Gutenberg School of Management and Economics

& Research Unit “Interdisciplinary Public Policy”

Discussion Paper Series

A Branch-and-Cut Algorithm for the Soft-Clustered Vehicle-Routing Problem

Katrin Heßler, Stefan Irnich

January 8, 2020

Discussion paper number 2001

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

Contact Details:

Katrin Heßler
Logistikmanagement
Johannes Gutenberg University Mainz
Jakob-Welder-Weg 9
55128 Mainz
Germany

khessler@uni-mainz.de

Stefan Irnich
Logistikmanagement
Johannes Gutenberg University Mainz
Jakob-Welder-Weg 9
55128 Mainz
Germany

irnich@uni-mainz.de

A Branch-and-Cut Algorithm for the Soft-Clustered Vehicle-Routing Problem

Katrin Heßler[*,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 is a variant of the classical capacitated vehicle-routing problem (CVRP) in which customers are partitioned into clusters and all customers of the same cluster must be served by the same vehicle. We introduce a novel symmetric formulation of the problem in which the clustering part is modeled with an asymmetric sub-model. We solve the new model with a branch-and-cut algorithm exploiting some known valid inequalities for the CVRP that can be adapted. In addition, we derive problem-specific cutting planes and new heuristic and exact separation procedures. For square grid instances in the Euclidean plane, we provide lower-bounding techniques and a reduction scheme that is also applicable to the respective traveling salesman problem. In comprehensive computational test on standard benchmark instances, we compare the different formulations and separation strategies in order to determine a best performing algorithmic setup. The computational results with this branch-and-cut algorithm show that several previously open instances can now be solved to proven optimality.

Key words: vehicle routing, clustered customers, branch-and-cut

1. Introduction

The *soft-clustered vehicle-routing problem* (SoftCluVRP) is a variant of the classical *capacitated vehicle-routing problem* (CVRP, Toth and Vigo 2014) in which customers are partitioned into clusters, and all customers of the same cluster must be served by the same vehicle. In contrast to the hard-clustered variant, where a cluster must be served completely before the next cluster is served, we consider the variant in which visits to customers of the same cluster can be interrupted by visits to customers of another cluster.

Both the CVRP and SoftCluVRP are defined over a complete undirected graph $G = (V, E)$ with the vertex set $V = \{0, 1, 2, \ldots, n\}$ and the edge set E. The vertex 0 denotes the *depot* and the other vertices $C = \{1, 2, \ldots, n\}$ denote the *customers*. A homogeneous fleet of m vehicles with capacity Q is hosted at the depot 0. For each edge $\{i, j\} \in E$, non-negative routing costs c_{ij} are given. A route $r = (i_0, i_1, \ldots, i_s, i_{s+1})$ is a cycle in G passing through the depot, i.e., $i_0 = i_{s+1} = 0$ and $i_1, \ldots, i_s$ are customers ($s \geq 1$). In the CVRP, a route is feasible if (i) all customers $i_1, \ldots, i_s$ are different and (ii) capacity constraints $\sum_{j=1}^{s} d_{i_j} \leq Q$ hold for given positive customer-specific demands d_i for $i \in C$.

The clustered variants require the definition of a partitioning of the vertex set: Let $V = V_0 \cup V_1 \cup V_2 \cup \ldots \cup V_N$ be such a partitioning, where $V_0 = \{0\}$ denotes the *depot cluster*. The *customer clusters* $V_h \subset C$ are index by $h \in H = \{1, 2, \ldots, N\}$ and are disjoint, i.e., $V_h \cap V_{h'} = \emptyset$ for all $h \neq h' \in H$. For any customer $i \in C$, the associated cluster is denoted by $h(i) \in H$, i.e., $i \in V_{h(i)}$. Each cluster V_h has an associated positive demand d_h and we define $d_0 = 0$ for the depot cluster V_0. For a route $r = (i_0, i_1, \ldots, i_s, i_{s+1})$, it is convenient to define $H(r)$ as the customer clusters *touched* by the route r, i.e., $H(r) = \{h \in H : V_h \cap \{i_1, \ldots, i_s\} \neq \emptyset\}$.

[*] Corresponding author.
Email addresses: khessler@uni-mainz.de (Katrin Heßler), irnich@uni-mainz.de (Stefan Irnich)

In the SoftCluVRP, a route is feasible if (i) all customers $i_1, \dots, i_s$ are different and (ii) capacity constraints $\sum_{h \in H(r)} d_h \leq Q$ hold (summing of the customer clusters touched by the route), and (iii) the soft-cluster constraints are fulfilled, i.e.,

$$V_h \subseteq \{i_1, \dots, i_s\} \qquad \forall h \in H(r).$$

Note that the *clustered vehicle-routing problem* (CluVRP, Battarra et al. 2014) requires that the hard-cluster constraints hold, i.e., for each $h \in H(r)$ there must exist an index $k \in \{1, 2, \dots, s - |V_h| + 1\}$ such that $V_h = \{i_k, i_{k+1}, \dots, i_{k+|V_h|-1}\}$.

In all cases (CVRP, CluVRP, SoftCluVRP), the task is to determine a cost-minimal set of m routes that together serve all customers, and herewith also all clusters, exactly once.

The very recent paper by Hintsch and Irnich (2020) surveys the pertinent literature. Therefore, we omit a comprehensive survey on clustered VRPs but briefly mention the most important findings: For the exact solution of the CluVRP, Battarra et al. (2014) developed a very competitive two-level exact algorithm computing optimal Hamiltonian paths through clusters for several entry and exit points in the subproblem level and combining them to route plans in the master level. The most powerful metaheuristics for the CluVRP are those of Vidal et al. (2015) and Hintsch and Irnich (2018) and they follow a similar two-level principle. For the problem considered in the paper at hand, the SoftCluVRP, the only tailored exact solution approach is the branch-and-price algorithm of Hintsch and Irnich (2020) using a MIP-based approach in the pricing subproblem instead of a shortest-path dynamic programming labeling algorithm. These results will serve as a benchmark for the newly developed branch-and-cut algorithm.

The only newer results not surveyed in (Hintsch and Irnich 2020) are the following: Hintsch et al. (2019) have developed a branch-price-and-cut for the soft-clustered variant of the capacitated arc-routing problem. Again, the pricing subproblem is solved by branch-and-cut as well as metaheuristics. Additional cutting planes on the route-based master formulation help to strengthen the linear relaxation. Hintsch (2019) has developed a *large-neighborhood search* (LNS) metaheuristic, which is the currently best-performing heuristic approach for the SoftCluVRP. We later compare against these results.

The contributions of the paper at hand are the following:

1. We introduce the first two-index formulation for the exact solution of the SoftCluVRP. The formulation decomposes into a standard routing part, a novel part ensuring vehicle-capacity and soft-cluster feasible solutions using a directed cluster graph, and a simple coupling between the two parts.
2. We analyze the impact of the fleet-size constraint on the validity of the formulation. Some additional constraints are mandatory to cope with non-minimal fleets. In addition, we exploit some non-trivial redundancies of the basic formulation.
3. The soft-cluster requirement leads to a new type of capacity cuts. These cuts are known for the CVRP and we derive a variant of the capacity cuts valid for the SoftCluVRP that are stronger than the straightforward adaptation of the capacity cuts for the CVRP.
4. Some SoftCluVRP instances from the benchmark of Golden et al. (1998) (adopted by Battarra et al. (2014)) are constructed on a square grid network. We provide lower-bounding techniques for SoftCluVRP instances that are especially strong for grid-based instances. For square grid instances in the Euclidean plane, we develop preprocessing techniques that allow to substantially reduce the edge set and the corresponding routing variables of the formulation.
5. With comprehensive computational test, we determine a competitive setup combining the multiple branch-and-cut components. The computational results on benchmark instances using SoftCluVRP benchmark instances from the literature show that the resulting branch-and-cut algorithm is competitive.

The remainder of this paper is structured as follows. In the next section, we present the new formulation for the SoftCluVRP. The components of the branch-and-cut algorithm including the families of valid inequalities and their heuristic and exact separation are detailed in Section 3. Lower-bounding and reduction techniques for the grid-based instances are presented in Section 4. In Section 5, we present the computational experiments, in which we configure the final branch-and-cut algorithm, analyze the influence of reduction for the grid instances on computational performance, and compare the results on all benchmark instances against those from the literature. Final conclusions are drawn in Section 6.

2. Two-Index Formulation

Since the SoftCluVRP is a relatively new problem, only a few models can be found in the scientific literature. A three-index formulation for the symmetric SoftCluVRP was recently presented by Hintsch and Irnich (2020), while another three-index formulation was presented by Defryn and Sörensen (2017) for the asymmetric version of the SoftCluVRP. These formulations use routing variables with a third index, say $k \in K = \{1, 2, \dots, m\}$, to refer to a specific vehicle (the first two indices describe the endpoints of a direct connection). The major drawback of three-index formulations is that they grow linearly with the fleet size and, more severely, they are inherently symmetric with respect to the numbering of the vehicles. Indeed, for a given solution, any permutation of the vehicle indices $k \in K$ produces one of $|K|!$ equivalent solutions. This makes a branch-and-bound-based approach as used in MIP solvers ineffective (Fischetti et al. 1995). The addition of symmetry-breaking constraints can only very partially mitigate the ineffectiveness of the MIP solver's branching decisions (Adulyasak et al. 2014).

Hintsch and Irnich (2020) also derive an extensive route-based formulation (a set-partitioning type of model) from the aforementioned three-index formulation via Dantzig-Wolfe decomposition and subsequent aggregation over vehicles. The drawback of this type of formulation is that a sophisticated branch-and-price algorithm is needed to cope with the huge number of route variables.

The idea of the new formulation we present in the following is to exploit that already well-performing standard models for the CVRP are known. We use the symmetric formulation of Laporte et al. (1985) that has non-negative integer routing variables x_{ij} for all edges $\{i, j\} \in E$. Note that all benchmark sets for the SoftCluVRP comprise symmetric instances.

To enforce the clustering constraints, we assume that the elements of the cluster index set H are completely ordered, which holds true if, e.g., $H \subset \mathbb{N}$. We introduce the directed acyclic *cluster graph* $D = (H, A)$ with the arc set $A = \{(h, h') : h, h' \in H, h < h'\}$. The new formulation uses additional binary variables y_a for each $a = (h, h') \in A$ that indicates whether clusters V_h and $V_{h'}$ are served by the same vehicle ($y_a = 1$), or not ($y_a = 0$). While routing variables are symmetric, we intentionally model with asymmetric y-variables. Each component in the subgraph of D spanned by the positive y-variables, i.e., by $A_{y=1} = \{a \in A : y_a = 1\}$, represents a subset of clusters served by one vehicle.

The orientation in the cluster graph enables an *Miller-Tucker-Zemlin* (MTZ)-based (Miller et al. 1960) modeling approach with continuous resource variables u_h for $h \in H$ that accumulate the demand served by each route. The new formulation is:

$$\min \sum_{\{i,j\} \in E} c_{ij} x_{ij} \tag{1a}$$

$$\text{subject to} \sum_{\{i,j\} \in \delta(i)} x_{ij} = 2 \qquad \forall i \in C \tag{1b}$$

$$\sum_{\{0,j\} \in \delta(0)} x_{0j} = 2m \tag{1c}$$

$$\sum_{\{i,j\} \in \delta(S)} x_{ij} \geq 2r(S) \qquad \forall S \subseteq C, S \neq \emptyset \tag{1d}$$

$$x_{ij} \in \{0, 1\} \qquad \forall \{i, j\} \in E \setminus \delta^*(0) \tag{1e}$$

$$x_{0j} \in \{0, 1, 2\} \qquad \forall \{0, j\} \in \delta^*(0) \tag{1f}$$

$$x_{ij} \leq y_{h(i),h(j)} \qquad \forall \{i, j\} \in E \setminus \delta(0) : h(i) < h(j) \tag{1g}$$

$$u_h - u_{h'} + Q y_{hh'} \leq Q - d_{h'} \qquad \forall (h, h') \in A \tag{1h}$$

$$d_h \leq u_h \leq Q \qquad \forall h \in H \tag{1i}$$

$$y_{hh'} \geq y_{hh''} + y_{h'h''} - 1 \qquad \forall (h, h'), (h', h'') \in A \tag{1j}$$

$$y_{h'h''} \geq y_{hh'} + y_{hh''} - 1 \qquad \forall (h,h'),(h',h'') \in A \tag{1k}$$

$$y_{hh''} \geq y_{hh'} + y_{h'h''} - 1 \qquad \forall (h,h'),(h',h'') \in A \tag{1l}$$

$$y_a \in \{0,1\} \qquad \forall a \in A \tag{1m}$$

The first part (1a)–(1f) is the CVRP formulation of Laporte et al. (1985): The objective (1a) minimizes the routing costs. Constraints (1b) ensure that each customer is visited once, and constraints (1c) ensure that exactly m vehicles leave and return to the depot. In the *capacity cuts* (1d), the set $\delta(S)$ is the set of edges with exactly one endpoint in S, and the number $r(S)$ describes the minimum number of vehicles needed to feasibly serve the customer subset S. In the CVRP, it suffices to bound $r(S)$ from below by computing $\lceil d(S)/Q \rceil$, where $d(S)$ is the sum of the demands of all customers in S. Since in the SoftCluVRP the demand is associated with clusters, we can arbitrarily distribute the demand d_h of every cluster V_h onto its customers, e.g., defining $d_i = d_h/|V_h|$ for all $i \in V_h$ and $h \in H$. We discuss the role of the capacity cuts in more detail in Section 3.1. The capacity cuts prohibit subtours not including the depot as well as subtours that serve more customers than possible when respecting vehicle capacity. The domains of the routing variables are given by (1e) and (1f). Note that a back-and-forth route $(0,j,0)$ is only feasible if $j \in C$ is a customer that forms a singleton cluster, i.e., $H_{h(j)} = \{j\}$. Therefore, we define $\delta^*(0) = \{\{0,j\} \in \delta(0) : H_{h(j)} = \{j\}\}$, where $\delta(0)$ is the set of all edges incident to the depot 0.

The last part (1h)–(1m) of the model provides a description of feasible combinations of clusters to be served by the same vehicle. The MTZ-like constraints (1h) impose $u_{h'} \geq u_h + d_{h'}$ for $y_{hh'} = 1$. It is crucial here that the set H is ordered and that (1h) is imposed only for one direction, i.e., for $(h,h') \in A$ and not for $(h',h) \notin A$, because otherwise the two constraints and $y_{hh'} = y_{h'h} = 1$ would directly imply the contraction $u_{h'} > u_{h'}$ and $u_{h'} < u_{h'}$. The constraints (1i) describe the domain of the u-variables and guarantee that the capacity Q is not exceeded. The constraints (1j)–(1l) are transitivity-enforcing constraints for the y-variables.

The coupling between the x- and y-variables is established via constraints (1g).

Proposition 1. *If m is the minimum number of vehicles needed to serve all customers C, then every feasible solution to formulation (1) is a feasible solution to the given SoftCluVRP instance.*

Proof. Note first that the minimum number of vehicles needed to serve C can be obtained as the solution value $m_{\min}$ of a bin-packing instance with bins of capacity Q and items with weights $(d_h)_{h \in H}$.

A feasible solution to model (1) may have the following defect: The y-variables can impose a connected component $O \subset H$ of the cluster graph D that is served by more than one vehicle, i.e., the x-variables impose more than one route in $\{0\} \cup \bigcup_{h \in O} V_h$. However, the connected component O respects the capacity constraint, i.e., $d(O) \leq Q$ holds true due to (1h)–(1m). As all components of D imposed by the y-variables together partition the set H into a feasible bin-packing solution, the number of connected components cannot be smaller than $m_{\min}$. If $m = m_{\min}$, the pigeonhole principle tells us that only one vehicle can serve each connected component. Therefore, a feasible SoftCluVRP solution results. □

A trivial improvement to formulation (1) is to add

$$y_{hh'} = 0 \qquad \forall (h,h') \in A : d_h + d_{h'} > Q. \tag{1n}$$

This can also be established by eliminating the y-variable that are set to zero from formulation (1), modifying the affected constraints (1g) and (1j)–(1l), and eliminating redundant constraints (1h).

Non-minimal Fleet. In many vehicle-routing problems, the primary objective is to minimize the number of vehicles. Therefore, Proposition 1 shows that formulation (1) is valid and relevant for the standard application, i.e., when $m = m_{\min}$.

Minimizing the number of vehicles and minimizing routing costs are in general conflicting objectives. With the focus on the second objective (routing costs), a relaxed version of the SoftCluVRP is one in which the minimum fleet-size constraint $\sum_{\{0,j\} \in \delta(0)} x_{0j} = 2m_{\min}$ is replaced by $\sum_{\{0,j\} \in \delta(0)} x_{0j} \leq 2m$ with a fleet-size limit $m > m_{\min}$. We denote this relaxation by SoftCluVRP$^{\leq m}$. The following proposition states that formulation (1) is also valid for the SoftCluVRP$^{\leq m}$ under some mild assumptions.

Proposition 2. *If the* depot-triangle inequality *holds for the routing costs of a given SoftCluVRP*$^{\leq m}$ *instance, i.e.,* $c_{ij} \leq c_{0i} + c_{0j}$ *for all* $i, j \in C$*, then there exists, for every feasible solution to formulation* (1)*, a feasible solution to the SoftCluVRP*$^{\leq m}$ *with identical or lower cost.*

Proof. As shown in the proof of Proposition 1, a feasible solution to formulation (1) may only have the defect that more than one route/vehicle serves a connected component O in the subgraph of D spanned by the positive y-variables. In this case, a pair of edges $\{0, i\}$ and $\{0, j\}$ belonging to two different routes can be replaced by the edges $\{i, j\}$ so that the two routes are merged into one. This route is feasible, because constraints (1h)–(1m) ensure $d(O) \leq Q$. Moreover, the depot-triangle inequality implies that after the replacement the new route is never more costly than the two merged routes. Iterative replacements finally lead to a single feasible route per component. The constructed new solution is then feasible for the SoftCluVRP$^{\leq m}$. □

If neither the assumptions of Proposition 1 nor of Proposition 2 are fulfilled, it is still possible to use formulation (1) for the SoftCluVRP$^{\leq m}$. In this case, the following class of *single-route inequalities* must be added to model (1):

$$\sum_{\substack{\{0,i\}\in\delta(0),\\ h(i)\in H(T)}} x_{0i} + 2\sum_{a\in A(T)} y_a \leq 2|H(T)| \qquad \forall T = (H(T), A(T)) \text{ tree in } D \tag{2}$$

Regarding the validity of (2), consider an integer feasible solution $(\bar{x}, \bar{y})$ to the SoftCluVRP or SoftCluVRP$^{\leq m}$. For an arbitrary tree $T = (H(T), A(T))$ in D, the customers $\bigcup_{h\in H(T)} V_h$ are served by a certain number of vehicles, say $m(T)$ vehicles. This implies

$$\sum_{\substack{\{0,i\}\in\delta(0),\\ h(i)\in H(T)}} \bar{x}_{0i} \leq 2m(T).$$

Moreover, the subgraph $T_{\bar{y}}$ of the tree spanned by arcs a with $\bar{y}_a = 1$ must decompose into $m(T)$ or more components. The latter implies that

$$\sum_{a\in A(T)} \bar{y}_a \leq |H(T)| - m(T) \tag{3}$$

holds true. Adding the first and twice the second inequality yields the inequality (2) for $(\bar{x}, \bar{y})$.

Note that inequalities (3), for all trees $(H(T), A(T))$, can be used to ensure capacity-feasible solutions, i.e., they can replace the MTZ-part (1h)–(1i) of formulation (1). We denote (3) as *tree-capacity constraints* in the following.

Proposition 3. *For every feasible solution to model* (1)*–*(2) *with a relaxed fleet-size constraint* $\sum_{\{0,j\}\in\delta(0)} x_{0j} \leq 2m$ *instead of* (1c) *defined by an arbitrary value* $m \geq m_{\min}$*, the solution is also feasible for the SoftCluVRP or SoftCluVRP*$^{\leq m}$*.*

Proof. Consider a feasible solution $(\bar{x}, \bar{y})$ to model (1)–(2) with a relaxed fleet-size constraint. The positive $\bar{y}$-values decompose D into a number of components. Consider an arbitrary component $O = \{h_1, h_2, \ldots, h_{|O|}\} \subset H$ and its ordered elements $h_1 < h_2 < \cdots < h_{|O|}$. Since O is a single connected component, the transitivity constraints (1j) and (1k) impose $\bar{y}_{h_1,h_2} = \bar{y}_{h_2,h_3} = \cdots = \bar{y}_{h_{|O|-1},h_{|O|}} = 1$. The path $(h_1, h_2, \ldots, h_{|O|})$ is also a tree T. For this tree $T = (O, \{(h_k, h_{k+1}) : k = 1, \ldots, |O|-1\})$, inequality (2) imposes $\sum_{(0,i)\in\delta(0),h(i)\in O} \bar{x}_{0i} \leq 2$ showing that the $\bar{x}$-values define only a single route serving component O. □

Redundancy. Formulation (1) has some redundant transitivity constraints. We denote by (R) the model (1) without constraints (1l). The relationship between the two formulations is characterized in the following two propositions:

Proposition 4. *Formulations (1) and (R) have the same set of integer solutions regarding the projection onto the x-variables.*

For the sake of clarity, all longer proofs (such as the one for Proposition 4) have been moved to the Appendix section.

Proposition 5. *The linear relaxations of formulations (1) and (R) have the same set of solutions regarding the projection onto the x-variables.*

3. Branch-and-Cut Algorithm

Formulations (1) and (1)–(2) are not directly solvable with a MIP solver, because they contain some large-sized families of constraints. This section describes how constraints of these families can be added dynamically using separation procedures. We distinguish between (possibly infeasible) integer solutions $(\bar{x}, \bar{y}) \in \mathbb{Z}^{|E|+|A|}$ and fractional solutions $(\bar{x}, \bar{y}) \in \mathbb{R}^{|E|+|A|}$ for

- capacity cuts (1d) (exponential in $|V|$),
- single-route inequalities (2) (exponential in $|H|$),
- tree-capacity constraints (3) (exponential in $|H|$),
- transitivity constraints (1j) and (1k) (cubic in $|H|$).

In the branch-and-cut implementation, we consider an inequality as violated only if the degree of violation (difference between right-hand and left-hand side) exceeds $\varepsilon = 10^{-4}$.

3.1. Capacity Cuts

For an integer solution $(\bar{x}, \bar{y}) \in \mathbb{Z}^{|E|+|A|}$, we determine the graph $G_{\bar{x}}$ spanned by the positive $\bar{x}$-variables. Subsequently, we remove the depot vertex 0 leading to the induced graph $G_{\bar{x}}[V \setminus \{0\}] = G_{\bar{x}}[C]$. The connected components of $G_{\bar{x}}[C]$ are subsets of customers served together. Each such subset $S \subset V \setminus \{0\}$ may violate the associated capacity constraint, i.e., if the component forms a cycle (=subtour not including the depot). Since the LHS of the capacity cut is equal to zero in this case, the capacity cut (1d) is violated independent of the value of $r(S)$.

However, we strive for a tight value $r(S)$ in order to add a strong valid inequality. It was already mentioned in Section 2 that different lower bounds on the minimum number of vehicles needed to serve some customers can be computed by arbitrarily distributing the cluster's demands onto the associated customers. For a given customer subset $S \subset V \setminus \{0\}$, we define the *clusters touched* as

$$H(S) = \{h \in H : S \cap V_h \neq \varnothing\}. \tag{4}$$

An optimal distribution of the demand is one that assigns the entire cluster demand for $h \in H(S)$ to only a single customer $i_h \in S \cap V_h$. This is summarized in the following proposition:

Proposition 6. *For any $S \subset V \setminus \{0\}$ with $S \neq \varnothing$, a largest lower bound $r(S)$ on the minimum number of vehicles needed to serve the customers S results from considering the demands $(d_h)_{h \in H(S)}$:*

(i) *the exact value $r(S)$ results from solving a bin-packing problem with bins of capacity Q and weights $(d_h)_{h \in H(S)}$;*

(ii) *a valid lower bound for $r(S)$ sufficient for formulation (1) is given by $\left\lceil \sum_{h \in H(S)} d_h / Q \right\rceil$.*

For a fractional solution $(\bar{x}, \bar{y}) \in \mathbb{R}^{|E|+|A|}$, we apply the following series of heuristic separation procedures. First, we again consider $G_{\bar{x}}[C]$ and its components again. For each component $S \subset C$, we test $\bar{x}(\delta(S)) < 2\left\lceil \sum_{h \in H(S)} d_h / Q \right\rceil$ (and herewith $\bar{x}(\delta(S)) < 2r(S)$) and, if true, a violated capacity cut (1d) is found.

Second, we apply two heuristic procedures that work according to the subset-first check-second principle, i.e., promising subsets S are computed first and the violation $\bar{x}(\delta(S)) < 2r(S)$ is checked for each computed subset S. The first heuristic is the probabilistic graph contraction algorithm of Karger (1993), summarized in Algorithm 1. The idea is to iterative contract edges of G where edges $e \in E$ with a higher value $\bar{x}_e$ are

Algorithm 1: Karger's contraction algorithm

input : graph $G = (V, E)$
output: sets to check S

for $e \in E$ **do**
  $p_e \leftarrow \bar{x}_e / \sum_{f \in E} \bar{x}_f$;
for $|V|$ *iterations* **do**
  $G' \leftarrow G$;
  while G' *has more than two vertices* **do**
    Choose an edge $e \in E(G')$ with probability p_e;
    Contract e into a single vertex, i.e., $G' \leftarrow G'/e$;
  $(S, \bar{S}) \leftarrow$ subsets of vertices represented by the two remaining vertices of G', $0 \in \bar{S}$;
  Compute $r(S)$;
  Check S regarding $\bar{x}(\delta(S)) < 2r(S)$;

chosen with a higher probability (proportional to $\bar{x}_e$). In the contraction step for an edge $\{i, j\} = e \in E$, the vertices i and j, and later subsets S_i and S_j containing i and j, respectively, are replaced by the union $S_i \cup S_j$. Edges $\{k, S_i\}$ and $\{k, S_j\}$ with $k \in V \setminus (S_i \cup S_j)$ are merged into a single edge with weight $\bar{x}_{k,S_i} + \bar{x}_{k,S_j}$. The graph contraction algorithm and the testing of violated SEC is repeated $|V|$ times, and a most-violated capacity cut is added.

The second heuristic uses the heuristic separation procedures for the CVRP publicly available in the library of Lysgaard et al. (2004). Potential subsets S result from the solution of some max-flow/min-cut problems. As the library is tailored to the CVRP, we distribute cluster demands d_h equally onto the customers $i \in V_h$. Both heuristic procedures (Karger, Lysgaard) are used independently.

If no violated capacity cut has been found with the heuristics, we apply the following exact MIP-based separation algorithm: the MIP simultaneously determines the subset $S \subset C$, computes the lower bound on $r(S)$ given by Proposition 6(ii), and maximizes the violation (if any) $2r(S) - \bar{x}(\delta(S))$. The MIP generalizes ideas first presented by Ahr (2004) and later refined by Martinelli et al. (2013) for exactly separating capacity cuts for the capacitated arc-routing problem. The formulation of the separation problem uses five types of variables: Binary variables s_i for $i \in V$ are indicator variables describing whether the vertex i belongs to the unknown set S. Similarly, binary variables y_h describe the clusters $H(S)$ touched by S, i.e., $H(S) = \{h \in H : y_h = 1\}$. Variables z_e for $e \in E$ are indicators for the cut set, i.e., $z_e = 1$ iff $e \in \delta(S)$. Moreover, the integer variable r describes (the lower bound on) $r(S)$ and the non-negative continuous variable $f < 1$ describes the fractional difference between $\lceil d(S)/Q \rceil$ and $d(S)/Q$.

$$\max \quad 2r - \sum_{e \in E} \bar{x}_e z_e \tag{5a}$$

$$\text{subject to} \quad s_0 = 0 \tag{5b}$$

$$z_e - s_i + s_j \geq 0 \qquad \forall e = \{i, j\} \in E, i \neq 0 \tag{5c}$$

$$z_e - s_j + s_i \geq 0 \qquad \forall e = \{i, j\} \in E, j \neq 0 \tag{5d}$$

$$s_i + s_j - z_e \geq 0 \qquad \forall e = \{i, j\} \in E \tag{5e}$$

$$s_i + s_j + z_e \leq 2 \qquad \forall e = \{i, j\} \in E \setminus \delta(0) \tag{5f}$$

$$\sum_{i \in V_h} s_i - y_h \geq 0 \qquad \forall h \in H \tag{5g}$$

$$r = \sum_{h \in H} (d_h/Q) y_h + f \tag{5h}$$

$$s_i \in \{0, 1\} \qquad \forall i \in V \setminus \{0\} \tag{5i}$$

$$0 \le z_e \le 1 \qquad \forall e \in E \tag{5j}$$
$$y_h \in \{0,1\} \qquad \forall h \in H \tag{5k}$$
$$r \ge 0, \text{integer} \tag{5l}$$
$$0 \le f \le 1 - 1/Q \tag{5m}$$

The objective (5a) minimizes the violation of a capacity cut described by $S = \{i \in V : s_i = 1\}$. Forcing $s_0 = 0$ ensures that $S \subset V \setminus \{0\} = C$ holds. The coupling of the z-variables with the s-variables is established via (5c)–(5f), where (5c) and (5d) force z_e to one if $s_i \neq s_j$, i.e., $e = \{i, j\} \in \delta(S)$, while (5e) and (5f) force z_e to zero if $s_i = s_j$. To correctly consider that a cluster V_h for some $h \in H$ is touched, constraints (5g) couple the vertex and cluster indicator variables. The correct value of r is guaranteed with constraint (5h). The domains of the variables are described by (5i)–(5m).

Repeatedly solving the exact separation formulation (5) with a MIP solver consumes considerable computation time. Therefore, the exact separation is only used at the root node of the branch-and-cut algorithm. Moreover, we do not call the exact separation routine if the lower bound is not improved by more than 0.01 % within the last ten iterations.

3.2. *Single-Route Inequalities*

Recall that single-route inequalities (2) are mandatory only if the fleet is larger than needed and the depot-triangle inequality does not hold (cf. Propositions 1 and 2). As we found them violated only rarely, we inspect only integer solution $(\bar{x}, \bar{y}) \in \mathbb{Z}^{|E|+|A|}$. For every connected component O of the digraph $D_{\bar{y}}$ spanned by the arcs a with $\bar{y}_a = 1$, let $H(O)$ be the vertex set of the component O. We can take any spanning tree $T = (H(O), A(O))$ spanning $H(O)$ in $D_{\bar{y}}$ and check whether (2) is violated which is the case if more than two edges $\{0, i\} \in \delta(0)$ are chosen.

3.3. *Tree-Capacity Constraints*

The tree-capacity constraints (3) can replace the MTZ constraints (1h)–(1i). Note that there are a quadratic number of MTZ constraints, while the number of tree-capacity constraints is exponential in $|H|$. Therefore, the latter family of constraints must be added dynamically.

For an integer solution $(\bar{x}, \bar{y}) \in \mathbb{Z}^{|E|+|A|}$, we consider all connected components O of $D_{\bar{y}}$ spanned by the arcs a with $\bar{y}_a = 1$ and spanning trees $T = (H(O), A(O))$ (as in the previous Section 3.2). For the sake of acceleration, the value $m(T)$ is approximated by $\lceil d(O)/Q \rceil$ instead of solving a bin-packing problem. As the number of components is small, all violated tree-capacity constraints (3) are added at the same time.

For fractional solutions $(\bar{x}, \bar{y}) \in \mathbb{R}^{|E|+|A|}$, we use a heuristic inspired by the procedure for integer solutions. Also here we consider all connected components O of $D_{\bar{y}}$. Within each component O, the tree T maximizing the left-hand-side of constraints (3) is computed as a maximum spanning tree using Kruskal's algorithm.

A special case of the tree-capacity constraints are constraints

$$\sum_{h' \in H : h < h'} y_{hh'} + \sum_{h' \in H : h' < h} y_{h'h} \le N - 1 \qquad \text{for all } h \in H$$

because all ingoing and outgoing arcs of h form a spanning tree in D. We add these $N = |H|$ constraints at initialization to accelerate the solution process.

Overall, later computational experiments will compare three setups: (1) only MTZ constraints statically added at the beginning, (2) only tree-capacity constraints added dynamically, and (3) the combination of static MTZ and dynamic tree-capacity constraints. The last strategy may lead to a faster branch-and-cut algorithm as neither MTZ dominate tree-capacity constraints on fractional solutions, nor vice versa.

3.4. Transitivity Constraints

Propositions 4 and 5 have proven that the transitivity constraints (1l) are completely redundant. However, the two groups (1j) and (1k) of transitivity constraints are each of cubic size in $|H|$. Therefore, one may either add transitivity constraints to the model right from the beginning or add them only when violated. In the later computational experiments, we test the following two strategies: Either, all transitivity constraints (1j) and (1k) are present in formulation (1). This is the *static* case.

Alternatively, formulation (1) is initialized without constraints (1j) and (1k), and only violated constraints are separated and added dynamically. The identification of violated transitivity constraints can be done by straightforward direct inspection consuming $|H|^3$ time.

In pretests we found that many transitivity constraints are violated at the same time. In order to not add too many constraints simultaneously, we separate them in batches. In every round, only those violated transitivity constraints (1j) and (1k) defined by $h < h' < h''$ are added for which the distance in D between h and h'' is minimal (counting the number of arcs). We use this selection rule for integer as well as fractional solutions, where in the latter case the distance-based rule does not at all consider the degree of violation (except for the cut tolerance ε).

4. Square Grid Instances

In the VRP benchmark set of Golden et al. (1998) that is also used for the SoftCluVRP, the instances have a specific structure. The customer vertices are located in a systematic and non-random fashion (see also Section 5). In the groups `Golden1` to `Golden8`, customers are located on concentric circles. In the groups `Golden9` to `Golden16`, customers are located on a square grid. Finally, in the groups `Golden17` to `Golden20`, the customers form a star.

For grid-based instances, we prove in Section 4.1 some properties of optimal solutions that allow the reduction of the edge set E without loosing optimality. Moreover, we derive in Section 4.2 simple-to-compute lower bounds that are especially effective for the grid-based instances.

4.1. Reduction of the Edge Set

The set of edges E of the grid-based instances can be reduced using the following theorem.

Theorem 1. *Let an instance of a* Euclidean traveling salesman problem *(EucTSP) be given, where all vertices are points of a square grid.*

If there exists a (3×3)*-vertex block, see Figure 1, then the vertex in the middle of the block (depicted as a diamond* $\diamond$ *in Figure 2) is connected to two block neighbors in every optimal Euclidean EucTSP tour. Therefore, in any optimal solution, two of the eight blue edges in Figure 2 are selected.*

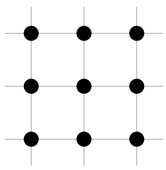


Figure 1: A (3×3)-vertex block.

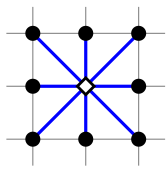


Figure 2: Edges of $\delta(\diamond)$ of an optimal TSP tour.

In the following, a vertex in a Euclidean SoftCluVRP defined over a square grid is denoted as a *middle vertex* if there exists a (3×3)-vertex block that is completely contained in one of the clusters. All edges $\{i, j\} \in E$ that connect a middle vertex i with a non-neighboring vertex j (different from one of the eight neighbors depicted in Figure 2) are denoted as *long edges*.

Corollary 1. *An optimal solution of a Euclidean SoftCluVRP defined over a square grid does not contain long edges.*

Proof. Follows directly from Theorem 1. □

The instances with vertices located on a circle (classes 1–8 of the benchmark of Golden et al. (1998)) cannot be reduced as suggested in Corollary 1. A counterexample is given in Figure 3.

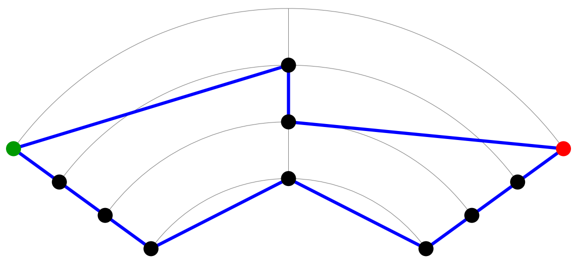


Figure 3: Instance with an optimal EucTSP tour with vertices on a concentric circles. The black vertices • form a 3×3-vertex block. The vertex in the middle of the (3×3)-vertex block is connected to the red vertex • that is not part of the block.

4.2. Lower Bounds

We present two different lower-bounding techniques for grid-based instances constructed as those of classes 9–16 of the benchmark set of Golden et al. (1998).

First, we exploit that any cluster is connected to the depot with not more than two edges. Therefore, connecting the depot to the closest customer vertices (allowing double connections for edges in $\delta^*(0)$) with the additional restriction that no cluster is connected more than two times yields a lower bound on the cost of depot-edges. Moreover, each customer vertex is connected to other vertices with distance of at least 1. Hence, the sum of connections to the depot plus the sum of distance 1 connections provides a valid lower bound, in the following referred to as the *grid lower bound*.

Second, we reuse the same idea that every cluster is connected to the depot with not more than two edges to formulate a relaxed model. The model relaxes formulation (1) and adds the condition on the depot connections. It reads as follows:

$$\min \quad \sum_{\{i,j\}\in E} c_{ij} x_{ij} \tag{6a}$$

$$\text{subject to} \quad \text{(1b)–(1f)}$$

$$\sum_{i \in V_h} x_{0i} \leq 2 \qquad \forall h \in H \tag{6b}$$

For the grid instances, formulation (6) provides the same or a better lower bound as the simple grid lower bound explained first, but the computational effort is higher. We refer to the second bound as the *relax lower bound*. Note that this latter bound is generally applicable to all SoftCluVRP instances.

5. Computational Results

The computational experiments are based on the same benchmark instances as considered by Hintsch and Irnich (2020). All benchmarks use CVRP instances and define an additional parameter θ that specifies the average cluster size. Clusters are then constructed in various way (for details see Fischetti et al. 1997; Bektaş et al. 2011).

The first set of 158 small- and medium-sized instances is based on the `GVRP` benchmarks `A`, `B`, `P`, and `GC` with $\theta \in \{2, 3\}$. The instances with 16 to 262 vertices and 6 to 131 clusters were generated by Bektaş et al. (2011). The second set of 220 large-scale instances is based on the `Golden` instances of Golden et al. (1998) with $\theta \in \{5, \ldots, 15\}$ and were generated by Battarra et al. (2014). The instances are divided into 20 groups denoted by `Golden1` to `Golden20` with 201 to 484 vertices and 14 to 97 clusters.

Unfortunately, the distances in the grid-based instances (groups `Golden9` to `Golden16`) are computed as Euclidean distances rounded to the next integer value. As a consequence, neither distances fulfill the triangle inequality nor Corollary 1 is directly applicable. To anyway test the reduction techniques described in Section 4.1, we generated 90 additional but smaller `Grid` instances as follows: Six instances were generated for each combination of grid size $d \times d$ with $d \in \{10, 12, 14\}$ and number $N \in \{6, 8, 10, 12, 14\}$ of clusters. In all instances, the minimal distance between vertices is 1. The depot is either in the middle or at the corner of the grid. To define the clusters, one vertex is randomly assigned to each cluster at initialization. As long as vertices are unassigned, such a vertex with at least one already assigned vertex in the neighborhood is chosen at random. This vertex is then assigned to a randomly chosen cluster of its neighborhood (see vertex $\diamond$ with neighbors $\bullet$ in Figure 2 for the definition of the neighborhood). Demands of the clusters are equally distributed on the interval $[10, 50]$ and the vehicle capacity is set to $Q = 100$. The number of vehicles is defined as $m = \lceil \sum_h d_h / Q \rceil$. All distances are computed as nontruncated Euclidean distances. The instances are online available at https://logistik.bwl.uni-mainz.de/benchmarks/.

The branch-and-cut algorithm was implemented in C++ using CPLEX 12.8.0 with Concert Technology and compiled into 64-bit single-thread code with Microsoft Visual Studio 2015. Experiments are carried out on a 64-bit Microsoft Windows 7 personal computer with an Intel® Core™ i7-5930k CPU clocked at 3.5 GHz and 64 GB of RAM. CPLEX's default values are kept for all parameters. Unless stated otherwise, computation times are limited to a maximum of 3600 seconds (1 hour).

5.1. Comparison of Cutting Strategies

In a first experiment, we try to find a reasonable cutting strategy for the final branch-and-cut algorithm. In Section 3, we discussed possible separation strategies for capacity cuts, single-route inequalities, tree-capacity constraints, and transitivity constraints. A synopsis of the strategies is presented in Table 1. We distinguish between constraints for model (1) (first section of the table) and the mandatory single-route inequalities for model (1)–(2) needed for fixed non-minimal fleets and when the depot-triangle inequality is not satisfied (second section). Note that in several instances of the `Golden` benchmark the fleet is larger than the minimum fleet size.

Table 1: Summary of Cutting Strategies

Constraint type	Abbrev.	Separation strategies	Remark
Transitivity constraints (1j)–(1k)	Trans	(2): `static`, `dyn`	int: mandatory
MTZ constraints (1h)–(1i)	MTZ	(2): `none`, `static`	int: mandatory, redundant if Tree:`dyn`
Capacity cuts (1d) separated			int: mandatory
with Lysgaard library	LysCC	(3): `none`, `root only`, <u>`dyn`</u>	fract: optional
with Karger's contr. alg.	ProbCC	(3): `none`, `root only`, <u>`dyn`</u>	fract: optional
with MIP (5)	MIP CC	(3): `none`, <u>`root only`</u>, `dyn`	fract: optional
Tree-capacity constraints (3)	Tree	(2): `none`, <u>`dyn`</u>	int: mandatory, redundant if MTZ:`static`
Single-route inequalities (2)	Single	(1): <u>`dyn`</u>	int: mandatory only for SoftCluVRP$^{\leq m}$ w/o depot-triangle inequality

Note: Default strategies are <u>underlined</u>.

Table 1 offers $2^3 \cdot 3^3 = 216$ possible configurations by combining separation strategies that either completely switch off a separation procedure (denoted by "`none`"), add all inequalities at initialization ("`static`"), or add inequalities dynamically using separation procedures. In the latter case, we compare dynamic separation at the root node of the branch-and-bound tree *only* ("`root only`") and separation at all tree nodes ("`dyn`"). There are some invalid configurations, e.g., combining MTZ:`none` and Tree:`none`, that cannot ensure capacity feasible solutions. We omit all invalid configurations.

Empirically testing all possible configurations is hardly possible given that the benchmark sets are large and the previous literature allowed one hour of computation time per instance. We address this issue in the following way: First, we define some *default separation strategies* (denoted by "`default`") that worked well in pretests. We did however not find good default strategies for the transitivity constraints and MTZ constraints, but default strategies for the remaining inequalities (those underlined in Table 1). As a consequence, we compare separation strategies that consider:

- all four combinations of either Trans:`static` or Trans:`dyn` and MTZ:`none` or MTZ:`static`; (4 strategies)
- use all default strategies (`default`) or vary exactly one of the default parameters for all other valid inequalities and their separation. (8 strategies)

In the latter case, the parameter altered is either LysCC:`none`, LysCC:`root only`, ProbCC:`none`, ProbCC:`root only`, MIP CC:`none`, MIP CC:`dyn`, or Tree:`none`. Overall, there remain $4 \cdot 8 - 1 = 31$ separation strategies to compare (one strategy is invalid). Second, we restrict the instance set to `GVRP` instances of the groups `A`, `B`, `P`, the `GC` instances with $n = 100$ customers, and the two smallest `Golden` instances of each group `Golden1` to `Golden20`. The selection comprises 190 instances for this experiment.

We compare the different B&C algorithms resulting from the 31 different separation strategies with the help of performance profiles as suggested by Dolan and Moré (2002). For a set of algorithms $\mathcal{A}$ (the 31 B&C algorithms in our case), the performance profile $\rho_A(\tau)$ of an algorithm $A \in \mathcal{A}$ describes the ratio of instances that can be solved by A within a factor τ compared to the fastest algorithm, i.e.

$$\rho_A(\tau) = \frac{\left|\left\{I \in \mathcal{I} : t_I^A / t_I^* \leq \tau\right\}\right|}{|\mathcal{I}|},$$

in which $\mathcal{I}$ is the set of instances, t_I^A is the computation time of algorithm A when applied to instance $I \in \mathcal{I}$, and t_I^* is the smallest computation time among all algorithms of set $\mathcal{A}$ for instance I. Unsolved instances are taken into account with $t_I^A = \infty$ (assuming also that $t_I^* = \infty$ gives $t_I^A / t_I^* = \infty$). Note that $\rho_A(1)$ is the percentage of instances for which algorithm A is the fastest, and $\rho_A(\infty)$ is the percentage of instances that are solved by algorithm A within the time limit.

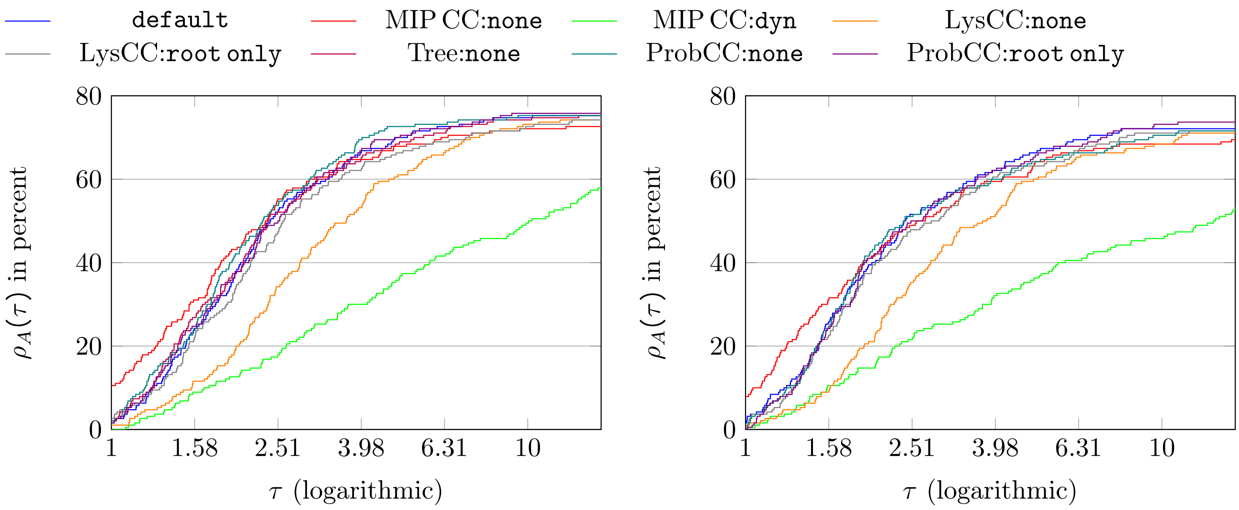
Figure 4 displays the performance profiles of the B&C algorithms using the 31 different cutting strategies. For the sake of clarity, we have decided to group the profiles in different way. In the upper part, the four Subfigures 4a–4d group by values of Trans and MTZ, i.e., Trans:`static` or Trans:`dyn` and MTZ:`none` or MTZ:`static`. Within each of these subfigures, the compared eight (seven in Subfigure 4b) strategies that result from the default (`default`) and the variation of exactly one of the default parameters.

The result from each subfigure is simple to summarize. In all four subfigures, the strategy A = MIP CC:`none` is the one that is the fastest for the highest number of instances (compare the values $\rho_{\text{MIP CC:}\texttt{none}}(1)$ and $\rho_A(1)$). The reason is that completely switching off the MIP-based separation of the capacity cuts accelerates the B&C substantially, but it comes at the cost of providing also less optimality proofs (compare the values $\rho_{\text{MIP CC:}\texttt{none}}(\tau)$ and $\rho_A(\tau)$ for $\tau \approx 15$). Therefore, the strategy MIP CC:`none` is not always the winning strategy. Having said this, we can identify the strategy ProbCC:`none` as the best one in Subfigure 4a, ProbCC:`root only` in Subfigure 4b, Tree:`none` in Subfigure 4c, and MIP CC:`none` in Subfigure 4d.

These four strategies are finally compared in Subfigure 4e: There is no strategy that dominates all other strategies. The strategy Trans:`dyn`, MTZ:`static`, Tree:`none` solves the largest number of instances (within the time limit). This strategy is (compare the figures) not the fastest algorithm most of the time. Analyzing results in more detail, we find that the strategy Trans:`dyn`, MTZ:`none`, MIP CC:`none` is the fastest variant most of the time for small-sized. It is however inferior for larger-sized instances. Therefore, we have decided to choose the strategy

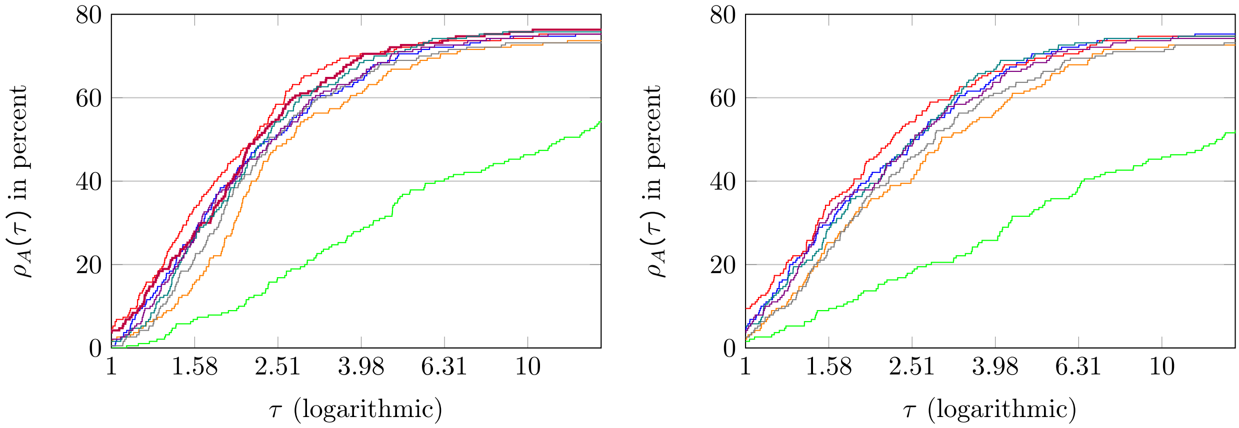
	Trans:`dyn`, MTZ:`static`, MIP CC:`root only`	
with default values	LysCC:`dyn`, ProbCC:`dyn`, and Tree:`none`.	(B&C)

as the one that we use as the *reference B&C algorithm* in the following computational experiments. This is the algorithm that we refer to as *B&C*.



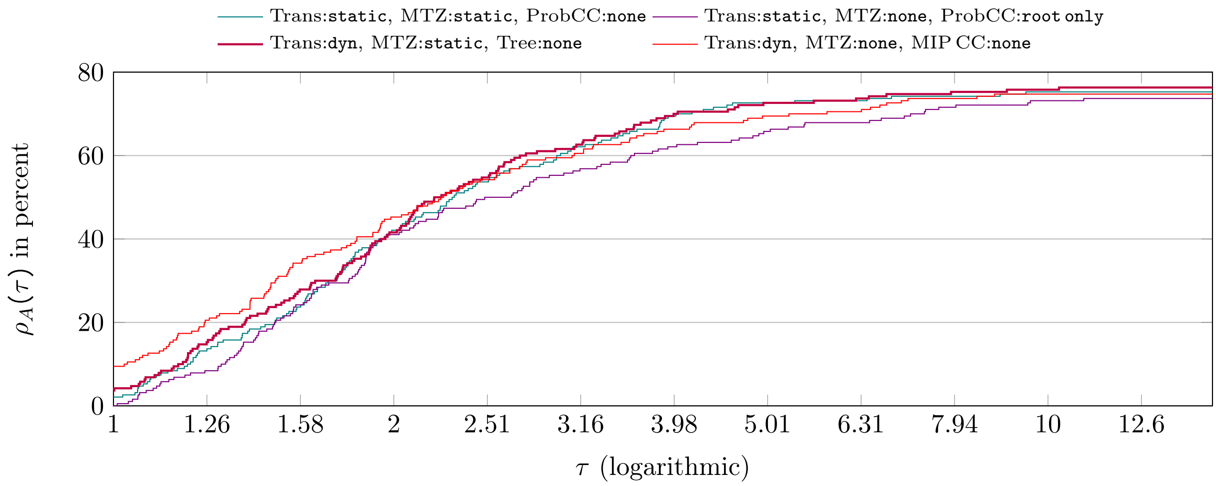
(a) Trans:`static`, MTZ:`static`

(b) Trans:`static`, MTZ:`none`



(c) Trans:`dyn`, MTZ:`static`

(d) Trans:`dyn`, MTZ:`none`



(e) Performance profile for the different settings.

Figure 4: Performance profiles for different settings. `default` is defined as the version in which MIP CC:`root only`, LysCC:`dyn`, Tree:`dyn`, ProbCC:`dyn`. The legend specifies which parameter of the `default`-version is changed.

5.2. *Savings-based Upper Bounds*

Pretests have shown that tight upper bounds are very helpful for initializing the B&C algorithm. We employ the savings-based algorithm of Hintsch (2019; p. 6), which is tailored to the SoftCluVRP. It works as follows: In contrast to the classical savings algorithm, there is one initial route for each cluster visiting all its customers. For each $h \in H$, this route is a TSP tour over the vertex set $V_h \cup \{0\}$ computed with the help of a combined *iterated local search* and *variable neighborhood decent* (ILS/VND) (Hintsch 2019; Section 2.1). Additionally, the same is done for each pair $(g, h) \in H \times H$. Such a route visits all customers $V_g \cup V_h$ and the depot 0. Let the cost of the resulting routes be $\hat{c}_h$ and $\hat{c}_{g,h}$, respectively. Savings values are calculated for each pair $(g, h) \in H \times H$ as $sav_{g,h} = \hat{c}_g + \hat{c}_h - \hat{c}_{g,h}$. Subsets of clusters to be served by one route are constructed now as in the classical savings algorithm: the largest (feasible) savings value $sav_{g,h}$ is chosen first. The two clusters are then merged, i.e., their demand is added. A saving becomes infeasible if either the vehicle capacity Q is exceeded by the total demand of both routes or both clusters are already part of the same route. The savings algorithm repeats these steps until the number of routes matches the number of vehicles or all remaining savings become infeasible.

In case the number of routes exceeds the number of vehicles m, we compute alternative subsets of clusters to be served by one route as a bin-packing solution with items of weight $d_h, h \in H$, and capacity Q. We use the arc-flow model of Valério de Carvalho (1999) for this purpose.

Finally, for each set of clusters served by a route, we construct a route with the combined ILS/VND. Such a route visits all customers belonging to the given clusters and the depot 0.

5.3. *Results for the GVRP Instances*

In this section, we compare the new B&C algorithm against the branch-and-price algorithm of Hintsch and Irnich (2020) using the `GVRP` benchmark. As mentioned before, the B&C algorithm benefits from tight upper bounds available at initialization. To highlight this effect, we used the reference B&C initialized with the savings-based upper bound of Section 5.2 and the same B&C algorithm but with the *best known solution* (BKS) as upper bound.

The results are summarized in Table 2, where the table entries have the following meaning:

- **#opt:** number of instances solved to proven optimality within 1 hour (3600 seconds);
- **time $\bar{T}$:** average computation time in seconds; unsolved instances are taken into account with the time limit TL of 1 hour (3600 seconds);
- **gap:** gap $100 \cdot (UB - LB)/LB$, i.e., gap in percent;
- **Total:** Aggregated result of Hintsch and Irnich (2020) and our default B&C algorithm (see B&C below);
- **Total*:** Best known results (including other algorithms and higher computation times);

- HI20: branch-and-price of Hintsch and Irnich (2020);
- B&C: reference B&C with upper bound of the savings-based heuristic (see Section 5.2);
- ⋄B&C: reference B&C, but with BKS provided as upper bound UB;
- simple LB grid or relax lower bound; the latter with model (6) run for a maximum of 600 seconds (see Section 4.2); BKS provided as UB.

Overall, the branch-and-price of Hintsch and Irnich (2020) computes the largest number of proven optima and it is slightly faster than the B&C algorithm. However, results are clearly mixed over the different groups of benchmark instances. Both versions of the B&C algorithm are much faster for class `B`, which can be attributed to the special structure of the `B` instances: the customer vertices are truly *clustered* and not scattered, i.e., almost uniformly distributed like all others (Augerat 1995).

The results show that the branch-and-price and B&C algorithms are complementing each other. The column *Total* underlines this statement, because the reference B&C algorithm computes optima for five non-solved instances of HI20. Comparing results with extended computation times and other algorithms employed in Hintsch and Irnich (2020) another four (five) previously open instances are solved now with the B&C (⋄B&C) algorithm. At the end, only seven of the 158 `GVRP` instances remain unsolved.

We provide instance-by-instance results with additional information in the Online Appendix.

Table 2: Results for the GVRP instances.

	HI20		Reference B&C algorithm						Total	Total*
			B&C			⋄B&C				
Set (#inst.)	#opt	time $\bar{T}$	#opt	time $\bar{T}$	gap	#opt	time $\bar{T}$	gap	#opt	#opt
GVRP-2										
A (27)	**26**	**713.2**	21	1000.3	1.4	26	605.2	0.1	26	27
B (23)	17	1104.5	**22**	**342.2**	2.9	23	156.5	0.0	22	23
P (24)	**23**	**474.5**	15	1482.3	4.1	16	1355.4	0.8	23	24
GC (5)	**1**	**2993.8**	0	*TL*	50.9	0	*TL*	3.6	1	1
GVRP-3										
A (27)	**27**	**83.0**	26	256.1	0.2	26	207.9	0.1	27	27
B (23)	**23**	160.1	**23**	**10.8**	0.0	23	7.5	0.0	23	23
P (24)	**24**	**106.6**	23	253.7	0.1	23	244.9	0.0	24	24
GC (5)	**1**	3221.4	**1**	3084.8	9.7	1	2937.0	2.5	1	2
Total (158)	**142**	**605.1**	131	741.3	8.7	138	612.8	0.9	147	151

5.4. Results for the Golden Instances

In this section, we use the Golden benchmark to compare the new B&C algorithm against the branch-and-price algorithm of Hintsch and Irnich (2020). The results are summarized in Table 3.

The branch-and-price of Hintsch and Irnich (2020) is for all twenty classes better than the B&C algorithms. We have two possible explanations for the rather poor performance of the B&C algorithm on the Golden benchmark: First, the instances are much larger than the GVRP instances with vertices symmetrically distributed (on a circle, square grid, and star, see Section 4). The results of the previous section showed that the B&C algorithm is strong for truly clustered instances. The Golden instances are however not nicely clustered.

Second, the capacity restriction is most of the time not tight. In fact, for almost all instances, the number of required vehicles can be reduced. Indeed, instances of the SoftCluVRP$^{\leq m}$ most of the time have optimal solutions with strictly lower costs. To show this, we perform an additional run for the Golden instances where the number of vehicles is limited but not fixed to m. For the SoftCluVRP$^{\leq m}$, more instances are solved optimally (26 instead of 21) in less time (on average) compared to the SoftCluVRP with fixed fleet size. Detailed results can be found in the Online Appendix.

On the positive side, the ⋄B&C algorithm always provides a valid lower bound so that the overall average gap for the Golden instances is approximately 2 %. In contrast, the branch-and-price algorithm failed to provide a valid lower bound in approximately one third of the cases (only 137 of 220 with LB). This is clearly a point in favor of the B&C algorithm.

What we can also see from Table 3 and the *simple LB* section is that sometimes the lower bounding procedures of Section 4.2 are very effective. For three classes, the *simple LB* computation is successful and provides a proof of optimality for more instances than could be solved with either branch-and-price or B&C. Overall, the new lower bounds allow proving optimality for nine previously open instances.

5.5. Results for Square Grid Instances

In this final experiment, we analyze and quantify the impact that the reduction technique of Section 4.1 has on the performance of the B&C algorithm. We use the 90 self-generated instances with customers located on a square grid and nontruncated Euclidean distances.

Both B&C algorithms that are compared use the default B&C once for the reduced and once for the non-reduced edge set. The number of columns and rows of the model is reduced on average by 17.8% and 16.7%, respectively. As before, we compare both variants with the help of performance profiles, see Figure 5.

Table 3: Results for the `Golden` instances.

			Reference B&C algorithm									
	HI20		B&C			⋄B&C			simple LB		Total	Total*
Set (#inst.)	#opt	time $\bar{T}$	#opt	time $\bar{T}$	gap	#opt	time $\bar{T}$	gap	#opt	gap	#opt	#opt
`Golden`												
1 (11)	**1**	**3300.4**	0	*TL*	5.4	0	*TL*	2.1	0	4.6	1	1
2 (11)	**1**	**3530.5**	0	*TL*	5.7	0	*TL*	2.1	0	3.3	1	1
3 (11)	0	*TL*	0	*TL*	6.1	0	*TL*	2.0	0	2.8	0	0
4 (11)	0	*TL*	0	*TL*	5.7	0	*TL*	1.9	0	2.1	0	0
5 (11)	**10**	**832.4**	6	2486.6	3.7	6	2216.4	1.3	0	5.8	10	11
6 (11)	**5**	**2702.1**	1	3558.2	5.5	1	3415.7	2.1	0	3.2	5	5
7 (11)	0	*TL*	0	*TL*	4.9	0	*TL*	1.9	0	2.6	0	4
8 (11)	0	*TL*	0	*TL*	6.1	0	*TL*	1.8	0	2.6	0	0
9 (11)	**8**	**1445.5**	0	*TL*	4.8	0	*TL*	4.4	7	0.4	9	10
10 (11)	**6**	**2453.1**	0	*TL*	8.8	0	*TL*	3.6	0	0.7	6	7
11 (11)	**1**	**3514.6**	0	*TL*	5.2	0	*TL*	4.0	2	0.4	3	9
12 (11)	0	*TL*	0	*TL*	8.5	0	*TL*	3.1	0	0.8	0	0
13 (11)	**9**	**1118.8**	4	2963.2	1.6	4	3021.6	0.3	6	0.2	11	11
14 (11)	**4**	**2860.9**	0	*TL*	3.4	0	*TL*	0.6	0	0.2	4	8
15 (11)	**1**	**3561.5**	0	*TL*	3.2	0	*TL*	0.6	6	0.1	6	8
16 (11)	0	*TL*	0	*TL*	3.4	0	*TL*	0.8	3	0.2	3	4
17 (11)	**8**	**1314.4**	**8**	1629.7	1.0	8	1654.5	0.2	0	2.0	8	9
18 (11)	**6**	**2338.4**	2	3259.3	4.1	3	3165.8	0.9	0	2.0	6	8
19 (11)	**5**	**2741.5**	0	*TL*	6.6	0	*TL*	3.4	0	2.8	5	8
20 (11)	**3**	**3032.7**	0	*TL*	6.5	0	*TL*	5.1	0	2.6	3	4
Total (220)	**68**	**2817.3**	21	3394.9	5.0	22	3373.7	2.1	24	2.0	81	108

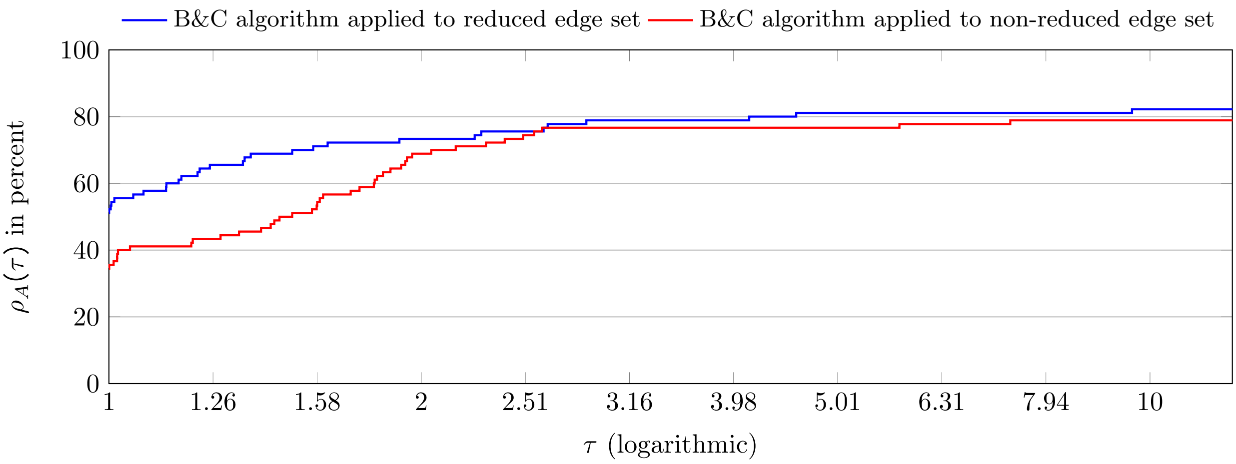


Figure 5: Performance profile for the reduced and non-reduced edge set.

The profiles clearly indicate that the reduction procedure is effective and has a relatively strong impact. Comparing both algorithms at $\tau = 1$, the algorithm with reduced (non-reduced) edge set is in 51% (34%) of the cases the fastest. For higher τ-values, the difference between both algorithms decreases but in total three more instances can be solved by the B&C algorithm that is applied to the reduced graph.

Results summarized in Table 4 also confirm the positive effect of the edge reduction: Average time and gap are overall smaller for the reduced version. In total 75 of 90 instances can be solved to proven optimality. Detailed results including number of columns and rows of the model can be found in the Online Appendix.

Table 4: Results for square `Grid` instances.

		reduced			non-reduced			Total	
Set (#inst.)	n+1	#opt	time $\bar{T}$	gap	#opt	time $\bar{T}$	gap	#opt	gap
`Grid`									
1-30 (30)	121	**28**	**605.8**	**0.4**	27	643.1	0.5	28	0.3
31-60 (30)	169	**24**	1091.6	**1.8**	23	**1084.2**	2.8	25	1.6
61-90 (30)	225	**22**	**1470.3**	5.4	21	1487.2	**5.1**	22	4.6
Total (90)		**74**	**1055.9**	**2.6**	71	1071.5	2.8	75	2.2

6. Conclusions

In this paper, we provided a first two-index formulation for the soft-clustered vehicle-routing problem (SoftCluVRP). The novelty of the model is the very clear separation of the standard routing part as known for the capacitated VRP from the part that ensures routes that respect the soft-cluster constraints. This latter part of the model uses an asymmetric and directed cluster graph so that MTZ-like constraints can be used here. Overall, the formulation is rather simple to use with modern MIP solvers, because only capacity constraints have to be added dynamically, i.e., in a cutting-plane fashion. All other valid inequalities that we presented are not mandatory when the fleet is not made artificially larger than necessary.

On the theoretical side, there are several contributions: First, we analyzed the impact of the fleet-size constraint and derived new constraints that allow to cope with non-minimal fleets. Second, we proved that one third of the transitivity constraints are actually redundant, for the complete integer formulation as well as for its linear relaxation. Third, we presented new capacity cuts that are stronger than the straightforward adaptation of the well-known version for the capacitated VRP. Finally, we prove a deep result that SoftCluVRPs defined on a square grid are reducible.

On the algorithmic side, the new branch-and-cut (B&C) algorithm is complementary to the only other exact solution approach from the literature that is based on branch-and-price. While not competitive on all instance classes, the B&C algorithm is particularly useful for SoftCluVRP instances that are truly clustered. A B&C approach is also much simpler to implement compared to the sophisticated branch-and-price of Hintsch and Irnich (2020). Overall, the B&C algorithm and lower-bounding strategies deliver new provably optimal solutions to five `GVRP` and nine `Golden` benchmark instances that were unsolved before.

Appendix

Proof of Proposition 4:

Let $(\bar{x}, \bar{y}, \bar{u})$ be an integer feasible solution to model (R). We consider the connected components of the digraph $D_{\bar{y}}$ spanned by the arcs a with $\bar{y}_a = 1$. Let $(H(O), A(O))$ be one of these components and let $O = \{h_1, h_2, \dots, h_{|O|}\}$ with $h_1 < h_2 < \cdots < h_{|O|}$. In a first step, we show that

$$(h_i, h_{i+1}) \in A(O), \text{ i.e., } \bar{y}_{h_i,h_{i+1}} = 1 \qquad \text{for all } i = 1, 2, \dots, |O| - 1 \tag{7}$$

holds true. If the opposite were true, i.e., $(h_i, h_{i+1}) \notin A(O)$, we consider a h_i-h_{i+1}-path P in $(H(O), A(O))$ with a minimum number of arcs (disregarding the direction of the arcs). Note that P must exist by definition of a connected component and that P and (h_i, h_{i+1}) form a cycle (again disregarding the direction of the arcs). We now consider the minimum and maximum vertices $h_{\min} = \min H(P)$ and $h_{\max} = \max H(P)$ w.r.t. the <-relation, respectively. Note that $h_{\min} = h_i$ and $h_{\max} = h_{i+1}$ at the same time is not possible.

If $h_{\min} < h_i$, the vertex $h_{\min}$ has an out-degree of 2 in P, i.e., there exist two arcs $(h_{\min}, h)$ and $(h_{\min}, h') \in A(O)$ with $h < h'$. Transitivity constraints (1k) for $h_{\min} < h < h'$ then imply that also $\bar{y}_{hh'} = 1$, i.e, $(h, h') \in A(O)$. This however contradicts with the minimality of P, because replacing (h_{min}, h) and (h_{min}, h') by (h, h') in P would create a shorter h_i-h_j-path.

If $h_{\max} > h_{i+1}$, the vertex $h_{\max}$ has an in-degree of 2 in P. The same type of argument can now be used together with the transitivity constraints (1j) leading to a shorter h_i-h_j-path contradicting with the minimality of P. Therefore, (7) must hold true.

In a second step, we show that the $\bar{y}$-values can be set to one within each component without violating any constraint. Let $(h_i, h_j) \in A(O)$ be an arc with $\bar{y}_{h_i h_j} = 0$. Because of (7), we know that $j > i + 1$ holds true. Summing up (1h) over the arcs $(h_i, h_{i+1}), (h_{i+1}, h_{i+2}), \dots, (h_{j-1}, h_j)$ gives $\bar{u}_{h_i} - \bar{u}_{h_j} \le -\sum_{k=i+1}^{j} d_{h_k} \le -d_{h_j}$. This proves that (1h) for the arc $(h_i, h_j) \in A$ is also fulfilled even after increasing $\bar{y}_{h_i,h_j}$ to 1. Hence, all arcs with both endpoints in O can be increased without violating (1i).

Note that the new values $\bar{y}$ for the y-variables do not violate any transitivity constraint (1j)–(1k) because the positive $\bar{y}$ represent the transitive closure over O.

Proof of Proposition 5:

Let $(\bar{x}, \bar{y}, \bar{u})$ be a feasible solution to the linear relaxation of model (R). The feasible solution $(\bar{x}, \bar{\bar{y}}, \bar{u})$ to the linear relaxation of model (1) is constructed as follows. We consider the connected components of the digraph $D_{\bar{y}}$ spanned by the arcs a with $\bar{y}_a > 0$. Note that $\bar{y}_{h,h'} = 0$ for h and h' in different components. Accordingly, we also set $\bar{\bar{y}}_{h,h'} = 0$.

Within each component, the $\bar{\bar{y}}$-values are defined recursively. For this purpose, let $(H(O), A(O))$ be one of these components and let $C = \{h_1, h_2, \dots, h_{|O|}\}$ with $h_1 < h_2 < \cdots < h_{|O|}$. We define

$$\begin{aligned}
\bar{\bar{y}}_{h_i h_{i+1}} &:= \bar{y}_{h_i h_{i+1}} && \text{for all } i = 1, \dots, |O| - 1;\\
\bar{\bar{y}}_{h_i h_{i+2}} &:= \max\{\bar{y}_{h_i h_{i+2}}, \bar{y}_{h_i h_{i+1}} + \bar{y}_{h_{i+1} h_{i+2}} - 1\} && \text{for all } i = 1, \dots, |O| - 2;
\end{aligned}$$

and for all $k = 3, \dots, |O| - 1$

$$\bar{\bar{y}}_{h_i h_{i+k}} := \max\{\bar{y}_{h_i h_{i+k}}, \max_{j \in \{1, \dots, k-1\}} \{\bar{\bar{y}}_{h_i h_{i+j}} + \bar{\bar{y}}_{h_{i+j} h_{i+k}} - 1\}\} \qquad \text{for all } i = 1, \dots, |O| - k.$$

We have to show that $(\bar{x}, \bar{\bar{y}}, \bar{u})$ is a feasible solution to the linear relaxation of (1). Since $\bar{\bar{y}} \geq \bar{y}$, it suffices to show that this solution fulfills the MTZ-constraints (1h) and transitivity constraints (1j)–(1l). Obviously, it suffices to check the validity of these constraints for arcs or pairs of arcs $(h_i, h_{i+k}) \in A(O)$.

Regarding MTZ-constraints (1h), if $\bar{\bar{y}}_{h_i,h_{i+k}} = \bar{y}_{h_i,h_{i+k}}$ then (1h) is fulfilled because $(\bar{x}, \bar{y}, \bar{u})$ is feasible. In addition, this solves the subcases $k = 1$, i.e., arcs (h_i, h_{i+1}). Otherwise, we first consider the case of $k = 2$ and $\bar{\bar{y}}_{h_i h_{i+2}} = \bar{y}_{h_i h_{i+1}} + \bar{y}_{h_{i+1} h_{i+2}} - 1$. It follows

$$\begin{aligned} u_{h_i} - u_{h_{i+2}} + Q\bar{\bar{y}}_{h_i h_{i+2}} &= u_{h_i} - u_{h_{i+1}} + u_{h_{i+1}} - u_{h_{i+2}} + Q(\bar{y}_{h_i h_{i+1}} + \bar{y}_{h_{i+1} h_{i+2}} - 1) \\ &= (u_{h_i} - u_{h_{i+1}} + Q\bar{y}_{h_i h_{i+1}}) + (u_{h_{i+1}} - u_{h_{i+2}} + Q\bar{y}_{h_{i+1} h_{i+2}}) - Q \\ &\overset{(1h)}{\leq} Q - d_{h_{i+1}} + Q - d_{h_{i+2}} - Q \leq Q - d_{h_{i+2}}. \end{aligned}$$

where we exploit (1h) for the $\bar{y}$-values.

For $k > 2$, we assume that the values of $\bar{\bar{y}}_{h_i h_{i+k}}$ results from an index $j \in \{1, \dots k-1\}$ with $\bar{\bar{y}}_{h_i h_{i+k}} = \bar{\bar{y}}_{h_i h_{i+j}} + \bar{\bar{y}}_{h_{i+j} h_{i+k}} - 1$. By induction over k, we get

$$\begin{aligned} u_{h_i} - u_{h_{i+k}} + Q\bar{\bar{y}}_{h_i h_{i+k}} &= u_{h_i} - u_{h_{i+j}} + u_{h_{i+j}} - u_{h_{i+k}} + Q(\bar{\bar{y}}_{h_i h_{i+j}} + \bar{\bar{y}}_{h_{i+j} h_{i+k}} - 1) \\ &= (u_{h_i} - u_{h_{i+j}} + Q\bar{\bar{y}}_{h_i h_{i+j}}) + (u_{h_{i+j}} - u_{h_{i+k}} + Q\bar{\bar{y}}_{h_{i+j} h_{i+k}}) - Q \\ &\overset{(1h)}{\leq} Q - d_{h_{i+j}} + Q - d_{h_{i+k}} - Q \leq Q - d_{h_{i+k}}. \end{aligned}$$

where we exploit (1h) for smaller k by induction hypothesis. This completes all case for the MTZ-constraints (1h).

Next we prove that the first two classes of transitivity constraints (1j) and (1k) hold true for the $\bar{\bar{y}}$-values. The proof is also by induction over $k \geq 2$ for pairs of arcs (h_i, h_{i+j}) and $(h_{i+j}, h_{i+k}) \in A(O)$, i.e., with $h_i < h_{i+j} < h_{i+k}$. For $k = 2$, the arc pair is (h_i, h_{i+1}) and (h_{i+1}, h_{i+2}) so that the result directly follows because $\bar{\bar{y}}_{h_i h_{i+l}} = \bar{y}_{h_i h_{i+l}}$ and $\bar{\bar{y}}_{h_{i+1}, h_{i+2}} = \bar{y}_{h_{i+1}, h_{i+2}}$.

For $k > 2$, we show w.l.o.g. that transitivity constraints (1j) are fulfilled (the proof for the second class of transitivity constraints (1k) works analogously). Let $l \in \{1, \dots, k-1\}$. We distinguish three cases.

The first case is $\bar{\bar{y}}_{h_i h_{i+k}} = \bar{y}_{h_i h_{i+k}}$ and $\bar{\bar{y}}_{h_{i+l} h_{i+k}} = \bar{y}_{h_{i+l} h_{i+k}}$. Then,

$$\bar{\bar{y}}_{h_i h_{i+l}} \geq \bar{y}_{h_i h_{i+l}} \overset{(1j)}{\geq} \bar{y}_{h_i h_{i+k}} + \bar{y}_{h_{i+l} h_{i+k}} - 1 = \bar{\bar{y}}_{h_i h_{i+k}} + \bar{\bar{y}}_{h_{i+l} h_{i+k}} - 1. \tag{8}$$

In the second case, let $\bar{\bar{y}}_{h_i h_{i+k}} = \bar{y}_{h_i h_{i+k}}$ and $\bar{\bar{y}}_{h_{i+l} h_{i+k}} > \bar{y}_{h_{i+l} h_{i+k}}$. The latter implies (using the definition of the $\bar{\bar{y}}$-values) that there exist indices $l_1 < l_2 < \cdots < l_s$ with $l_1 > l$ and $l_s < k$ for an $s \in \{1, \dots, k-l\}$ such that $\bar{\bar{y}}_{h_{i+l} h_{i+k}}$ can be expressed by $\bar{y}$-values in the following form:

$$\bar{\bar{y}}_{h_{i+l} h_{i+k}} = \bar{y}_{h_{i+l} h_{i+l_1}} + \bar{y}_{h_{i+l^1} h_{i+l_2}} + \cdots + \bar{y}_{h_{i+l_{s-1}} h_{i+k}} + \bar{y}_{h_{i+l_s} h_{i+k}} - s + 1 \tag{9}$$

Iteratively exploiting constraints (1j) for the $\bar{y}$-values yields

$$\begin{aligned} &\bar{\bar{y}}_{h_i h_{i+l}} \geq \bar{y}_{h_i h_{i+l}} \overset{(1j)}{\geq} \bar{y}_{h_i h_{i+l_1}} + \bar{y}_{h_{i+l} h_{i+l_1}} - 1 \overset{(1j)}{\geq} \bar{y}_{h_i h_{i+l_2}} + \bar{y}_{h_{i+l_1} h_{i+l_2}} + \bar{y}_{h_{i+l} h_{i+l_1}} - 2 \\ &\overset{(1j)}{\geq} \dots \overset{(1j)}{\geq} \bar{y}_{h_i h_{i+k}} + \bar{y}_{h_{i+l_s} h_{i+k}} + \bar{y}_{h_{i+l_{s-1}} h_{i+l_s}} + \dots + \bar{y}_{h_{i+l_1} h_{i+l_2}} + \bar{y}_{h_{i+l} h_{i+l_1}} - s \overset{(9)}{\geq} \bar{\bar{y}}_{h_i h_{i+k}} + \bar{\bar{y}}_{h_{i+l} h_{i+k}} - 1. \end{aligned}$$

In the third and last case, let $\bar{\bar{y}}_{h_i h_{i+k}} > \bar{y}_{h_i h_{i+k}}$. Again, using the definition of the $\bar{\bar{y}}$-values, we know that there exists a $j \in \{1, \dots, k-1\}$ such that

$$\bar{\bar{y}}_{h_i h_{i+k}} = \bar{\bar{y}}_{h_i h_{i+j}} + \bar{\bar{y}}_{h_{i+j} h_{i+k}} - 1. \tag{10}$$

By induction hypothesis, constraints

$$\bar{\bar{y}}_{h_i h_{i+l}} \geq \bar{\bar{y}}_{h_i h_{i+j}} + \bar{\bar{y}}_{h_{i+j} h_{i+l}} - 1, \tag{11a}$$

$$\bar{\bar{y}}_{h_{i+j}h_{i+l}} \geq \bar{\bar{y}}_{h_{i+j}h_{i+k}} + \bar{\bar{y}}_{h_{i+l}h_{i+k}} - 1 \tag{11b}$$

are satisfied, yielding

$$\bar{\bar{y}}_{h_i h_{i+l}} \overset{(11a)}{\geq} \bar{\bar{y}}_{h_i h_{i+j}} + \bar{\bar{y}}_{h_{i+j}h_{i+l}} - 1 \overset{(11b)}{\geq} \bar{\bar{y}}_{h_i h_{i+j}} + \bar{\bar{y}}_{h_j h_{i+k}} + \bar{\bar{y}}_{h_l h_{i+k}} - 2 \overset{(10)}{=} \bar{\bar{y}}_{h_i h_{i+k}} + \bar{\bar{y}}_{h_{i+l}h_{i+k}} - 1.$$

This completes the proof for the transitivity constraints (1j) and (1k).

Finally, the validity of the third class of transitivity constraints (1l) directly results from the definition of the $\bar{\bar{y}}$-values.

Proof of Theorem 1:

Our proof is by contradiction, i.e., we assume an optimal EucTSP tour in which the vertex in the middle of the (3×3)-vertex block (the vertex $\diamond$) is connected to a vertex $x = (x_1, x_2)$ that is not part of the (3×3)-vertex block.

According to Flood (1956), an optimal EucTSP tour is without any crossings in the Euclidean plane. Therefore, the vertex $\diamond$ cannot be connected to any other vertex on the horizontal or vertical line crossing $\diamond$ outside the (3×3)-vertex block. For the same reason, the vertex $\diamond$ cannot be connected to any points on the diagonals (45° and 135°) outside the 3×3-vertex block (the diagonals cross $\diamond$ and the top-left and bottom-right vertex of the (3×3)-vertex block or the top-right and bottom-left vertices of the block).

Exploiting symmetry, we can assume that x is one of the green vertices $\bullet$ depicted in Table 5. Moreover, the vertex $\star$, right of $\diamond$, must be connected to two other vertices; these are denoted by $y = (y_1, y_2)$ and $z = (z_1, z_2)$. Possible y-coordinates have a gray background. The coordinates of x, y, and z are measured according to a coordinate system in which $\star$ is the origin. W.l.o.g. the minimal distance is one so that $\diamond = (-1, 0)$, $\star = (0, 0)$, and $x, y, z \in \mathbb{Z}^2$.

Because crossings are prohibited, the vertex $\star$ must be connected to green or purple vertices (depicted as $\bullet$ and $\bullet$, see Cases 1–3 in Table 5) or to the vertex $\diamond$ (see Cases 4–7 in Table 5).

For all seven cases, we can construct a (strictly) shorter EucTSP tour proving that the original tour was not optimal. Table 5 summarizes all cases, where below each grid the corresponding distances are calculated and y-vertices are marked with a gray background.

In the following, the inequalities of all cases are derived in detail. Simple calculations (square and estimate) show that

$$\sqrt{(x_1+1)^2 + x_2^2} \geq \frac{1}{\sqrt{2}} + \sqrt{x_1^2 + x_2^2} \tag{*}$$

holds true for $x_1 \geq x_2 \geq 0$. This inequality is helpful for some intermediate steps. Note that in general $x_1, x_2 \geq 1$ and $x_1 \geq x_2$.

Case 1: Note first that in case of $x_1 \geq y_1$ and $x_2 \leq y_2$ two lines of x and y would cross. Hence, this case does not need to be considered. Three other cases remain: First, if $x_1 \geq y_1$ and $x_2 \geq y_2$ then $\sqrt{y_1^2 + y_2^2} \geq 1$ and $\sqrt{(x_1+1)^2 + x_2^2} > \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2}$. Second, if $x_1 \leq y_1$ and $x_2 \leq y_2$ then $\sqrt{y_1^2 + y_2^2} > \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2}$ and $\sqrt{(x_1+1)^2 + x_2^2} > 1$.

Third, if $x_1 \leq y_1$ and $x_2 \geq y_2$ then we distinguish three subcases: If $y = (1, 0)$ the inequality holds true because $1 + \sqrt{(x_1+1)^2 + x_2^2} > 1 + \sqrt{(x_1-1)^2 + x_2^2}$. If $\sqrt{(x_1+1)^2 + x_2^2} \geq \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2}$ and $y_1, y_2 \geq 1$, the inequality holds true because $\sqrt{y_1^2 + y_2^2} > 1$. Otherwise, let $\sqrt{(x_1+1)^2 + x_2^2} < \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2}$ and $y_1, y_2 \geq 1$. We consider the triangle defined by the vertices $\mathbf{0}$, x, and y depicted in Figure 6. Let β be the angle between the edges (x, y) and $(\mathbf{0}, x)$. Since $x_1 \geq x_2$ and $y_2 \geq 1$, the angle is obtuse, i.e., $\beta > 90$. Let $v = (v_1, v_2)$ be the intersection of the line crossing $\mathbf{0}$ and y and the line forming a 90-degree angle with vertices x and y. It follows that $\sqrt{y_1^2 + y_2^2} > \sqrt{(v_1 - y_1)^2 + (v_2 - y_2)^2} > \cos(\beta)\sqrt{(v_1 - y_1)^2 + (v_2 - y_2)^2} = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2}$.

Case 2: Obviously, $\sqrt{y_1^2 + y_2^2} > \sqrt{(|y_1| - 1)^2 + y_2^2}$ and $\sqrt{(x_1+1)^2 + x_2^2} > \sqrt{x_1^2 + x_2^2}$.

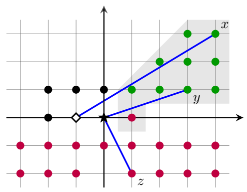
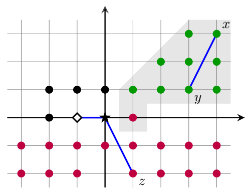
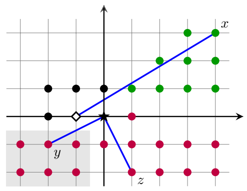
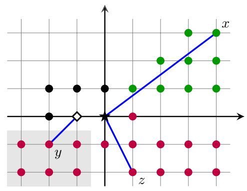
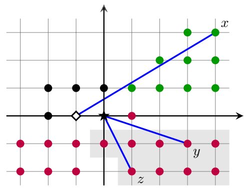
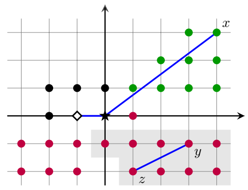
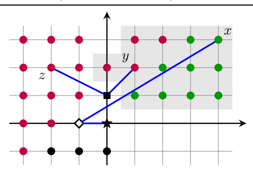
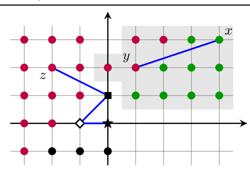
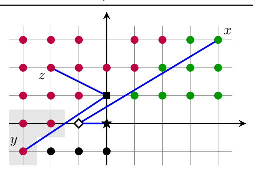
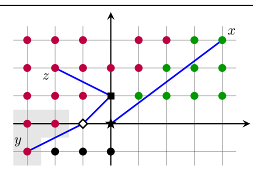
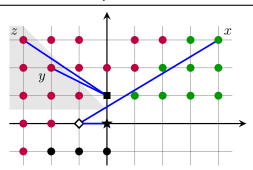
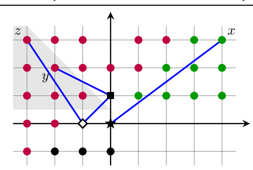
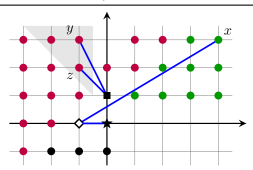
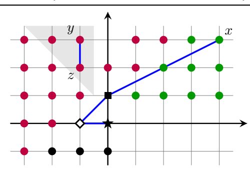
Case	Assumption	TSP tour		Improved TSP tour
1	$z - \star$, $\star - y$ $y_1 \geq 1, y_2 \geq 0$			
		$\sqrt{y_1^2 + y_2^2} + \sqrt{(x_1+1)^2 + x_2^2}$	$>$	$1 + \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2}$
2	$z - \star$, $\star - y$ $y_1 \leq -1, y_2 \leq -1$			
		$\sqrt{y_1^2 + y_2^2} + \sqrt{(x_1+1)^2 + x_2^2}$	$>$	$\sqrt{(\lvert y_1\rvert - 1)^2 + y_2^2} + \sqrt{x_1^2 + x_2^2}$
3	$z - \star$, $\star - y$ $z_1 \geq 1, z_2 \leq -1$ $y = (0,-1)$ or $y_1 \geq 1,\ y_2 \leq -1$			
		$\sqrt{(x_1+1)^2 + x_2^2} + \sqrt{y_1^2 + y_2^2} + \sqrt{z_1^2 + z_2^2}$	$>$	$1 + \sqrt{x_1^2 + x_2^2} + \sqrt{(y_1 - z_1)^2 + (y_2 - z_2)^2}$
4	$\diamond - \star$ $y_1, y_2 \geq 1$ or $y = (0,2)$			
		$\sqrt{y_1^2 + (y_2-1)^2} + \sqrt{(x_1+1)^2 + x_2^2}$	$>$	$\sqrt{2} + \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2}$
5	$\diamond - \star$ $y_1 \leq -2,\ y_2 \leq 0$			
		$1 + \sqrt{y_1^2 + (\lvert y_2\rvert + 1)^2} + \sqrt{(x_1+1)^2 + x_2^2}$	$>$	$\sqrt{2} + \sqrt{(\lvert y_1\rvert - 1)^2 + y_2^2} + \sqrt{x_1^2 + x_2^2}$
6	$\diamond - \star$ $y_1 \leq -1,\ y_2 \geq 1$ $\lvert y_1\rvert \geq y_2$			
		$1 + \sqrt{y_1^2 + (y_2-1)^2} + \sqrt{(x_1+1)^2 + x_2^2}$	$>$	$\sqrt{2} + \sqrt{(\lvert y_1\rvert - 1)^2 + y_2^2} + \sqrt{x_1^2 + x_2^2}$
7	$\diamond - \star$ $y_1 \leq -1,\ y_2 \geq 1$ $\lvert y_1\rvert < y_2$ $\lvert z_1\rvert < z_2$			
		$\sqrt{y_1^2 + (y_2-1)^2} + \sqrt{z_1^2 + (z_2-1)^2} + \sqrt{(x_1+1)^2 + x_2^2}$	$>$	$\sqrt{2} + \sqrt{(y_1 - z_1)^2 + (y_2 - z_2)^2} + \sqrt{x_1^2 + x_2^2}$

Table 5: Original and improved TSP tours.
Note: The horizontal axis is the first axis for x_1, y_1, and z_1, while the vertical axis is the second axis for x_2, y_2, and z_2.

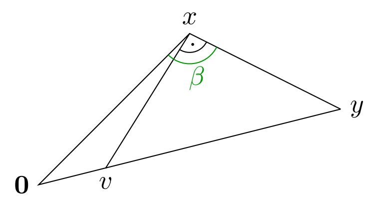


Figure 6: Triangle described in Case 1.

Case 3: For the x-part, we find again $\sqrt{(x_1+1)^2+x_2^2} > \sqrt{x_1^2+x_2^2}$.

For the y-part, we consider the four different cases that distinguish the positions of y relative to z: First, if $y_1 \geq z_1$ and $y_2 \leq z_2$ then $\sqrt{y_1^2+y_2^2} \geq \sqrt{(y_1-z_1)^2+(y_2-z_2)^2}$. Second, if $y_1 \leq z_1$ and $y_2 \geq z_2$ then $\sqrt{z_1^2+z_2^2} \geq \sqrt{(y_1-z_1)^2+(y_2-z_2)^2}$. Third, if $y_1 \geq z_1$ and $y_2 \geq z_2$, it follows that $\sqrt{y_1^2+y_2^2}+\sqrt{z_1^2+z_2^2} \geq y_1+\sqrt{1+z_2^2} \geq 1+\sqrt{(y_1-1)^2+z_2^2} \geq 1+\sqrt{(y_1-z_1)^2+(y_2-z_2)^2}$. Fourth and finally, if $y_1 \leq z_1$ and $y_2 \leq z_2$, it follows that $\sqrt{y_1^2+y_2^2}+\sqrt{z_1^2+z_2^2} \geq y_2+\sqrt{z_1^2+1} \geq 1+\sqrt{z_1^2+(y_2-1)^2} \geq 1+\sqrt{(y_1-z_1)^2+(y_2-z_2)^2}$. Note that the inequalities in the third and fourth case result again from squaring and estimating (as done for (*)).

Case 4: The only possibility for $y_1 = 0$ is the point $y = (0,2)$. We consider two subcases for x_2: On the one hand, we assume that $x_2 = 1$. It follows that $1+\sqrt{(x_1+1)^2+1} > \sqrt{2}+\sqrt{x_1^2+1}$ [it is simple to show that the larger x_1, the smaller is the difference between LHS and RHS; the most critical case is therefore $x_1 = 1$, in which case the two sides evaluate to $1+\sqrt{5} > \sqrt{2}+\sqrt{2}$]. On the other hand, we assume $x_2 \geq 2$. Then, $1+\sqrt{(x_1+1)^2+x_2^2} \overset{(*)}{\geq} 1+\frac{1}{\sqrt{2}}+\sqrt{x_1^2+x_2^2} > \sqrt{2}+\sqrt{x_1^2+(x_2-2)^2}$.

If $y_1, y_2 \geq 1$, we consider four subcases according to the positions of x relative to y: First, in case of $x_1 \geq y_1$ and $x_2 \leq y_2$ two lines of x and y would cross so that this case does not need to be considered. Second, if $x_1 \geq y_1$ and $x_2 \geq y_2$ then $\sqrt{y_1^2+(y_2-1)^2} \geq 1$ and $\sqrt{(x_1+1)^2+x_2^2} \overset{(*)}{\geq} \frac{1}{\sqrt{2}}+\sqrt{x_1^2+x_2^2} > \frac{1}{\sqrt{2}}+\sqrt{(x_1-y_1)^2+(x_2-y_2)^2}$. Third, if $x_1 \leq y_1$ and $x_2 \leq y_2$ then $\sqrt{(x_1+1)^2+x_2^2} > \sqrt{2}$ and $\sqrt{y_1^2+(y_2-1)^2} > \sqrt{(y_1-x_1)^2+(y_2-x_2)^2}$. Fourth and finally, if $x_1 \geq y_1$ and $x_2 \leq y_2$ then it follows by simple calculations that $\sqrt{y_1^2+(y_2-1)^2}+\sqrt{(x_1+1)^2+x_2^2} \geq \sqrt{1+(y_2-1)^2}+x_1+1 > \sqrt{2}+\sqrt{(x_1-1)^2+(y_2-1)^2} \geq \sqrt{2}+\sqrt{(x_1-y_1)^2+(x_2-y_2)^2}$.

Case 5: Obviously, $\sqrt{y_1^2+(|y_2|+1)^2} > \sqrt{(|y_1|-1)^2+y_2^2}$ and $\sqrt{(x_1+1)^2+x_2^2} \overset{(*)}{\geq} \frac{1}{\sqrt{2}}+\sqrt{x_1^2+x_2^2} > \sqrt{2}-1+\sqrt{x_1^2+x_2^2}$.

Case 6: For the x-part we know $1+\sqrt{(x_1+1)^2+x_2^2} > \sqrt{2}+\sqrt{x_1^2+x_2^2}$. For the y-part, we exploit the precondition $|y_1| \geq y_2 \geq 1$. It follows by simple estimations that $\sqrt{y_1^2+(y_2-1)^2} \geq \sqrt{(|y_1|-1)^2+y_2^2}$.

Case 7: If $y_1 \geq 1$ and $y_2 \geq 2$ then $\sqrt{y_1^2+(y_2-1)^2} \geq 1+\sqrt{(y_1-1)^2+(y_2-2)^2}$. It follows that $\sqrt{y_1^2+(y_2-1)^2}+\sqrt{z_1^2+(z_2-1)^2}+\sqrt{(x_1+1)^2+x_2^2} \overset{(*)}{\geq} 2+\sqrt{(y_1-1)^2+(y_2-2)^2}+\sqrt{(z_1-1)^2+(z_2-2)^2}+\frac{1}{\sqrt{2}}+\sqrt{x_1^2+x_2^2} > \sqrt{2}+\sqrt{(y_1-z_1)^2+(y_2-z_2)^2}+\sqrt{x_1^2+x_2^2}$.

Two final remarks are due: The difference between the left-hand and right-hand side of the >-inequalities is in all the (sub)cases greater than 0.2. Hence, the tours depicted on the right-hand side are always strictly improving.

Moreover, the improved tours of Cases 5 and 6 are certainly not optimal: The middle vertex ⋄ of the (3×3)-vertex block is still connected to an outer vertex. Hence, these tours can be further shortened using one of the Cases 1, 2, or 3.

References

Yossiri Adulyasak, Jean-François Cordeau, and Raf Jans. Formulations and branch-and-cut algorithms for multivehicle production and inventory routing problems. *INFORMS Journal on Computing*, 26(1):103–120, 2014. doi: 10.1287/ijoc.2013.0550.

Dino Ahr. *Contributions to Multiple Postmen Problems*. Ph.d. dissertation, Department of Computer Science, Heidelberg University, Heidelberg, Germany, 2004.

Philippe Augerat. *Approche polyèdrale du problème de tournées de véhicules*. PhD thesis, Institut National Polytechnique de Grenoble, Grenoble, France, 1995.

Maria Battarra, Güneş Erdoğan, and Daniele Vigo. Exact algorithms for the clustered vehicle routing problem. *Operations Research*, 62(1):58–71, 2014. doi: 10.1287/opre.2013.1227.

Tolga Bektaş, Güneş Erdoğan, and Stefan Røpke. Formulations and branch-and-cut algorithms for the generalized vehicle routing problem. *Transportation Science*, 45(3):299–316, 2011. doi: 10.1287/trsc.1100.0352.

Christof Defryn and Kenneth Sörensen. A fast two-level variable neighborhood search for the clustered vehicle routing problem. *Computers & Operations Research*, 83:78–94, 2017. doi: 10.1016/j.cor.2017.02.007.

Elizabeth D. Dolan and Jorge J. Moré. Benchmarking optimization software with performance profiles. *Mathematical Programming*, 91(2):201–213, 2002. doi: 10.1007/s101070100263.

Mateo Fischetti, Juan J. Salazar Gonzáles, and Paolo Toth. A branch-and-cut algorithm for the symmetric generalized traveling salesman problem. *Operations Research*, 45(3):378–394, 1997. doi: 10.1287/opre.45.3.378.

Matteo Fischetti, Juan J. Salazar González, and Paolo Toth. Experiments with a multi-commodity formulation for the symmetric capacitated vehicle routing problem. In *Proceedings of the 3rd Meeting of the EURO Working Group on Transportation*, 1995.

Merrill M. Flood. The traveling-salesman problem. *Operations Research*, 4(1):61–75, 1956. URL https://www.jstor.org/stable/167517.

Bruce L. Golden, Edward A. Wasil, James P. Kelly, and I-Ming Chao. The impact of metaheuristics on solving the vehicle routing problem: Algorithms, problem sets, and computational results. In *Fleet Management and Logistics*, pages 33–56. Springer US, 1998. doi: 10.1007/978-1-4615-5755-5_2.

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

Timo Hintsch and Stefan Irnich. Large multiple neighborhood search for the clustered vehicle-routing problem. *European Journal of Operational Research*, 270(1):118–131, October 2018. doi: 10.1016/j.ejor.2018.02.056.

Timo Hintsch and Stefan Irnich. Exact solution of the soft-clustered vehicle-routing problem. *European Journal of Operational Research*, 280(1):164–178, 2020. doi: 10.1016/j.ejor.2019.07.019.

Timo Hintsch, Stefan Irnich, and Lone Kiilerich. Branch-price-and-cut for the soft-clustered capacitated arc-routing problem. Technical Report LM-2019-02, Chair of Logistics Management, Gutenberg School of Management and Economics, Johannes Gutenberg University Mainz, Mainz, Germany, 2019.

David R. Karger. Global min-cuts in RNC, and other ramifications of a simple min-out algorithm. In *Proceedings of the Fourth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '93, pages 21–30, Philadelphia, PA, USA, 1993. Society for Industrial and Applied Mathematics. ISBN 0-89871-313-7. URL http://dl.acm.org/citation.cfm?id=313559.313605.

Gilbert Laporte, Yves Nobert, and Martin Desrochers. Optimal routing under capacity and distance restrictions. *Operations Research*, 33(5):1050–1073, 1985. doi: 10.2307/170853.

Jens Lysgaard, Adam N. Letchford, and Richard W. Eglese. A new branch-and-cut algorithm for the capacitated vehicle routing problem. *Mathematical Programming*, 100(2):423–445, 2004. doi: 10.1007/s10107-003-0481-8.

Rafael Martinelli, Marcus Poggi, and Anand Subramanian. Improved bounds for large scale capacitated arc routing problem. *Computers & Operations Research*, 40(8):2145–2160, 2013. doi: 10.1016/j.cor.2013.02.013.

C.E. Miller, A.W. Tucker, and R.A. Zemlin. Integer programming formulations of traveling salesman problems. *Journal of the Association for Computing Machinery (JACM)*, 7:326–329, 1960. doi: 10.1145/321043.321046.

Paolo Toth and Daniele Vigo, editors. *Vehicle Routing*, volume 18 of *MOS-SIAM Series on Optimization*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 2014. ISBN 978-1-61197-358-7.

José M. Valério de Carvalho. Exact solution of bin-packing problems using column generation and branch-and-bound. *Annals of Operations Research*, 86:629–659, 1999. doi: 10.1023/A:1018952112615.

Thibaut Vidal, Maria Battarra, Anand Subramanian, and Güneş Erdoğan. Hybrid metaheuristics for the clustered vehicle routing problem. *Computers & Operations Research*, 58:87–99, June 2015. doi: 10.1016/j.cor.2014.10.019.

Online Appendix

In this Appendix, we present instance-by-instance results. The entries in the Tables 6–8 have the following meaning:

Group:	Subset of instances: `A`, `B`, `P`, `C`, and `G` for the `GVRP` instances; `Golden1-Golden20` for the `Golden` instances
No.:	Number of the instance for the self-generated, smaller-sized `Grid` instances
n:	Number of customers
k:	Number of vehicles in the original CVRP instance
N:	Number of customer clusters
m:	Number of vehicles for the SoftCluVRP
UB:	Upper bound; bold if $LB = UB$, i.e., optimality is proven
LB:	Lower bound; bold ditto

Moreover, we indicate which SoftCluVRP algorithm has computed/proven a(n) upper/lower bound for the first time ("first found by"):

HI20:	branch-and-price of Hintsch and Irnich (2020)
H19:	metaheuristic of Hintsch (2019)
DS17:	metaheuristic of Defryn and Sörensen (2017)
BEV19:	exact algorithm of Battarra et al. (2014)
B&C:	Trans:`dyn`, MTZ:`static`, Tree:`none`
*B&C:	Trans:`dyn`, MTZ:`static`, Tree:`none` with a time limit of 36,000 seconds (10 hours)
⋄B&C:	Trans:`dyn`, MTZ:`static`, Tree:`none`, BKS provided as UB
•B&C:	Trans:`static`, MTZ:`none`
†B&C:	Trans:`static`, MTZ:`static`, ProbCC:`none`
pretest:	Pretests with one of the algorithms described in Section 5.1
grid:	grid lower bound from Section 4.2
relax:	relax lower bound from model (6) with a time limit of 600 seconds, see Section 4.2

Table 6 displays the results for the `GVRP` instances, Table 7 for the `Golden` instances, and Table 8 for the self-generated `Grid` instances.

Table 6: Detailed results for the GVRP instances.

Instance					Results					
							first found by		time T	
Group	$n+1$	k	N	m	UB	LB	UB	LB	HI20	B&C
GVRP-2										
A	32	5	16	2	**595**	**595**	HI20	HI20	7	140
A	33	5	17	3	**528**	**528**	HI20	HI20	12	4
A	33	6	17	3	**561**	**561**	HI20	HI20	5	5
A	34	5	17	3	**568**	**568**	HI20	HI20	13	3
A	36	5	18	2	**596**	**596**	HI20	HI20	65	101
A	37	5	19	3	**573**	**573**	HI20	HI20	6	4
A	37	6	19	3	**660**	**660**	HI20	HI20	4	10
A	38	5	19	3	**547**	**547**	HI20	HI20	1	2
A	39	5	20	3	**659**	**659**	HI20	HI20	78	9
A	39	6	20	3	**676**	**676**	HI20	HI20	78	375
A	44	6	22	3	**723**	**723**	HI20	HI20	23	42
A	45	6	23	4	**679**	**679**	HI20	HI20	4	1
A	45	7	23	4	**774**	**774**	HI20	HI20	242	1856
A	46	7	23	4	**708**	**708**	HI20	HI20	209	49
A	48	7	24	4	**784**	**784**	HI20	HI20	1431	149
A	53	7	27	4	**732**	**732**	HI20	HI20	285	56
A	54	7	27	4	**806**	**806**	HI20	HI20	265	1147
A	55	9	28	5	**778**	**778**	HI20	HI20	84	68
A	60	9	30	5	**877**	**877**	HI20	HI20	2010	262
A	61	9	31	5	**749**	**749**	HI20	HI20	142	432
A	62	8	31	4	**849**	**849**	HI20	HI20	839	692
A	63	9	32	5	**1043**	**1043**	HI20	HI20	3159	*TL*
A	63	10	32	5	**895**	**895**	HI20	HI20	512	*TL*
A	64	9	32	5	**895**	**895**	HI20	HI20	1132	*TL*
A	65	9	33	5	**825**	**825**	HI20	HI20	2544	*TL*
A	69	9	35	5	**857**	**857**	HI20	HI20	2506	*TL*
A	80	10	40	5	**1115**	**1115**	HI20	HI20	*TL*	*TL*
B	31	5	16	3	**451**	**451**	HI20	HI20	7	1
B	34	5	17	3	**495**	**495**	HI20	HI20	26	<1
B	35	5	18	3	**654**	**654**	HI20	HI20	27	<1
B	38	6	19	3	**479**	**479**	HI20	HI20	3	2
B	39	5	20	3	**378**	**378**	HI20	HI20	5	<1
B	41	6	21	3	**514**	**514**	HI20	HI20	13	50
B	43	6	22	3	**522**	**522**	HI20	HI20	897	44
B	44	7	22	4	**562**	**562**	HI20	HI20	363	3
B	45	5	23	3	**542**	**542**	HI20	HI20	7	1
B	45	6	23	4	**506**	**506**	HI20	HI20	141	246
B	50	7	25	4	**495**	**495**	HI20	HI20	1	<1
B	50	8	25	5	**954**	**954**	HI20	B&C	*TL*	502
B	51	7	26	4	**672**	**672**	HI20	HI20	123	2
B	52	7	26	4	**485**	**485**	HI20	HI20	224	2
B	56	7	28	4	**520**	**520**	HI20	B&C	*TL*	61
B	57	7	29	4	**776**	**776**	H19	B&C	*TL*	7

Continued on next page

Instance					Results					
							first found by		time T	
Group	$n+1$	k	N	m	UB	LB	UB	LB	HI20	B&C
B	57	9	29	5	**983**	**983**	HI20	HI20	251	1204
B	63	10	32	5	**865**	**865**	HI20	HI20	1402	722
B	64	9	32	5	**550**	**550**	HI20	HI20	21	5
B	66	9	33	5	**849**	**849**	H19	B&C	*TL*	207
B	67	10	34	5	**721**	**721**	H19	B&C	*TL*	1192
B	68	9	34	5	**745**	**745**	HI20	HI20	293	21
B	78	10	39	5	**842**	**842**	HI20	HI20	*TL*	*TL*
P	16	8	8	5	**299**	**299**	HI20	HI20	<1	<1
P	19	2	10	2	**195**	**195**	HI20	HI20	<1	<1
P	20	2	10	2	**208**	**208**	HI20	HI20	<1	1
P	21	2	11	2	**208**	**208**	HI20	HI20	<1	<1
P	22	2	11	2	**209**	**209**	HI20	HI20	<1	1
P	22	8	11	5	**397**	**397**	HI20	HI20	<1	1
P	23	8	12	5	**369**	**369**	HI20	HI20	2	5
P	40	5	20	3	**401**	**401**	HI20	HI20	10	13
P	45	5	23	3	**443**	**443**	HI20	HI20	5	24
P	50	7	25	4	**464**	**464**	HI20	HI20	119	173
P	50	8	25	4	**501**	**501**	HI20	HI20	230	*TL*
P	50	10	25	5	**512**	**512**	HI20	HI20	56	358
P	51	10	26	6	**548**	**548**	HI20	HI20	21	114
P	55	7	28	4	**477**	**477**	HI20	HI20	8	284
P	55	8	28	4	**484**	**484**	HI20	HI20	40	2096
P	55	10	28	5	**514**	**514**	HI20	HI20	5	105
P	55	15	28	8	**684**	**684**	HI20	HI20	70	*TL*
P	60	10	30	5	**575**	**575**	HI20	HI20	725	*TL*
P	60	15	30	8	**700**	**700**	HI20	HI20	216	*TL*
P	65	10	33	5	**616**	**616**	HI20	HI20	291	*TL*
P	70	10	35	5	**643**	**643**	HI20	HI20	934	*TL*
P	76	4	38	2	**557**	**557**	HI20	HI20	2745	*TL*
P	76	5	38	3	**571**	**571**	HI20	HI20	2311	*TL*
P	101	4	51	2	**645**	**645**	H19	HI20	*TL*	*TL*
G	101	10	51	5	**628**	**628**	HI20	HI20	569	*TL*
C	121	7	61	4	799	782	H19	*B&C	*TL*	*TL*
C	151	12	76	6	805	793	H19	HI20	*TL*	*TL*
C	200	16	100	8	944	910	H19	⋄B&C	*TL*	*TL*
C	262	25	131	12	3655	3355	H19	B&C	*TL*	*TL*
GVRP-3										
A	32	5	11	2	**515**	**515**	DS17	HI20	1	<1
A	33	5	11	2	**461**	**461**	DS17	HI20	1	1
A	33	6	11	2	**554**	**554**	DS17	HI20	2	3
A	34	5	12	2	**538**	**538**	DS17	HI20	6	1
A	36	5	12	2	**543**	**543**	DS17	HI20	6	1
A	37	5	13	2	**545**	**545**	HI20	HI20	11	2
A	37	6	13	2	**605**	**605**	DS17	HI20	6	8
A	38	5	13	2	**507**	**507**	BEV19	HI20	5	<1

Continued on next page

					Results					
Instance							first found by		time T	
Group	$n+1$	k	N	m	UB	LB	UB	LB	HI20	B&C
A	39	5	13	2	**588**	**588**	DS17	HI20	13	3
A	39	6	13	2	**603**	**603**	DS17	HI20	6	4
A	44	6	15	2	**691**	**691**	DS17	HI20	27	246
A	45	6	15	3	**652**	**652**	DS17	HI20	2	4
A	45	7	15	3	**661**	**661**	DS17	HI20	29	11
A	46	7	16	3	**642**	**642**	DS17	HI20	12	18
A	48	7	16	3	**680**	**680**	DS17	HI20	57	34
A	53	7	18	3	**627**	**627**	DS17	HI20	14	<1
A	54	7	18	3	**699**	**699**	DS17	HI20	143	41
A	55	9	19	3	**645**	**645**	DS17	HI20	9	13
A	60	9	20	3	**762**	**762**	DS17	HI20	29	1127
A	61	9	21	4	**671**	**671**	DS17	HI20	23	75
A	62	8	21	3	**771**	**771**	DS17	HI20	404	48
A	63	9	21	3	**837**	**837**	DS17	HI20	43	432
A	63	10	21	4	**779**	**779**	DS17	HI20	13	61
A	64	9	22	3	**767**	**767**	DS17	HI20	585	934
A	65	9	22	3	**693**	**693**	DS17	HI20	14	17
A	69	9	23	3	**794**	**794**	DS17	HI20	603	*TL*
A	80	10	27	4	**944**	**944**	DS17	HI20	178	228
B	31	5	11	2	**375**	**375**	BEV19	HI20	4	<1
B	34	5	12	2	**415**	**415**	DS17	HI20	5	<1
B	35	5	12	2	**557**	**557**	DS17	HI20	18	<1
B	38	6	13	2	**427**	**427**	DS17	HI20	3	1
B	39	5	13	2	**317**	**317**	DS17	HI20	<1	<1
B	41	6	14	2	**469**	**469**	DS17	HI20	12	<1
B	43	6	15	2	**405**	**405**	DS17	HI20	8	1
B	44	7	15	3	**443**	**443**	DS17	HI20	7	2
B	45	5	15	2	**489**	**489**	DS17	HI20	3	<1
B	45	6	15	2	**386**	**386**	DS17	HI20	4	5
B	50	7	17	3	**464**	**464**	DS17	HI20	16	2
B	50	8	17	3	**661**	**661**	DS17	HI20	5	8
B	51	7	17	3	**578**	**578**	DS17	HI20	17	1
B	52	7	18	3	**427**	**427**	BEV19	HI20	11	1
B	56	7	19	3	**420**	**420**	DS17	HI20	16	3
B	57	7	19	3	**622**	**622**	DS17	HI20	437	1
B	57	9	19	3	**746**	**746**	DS17	HI20	1606	24
B	63	10	21	3	**685**	**685**	BEV19	HI20	21	3
B	64	9	22	4	**524**	**524**	DS17	HI20	32	6
B	66	9	22	3	**683**	**683**	DS17	HI20	252	74
B	67	10	23	4	**619**	**619**	DS17	HI20	72	7
B	68	9	23	3	**582**	**582**	DS17	HI20	25	40
B	78	10	26	4	**704**	**704**	DS17	HI20	1109	68
P	16	8	6	4	**251**	**251**	DS17	HI20	<1	<1
P	19	2	7	1	**170**	**170**	DS17	HI20	<1	<1
P	20	2	7	1	**177**	**177**	DS17	HI20	<1	<1
P	21	2	7	1	**179**	**179**	DS17	HI20	<1	<1

Continued on next page

Instance					Results					
							first found by		time T	
Group	$n+1$	k	N	m	UB	LB	UB	LB	HI20	B&C
P	22	2	8	1	**183**	**183**	DS17	HI20	<1	<1
P	22	8	8	4	**365**	**365**	BEV19	HI20	<1	<1
P	23	8	8	3	**270**	**270**	DS17	HI20	1	1
P	40	5	14	2	**381**	**381**	DS17	HI20	8	2
P	45	5	15	2	**422**	**422**	DS17	HI20	2	3
P	50	7	17	3	**430**	**430**	DS17	HI20	4	9
P	50	8	17	3	**441**	**441**	DS17	HI20	3	23
P	50	10	17	4	**471**	**471**	DS17	HI20	5	40
P	51	10	17	4	**493**	**493**	DS17	HI20	2	11
P	55	7	19	3	**454**	**454**	HI20	HI20	21	94
P	55	8	19	3	**454**	**454**	HI20	HI20	9	51
P	55	10	19	4	**481**	**481**	DS17	HI20	4	21
P	55	15	19	6	**572**	**572**	DS17	HI20	9	86
P	60	10	20	4	**534**	**534**	HI20	HI20	61	703
P	60	15	20	5	**591**	**591**	DS17	HI20	39	967
P	65	10	22	4	**575**	**575**	HI20	HI20	8	43
P	70	10	24	4	**602**	**602**	DS17	HI20	30	240
P	76	4	26	2	**556**	**556**	HI20	HI20	382	50
P	76	5	26	2	**556**	**556**	DS17	HI20	71	143
P	101	4	34	2	**649**	**649**	DS17	HI20	1899	*TL*
G	101	10	34	4	**598**	**598**	DS17	HI20	1707	1024
C	121	7	41	3	680	673	H19	*B&C	*TL*	*TL*
C	151	12	51	4	**756**	**756**	HI20	HI20	*TL*	*TL*
C	200	16	67	6	865	858	H19	HI20	*TL*	*TL*
C	262	25	88	9	3178	2974	H19	⋄B&C	*TL*	*TL*

Table 7: Detailed results for the Golden instances.

Instance				Results								
						first found by		time T		SoftCluVRP$^{\leq m}$		
Group	$n+1$	N	m	UB	LB	UB	LB	HI20	B&C	UB	LB	time T
Golden1	241	17	4	**4640**	**4640**	HI20	HI20	304	*TL*	**4531**	**4531**	2709
Golden1	241	18	4	4645	4636	HI20	HI20	*TL*	*TL*	4539	4535	*TL*
Golden1	241	19	4	4650	4647	HI20	HI20	*TL*	*TL*	4597	4523	*TL*
Golden1	241	21	4	4650	4640	HI20	HI20	*TL*	*TL*	4582	4518	*TL*
Golden1	241	22	4	4650	4638	H19	HI20	*TL*	*TL*	4595	4517	*TL*
Golden1	241	25	4	4650	4614	H19	HI20	*TL*	*TL*	4628	4505	*TL*
Golden1	241	27	4	4652	4624	H19	HI20	*TL*	*TL*	4652	4478	*TL*
Golden1	241	31	4	4665	4632	H19	HI20	*TL*	*TL*	4665	4474	*TL*
Golden1	241	35	4	4619	4583	H19	HI20	*TL*	*TL*	4619	4441	*TL*
Golden1	241	41	4	4619	4525	H19	B&C	*TL*	*TL*	4619	4470	*TL*
Golden1	241	49	4	4607	4525	H19	B&C	*TL*	*TL*	4607	4470	*TL*
Golden2	321	22	4	7394	7393	H19	HI20	*TL*	*TL*	7394	7135	*TL*
Golden2	321	23	4	**7369**	**7369**	HI20	HI20	2836	*TL*	7369	7129	*TL*
Golden2	321	25	4	7367	7366	H19	HI20	*TL*	*TL*	7367	7126	*TL*
Golden2	321	27	4	7333	7329	H19	HI20	*TL*	*TL*	7333	7080	*TL*
Golden2	321	30	4	7329	7162	H19	B&C	*TL*	*TL*	7329	7107	*TL*
Golden2	321	33	4	7311	7162	H19	B&C	*TL*	*TL*	7311	7107	*TL*
Golden2	321	36	4	7293	7162	H19	•B&C	*TL*	*TL*	7293	7107	*TL*
Golden2	321	41	4	7283	7161	H19	B&C	*TL*	*TL*	7283	7106	*TL*
Golden2	321	46	4	7284	7161	H19	B&C	*TL*	*TL*	7284	7106	*TL*
Golden2	321	54	4	7274	7160	H19	B&C	*TL*	*TL*	7274	7105	*TL*
Golden2	321	65	4	7261	7160	H19	B&C	*TL*	*TL*	7261	7105	*TL*
Golden3	401	27	4	10077	10064	H19	HI20	*TL*	*TL*	10077	9733	*TL*
Golden3	401	29	4	10018	9795	H19	pretest	*TL*	*TL*	10018	9727	*TL*
Golden3	401	31	4	10002	9783	H19	⋄B&C	*TL*	*TL*	10002	9723	*TL*
Golden3	401	34	4	9995	9772	H19	†B&C	*TL*	*TL*	9995	9713	*TL*
Golden3	401	37	4	9986	9762	H19	⋄B&C	*TL*	*TL*	9986	9708	*TL*
Golden3	401	41	4	9926	9763	H19	•B&C	*TL*	*TL*	9926	9712	*TL*
Golden3	401	45	4	9936	9759	H19	B&C	*TL*	*TL*	9936	9704	*TL*
Golden3	401	51	4	9916	9742	H19	B&C	*TL*	*TL*	9916	9687	*TL*
Golden3	401	58	4	9910	9741	H19	B&C	*TL*	*TL*	9910	9686	*TL*
Golden3	401	67	4	9901	9741	H19	B&C	*TL*	*TL*	9901	9686	*TL*
Golden3	401	81	4	9868	9740	H19	B&C	*TL*	*TL*	9868	9685	*TL*
Golden4	481	33	4	12741	12409	H19	pretest	*TL*	*TL*	12741	12331	*TL*
Golden4	481	35	4	12740	12427	H19	pretest	*TL*	*TL*	12740	12325	*TL*
Golden4	481	37	4	12645	12376	H19	⋄B&C	*TL*	*TL*	12645	12323	*TL*
Golden4	481	41	4	12568	12375	H19	relax	*TL*	*TL*	12568	12310	*TL*
Golden4	481	44	4	12566	12375	H19	relax	*TL*	*TL*	12566	12315	*TL*
Golden4	481	49	4	12566	12375	H19	relax	*TL*	*TL*	12566	12324	*TL*
Golden4	481	54	4	12525	12367	H19	relax	*TL*	*TL*	12525	12307	*TL*
Golden4	481	61	4	12558	12367	H19	relax	*TL*	*TL*	12558	12294	*TL*
Golden4	481	69	4	12573	12347	H19	B&C	*TL*	*TL*	12573	12292	*TL*
Golden4	481	81	4	12555	12339	H19	B&C	*TL*	*TL*	12555	12282	*TL*

Continued on next page

Instance				Results								
						first found by		time T		SoftCluVRP$^{\leq m}$		
Group	$n+1$	N	m	UB	LB	UB	LB	HI20	B&C	UB	LB	time T
Golden4	481	97	4	12528	12337	H19	B&C	*TL*	*TL*	12528	12282	*TL*
Golden5	201	14	4	**6970**	**6970**	HI20	HI20	212	533	**6742**	**6742**	377
Golden5	201	15	3	**6742**	**6742**	HI20	HI20	280	582	**6742**	**6742**	241
Golden5	201	16	3	**6742**	**6742**	HI20	HI20	142	922	**6742**	**6742**	261
Golden5	201	17	3	**6862**	**6862**	HI20	HI20	380	1731	**6862**	**6862**	783
Golden5	201	19	4	**6874**	**6874**	HI20	HI20	180	2907	**6735**	**6735**	2485
Golden5	201	21	4	**6816**	**6816**	HI20	HI20	666	2679	**6637**	**6637**	1136
Golden5	201	23	4	**6750**	**6750**	HI20	HI20	260	*TL*	**6637**	**6637**	3352
Golden5	201	26	4	**6704**	**6704**	HI20	HI20	647	*TL*	**6521**	**6521**	1959
Golden5	201	29	4	**6704**	**6704**	HI20	HI20	779	*TL*	**6521**	**6521**	*TL*
Golden5	201	34	4	**6684**	**6684**	HI20	HI20	*TL*	*TL*	6567	6389	*TL*
Golden5	201	41	4	**6557**	**6557**	HI20	HI20	2010	*TL*	6557	6317	*TL*
Golden6	281	19	3	**8115**	**8115**	HI20	HI20	1110	3140	**8115**	**8115**	1824
Golden6	281	21	3	**8119**	**8119**	HI20	HI20	901	*TL*	8119	8068	*TL*
Golden6	281	22	3	**8107**	**8107**	HI20	HI20	1053	*TL*	8107	8047	*TL*
Golden6	281	24	4	**8316**	**8316**	HI20	HI20	2491	*TL*	8267	8008	*TL*
Golden6	281	26	4	**8249**	**8249**	HI20	HI20	2568	*TL*	8225	7987	*TL*
Golden6	281	29	4	8244	8234	H19	HI20	*TL*	*TL*	8244	7966	*TL*
Golden6	281	32	4	8179	8175	H19	HI20	*TL*	*TL*	8179	7955	*TL*
Golden6	281	36	4	8179	8178	H19	HI20	*TL*	*TL*	8179	7947	*TL*
Golden6	281	41	4	8204	7995	H19	•B&C	*TL*	*TL*	8204	7938	*TL*
Golden6	281	47	4	8179	7970	H19	relax	*TL*	*TL*	8179	7913	*TL*
Golden6	281	57	4	8204	7960	H19	B&C	*TL*	*TL*	8204	7908	*TL*
Golden7	361	25	3	**9318**	**9318**	HI20	HI20	*TL*	*TL*	9318	9173	*TL*
Golden7	361	26	3	**9295**	**9295**	HI20	HI20	*TL*	*TL*	9295	9173	*TL*
Golden7	361	28	3	9271	9150	H19	†B&C	*TL*	*TL*	9271	9151	*TL*
Golden7	361	31	4	**9418**	**9418**	HI20	HI20	*TL*	*TL*	9418	9159	*TL*
Golden7	361	33	4	9395	9215	H19	•B&C	*TL*	*TL*	9395	9155	*TL*
Golden7	361	37	4	**9395**	**9395**	HI20	HI20	*TL*	*TL*	9395	9161	*TL*
Golden7	361	41	4	9386	9198	H19	⋄B&C	*TL*	*TL*	9386	9142	*TL*
Golden7	361	46	4	9368	9177	H19	†B&C	*TL*	*TL*	9368	9111	*TL*
Golden7	361	52	4	9365	9173	H19	⋄B&C	*TL*	*TL*	9365	9118	*TL*
Golden7	361	61	4	9316	9157	H19	B&C	*TL*	*TL*	9316	9102	*TL*
Golden7	361	73	4	9302	9157	H19	B&C	*TL*	*TL*	9302	9102	*TL*
Golden8	441	30	4	10409	10190	H19	pretest	*TL*	*TL*	10409	10122	*TL*
Golden8	441	32	4	10409	10197	H19	relax	*TL*	*TL*	10409	10120	*TL*
Golden8	441	34	4	10409	10177	H19	⋄B&C	*TL*	*TL*	10409	10121	*TL*
Golden8	441	37	4	10360	10198	H19	⋄B&C	*TL*	*TL*	10360	10142	*TL*
Golden8	441	41	4	10360	10219	H19	relax	*TL*	*TL*	10360	10155	*TL*
Golden8	441	45	4	10385	10198	H19	•B&C	*TL*	*TL*	10385	10141	*TL*
Golden8	441	49	4	10399	10195	H19	B&C	*TL*	*TL*	10399	10139	*TL*
Golden8	441	56	4	10371	10195	H19	B&C	*TL*	*TL*	10371	10139	*TL*
Golden8	441	63	4	10361	10195	H19	B&C	*TL*	*TL*	10361	10139	*TL*
Golden8	441	74	4	10356	10195	H19	B&C	*TL*	*TL*	10356	10139	*TL*
Golden8	441	89	4	10281	10195	H19	B&C	*TL*	*TL*	10281	10139	*TL*

Continued on next page

				Results								
Instance						first found by		time T		SoftCluVRP$^{\leq m}$		
Group	$n+1$	N	m	UB	LB	UB	LB	HI20	B&C	UB	LB	time T
Golden9	256	18	4	**281**	**281**	HI20	HI20	1287	*TL*	276	269	*TL*
Golden9	256	19	4	**279**	**279**	HI20	HI20	209	*TL*	276	269	*TL*
Golden9	256	20	4	**276**	**276**	HI20	HI20	112	*TL*	273	269	*TL*
Golden9	256	22	4	**276**	**276**	HI20	HI20	217	*TL*	276	269	*TL*
Golden9	256	24	4	**276**	**276**	HI20	HI20	175	*TL*	270	269	*TL*
Golden9	256	26	4	**273**	**273**	HI20	HI20	465	*TL*	273	266	*TL*
Golden9	256	29	4	**273**	**273**	HI20	HI20	985	*TL*	273	266	*TL*
Golden9	256	32	4	**273**	**273**	HI20	HI20	1650	*TL*	273	266	*TL*
Golden9	256	37	4	**273**	**273**	HI20	HI20	*TL*	*TL*	273	266	*TL*
Golden9	256	43	4	**270**	**270**	H19	HI20	*TL*	*TL*	270	262	*TL*
Golden9	256	52	4	269	268	H19	HI20	*TL*	*TL*	269	262	*TL*
Golden10	324	22	4	**346**	**346**	HI20	HI20	923	*TL*	346	345	*TL*
Golden10	324	24	4	**346**	**346**	HI20	HI20	1014	*TL*	346	345	*TL*
Golden10	324	25	4	**346**	**346**	HI20	HI20	1114	*TL*	346	345	*TL*
Golden10	324	27	4	**346**	**346**	HI20	HI20	1360	*TL*	346	345	*TL*
Golden10	324	30	4	**347**	**347**	HI20	HI20	1848	*TL*	347	345	*TL*
Golden10	324	33	4	**344**	**344**	HI20	HI20	2725	*TL*	344	341	*TL*
Golden10	324	36	4	**344**	**344**	HI20	HI20	*TL*	*TL*	344	341	*TL*
Golden10	324	41	4	346	340	H19	grid	*TL*	*TL*	346	340	*TL*
Golden10	324	47	4	344	340	H19	grid	*TL*	*TL*	344	340	*TL*
Golden10	324	54	4	340	338	H19	grid	*TL*	*TL*	340	338	*TL*
Golden10	324	65	4	335	334	H19	grid	*TL*	*TL*	335	334	*TL*
Golden11	400	27	5	**434**	**434**	HI20	HI20	*TL*	*TL*	434	423	*TL*
Golden11	400	29	5	**434**	**434**	HI20	HI20	*TL*	*TL*	434	423	*TL*
Golden11	400	31	5	**433**	**433**	HI20	HI20	2661	*TL*	433	423	*TL*
Golden11	400	34	5	**427**	**427**	HI20	HI20	*TL*	*TL*	427	416	*TL*
Golden11	400	37	5	**427**	**427**	H19	HI20	*TL*	*TL*	427	416	*TL*
Golden11	400	40	5	**425**	**425**	H19	HI20	*TL*	*TL*	425	415	*TL*
Golden11	400	45	5	**425**	**425**	H19	HI20	*TL*	*TL*	425	415	*TL*
Golden11	400	50	5	423	422	H19	grid	*TL*	*TL*	423	415	*TL*
Golden11	400	58	5	**422**	**422**	H19	grid	*TL*	*TL*	422	415	*TL*
Golden11	400	67	5	**422**	**422**	H19	grid	*TL*	*TL*	422	415	*TL*
Golden11	400	80	5	417	416	H19	grid	*TL*	*TL*	417	410	*TL*
Golden12	484	33	5	512	507	H19	grid	*TL*	*TL*	512	500	*TL*
Golden12	484	35	5	512	507	H19	grid	*TL*	*TL*	512	500	*TL*
Golden12	484	38	5	511	507	H19	grid	*TL*	*TL*	511	500	*TL*
Golden12	484	41	5	512	507	H19	grid	*TL*	*TL*	512	500	*TL*
Golden12	484	44	5	511	507	H19	grid	*TL*	*TL*	511	500	*TL*
Golden12	484	49	5	511	507	H19	grid	*TL*	*TL*	511	500	*TL*
Golden12	484	54	5	510	507	H19	grid	*TL*	*TL*	510	500	*TL*
Golden12	484	61	5	510	507	H19	grid	*TL*	*TL*	510	500	*TL*
Golden12	484	70	5	509	506	H19	grid	*TL*	*TL*	509	499	*TL*
Golden12	484	81	5	502	498	H19	grid	*TL*	*TL*	502	493	*TL*
Golden12	484	97	5	502	498	H19	grid	*TL*	*TL*	502	493	*TL*
Golden13	253	17	4	**530**	**530**	HI20	HI20	116	2378	**519**	**519**	506

Continued on next page

				Results								
Instance						first found by		time T		SoftCluVRP$^{\leq m}$		
Group	$n+1$	N	m	*UB*	*LB*	*UB*	*LB*	HI20	B&C	*UB*	*LB*	time T
Golden13	253	19	4	**521**	**521**	HI20	HI20	189	983	**516**	**516**	26
Golden13	253	20	4	**521**	**521**	HI20	HI20	192	538	**516**	**516**	13
Golden13	253	22	4	**523**	**523**	HI20	HI20	203	*TL*	**516**	**516**	48
Golden13	253	23	4	**523**	**523**	HI20	HI20	215	3496	**516**	**516**	73
Golden13	253	26	4	**523**	**523**	HI20	HI20	118	*TL*	**516**	**516**	74
Golden13	253	29	4	**522**	**522**	HI20	HI20	1483	*TL*	**516**	**516**	1551
Golden13	253	32	4	**521**	**521**	HI20	HI20	286	*TL*	**516**	**516**	*TL*
Golden13	253	37	4	**521**	**521**	HI20	HI20	2305	*TL*	521	516	*TL*
Golden13	253	43	4	**521**	**521**	HI20	HI20	*TL*	*TL*	521	516	*TL*
Golden13	253	51	4	**521**	**521**	H19	HI20	*TL*	*TL*	521	516	*TL*
Golden14	321	22	4	**665**	**665**	HI20	HI20	1814	*TL*	665	653	*TL*
Golden14	321	23	4	**662**	**662**	HI20	HI20	752	*TL*	655	652	*TL*
Golden14	321	25	4	**660**	**660**	HI20	HI20	637	*TL*	653	652	*TL*
Golden14	321	27	4	**660**	**660**	HI20	HI20	3067	*TL*	**652**	**652**	3067
Golden14	321	30	4	**660**	**660**	HI20	HI20	*TL*	*TL*	**652**	**652**	*TL*
Golden14	321	33	4	**660**	**660**	H19	HI20	*TL*	*TL*	660	652	*TL*
Golden14	321	36	4	**658**	**658**	H19	HI20	*TL*	*TL*	658	652	*TL*
Golden14	321	41	4	**658**	**658**	HI20	HI20	*TL*	*TL*	658	652	*TL*
Golden14	321	46	4	658	657	H19	grid	*TL*	*TL*	658	652	*TL*
Golden14	321	54	4	658	657	H19	grid	*TL*	*TL*	658	652	*TL*
Golden14	321	65	4	658	657	H19	grid	*TL*	*TL*	658	652	*TL*
Golden15	397	27	4	**815**	**815**	H19	HI20	*TL*	*TL*	815	813	*TL*
Golden15	397	29	4	**815**	**815**	H19	HI20	*TL*	*TL*	815	813	*TL*
Golden15	397	31	4	**813**	**813**	HI20	HI20	3176	*TL*	**813**	**813**	*TL*
Golden15	397	34	4	**813**	**813**	H19	HI20	*TL*	*TL*	**813**	**813**	*TL*
Golden15	397	37	4	815	813	H19	grid	*TL*	*TL*	815	813	*TL*
Golden15	397	40	4	815	813	H19	grid	*TL*	*TL*	815	813	*TL*
Golden15	397	45	5	817	815	H19	relax	*TL*	*TL*	817	808	*TL*
Golden15	397	50	5	**815**	**815**	H19	relax	*TL*	*TL*	815	808	*TL*
Golden15	397	57	5	**815**	**815**	H19	relax	*TL*	*TL*	815	808	*TL*
Golden15	397	67	5	**815**	**815**	H19	relax	*TL*	*TL*	815	808	*TL*
Golden15	397	80	5	**815**	**815**	H19	relax	*TL*	*TL*	815	808	*TL*
Golden16	481	33	5	993	990	H19	grid	*TL*	*TL*	993	980	*TL*
Golden16	481	35	5	**993**	**993**	H19	HI20	*TL*	*TL*	993	980	*TL*
Golden16	481	37	5	993	992	H19	HI20	*TL*	*TL*	993	980	*TL*
Golden16	481	41	5	993	990	H19	grid	*TL*	*TL*	993	980	*TL*
Golden16	481	44	5	993	990	H19	grid	*TL*	*TL*	993	980	*TL*
Golden16	481	49	5	989	987	H19	grid	*TL*	*TL*	989	979	*TL*
Golden16	481	54	5	985	984	H19	grid	*TL*	*TL*	985	977	*TL*
Golden16	481	61	5	985	984	H19	grid	*TL*	*TL*	985	977	*TL*
Golden16	481	69	5	**984**	**984**	H19	grid	*TL*	*TL*	984	977	*TL*
Golden16	481	81	5	**984**	**984**	H19	grid	*TL*	*TL*	984	977	*TL*
Golden16	481	97	5	**984**	**984**	H19	grid	*TL*	*TL*	984	977	*TL*
Golden17	241	17	3	**386**	**386**	HI20	HI20	132	417	**386**	**386**	269
Golden17	241	18	3	**385**	**385**	HI20	HI20	290	341	**385**	**385**	290

Continued on next page

Instance				Results								
						first found by		time T		SoftCluVRP$^{\leq m}$		
Group	$n+1$	N	m	UB	LB	UB	LB	HI20	B&C	UB	LB	time T
Golden17	241	19	3	**385**	**385**	HI20	HI20	220	342	**385**	**385**	295
Golden17	241	21	3	**385**	**385**	HI20	HI20	457	625	**385**	**385**	316
Golden17	241	22	3	**385**	**385**	HI20	HI20	372	750	**385**	**385**	570
Golden17	241	25	3	**382**	**382**	HI20	HI20	487	1093	**382**	**382**	619
Golden17	241	27	3	**382**	**382**	HI20	HI20	1039	1851	**382**	**382**	1774
Golden17	241	31	4	**390**	**390**	HI20	HI20	661	1707	**382**	**382**	2582
Golden17	241	35	4	390	389	H19	HI20	*TL*	*TL*	381	379	*TL*
Golden17	241	41	4	**388**	**388**	HI20	HI20	*TL*	*TL*	382	378	*TL*
Golden17	241	49	4	387	386	H19	HI20	*TL*	*TL*	387	376	*TL*
Golden18	301	21	4	**558**	**558**	HI20	HI20	694	1571	552	547	*TL*
Golden18	301	22	4	**558**	**558**	HI20	HI20	781	1881	**553**	**553**	*TL*
Golden18	301	24	4	**558**	**558**	HI20	HI20	831	*TL*	553	548	*TL*
Golden18	301	26	4	**562**	**562**	HI20	HI20	974	*TL*	552	544	*TL*
Golden18	301	28	4	**558**	**558**	HI20	HI20	*TL*	*TL*	551	541	*TL*
Golden18	301	31	4	**554**	**554**	HI20	HI20	2450	*TL*	554	541	*TL*
Golden18	301	34	4	**554**	**554**	HI20	HI20	1992	*TL*	554	540	*TL*
Golden18	301	38	4	**555**	**555**	HI20	HI20	*TL*	*TL*	551	540	*TL*
Golden18	301	43	4	558	550	H19	⋄B&C	*TL*	*TL*	558	540	*TL*
Golden18	301	51	4	555	549	H19	†B&C	*TL*	*TL*	555	540	*TL*
Golden18	301	61	4	556	548	H19	•B&C	*TL*	*TL*	556	538	*TL*
Golden19	361	25	10	**886**	**886**	HI20	HI20	538	*TL*	738	730	*TL*
Golden19	361	26	10	**888**	**888**	HI20	HI20	1208	*TL*	763	725	*TL*
Golden19	361	28	4	**741**	**741**	HI20	HI20	1479	*TL*	741	730	*TL*
Golden19	361	31	4	**735**	**735**	HI20	HI20	*TL*	*TL*	735	728	*TL*
Golden19	361	33	4	**727**	**727**	HI20	HI20	2719	*TL*	727	723	*TL*
Golden19	361	37	5	**732**	**732**	HI20	HI20	2612	*TL*	732	716	*TL*
Golden19	361	41	5	**730**	**730**	HI20	HI20	*TL*	*TL*	730	714	*TL*
Golden19	361	46	5	730	721	H19	B&C	*TL*	*TL*	730	713	*TL*
Golden19	361	52	5	**730**	**730**	HI20	HI20	*TL*	*TL*	730	712	*TL*
Golden19	361	61	5	737	721	H19	B&C	*TL*	*TL*	737	713	*TL*
Golden19	361	73	5	736	721	H19	B&C	*TL*	*TL*	736	712	*TL*
Golden20	421	29	11	**1170**	**1170**	HI20	HI20	1099	*TL*	1052	971	*TL*
Golden20	421	31	12	**1183**	**1183**	HI20	HI20	1080	*TL*	1088	966	*TL*
Golden20	421	33	12	**1175**	**1175**	HI20	HI20	2381	*TL*	1162	966	*TL*
Golden20	421	36	5	**1005**	**1005**	H19	HI20	*TL*	*TL*	1005	963	*TL*
Golden20	421	39	5	991	971	H19	B&C	*TL*	*TL*	991	961	*TL*
Golden20	421	43	5	990	971	H19	†B&C	*TL*	*TL*	990	962	*TL*
Golden20	421	47	5	988	970	H19	⋄B&C	*TL*	*TL*	988	961	*TL*
Golden20	421	53	5	988	970	H19	relax	*TL*	*TL*	988	961	*TL*
Golden20	421	61	5	987	970	H19	relax	*TL*	*TL*	987	961	*TL*
Golden20	421	71	5	986	970	H19	relax	*TL*	*TL*	986	961	*TL*
Golden20	421	85	5	980	969	H19	relax	*TL*	*TL*	980	960	*TL*

Table 8: Detailed results for square `Grid` instances.

Instance				Results											
				reduced						non-reduced					
No.	n	N	m	UB	LB	time T	#nodes	#cols	#rows	UB	LB	time T	#nodes	#cols	#rows
1	121	6	2	**134.9**	**134.9**	11	55	6.240	4.428	**134.9**	**134.9**	22	89	7.281	5.211
2	121	6	2	**135.5**	**135.5**	10	48	5.383	3.836	**135.5**	**135.5**	18	17	7.281	4.938
3	121	6	2	**130.7**	**130.7**	14	273	5.192	3.967	**130.7**	**130.7**	13	82	7.281	5.666
4	121	6	2	**123.7**	**123.7**	7	104	6.144	4.730	**123.7**	**123.7**	4	102	7.281	5.571
5	121	6	2	**124.9**	**124.9**	3	17	5.846	4.571	**124.9**	**124.9**	6	544	7.281	5.735
6	121	6	2	**128.4**	**128.4**	11	28	6.958	4.949	**128.4**	**128.4**	10	253	7.281	5.191
7	121	8	3	**144.7**	**144.7**	82	1.252	6.657	5.598	**144.7**	**144.7**	69	839	7.296	6.132
8	121	8	3	**151.7**	**151.7**	256	6.573	6.357	5.037	**151.7**	**151.7**	114	904	7.296	5.769
9	121	8	3	**151.7**	**151.7**	30	29	6.069	4.881	**151.7**	**151.7**	58	309	7.296	5.779
10	121	8	3	**128.9**	**128.9**	16	660	6.663	5.587	**128.9**	**128.9**	19	423	7.296	6.122
11	121	8	3	**131.6**	**131.6**	28	29	7.080	5.967	**131.6**	**131.6**	21	60	7.296	6.161
12	121	8	4	**134.0**	**134.0**	28	21	6.263	4.677	**134.0**	**134.0**	47	27	7.296	5.416
13	121	10	3	**147.6**	**147.6**	156	2.461	7.207	6.137	**147.6**	**147.6**	247	4.072	7.315	6.231
14	121	10	3	**148.0**	**148.0**	470	4.045	6.891	5.710	148.0	147.9	413	6.264	7.315	6.052
15	121	10	4	**157.8**	**157.8**	140	2.283	6.783	5.769	**157.8**	**157.8**	240	2.931	7.315	6.243
16	121	10	4	**137.0**	**137.0**	65	414	6.993	5.968	**137.0**	**137.0**	64	393	7.315	6.265
17	121	10	4	**134.4**	**134.4**	34	629	6.991	5.643	**134.4**	**134.4**	54	197	7.315	5.859
18	121	10	3	**127.7**	**127.7**	68	5.830	6.991	6.093	**127.7**	**127.7**	26	633	7.315	6.360
19	121	12	5	181.8	181.7	859	10.619	6.696	5.747	181.8	181.7	758	6.614	7.338	6.278
20	121	12	5	**185.8**	**185.8**	2607	19.868	7.338	6.490	**185.8**	**185.8**	2607	19.868	7.338	6.490
21	121	12	4	168.7	164.6	*TL*	34.449	6.284	5.299	168.7	162.0	*TL*	25.116	7.338	6.089
22	121	12	4	**144.7**	**144.7**	217	3.668	6.588	5.675	**144.7**	**144.7**	341	7.769	7.338	6.326
23	121	12	4	142.8	142.7	529	7.727	6.912	6.015	**142.8**	**142.8**	435	4.093	7.338	6.388
24	121	12	4	**149.3**	**149.3**	554	7.700	7.338	6.265	**149.3**	**149.3**	552	8.939	7.338	6.265
25	121	14	5	**174.1**	**174.1**	2224	26.489	7.046	6.207	174.5	169.3	*TL*	33.923	7.365	6.477
26	121	14	5	**166.2**	**166.2**	1506	21.842	7.365	6.708	**166.2**	**166.2**	1507	21.842	7.365	6.708
27	121	14	4	166.2	150.4	*TL*	18.496	7.365	6.789	161.8	152.2	*TL*	25.457	7.365	6.789
28	121	14	5	**136.6**	**136.6**	96	1.249	7.365	6.590	**136.6**	**136.6**	76	802	7.365	6.590
29	121	14	4	**137.4**	**137.4**	768	14.998	6.934	6.000	**137.4**	**137.4**	292	4.003	7.365	6.378
30	121	14	5	**144.9**	**144.9**	186	4.409	6.828	5.968	**144.9**	**144.9**	483	17.003	7.365	6.398
31	169	6	2	**183.2**	**183.2**	41	116	11.406	8.813	**183.2**	**183.2**	65	262	14.217	11.262
32	169	6	3	**191.7**	**191.7**	49	35	10.575	8.497	**191.7**	**191.7**	71	63	14.217	11.586
33	169	6	2	**189.9**	**189.9**	45	374	10.337	7.782	**189.9**	**189.9**	64	110	14.217	10.614
34	169	6	2	**175.1**	**175.1**	24	53	7.885	5.638	**175.1**	**175.1**	9	9	14.217	8.928
35	169	6	3	**178.0**	**178.0**	21	31	13.006	10.166	**178.0**	**178.0**	23	19	14.217	11.165
36	169	6	2	171.7	171.6	11	73	11.402	9.128	**171.7**	**171.7**	8	21	14.217	11.387
37	169	8	3	217.8	211.3	*TL*	42.500	13.012	11.147	**217.0**	**217.0**	1220	1.848	14.232	12.215
38	169	8	2	**184.2**	**184.2**	79	312	12.701	10.475	**184.2**	**184.2**	58	289	14.232	11.833
39	169	8	3	**194.0**	**194.0**	135	2.243	10.744	8.805	**194.0**	**194.0**	71	235	14.232	11.609
40	169	8	3	**180.2**	**180.2**	31	244	12.416	10.200	**180.2**	**180.2**	73	1.547	14.232	11.756
41	169	8	3	**178.5**	**178.5**	56	670	12.580	10.461	**178.5**	**178.5**	56	800	14.232	11.906
42	169	8	3	**184.3**	**184.3**	73	645	11.871	9.374	**184.3**	**184.3**	96	1.004	14.232	11.224
43	169	10	4	**218.7**	**218.7**	2026	6.464	12.588	11.021	**218.7**	**218.7**	443	2.679	14.251	12.430
44	169	10	4	**221.3**	**221.3**	525	3.100	13.186	11.171	**221.3**	**221.3**	1069	3.781	14.251	12.108
45	169	10	3	209.6	197.8	*TL*	17.723	11.754	9.900	217.3	187.9	*TL*	17.151	14.251	12.080
46	169	10	3	**183.1**	**183.1**	215	3.825	11.040	8.812	**183.1**	**183.1**	310	5.254	14.251	11.033
47	169	10	3	190.8	184.5	*TL*	27.593	13.179	11.516	189.3	181.1	*TL*	34.370	14.251	12.487
48	169	10	3	**178.6**	**178.6**	63	985	12.005	10.323	**178.6**	**178.6**	362	5.104	14.251	12.298
49	169	12	4	**222.4**	**222.4**	3559	18.086	12.463	10.411	222.4	217.6	*TL*	14.217	14.274	11.743
50	169	12	5	**237.8**	**237.8**	923	7.220	11.471	9.600	**237.8**	**237.8**	941	8.284	14.274	11.774
51	169	12	4	**213.4**	**213.4**	1688	16.525	12.310	10.715	214.7	204.2	*TL*	14.793	14.274	12.365
52	169	12	4	**186.0**	**186.0**	420	2.353	11.763	9.977	**186.0**	**186.0**	968	3.848	14.274	12.060
53	169	12	4	**184.0**	**184.0**	94	502	13.963	12.297	**184.0**	**184.0**	133	1.479	14.274	12.589
54	169	12	5	**193.4**	**193.4**	198	1.217	11.730	10.291	**193.4**	**193.4**	202	1.789	14.274	12.416
55	169	14	5	241.8	220.4	*TL*	17.207	12.501	10.997	244.0	218.8	*TL*	10.514	14.301	12.560
56	169	14	5	239.2	220.3	*TL*	14.034	13.846	12.227	243.6	209.2	*TL*	12.467	14.301	12.602
57	169	14	5	254.1	207.1	*TL*	5.475	13.990	12.422	273.3	210.7	*TL*	4.615	14.301	12.706
58	169	14	5	**200.2**	**200.2**	335	3.739	14.143	12.149	**200.2**	**200.2**	331	2.508	14.301	12.304
59	169	14	5	**192.5**	**192.5**	309	3.158	14.301	12.997	**192.5**	**192.5**	309	3.158	14.301	12.997
60	169	14	4	**181.5**	**181.5**	228	2.394	11.347	9.844	**181.5**	**181.5**	444	3.796	14.301	12.044
61	225	6	2	**238.0**	**238.0**	3084	143.818	13.152	9.517	**238.0**	**238.0**	749	16.137	25.221	18.062
62	225	6	3	**268.2**	**268.2**	82	89	15.012	10.724	**268.2**	**268.2**	152	162	25.221	16.435
63	225	6	3	**259.9**	**259.9**	67	98	14.081	10.246	**259.9**	**259.9**	493	3.139	25.221	18.243
64	225	6	3	**238.8**	**238.8**	26	178	12.956	9.056	**238.8**	**238.8**	32	44	25.221	16.670
65	225	6	3	**237.5**	**237.5**	30	25	18.579	14.800	**237.5**	**237.5**	55	355	25.221	20.660
66	225	6	2	**229.3**	**229.3**	22	26	15.123	11.577	**229.3**	**229.3**	33	66	25.221	18.956
67	225	8	2	**245.5**	**245.5**	308	1.372	14.857	11.298	**245.5**	**245.5**	789	1.201	25.236	19.394
68	225	8	3	**255.5**	**255.5**	137	229	16.313	12.624	**255.5**	**255.5**	262	381	25.236	19.495
69	225	8	3	**249.0**	**249.0**	237	946	19.107	16.176	**249.0**	**249.0**	105	122	25.236	21.440

Continued on next page

Instance				Results											
				reduced						non-reduced					
No.	n	N	m	UB	LB	time T	#nodes	#cols	#rows	UB	LB	time T	#nodes	#cols	#rows
70	225	8	2	**229.9**	**229.9**	64	194	17.211	13.317	**229.9**	**229.9**	116	54	25.236	19.093
71	225	8	3	**237.1**	**237.1**	78	203	20.984	17.498	**237.1**	**237.1**	57	234	25.236	21.324
72	225	8	3	**232.0**	**232.0**	22	115	19.659	16.305	**232.0**	**232.0**	38	204	25.236	20.891
73	225	10	4	**279.0**	**279.0**	860	2.002	20.265	17.105	**279.0**	**279.0**	1852	3.746	25.255	21.286
74	225	10	3	252.9	242.8	TL	13.441	22.596	19.747	252.4	245.6	TL	10.978	25.255	22.084
75	225	10	3	**267.7**	**267.7**	2462	12.827	19.932	14.895	268.3	258.6	TL	10.959	25.255	18.826
76	225	10	4	**236.9**	**236.9**	97	519	22.168	19.294	236.9	236.8	98	508	25.255	22.111
77	225	10	4	**242.7**	**242.7**	1164	66.996	20.651	17.535	**242.7**	**242.7**	121	1.097	25.255	21.480
78	225	10	4	**242.4**	**242.4**	96	326	20.995	18.206	**242.4**	**242.4**	88	268	25.255	21.813
79	225	12	4	307.6	265.9	TL	5.107	21.421	17.484	308.5	259.5	TL	6.998	25.278	20.293
80	225	12	4	293.1	249.0	TL	7.661	21.616	18.943	283.9	254.7	TL	8.985	25.278	22.044
81	225	12	4	282.6	256.3	TL	6.601	21.804	19.103	285.6	257.6	TL	5.383	25.278	22.141
82	225	12	5	254.6	254.5	514	5.185	22.210	19.659	254.6	254.5	511	3.118	25.278	22.412
83	225	12	4	**243.7**	**243.7**	997	8.931	20.650	17.955	**243.7**	**243.7**	614	4.816	25.278	21.736
84	225	12	3	**239.3**	**239.3**	1011	9.041	24.853	22.162	**239.3**	**239.3**	1611	14.411	25.278	22.548
85	225	14	5	371.8	259.1	TL	6.599	23.430	20.701	360.9	267.6	TL	6.127	25.305	22.391
86	225	14	5	330.5	272.1	TL	10.855	21.054	18.595	325.0	257.7	TL	4.188	25.305	22.165
87	225	14	5	335.3	261.1	TL	4.371	22.429	20.178	335.3	264.6	TL	3.415	25.305	22.758
88	225	14	5	252.2	252.1	1242	13.298	21.057	18.070	252.2	252.1	1177	7.772	25.305	21.458
89	225	14	4	285.4	233.6	TL	10.966	22.424	20.345	272.6	233.1	TL	6.631	25.305	23.019
90	225	14	4	249.5	249.4	2711	18.971	22.008	19.684	249.5	249.4	3262	21.263	25.305	22.616
Total (90)						1056	8291	12832	10747			1072	5470	15611	12909