	25	131	12	3693	-	Vidal15	-	-	-	0	3600
C	100	10	51	5	**628**	2.18	B&P	626	628	628	4	569
C	120	7	61	4	807	-	Bat14	-	-	-	0	3600
C	150	12	76	6	816	-	Bat14	-	-	-	0	3600
C	199	16	100	8	955	-	Bat14	-	-	-	0	3600

Table 10: Detailed results for the `GVRP-2` instances, subsets `P` and `GC`.

Instance								Branch-and-Price results				
	n	k	N	m	BKS	*Gap to CluVRP*	*First found by*	*LB* root	*LB* tree	*UB*	#B&B nodes	Time T
A	31	5	11	2	**515**	1.34	DS17	515	515	515	1	<1
A	32	5	11	2	**461**	2.33	DS17	461	461	461	1	<1
A	32	6	11	2	**554**	1.42	DS17	551	554	554	3	2
A	33	5	12	2	**538**	1.65	DS17	529	538	538	11	6
A	35	5	12	2	**543**	7.65	DS17	535	543	543	8	6
A	36	5	13	2	**545**	4.22	B&P	545	545	545	16	11
A	36	6	13	2	**605**	1.63	DS17	603	605	605	3	6
A	37	5	13	2	**507**	0.00	Bat14	500	507	507	4	5
A	38	5	13	2	**588**	3.61	DS17	568	588	588	15	13
A	38	6	13	2	**603**	1.63	DS17	595	603	603	7	6
A	43	6	15	2	**691**	3.22	DS17	686	691	691	3	27
A	44	6	15	3	**652**	8.43	DS17	652	652	652	1	2
A	44	7	15	3	**661**	0.45	DS17	651	661	661	9	29
A	45	7	16	3	**642**	3.31	DS17	641	642	642	5	12
A	47	7	16	3	**680**	0.44	DS17	674	680	680	28	57
A	52	7	18	3	**627**	3.69	DS17	625	627	627	3	14
A	53	7	18	3	**699**	3.45	DS17	679	699	699	51	143
A	54	9	19	3	**645**	1.23	DS17	645	645	645	1	9
A	59	9	20	3	**762**	3.18	DS17	761	762	762	3	29
A	60	9	21	4	**671**	1.61	DS17	671	671	671	3	23
A	61	8	21	3	**771**	0.9	DS17	761	771	771	36	404
A	62	10	21	4	**779**	2.75	DS17	779	779	779	1	13
A	62	9	21	3	**837**	3.24	DS17	837	837	837	1	43
A	63	9	22	3	**767**	0.78	DS17	754	767	767	54	585
A	64	9	22	3	**693**	4.41	DS17	693	693	693	1	14
A	68	9	23	3	**794**	2.46	DS17	779	794	794	31	603
A	79	10	27	4	**944**	2.88	DS17	944	944	944	1	178
B	30	5	11	2	**375**	0.00	Bat14	366	375	375	13	4
B	33	5	12	2	**415**	0.24	DS17	397	415	415	19	5
B	34	5	12	2	**557**	0.89	DS17	526	557	557	44	18
B	37	6	13	2	**427**	0.93	DS17	427	427	427	1	3
B	38	5	13	2	**317**	1.25	DS17	317	317	317	1	<1
B	40	6	14	2	**469**	1.47	DS17	461	469	469	8	12
B	42	6	15	2	**405**	2.41	DS17	400	405	405	3	8
B	43	7	15	3	**443**	0.89	DS17	435	443	443	3	7
B	44	5	15	2	**489**	3.36	DS17	489	489	489	1	3
B	44	6	15	2	**386**	1.28	DS17	386	386	386	1	4
B	49	7	17	3	**464**	0.64	DS17	454	464	464	7	16
B	49	8	17	3	**661**	0.75	DS17	661	661	661	1	5
B	50	7	17	3	**578**	1.2	DS17	567	578	578	9	17
B	51	7	18	3	**427**	0.00	Bat14	421	427	427	3	11
B	55	7	19	3	**420**	3	DS17	416	420	420	4	16
B	56	7	19	3	**622**	1.89	DS17	582	622	622	138	437
B	56	9	19	3	**746**	0.93	DS17	709	746	746	505	1606
B	62	10	21	3	**685**	0.00	Bat14	685	685	685	1	21
B	63	9	22	4	**524**	0.38	DS17	522	524	524	5	32
B	65	9	22	3	**683**	0.58	DS17	666	683	683	30	252
B	66	10	23	4	**619**	1.12	DS17	610	619	619	11	72
B	67	9	23	3	**582**	1.02	DS17	582	582	582	1	25
B	77	10	26	4	**704**	2.36	DS17	679	704	704	91	1109

Table 11: Detailed results for the `GVRP-3` instances, subsets `A` and `B`.

Instance								Branch-and-Price results				
	n	*k*	*N*	*m*	BKS	*Gap to CluVRP*	*First found by*	*LB* root	*LB* tree	*UB*	#B&B nodes	Time *T*
P	15	8	6	4	**251**	0.79	DS17	251	251	251	1	<1
P	18	2	7	1	**170**	8.6	DS17	170	170	170	1	<1
P	19	2	7	1	**177**	11.5	DS17	177	177	177	1	<1
P	20	2	7	1	**179**	5.79	DS17	179	179	179	1	<1
P	21	2	8	1	**183**	9.41	DS17	183	183	183	1	<1
P	21	8	8	4	**365**	0.00	Bat14	365	365	365	1	<1
P	22	8	8	3	**270**	3.23	DS17	264	270	270	4	<1
P	39	5	14	2	**381**	3.79	DS17	380	381	381	8	8
P	44	5	15	2	**422**	4.09	DS17	422	422	422	1	2
P	49	7	17	3	**430**	3.8	DS17	430	430	430	1	4
P	49	8	17	3	**441**	4.13	DS17	441	441	441	1	3
P	49	10	17	4	**471**	4.07	DS17	470	471	471	3	5
P	50	10	17	4	**493**	8.19	DS17	493	493	493	1	2
P	54	7	19	3	**454**	1.73	B&P	453	454	454	4	21
P	54	8	19	3	**454**	3.61	B&P	454	454	454	2	9
P	54	10	19	4	**481**	3.8	DS17	481	481	481	2	4
P	54	15	19	6	**572**	3.87	DS17	570	572	572	9	9
P	59	10	20	4	**534**	3.26	B&P	530	534	534	21	61
P	59	15	20	5	**591**	3.27	DS17	585	591	591	14	39
P	64	10	22	4	**575**	7.11	B&P	575	575	575	1	8
P	69	10	24	4	**602**	6.38	DS17	602	602	602	1	30
P	75	4	26	2	**556**	4.3	B&P	554	556	556	20	382
P	75	5	26	2	**556**	4.3	DS17	556	556	556	1	71
P	100	4	34	2	**649**	4.42	DS17	648	649	649	6	1899
G	261	25	88	9	3196	-	DS17	-	-	-	0	3600
C	100	10	34	4	**598**	1.48	DS17	592	598	598	49	1707
C	120	7	41	3	681	-	DS17	-	-	-	0	3600
C	150	12	51	4	**756**	5.97	B&P†	-	-	-	0	3600
C	199	16	67	6	874	-	DS17	-	-	-	0	3600

Table 12: Detailed results for the `GVRP-3` instances, subsets `P` and `GC`.

I.3. Detailed Results for the `Golden-Bat` Instances

Detailed results for the `Golden-Bat` instances are given in Tables 13 to 16, analogous to Section I.2 (without the number of vehicles k in the original CVRP instance).

Instance							Branch-and-Price results				
	n	N	m	BKS	*Gap to CluVRP*	*First found by*	*LB* root	*LB* tree	*UB*	#B&B nodes	Time *T*
Golden1	240	17	4	**4640**	3.95	B&P	4640	4640	4640	1	304
Golden1	240	18	4	4645	-	B&P	4610	4614	4645	10	3600
Golden1	240	19	4	4650	-	B&P	4621	4623	4650	6	3600
Golden1	240	21	4	4650	-	B&P†	4621	4621	-	4	3600
Golden1	240	22	4	4677	-	B&P†	4625	4625	-	3	3600
Golden1	240	25	4	4734	-	DS17	4609	4609	-	1	3600
Golden1	240	27	4	4708	-	B&P†	4620	4620	-	2	3600
Golden1	240	31	4	4766	-	DS17	4632	4632	-	1	3600
Golden1	240	35	4	4720	-	DS17	-	-	-	0	3600
Golden1	240	41	4	4710	-	DS17	-	-	-	0	3600
Golden1	240	49	4	4670	-	DS17	-	-	-	0	3600
Golden2	320	22	4	7434	-	B&P†	-	-	-	0	3600
Golden2	320	23	4	**7369**	4.21	B&P	7369	7369	7369	1	2836
Golden2	320	25	4	7369	-	B&P†	-	-	-	0	3600
Golden2	320	27	4	7480	-	DS17	-	-	-	0	3600
Golden2	320	30	4	7485	-	DS17	-	-	-	0	3600
Golden2	320	33	4	7471	-	DS17	-	-	-	0	3600
Golden2	320	36	4	7447	-	DS17	-	-	-	0	3600
Golden2	320	41	4	7450	-	DS17	-	-	-	0	3600
Golden2	320	46	4	7497	-	DS17	-	-	-	0	3600
Golden2	320	54	4	7487	-	DS17	-	-	-	0	3600
Golden2	320	65	4	7477	-	DS17	-	-	-	0	3600
Golden3	400	27	4	10389	-	DS17	-	-	-	0	3600
Golden3	400	29	4	10232	-	DS17	-	-	-	0	3600
Golden3	400	31	4	10273	-	DS17	-	-	-	0	3600
Golden3	400	34	4	10292	-	DS17	-	-	-	0	3600
Golden3	400	37	4	10255	-	DS17	-	-	-	0	3600
Golden3	400	41	4	10275	-	DS17	-	-	-	0	3600
Golden3	400	45	4	10252	-	DS17	-	-	-	0	3600
Golden3	400	51	4	10292	-	DS17	-	-	-	0	3600
Golden3	400	58	4	10350	-	DS17	-	-	-	0	3600
Golden3	400	67	4	10289	-	DS17	-	-	-	0	3600
Golden3	400	81	4	10311	-	DS17	-	-	-	0	3600
Golden4	480	33	4	13119	-	DS17	-	-	-	0	3600
Golden4	480	35	4	13205	-	DS17	-	-	-	0	3600
Golden4	480	37	4	13092	-	DS17	-	-	-	0	3600
Golden4	480	41	4	13011	-	DS17	-	-	-	0	3600
Golden4	480	44	4	13115	-	DS17	-	-	-	0	3600
Golden4	480	49	4	13133	-	DS17	-	-	-	0	3600
Golden4	480	54	4	13124	-	DS17	-	-	-	0	3600
Golden4	480	61	4	13190	-	DS17	-	-	-	0	3600
Golden4	480	69	4	13294	-	DS17	-	-	-	0	3600
Golden4	480	81	4	13235	-	DS17	-	-	-	0	3600
Golden4	480	97	4	13350	-	DS17	-	-	-	0	3600
Golden5	200	14	4	**6970**	8.55	B&P	6970	6970	6970	1	212
Golden5	200	15	3	**6742**	9.19	B&P	6742	6742	6742	1	280
Golden5	200	16	3	**6742**	10	B&P	6742	6742	6742	1	142
Golden5	200	17	3	**6862**	7.69	B&P	6862	6862	6862	1	380
Golden5	200	19	4	**6874**	9.27	B&P	6874	6874	6874	1	180
Golden5	200	21	4	**6816**	10.27	B&P	6816	6816	6816	1	666
Golden5	200	23	4	**6750**	11.68	B&P	6750	6750	6750	1	260
Golden5	200	26	4	**6704**	11.32	B&P	6704	6704	6704	1	647
Golden5	200	29	4	**6704**	9.53	B&P	6704	6704	6704	1	779
Golden5	200	34	4	**6684**	10.03	B&P*	-	-	-	0	3600
Golden5	200	41	4	**6557**	9.45	B&P	6557	6557	6557	1	2010

Table 13: Detailed results for the `Golden-Bat` instances `1-5`.

Instance							Branch-and-Price results				
	n	*N*	*m*	BKS	*Gap to CluVRP*	*First found by*	*LB* root	*LB* tree	*UB*	#B&B nodes	Time *T*
Golden6	280	19	3	**8115**	5.9	B&P	8115	8115	8115	1	1110
Golden6	280	21	3	**8119**	5.9	B&P	8119	8119	8119	1	901
Golden6	280	22	3	**8107**	6.23	B&P	8107	8107	8107	1	1053
Golden6	280	24	4	**8316**	6.07	B&P	8316	8316	8316	1	2491
Golden6	280	26	4	**8249**	7.42	B&P	8249	8249	8249	1	2568
Golden6	280	29	4	8395	-	DS17	-	-	-	0	3600
Golden6	280	32	4	8290	-	DS17	-	-	-	0	3600
Golden6	280	36	4	8383	-	DS17	-	-	-	0	3600
Golden6	280	41	4	8405	-	DS17	-	-	-	0	3600
Golden6	280	47	4	8349	-	DS17	-	-	-	0	3600
Golden6	280	57	4	8461	-	DS17	-	-	-	0	3600
Golden7	360	25	3	**9318**	5.92	B&P†	-	-	-	0	3600
Golden7	360	26	3	**9295**	6	B&P†	-	-	-	0	3600
Golden7	360	28	3	9581	-	DS17	-	-	-	0	3600
Golden7	360	31	4	**9418**	6.02	B&P†	-	-	-	0	3600
Golden7	360	33	4	9685	-	DS17	-	-	-	0	3600
Golden7	360	37	4	**9395**	7.26	B&P†	-	-	-	0	3600
Golden7	360	41	4	9664	-	DS17	-	-	-	0	3600
Golden7	360	46	4	9642	-	DS17	-	-	-	0	3600
Golden7	360	52	4	9694	-	DS17	-	-	-	0	3600
Golden7	360	61	4	9713	-	DS17	-	-	-	0	3600
Golden7	360	73	4	9602	-	DS17	-	-	-	0	3600
Golden8	440	30	4	10651	-	DS17	-	-	-	0	3600
Golden8	440	32	4	10640	-	DS17	-	-	-	0	3600
Golden8	440	34	4	10682	-	DS17	-	-	-	0	3600
Golden8	440	37	4	10660	-	DS17	-	-	-	0	3600
Golden8	440	41	4	10692	-	DS17	-	-	-	0	3600
Golden8	440	45	4	10667	-	DS17	-	-	-	0	3600
Golden8	440	49	4	10732	-	DS17	-	-	-	0	3600
Golden8	440	56	4	10726	-	DS17	-	-	-	0	3600
Golden8	440	63	4	10747	-	DS17	-	-	-	0	3600
Golden8	440	74	4	10755	-	DS17	-	-	-	0	3600
Golden8	440	89	4	10714	-	DS17	-	-	-	0	3600
Golden9	255	18	4	**281**	6.33	B&P	280	281	281	19	1287
Golden9	255	19	4	**279**	6.69	B&P	279	279	279	2	209
Golden9	255	20	4	**276**	6.76	B&P	276	276	276	1	112
Golden9	255	22	4	**276**	4.83	B&P	276	276	276	1	217
Golden9	255	24	4	**276**	4.83	B&P	276	276	276	1	175
Golden9	255	26	4	**273**	5.21	B&P	273	273	273	3	465
Golden9	255	29	4	**273**	6.51	B&P	273	273	273	1	985
Golden9	255	32	4	**273**	8.08	B&P	273	273	273	1	1650
Golden9	255	37	4	**273**	7.14	B&P*	273	273	-	7	3600
Golden9	255	43	4	281	-	DS17	270	270	-	3	3600
Golden9	255	52	4	279	-	DS17	-	-	-	0	3600
Golden10	323	22	4	**346**	5.72	B&P	346	346	346	2	923
Golden10	323	24	4	**346**	4.16	B&P	346	346	346	3	1014
Golden10	323	25	4	**346**	3.62	B&P	346	346	346	6	1114
Golden10	323	27	4	**346**	4.16	B&P	346	346	346	4	1360
Golden10	323	30	4	**347**	5.45	B&P	347	347	347	2	1848
Golden10	323	33	4	**344**	7.77	B&P	344	344	344	1	2725
Golden10	323	36	4	**344**	10.65	B&P†	-	-	-	0	3600
Golden10	323	41	4	363	-	DS17	-	-	-	0	3600
Golden10	323	47	4	360	-	DS17	-	-	-	0	3600
Golden10	323	54	4	357	-	DS17	-	-	-	0	3600
Golden10	323	65	4	354	-	DS17	-	-	-	0	3600

Table 14: Detailed results for the Golden-Bat instances 6-10.

Instance							Branch-and-Price results				
	n	N	m	BKS	*Gap to CluVRP*	*First found by*	*LB* root	*LB* tree	*UB*	#B&B nodes	Time *T*
Golden11	399	27	5	**434**	5.03	B&P†	434	434	-	5	3600
Golden11	399	29	5	**434**	4.62	B&P*	434	434	-	3	3600
Golden11	399	31	5	**433**	4.84	B&P	433	433	433	1	2661
Golden11	399	34	5	**427**	6.15	B&P†	-	-	-	0	3600
Golden11	399	37	5	444	-	DS17	-	-	-	0	3600
Golden11	399	40	5	444	-	DS17	-	-	-	0	3600
Golden11	399	45	5	442	-	DS17	-	-	-	0	3600
Golden11	399	50	5	442	-	DS17	-	-	-	0	3600
Golden11	399	58	5	445	-	DS17	-	-	-	0	3600
Golden11	399	67	5	446	-	DS17	-	-	-	0	3600
Golden11	399	80	5	446	-	DS17	-	-	-	0	3600
Golden12	483	33	5	529	-	DS17	-	-	-	0	3600
Golden12	483	35	5	531	-	DS17	-	-	-	0	3600
Golden12	483	38	5	531	-	DS17	-	-	-	0	3600
Golden12	483	41	5	532	-	DS17	-	-	-	0	3600
Golden12	483	44	5	530	-	DS17	-	-	-	0	3600
Golden12	483	49	5	533	-	Bat14	-	-	-	0	3600
Golden12	483	54	5	535	-	Bat14	-	-	-	0	3600
Golden12	483	61	5	535	-	Vidal15	-	-	-	0	3600
Golden12	483	70	5	533	-	Vidal15	-	-	-	0	3600
Golden12	483	81	5	535	-	Vidal15	-	-	-	0	3600
Golden12	483	97	5	544	-	Vidal15	-	-	-	0	3600
Golden13	252	17	4	**530**	3.99	B&P	530	530	530	1	116
Golden13	252	19	4	**521**	5.1	B&P	521	521	521	1	189
Golden13	252	20	4	**521**	4.93	B&P	521	521	521	1	192
Golden13	252	22	4	**523**	4.56	B&P	523	523	523	1	203
Golden13	252	23	4	**523**	4.56	B&P	523	523	523	1	215
Golden13	252	26	4	**523**	3.51	B&P	523	523	523	1	118
Golden13	252	29	4	**522**	3.33	B&P	522	522	522	5	1483
Golden13	252	32	4	**521**	4.05	B&P	521	521	521	1	286
Golden13	252	37	4	**521**	4.4	B&P	521	521	521	3	2305
Golden13	252	43	4	**521**	5.79	B&P†	-	-	-	0	3600
Golden13	252	51	4	532	-	DS17	-	-	-	0	3600
Golden14	320	22	4	**665**	3.9	B&P	665	665	665	3	1814
Golden14	320	23	4	**662**	3.78	B&P	662	662	662	2	752
Golden14	320	25	4	**660**	2.65	B&P	660	660	660	1	637
Golden14	320	27	4	**660**	2.37	B&P	660	660	660	11	3067
Golden14	320	30	4	**660**	2.65	B&P†	660	660	-	5	3600
Golden14	320	33	4	672	-	DS17	660	660	-	3	3600
Golden14	320	36	4	668	-	DS17	-	-	-	0	3600
Golden14	320	41	4	**658**	4.64	B&P†	-	-	-	0	3600
Golden14	320	46	4	676	-	DS17	-	-	-	0	3600
Golden14	320	54	4	674	-	DS17	-	-	-	0	3600
Golden14	320	65	4	679	-	DS17	-	-	-	0	3600
Golden15	396	27	4	825	-	DS17	-	-	-	0	3600
Golden15	396	29	4	825	-	DS17	815	815	-	1	3600
Golden15	396	31	4	**813**	2.87	B&P	813	813	813	2	3176
Golden15	396	34	4	826	-	DS17	-	-	-	0	3600
Golden15	396	37	4	826	-	DS17	-	-	-	0	3600
Golden15	396	40	4	830	-	DS17	-	-	-	0	3600
Golden15	396	45	5	834	-	DS17	-	-	-	0	3600
Golden15	396	50	5	839	-	DS17	-	-	-	0	3600
Golden15	396	57	5	838	-	DS17	-	-	-	0	3600
Golden15	396	67	5	840	-	DS17	-	-	-	0	3600
Golden15	396	80	5	843	-	DS17	-	-	-	0	3600

Table 15: Detailed results for the `Golden-Bat` instances `11-15`.

Instance							Branch-and-Price results				
	n	N	m	BKS	*Gap to CluVRP*	*First found by*	*LB* root	*LB* tree	*UB*	#B&B nodes	Time *T*
Golden16	480	33	5	1013	-	DS17	-	-	-	0	3600
Golden16	480	35	5	1011	-	DS17	-	-	-	0	3600
Golden16	480	37	5	1007	-	DS17	-	-	-	0	3600
Golden16	480	41	5	1013	-	DS17	-	-	-	0	3600
Golden16	480	44	5	1018	-	DS17	-	-	-	0	3600
Golden16	480	49	5	1014	-	DS17	-	-	-	0	3600
Golden16	480	54	5	1012	-	DS17	-	-	-	0	3600
Golden16	480	61	5	1012	-	DS17	-	-	-	0	3600
Golden16	480	69	5	1012	-	Bat14	-	-	-	0	3600
Golden16	480	81	5	1017	-	DS17	-	-	-	0	3600
Golden16	480	97	5	1018	-	Bat14	-	-	-	0	3600
Golden17	240	17	3	**386**	7.66	B&P	386	386	386	1	132
Golden17	240	18	3	**385**	8.11	B&P	385	385	385	1	290
Golden17	240	19	3	**385**	8.77	B&P	385	385	385	1	220
Golden17	240	21	3	**385**	9.41	B&P	385	385	385	2	457
Golden17	240	22	3	**385**	9.2	B&P	385	385	385	1	372
Golden17	240	25	3	**382**	8.61	B&P	382	382	382	1	487
Golden17	240	27	3	**382**	7.73	B&P	382	382	382	3	1039
Golden17	240	31	4	**390**	7.36	B&P	390	390	390	1	661
Golden17	240	35	4	396	-	B&P†	389	389	-	2	3600
Golden17	240	41	4	**388**	5.83	B&P†	-	-	-	0	3600
Golden17	240	49	4	396	-	DS17	-	-	-	0	3600
Golden18	300	21	4	**558**	5.74	B&P	558	558	558	1	694
Golden18	300	22	4	**558**	6.06	B&P	558	558	558	1	781
Golden18	300	24	4	**558**	5.74	B&P	558	558	558	1	831
Golden18	300	26	4	**562**	4.75	B&P	562	562	562	1	974
Golden18	300	28	4	**558**	3.29	B&P*	-	-	-	0	3600
Golden18	300	31	4	**554**	4.15	B&P	554	554	554	1	2450
Golden18	300	34	4	**554**	4.81	B&P	554	554	554	1	1992
Golden18	300	38	4	**555**	5.29	B&P†	-	-	-	0	3600
Golden18	300	43	4	573	-	DS17	-	-	-	0	3600
Golden18	300	51	4	575	-	DS17	-	-	-	0	3600
Golden18	300	61	4	574	-	DS17	-	-	-	0	3600
Golden19	360	25	10	**886**	4.22	B&P	886	886	886	1	538
Golden19	360	26	10	**888**	3.9	B&P	888	888	888	1	1208
Golden19	360	28	4	**741**	8.29	B&P	741	741	741	1	1479
Golden19	360	31	4	**735**	9.37	B&P*	-	-	-	0	3600
Golden19	360	33	4	**727**	8.78	B&P	727	727	727	1	2719
Golden19	360	37	5	**732**	8.39	B&P	732	732	732	1	2612
Golden19	360	41	5	**730**	7.48	B&P†	-	-	-	0	3600
Golden19	360	46	5	752	-	DS17	-	-	-	0	3600
Golden19	360	52	5	**730**	8.75	B&P†	-	-	-	0	3600
Golden19	360	61	5	763	-	DS17	-	-	-	0	3600
Golden19	360	73	5	763	-	DS17	-	-	-	0	3600
Golden20	420	29	11	**1170**	4.1	B&P	1170	1170	1170	1	1099
Golden20	420	31	12	**1183**	3.98	B&P	1183	1183	1183	1	1080
Golden20	420	33	12	**1175**	2.73	B&P	1175	1175	1175	1	2381
Golden20	420	36	5	1033	-	DS17	-	-	-	0	3600
Golden20	420	39	5	1025	-	DS17	-	-	-	0	3600
Golden20	420	43	5	1017	-	DS17	-	-	-	0	3600
Golden20	420	47	5	1023	-	DS17	-	-	-	0	3600
Golden20	420	53	5	1022	-	DS17	-	-	-	0	3600
Golden20	420	61	5	1021	-	DS17	-	-	-	0	3600
Golden20	420	71	5	1025	-	DS17	-	-	-	0	3600
Golden20	420	85	5	1018	-	DS17	-	-	-	0	3600

Table 16: Detailed results for the `Golden-Bat` instances `16-20`.