Exact and heuristic solution of the consistent vehicle-routing problem

Goeke, Dominik; Roberti, Roberto; Schneider, Michael

published in
Transportation Science
2019

DOI (link to publisher)
10.1287/trsc.2018.0864

document version
Publisher's PDF, also known as Version of record

Transportation Science

Publication details, including instructions for authors and subscription information:
http://pubsonline.informs.org

Exact and Heuristic Solution of the Consistent Vehicle-Routing Problem

Dominik Goeke, Roberto Roberti, Michael Schneider

To cite this article:
Dominik Goeke, Roberto Roberti, Michael Schneider (2019) Exact and Heuristic Solution of the Consistent Vehicle-Routing Problem. Transportation Science 53(4):1023-1042. https://doi.org/10.1287/trsc.2018.0864

http://pubsonline.informs.org/journal/trsc/

TRANSPORTATION SCIENCE
Vol. 53, No. 4, July–August 2019, pp. 1023–1042
ISSN 0041-1655 (print), ISSN 1526-5447 (online)

Exact and Heuristic Solution of the Consistent Vehicle-Routing Problem

Dominik Goeke,[a] Roberto Roberti,[b] Michael Schneider[a]

[a] School of Business and Economics, RWTH Aachen University, 52062 Aachen, Germany; [b] Department of Information, Logistics and Innovation, VU Amsterdam, 1081 HV Amsterdam, Netherlands

Contact: goeke@dpo.rwth-aachen.de, http://orcid.org/0000-0003-1828-0285 (DG); r.roberti@vu.nl, http://orcid.org/0000-0002-2987-1593 (RR); schneider@dpo.rwth-aachen.de, http://orcid.org/0000-0002-4203-8926 (MS)

Received: November 17, 2017
Revised: May 17, 2018
Accepted: July 18, 2018
Published Online in Articles in Advance: June 28, 2019

https://doi.org/10.1287/trsc.2018.0864

Abstract. Providing consistent service by satisfying customer demands with the same driver (driver consistency) at approximately the same time (arrival-time consistency) allows companies in last-mile distribution to stand out among competitors. The consistent vehicle-routing problem (ConVRP) is a multiday problem addressing such consistency requirements along with traditional constraints on vehicle capacity and route duration. The literature offers several heuristics but no exact method for this problem. The state-of-the-art exact technique to solve VRPs—column generation (CG) applied to route-based formulations in which columns are generated via dynamic programming—cannot be successfully extended to the ConVRP because the linear relaxation of route-based formulations is weak. We propose the first exact method for the ConVRP, which can solve medium-sized instances with five days and 30 customers. The method solves, via CG, a formulation in which each variable represents the set of routes assigned to a vehicle over the planning horizon. As an upper bounding procedure, we develop a large neighborhood search (LNS) featuring a repair procedure specifically designed to improve the arrival-time consistency of solutions. Used as stand-alone heuristic, the LNS is able to significantly improve the solution quality on benchmark instances from the literature compared with state-of-the-art heuristics.

Supplemental Material: The online appendix is available at https://doi.org/10.1287/trsc.2018.0864.

Keywords: customer service • column generation • large neighborhood search • consistent vehicle routing problems

1. Introduction

Vehicle-routing problems (VRPs) with consistency considerations have received substantial interest in recent years because of the practical importance of providing consistent service in many industries, such as, for example, small package shipping, healthcare, or vendor-managed inventory systems (for a survey, see Kovacs et al. 2014). To boost customer satisfaction, customers should be served at roughly the same time (arrival-time consistency, ATC) by the same driver (driver consistency, DC) or at least a small set of familiar drivers each time they require service. Taking the driver's perspective, serving the same customers repeatedly makes the driver familiar with the geographic region and the characteristics of the customer and, thus, more efficient in fulfilling the tasks.

The most prominent variant of the class of VRPs with consistency considerations is the consistent VRP (ConVRP), introduced by Groër, Golden, and Wasil (2009). The ConVRP is a multiday VRP requiring that, in addition to the traditional constraints on vehicle capacity and route duration, the same driver serves the same customers at approximately the same time on each day that these customers require service, given by a maximum allowed difference between the arrival times on the different days. Originally, the problem is motivated from the delivery and collection operations at United Parcel Services, where strong emphasis is put on customer and employee satisfaction.

In the academic literature, the ConVRP has received adequate attention from the heuristic side. Groër, Golden, and Wasil (2009) develop a two-phase algorithm based on record-to-record travel, which first constructs template routes and then uses them to generate the daily routes by removing nonoccurring customers and inserting new ones. The template routes are based on a simple precedence principle, which states that, if two customers a and b are served by the same driver on a specific day, then the driver who serves them and the order in which they are served must be the same on all days on which they both require service. Since the publication of this article, four algorithms have been proposed that are able to solve ConVRP instances, three of which use the idea of a template that is adjusted to the individual days (Sungur et al. 2010; Tarantilis, Stavropoulou, and Repoussis 2012; Kovacs, Parragh,

and Hartl 2014) and one approach that applies search over all routes of all days (Kovacs et al. 2015).

Sungur et al. (2010) actually solve a different problem, called the courier delivery problem, which is modeled as a multiday VRP with soft time windows, using robust optimization and scenario-based stochastic programming to represent uncertainty in service time and probabilistic customers. With slight adaptations, their tabu search (TS) approach can provide solutions that adhere to the precedence principle of Groër, Golden, and Wasil (2009) and, thus, solve the ConVRP. Tarantilis, Stavropoulou, and Repoussis (2012) use a TS to improve the template routes and the resulting daily routes in a sequential manner. Kovacs, Parragh, and Hartl (2014) present an adaptive large neighborhood search (ALNS) that is solely applied to the template routes; the daily routes are improved using a truncated two-opt operator. In addition, the paper proposes a relaxed version of the problem called ConVRP with shiftable starting times, in which it is possible to delay the departure at the depot to better meet the ATC requirements. Finally, Kovacs et al. (2015) introduce the generalized ConVRP, which (1) allows each customer to be served by a set of drivers (instead of a single one), (2) features shiftable starting times and AM/PM time windows, and (3) does not integrate the maximum time differences between arrivals on different days as hard constraints but penalizes them in the objective function. The proposed large neighborhood search (LNS) works on entire solutions instead of a template and currently represents the state-of-the-art heuristic also for the standard ConVRP. Multiobjective variants of the ConVRP are investigated by Kovacs, Parragh, and Hartl (2015) and Lian, Milburn, and Rardin (2016) and are addressed by means of multidirectional LNS and local search, respectively. Feillet et al. (2014) present an ALNS to tackle a VRP with ATC in the context of transporting people with disabilities. DC might also be imposed by contractual arrangements between carriers and customers. In this context, Dayarian et al. (2015, 2016) study a multiperiod VRP arising in the dairy industry in which a routing plan with recourse action must be repeated in every period once negotiated.

To the best of our knowledge, no exact approach to the ConVRP has been proposed yet. The only two papers addressing consistency considerations in an exact fashion are owed to Subramanyam and Gounaris (2016, 2017), who study the consistent traveling-salesman problem (TSP); that is, only one route per day is planned and routes must adhere to the ATC requirements. Subramanyam and Gounaris (2016) present three mixed-integer, linear-programming (MILP) formulations and several classes of valid inequalities that are embedded in a branch-and-cut framework; they are able to solve all instances with up to five planning periods and 25 customers to guaranteed optimality and also some instances with up to 50 customers. Subramanyam and Gounaris (2017) decompose the problem into a sequence of single-period TSPs with time windows and solve the consistent TSP via branch and bound; they are able to solve instances with up to five planning periods and 100 customers, outperforming the results of Subramanyam and Gounaris (2016), but a few instances with 33 customers remain open.

The major contribution of this paper is twofold:

- We propose the first exact method for the ConVRP, which is able to solve small and medium-sized instances with up to five planning periods and 30 customers. Most of the state-of-the-art exact methods to solve VRPs are based on column generation (CG) applied to formulations in which each variable represents a feasible route, and the pricing problem is solved via dynamic programming (DP). However, these methods cannot be directly extended to solve the ConVRP because the linear relaxation of route-based formulations provides weak lower bounds because of the interdependency between the daily routes, which is caused by the required ATC at customers. Therefore, we propose an exact method based on CG applied to a formulation in which each variable represents the set of routes assigned to a vehicle over the planning horizon. The exact method initially takes into account DC only and addresses ATC at a later stage. We also show how our exact approach can be modified to handle variants of the ConVRP in which either (1) only the departure time at the depot is flexible or (2) the departure time at the depot is flexible and waiting at customers is allowed, and we investigate the benefits of adding such flexibility.
- As an upper bounding procedure, we develop an LNS featuring suitable penalty mechanisms to deal with infeasible solutions and a repair procedure specifically designed to improve the ATC of solutions. Used as a stand-alone heuristic, the LNS is able to significantly improve the solution quality on benchmark instances from the literature compared with state-of-the-art heuristics. This is especially true for instances that assume small service frequencies, that is, the probability that a customer requires service on a given day is relatively small, and that are, therefore, more difficult for template-based methods.

A minor contribution of this paper is to provide a new compact formulation for the ConVRP that improves upon the compact formulation presented by Groër, Golden, and Wasil (2009). The new compact formulation contains fewer variables and constraints and is able to provide optimal solutions of small-sized instances in significantly shorter computing times.

The organization of this paper is as follows. In Section 2, we formally define the ConVRP and the notation used throughout the paper. In Section 3, we introduce the new compact formulation. Section 4 describes the proposed exact method and Section 5

the LNS. Section 6 is devoted to the computational results. Conclusions and future research directions are given in Section 7.

2. Problem Definition

In the ConVRP, a set of customers $\mathcal{N}$ require delivery of a single commodity over a set $\mathcal{D}$ of days. The demand of customer $i \in \mathcal{N}$ on day $d \in \mathcal{D}$ is denoted by q_{id} (we assume that $q_{id} = 0$ if customer i does not require service on day d); $\mathcal{D}_i \subseteq \mathcal{D}$ indicates the subset of days on which customer i must be served (i.e., $\mathcal{D}_i = \{d \in \mathcal{D} \mid q_{id} > 0\}$), and $\mathcal{N}_d \subseteq \mathcal{N}$ is the subset of customers that must be served on day $d \in \mathcal{D}$ (i.e., $\mathcal{N}_d = \{i \in \mathcal{N} \mid q_{id} > 0\}$).

A homogeneous fleet $\mathcal{K}$ of capacitated vehicles based at a single depot, denoted by zero, is available to satisfy all customer requests. The capacity of the vehicles is given by Q. We indicate with $\mathcal{V}$ the set of customers plus the depot (i.e., $\mathcal{V} = \mathcal{N} \cup \{0\}$). The travel time t_{ij} between two locations $i, j \in \mathcal{V}$ is assumed to be deterministic and symmetric (i.e., $t_{ij} = t_{ji}$). The service time at customer $i \in \mathcal{N}$ on day $d \in \mathcal{D}$ is denoted by s_{id}. The maximum route duration of a vehicle on each day is T time units.

To respect DC, each customer must be served by the same driver/vehicle on every day of the planning horizon on which service is required. ATC is expressed by requiring that the service must take place roughly at the same time, so the difference between the latest and the earliest arrival time at each customer over the planning horizon cannot exceed the maximum allowed time difference L. As defined in Groër, Golden, and Wasil (2009), we assume that vehicles are not allowed to wait at a customer or the depot to meet ATC. The objective of the ConVRP is to find a set of routes for the vehicle fleet that minimizes the total vehicle operating time z, defined as the total travel and service time over the planning horizon.

3. A New Compact Formulation for the ConVRP

To the best of our knowledge, the only formulation proposed in the literature for the ConVRP is owed to Groër, Golden, and Wasil (2009), who introduced a compact formulation for the problem. In the following, we present a new compact formulation that uses fewer variables than the one of Groër, Golden, and Wasil (2009), namely at most $|\mathcal{V}^2| \cdot |\mathcal{K}| \cdot |\mathcal{D}| + |\mathcal{V}| \cdot |\mathcal{K}| + |\mathcal{V}| \cdot |\mathcal{D}|$ instead of at most $|\mathcal{V}^2| \cdot |\mathcal{K}| \cdot |\mathcal{D}| + |\mathcal{V}| \cdot |\mathcal{K}| \cdot |\mathcal{D}| + |\mathcal{V}| \cdot |\mathcal{D}|$. The computational efficiency of the two formulations is compared in Section 6.2, in which we show that the new formulation is always better in terms of both the quality of the lower bound provided by its linear relaxation and the computing time to find the optimal solutions of small-sized instances.

We represent the ConVRP on a directed multigraph $\mathcal{G} = (\mathcal{V}, \mathcal{A})$. The arc set $\mathcal{A}$ is defined as $\mathcal{A} = \cup_{d \in \mathcal{D}} \mathcal{A}_d$, where $\mathcal{A}_d = \{(0, j) \mid j \in \mathcal{N}_d\} \cup \{(i, 0) \mid i \in \mathcal{N}_d\} \cup \{(i, j) \mid i, j \in \mathcal{N}_d : i \neq j\}$. Let $\hat{t}_{ijd}$ be the modified travel time associated with arc $(i, j) \in \mathcal{A}_d$, $d \in \mathcal{D}$, defined as $\hat{t}_{ijd} = t_{ij}$ if $i = 0$ and $\hat{t}_{ijd} = t_{ij} + s_{id}$ otherwise. We define the following three sets of variables:

- $x_{ijkd} \in \{0, 1\}$: Binary variable equal to one if arc $(i, j) \in \mathcal{A}_d$ is traversed by vehicle $k \in \mathcal{K}$ on day $d \in \mathcal{D}$ (zero otherwise),
- $y_{ik} \in \{0, 1\}$: Binary variable equal to one if customer $i \in \mathcal{N}$ is served by vehicle $k \in \mathcal{K}$ (zero otherwise),
- $b_{id} \in \mathbb{R}_+$: Continuous variable indicating the arrival time at customer $i \in \mathcal{N}$ on day $d \in \mathcal{D}_i$.

As such, the ConVRP can be formulated as follows:

$$z = \min \sum_{d \in \mathcal{D}} \sum_{k \in \mathcal{K}} \sum_{(i,j) \in \mathcal{A}_d} \hat{t}_{ijd} x_{ijkd} \tag{1}$$

$$\text{s.t.} \sum_{(0,j) \in \mathcal{A}_d} x_{0jkd} \leq 1 \qquad k \in \mathcal{K} \quad d \in \mathcal{D} \tag{2}$$

$$\sum_{k \in \mathcal{K}} y_{ik} = 1 \qquad i \in \mathcal{N} \tag{3}$$

$$\sum_{i \in \mathcal{N}_d} q_{id} y_{ik} \leq Q \qquad k \in \mathcal{K} \quad d \in \mathcal{D} \tag{4}$$

$$\sum_{(i,j) \in \mathcal{A}_d} x_{ijkd} = \sum_{(j,i) \in \mathcal{A}_d} x_{jikd} \qquad j \in \mathcal{N} \quad k \in \mathcal{K} \quad d \in \mathcal{D}_j \tag{5}$$

$$\sum_{(i,j) \in \mathcal{A}_d} x_{ijkd} = y_{jk} \qquad j \in \mathcal{N} \quad k \in \mathcal{K} \quad d \in \mathcal{D}_j \tag{6}$$

$$b_{id} \leq T - (T - \hat{t}_{0id}) \sum_{k \in \mathcal{K}} x_{0ikd} \qquad d \in \mathcal{D} \quad i \in \mathcal{N}_d \tag{7}$$

$$b_{id} + (\hat{t}_{ijd} + T) \sum_{k \in \mathcal{K}} x_{ijkd} + (T - \hat{t}_{jid}) \sum_{k \in \mathcal{K}} x_{jikd} \leq b_{jd} + T \qquad d \in \mathcal{D} \quad i, j \in \mathcal{N}_d : i \neq j \tag{8}$$

$$b_{id} - b_{id'} \leq L \qquad i \in \mathcal{N} \quad d, d' \in \mathcal{D}_i : d \neq d' \tag{9}$$

$$x_{ijkd} \in \{0, 1\} \qquad k \in \mathcal{K} \quad d \in \mathcal{D} \quad (i, j) \in \mathcal{A}_d \tag{10}$$

$$y_{ik} \in \{0, 1\} \qquad i \in \mathcal{N} \quad k \in \mathcal{K} \tag{11}$$

$$\hat{t}_{0id} \leq b_{id} \leq T - \hat{t}_{i0d} \qquad i \in \mathcal{N} \quad d \in \mathcal{D}_i. \tag{12}$$

The objective function aims at minimizing the total operating time. Constraints (2) guarantee that each vehicle performs at most one route on each day. Constraints (3) assign each customer to exactly one vehicle. Constraints (4) guarantee that the capacity of the vehicles is not exceeded. Constraints (5) are flow-conservation constraints. Constraints (6) link $\boldsymbol{x}$ and $\boldsymbol{y}$ variables to ensure DC. Constraints (7) ensure that the arrival time at a customer i is not greater than the travel time from the depot to i if arc $(0, i)$ is traversed. Constraints (8) link $\boldsymbol{x}$ and $\boldsymbol{b}$ variables to set the arrival times based on the traversed arcs and prevent subtours. Constraints (9) model ATC requirements. Constraints (10)–(12) define the range of the decision variables and ensure that maximum route duration is respected.

Because formulation (1)–(12) is symmetric in the vehicles (the fleet is homogeneous), any permutation of

a feasible solution with respect to index $1 \le k \le |\mathcal{K}|$ is also a feasible solution. To break some symmetries, we add the following inequalities to (1)–(12), which order the vehicles by nonincreasing quantity delivered over the planning horizon:

$$\sum_{d\in\mathcal{D}}\sum_{i\in\mathcal{N}_d} q_{id}y_{ik} \le \sum_{d\in\mathcal{D}}\sum_{i\in\mathcal{N}_d} q_{id}y_{i,k-1} \qquad k\in\mathcal{K}\setminus\{1\}. \quad (13)$$

Adding subtour elimination constraints (SECs) to formulation (1)–(12) allows significantly decreasing the computing time to find an optimal solution of small ConVRP instances. Therefore, in the computational experiments in Section 6.2, the following set of generalized SECs (GSECs) are added to our formulation:

$$\sum_{(i,j)\in\mathcal{A}_d:\, i\in\mathcal{V}\setminus S,\, j\in S} Qx_{ijkd} \ge \sum_{i\in S} q_{id}y_{id} \qquad k\in\mathcal{K} \quad d\in\mathcal{D} \quad S\subset\mathcal{N}_d:|S|\ge 2. \quad (14)$$

Constraints (14) are clearly redundant when integrality constraints (10) and (11) are present, but they strengthen the linear relaxation of formulation (1)–(12).

4. An Exact Method for the ConVRP

In this section, we introduce a formulation of the ConVRP with exponentially many variables. Two lower bounds based on this formulation are presented in Section 4.1. An outline of the proposed exact method is provided in Section 4.2, and the different steps of the algorithm are detailed in Sections 4.3–4.6.

Let Ω be the set of all possible subsets of customers (hereafter called *clusters*) that can be served by a single vehicle over the planning horizon without violating capacity, route duration, DC, and ATC constraints. Moreover, let g_C be the minimum cost to serve cluster $C\in\Omega$ with a single vehicle; that is, g_C is the sum of the costs of the least-cost routes satisfying the listed constraints performed by the vehicle on each day $d\in\mathcal{D}$ to serve the customers $C\cap\mathcal{N}_d$. By introducing a binary decision variable ξ_C that is equal to one if cluster $C\in\Omega$ is assigned to a vehicle (zero otherwise), the ConVRP can be formulated as the following set-partitioning (SP) problem:

$$z(\text{SP}) = \min \sum_{C\in\Omega} g_C\xi_C \quad (15)$$

$$\sum_{C\in\Omega:\, i\in C} \xi_C = 1 \qquad i\in\mathcal{N} \quad (16)$$

$$\sum_{C\in\Omega} \xi_C \le |\mathcal{K}| \quad (17)$$

$$\xi_C\in\{0,1\} \qquad C\in\Omega. \quad (18)$$

The objective function (15) asks for minimizing the cost of the selected clusters. Constraints (16) ensure that each customer belongs to exactly one selected cluster. Constraint (17) ensures that at most $|\mathcal{K}|$ clusters are selected. Constraints (18) define variables ξ as binary. Each feasible solution of problems (15)–(18) is a subset of clusters of Ω. The ConVRP solution corresponding to that subset of clusters consists of the least-cost routes associated with costs g_C.

Note that SP contains exponentially many variables, so a CG approach must be applied to find an optimal ConVRP solution when solving problem SP. For many variants of the VRP, the state-of-the-art exact methods are based on CG (see, e.g., Jepsen et al. 2008; Baldacci, Mingozzi, and Roberti 2011; Dabia et al. 2013; Contardo and Martinelli 2014; and Pecin et al. 2017a, b). In all of these methods, the pricing problem is solved via DP. Unfortunately, DP cannot directly be applied to price out clusters in a CG approach based on formulation SP because of the number and the range of the state variables needed and the weakness of the dominance rules that can be applied.

However, it is possible to derive some tight lower bounds from formulation SP (see Section 4.1), which are used by our exact method to find an optimal ConVRP solution. In the remainder of the paper, we refer to our exact method as cluster column generation (CCG).

4.1. Lower Bounds Based on Formulation SP

Recall that, for a given cluster $C\in\Omega$, the minimum cost g_C to serve all customers in C with a single vehicle over the planning horizon is given by the sum of the costs of the routes that on each day $d\in\mathcal{D}$ serve all customers $\mathcal{N}_d\cap C$ without violating capacity, route duration, and ATC constraints. Because of the ATC requirement, the routes performed by a vehicle on each day are not necessarily the least-cost routes (i.e., the TSP) to serve all customers in the cluster.

Let $\hat{\Omega}\supseteq\Omega$ be the set of all possible clusters that can be served by a single vehicle over the planning horizon without violating capacity and route-duration constraints. Let $\hat{g}_C$ be the cost to serve all customers of the cluster $C\in\hat{\Omega}$ with a single vehicle so that capacity and route-duration constraints are respected but ATC constraints can be violated. It is easy to observe that cost $\hat{g}_C$ of cluster $C\in\hat{\Omega}$ is given by the sum of the cost of the TSPs to serve customers $C\cap\mathcal{N}_d$ on each day $d\in\mathcal{D}$.

A valid lower bound to the ConVRP is, therefore, given by the optimal value, $z(\text{LP}_0)$, of the following linear problem, hereafter called LP_0:

$$z(\text{LP}_0) = \min \sum_{C\in\hat{\Omega}} \hat{g}_C\xi_C \quad (19)$$

$$\text{s.t.} \sum_{C\in\hat{\Omega}:\, i\in C} \xi_C = 1 \qquad i\in\mathcal{N} \quad (20)$$

$$\sum_{C\in\hat{\Omega}} \xi_C \le |\mathcal{K}| \quad (21)$$

$$\xi_C \ge 0 \qquad C\in\hat{\Omega}. \quad (22)$$

The lower bound $z(\mathrm{LP}_0)$ can be improved by adding the following valid inequalities:

- Minimum number of vehicles:

$$\sum_{C\in\hat{\Omega}} \xi_C \geq \mathcal{H}_{\min}, \tag{23}$$

where $\mathcal{H}_{\min}$ is a lower bound on the minimum number of vehicles used in any optimal solution of the ConVRP.

- Subset-row (SR) inequalities that state, for each triplet of customers $\{i,j,h\}\in\mathcal{N}$, no more than one of the clusters serving at least two of the three customers $\{i,j,h\}$ can be selected:

$$\sum_{C\in\hat{\Omega}:|C\cap\{i,j,h\}|\geq 2} \xi_C \leq 1 \qquad \{i,j,h\}\in\mathcal{N}: i\neq j\neq h. \tag{24}$$

Inequalities (24) are a special case of the well-known SR inequalities introduced by Jepsen et al. (2008) and can be separated by complete enumeration.

In the following, we denote by $z(\mathrm{LP}_1)$ the optimal value of problem LP_0 plus inequalities (23) and by $z(\mathrm{LP}_2)$ the optimal value of problem LP_0 plus inequalities (23) and (24). Moreover, let $\alpha_i\in\mathbb{R}$ be the dual variable associated with constraint (20) of customer $i\in\mathcal{N}$, $\alpha_0\in\mathbb{R}_-$ the dual variable associated with constraint (21), $\beta\in\mathbb{R}_+$ the dual variable associated with constraint (23), and $\gamma_{ijh}\in\mathbb{R}_-$ the dual variable associated with constraint (24) of the triplet of customers $\{i,j,h\}\in\mathcal{N}$.

4.2. Overview of CCG

CCG consists of four main steps that can be outlined as follows:

Step 1. Initialization: An upper bound UB to the ConVRP and a lower bound $\mathcal{H}_{\min}$ to the number of vehicles in any optimal ConVRP solution are computed. The upper bound UB is computed by running 10 times the LNS described in Section 5.3, each time with a limit of 25,000 iterations (i.e., $\eta_{total}=25{,}000$). The lower bound $\mathcal{H}_{\min}$ on the minimum number of vehicles is computed by using an MILP (see Section 4.3).

Step 2. Generate the set of clusters $\hat{\Omega}$: The goal of this step is to generate the whole set of clusters $\hat{\Omega}$ (i.e., clusters that can be served by a single vehicle over the planning horizon without violating capacity and route-duration constraints). As described in Section 4.4, this can be done via DP. If it is not possible to generate the whole set $\hat{\Omega}$, then CCG stops without providing a proven optimal solution to the ConVRP. We denote the number of clusters after this step by $|\hat{\Omega}'|$.

Step 3. Remove nonoptimal clusters from $\hat{\Omega}$: This step aims at removing clusters that cannot belong to an optimal ConVRP solution from the set of clusters $\hat{\Omega}$ by iteratively computing optimal dual solutions of problems LP_1 and LP_2.

First, we use CG as described in Section 4.5 to compute an optimal LP_1 dual solution $(\boldsymbol{\alpha}^*,\boldsymbol{\beta}^*)$ of cost $z(\mathrm{LP}_1)$. Any cluster having reduced cost with respect to (w.r.t.) $(\boldsymbol{\alpha}^*,\boldsymbol{\beta}^*)$ greater than the corresponding gap (i.e., $\mathrm{UB}-z(\mathrm{LP}_1)$) is then removed from the set $\hat{\Omega}$ because it cannot belong to an optimal ConVRP solution. Second, using CG as described in Section 4.5, an optimal LP_2 dual solution $(\boldsymbol{\alpha}^*,\boldsymbol{\beta}^*,\boldsymbol{\gamma}^*)$ of cost $z(\mathrm{LP}_2)$ is computed. Any cluster having reduced cost w.r.t. $(\boldsymbol{\alpha}^*,\boldsymbol{\beta}^*,\boldsymbol{\gamma}^*)$ greater than the corresponding gap (i.e., $\mathrm{UB}-z(\mathrm{LP}_2)$) is then removed from the set $\hat{\Omega}$.

Step 3 is iterated as long as the set of clusters $\hat{\Omega}$ is reduced by using the optimal dual solutions $(\boldsymbol{\alpha}^*,\boldsymbol{\beta}^*)$ and $(\boldsymbol{\alpha}^*,\boldsymbol{\beta}^*,\boldsymbol{\gamma}^*)$. By iterating, we can generate alternative optimal dual solutions $(\boldsymbol{\alpha}^*,\boldsymbol{\beta}^*)$ and $(\boldsymbol{\alpha}^*,\boldsymbol{\beta}^*,\boldsymbol{\gamma}^*)$, that is, obtain different reduced cost w.r.t. to each cluster and, thus, potentially remove additional clusters. We denote the number of clusters left after this step as $|\hat{\Omega}''|$.

Step 4. Find an optimal ConVRP solution: Let $\overline{\Omega}\subseteq\Omega$ be a subset of clusters C for which cost g_C is known, such that $\overline{\Omega}\cap\hat{\Omega}=\emptyset$. The optimal value of the following problem $\overline{\mathrm{SP}}$ provides a valid lower bound to the ConVRP:

$$z(\overline{\mathrm{SP}}) = \min \sum_{C\in\overline{\Omega}} g_C\xi_C + \sum_{C\in\hat{\Omega}} \hat{g}_C\xi_C \tag{25}$$

$$\sum_{C\in\overline{\Omega}\cup\hat{\Omega}:\, i\in C} \xi_C = 1 \qquad i\in\mathcal{N} \tag{26}$$

$$\sum_{C\in\overline{\Omega}\cup\hat{\Omega}} \xi_C \leq |\mathcal{H}| \tag{27}$$

$$\xi_C\in\{0,1\} \qquad C\in\overline{\Omega}\cup\hat{\Omega}. \tag{28}$$

The objective function (25) aims at minimizing the total cost of the clusters selected from the two sets $\overline{\Omega}$ and $\hat{\Omega}$. Constraints (26) ensure that each customer belongs to exactly one of the selected clusters. Constraint (27) guarantees that at most $|\mathcal{H}|$ clusters are selected. Constraints (28) are integrality constraints.

Let $\Omega^*\subseteq\overline{\Omega}\cup\hat{\Omega}$ be the set of clusters in the optimal solution of $\overline{\mathrm{SP}}$. We can observe that whenever $\Omega^*\subseteq\overline{\Omega}$, then the clusters of the set Ω^* represent an optimal ConVRP solution because they take into account ATC and, by definition, DC.

To find an optimal ConVRP solution, the last step of CCG consists of iteratively solving $\overline{\mathrm{SP}}$ with a general purpose MILP solver until an optimal ConVRP is found while changing the sets of clusters $\overline{\Omega}$ and $\hat{\Omega}$. At each iteration, $\overline{\mathrm{SP}}$ is solved, the cost g_{C^*} of one of the clusters $C^*\in\Omega^*\cap\hat{\Omega}$ is computed (see Section 4.6), and cluster C^* is removed from $\hat{\Omega}$. If $z(\overline{\mathrm{SP}})+g_{C^*}-\hat{g}_{C^*}\leq\mathrm{UB}$, then cluster C^* is also added to $\overline{\Omega}$ because it can be part of an optimal ConVRP solution of cost between $z(\overline{\mathrm{SP}})$ and UB. At the first iteration, the set $\hat{\Omega}$ is inherited from Step 3, and the set $\overline{\Omega}$ is empty. Note that it may not be possible to serve a cluster C^* with a single vehicle while adhering to ATC; if so, cluster C^* is obviously not added to $\overline{\Omega}$.

Because the complexity of computing cost g_{C^*} for a given cluster C^* also depends on the number of customers in the cluster C^*, at each iteration the selected cluster $C^* \in \Omega^* \cap \hat{\Omega}$ is the one with the smallest number of customers.

4.3. Computing $\mathcal{H}_{\min}$ in Step 1

The lower bound $\mathcal{H}_{\min}$ on the minimum number of vehicles in any ConVRP solution is computed by solving the following MILP. Let $\varphi_{ik} \in \{0,1\}$ be a binary variable equal to one if customer $i \in \mathcal{N}$ is assigned to vehicle $k \in \mathcal{H}$ (zero otherwise), and let $\vartheta_k \in \{0,1\}$ be a binary variable equal to one if vehicle $k \in \mathcal{H}$ is used (zero otherwise). Then, $\mathcal{H}_{\min}$ can be computed as

$$\mathcal{H}_{\min} = \min \sum_{k \in \mathcal{H}} \vartheta_k \tag{29}$$

$$\text{s.t.} \sum_{k \in \mathcal{H}} \varphi_{ik} = 1 \qquad i \in \mathcal{N} \tag{30}$$

$$\sum_{i \in \mathcal{N}_d} q_{id} \varphi_{ik} \le Q \vartheta_k \qquad d \in \mathcal{D} \quad k \in \mathcal{H} \tag{31}$$

$$\varphi_{ik} \in \{0,1\} \qquad i \in \mathcal{N} \quad k \in \mathcal{H} \tag{32}$$

$$\vartheta_k \in \{0,1\} \qquad k \in \mathcal{H}. \tag{33}$$

The objective function (29) aims at minimizing the number of vehicles used. Constraints (30) ensure that each customer $i \in \mathcal{N}$ is assigned to exactly one vehicle. Constraints (31) guarantee that the capacity of each vehicle $k \in \mathcal{H}$ is respected on each day $d \in \mathcal{D}$. The range of the decision variables is defined by constraints (32) and (33).

For small and medium-sized ConVRP instances, problems (29)–(33) can be solved to optimality with a general purpose MILP solver in less than a second of computing time.

4.4. Generating the Set $\hat{\Omega}$ in Step 2

To generate the set $\hat{\Omega}$, we use a simple DP recursion that enumerates all feasible routes Φ_d for each day $d \in \mathcal{D}$ of the planning horizon, and then clusters are generated by combining the routes of the sets Φ_d. A route is feasible if the vehicle capacity Q is not exceeded and its duration does not exceed the maximum route duration T.

Let $f_d(S,i)$ be the cost of the min-cost path starting from the depot, visiting all customers of the set $S \subseteq \mathcal{N}_d$, and ending at customer $i \in S$ on day $d \in \mathcal{D}$. Functions $f_d(S,i)$ for each day $d \in \mathcal{D}$ can be computed via DP as follows. We initialize $f_d(\{i\},i) = \hat{t}_{0id}$ for each $i \in \mathcal{N}_d$. The recursion for computing functions $f_d(S,i)$ for each subset of customers $S \subseteq \mathcal{N}_d$ and each customer $i \in S$ is

$$f_d(S,i) = \min_{j \in S \setminus \{i\}} \{f_d(S \setminus \{i\}, j) + \hat{t}_{jid}\}.$$

Because routes have to respect the vehicle capacity Q and the maximum route duration T, there is no need to propagate functions $f_d(S,i)$ such that either $f_d(S,i) + \hat{t}_{i0d} > T$ or $\sum_{j \in S} q_{jd} > Q$. For the sake of simplicity, we assume, in the remainder of the section, that $f_d(S,i) = \infty$ if function $f_d(S,i)$ is not computed because of constraint violations.

The cost of the least-cost route to serve the subset of customers $S \subseteq \mathcal{N}_d$ on day $d \in \mathcal{D}$ is given by $\min_{i \in S}\{f_d(S,i) + \hat{t}_{i0d}\}$. From functions $f_d(S,i)$, it is possible to generate the set $\hat{\Omega}$. In particular, cluster $C \subseteq \mathcal{N}$ belongs to the set $\hat{\Omega}$ if $\min_{i \in C \cap \mathcal{N}_d}\{f_d(C \cap \mathcal{N}_d, i) + \hat{t}_{i0d}\} \le T$ for each day $d \in \mathcal{D}$; otherwise, cluster C does not belong to the set $\hat{\Omega}$.

The cost $\hat{g}_C$ of cluster $C \in \hat{\Omega}$ is given by

$$\hat{g}_C = \sum_{d \in \mathcal{D}} \left(\min_{i \in C \cap \mathcal{N}_d} \left\{ f_d(C \cap \mathcal{N}_d, i) + \hat{t}_{i0d} \right\} \right).$$

We can observe that $\hat{g}_C$ is the cost to serve all customers of the set C with a single vehicle over the planning horizon without necessarily satisfying the ATC constraints.

4.5. Computing Lower Bounds $z(\text{LP}_1)$ and $z(\text{LP}_2)$ in Step 3

In principle, because all variables are generated a priori, we could simply solve LP_1 as it is. However, it is computationally convenient to apply a simple CG algorithm that solves LP_1 by starting from a small set of clusters (we use a dummy cluster that contains all customers and has cost equal to UB) and then iteratively adding the 100 clusters with the highest negative reduced cost at a time until all clusters of the set $\hat{\Omega}$ have nonnegative reduced cost w.r.t. the dual solution $(\boldsymbol{\alpha}, \boldsymbol{\beta})$ of problem LP_1. The reduced cost $\hat{g}_C(\boldsymbol{\alpha}, \boldsymbol{\beta})$ of cluster $C \in \hat{\Omega}$ is computed as $\hat{g}_C(\boldsymbol{\alpha}, \boldsymbol{\beta}) = \hat{g}_C - \alpha_0 - \sum_{i \in C} \alpha_i - \beta$.

Once an optimal dual solution $(\boldsymbol{\alpha}^*, \boldsymbol{\beta}^*)$ of cost $z(\text{LP}_1)$ is found, all clusters $C \in \hat{\Omega}$ having reduced cost $\hat{g}_C(\boldsymbol{\alpha}^*, \boldsymbol{\beta}^*)$ greater than the gap left (i.e., $\text{UB} - z(\text{LP}_1)$) can be removed from the set $\hat{\Omega}$ because they cannot belong to an optimal ConVRP solution.

A similar CG procedure is applied to compute $z(\text{LP}_2)$. At the beginning, the master problem contains no SR inequalities (24) and just a dummy cluster; then, at each iteration, the 100 clusters having the highest negative reduced cost w.r.t. the dual solution $(\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma})$ of LP_2 are added along with the most violated SR inequality (24).

Once an optimal dual solution $(\boldsymbol{\alpha}^*, \boldsymbol{\beta}^*, \boldsymbol{\gamma}^*)$ of cost $z(\text{LP}_2)$ is found, all clusters $C \in \hat{\Omega}$ having reduced cost $\hat{g}_C(\boldsymbol{\alpha}^*, \boldsymbol{\beta}^*, \boldsymbol{\gamma}^*)$ greater than the gap left (i.e., $\text{UB} - z(\text{LP}_2)$) can be removed from the set $\hat{\Omega}$ because they cannot belong to an optimal ConVRP solution. The reduced cost $\hat{g}_C(\boldsymbol{\alpha}^*, \boldsymbol{\beta}^*, \boldsymbol{\gamma}^*)$ of cluster $C \in \hat{\Omega}$ is computed as

$$\hat{g}_C(\boldsymbol{\alpha}^*, \boldsymbol{\beta}^*, \boldsymbol{\gamma}^*) = \hat{g}_C - \alpha_0^* - \sum_{i \in C} \alpha_i^* - \beta^* - \sum_{\substack{\{i,j,h\} \in \mathcal{N}: \\ |\{i,j,h\} \cap C| \ge 2}} \gamma_{ijh}^*.$$

4.6. Computing Cost g_C in Step 4

The problem of computing cost g_C for a given cluster $C \in \Omega^* \cap \hat{\Omega}$ can be represented on a directed multigraph

$\mathcal{G}(C) = (\mathcal{V}(C), \mathcal{A}(C))$. The vertex set is defined as $\mathcal{V}(C) = C \cup \{0\}$, and the arc set $\mathcal{A}(C)$ is defined as $\mathcal{A}(C) = \cup_{d\in\mathcal{D}} \mathcal{A}_d(C)$, where $\mathcal{A}_d(C) = \{(0,j)|j\in\mathcal{N}_d\cap C\}\cup\{(i,0)|i\in\mathcal{N}_d\cap C\} \cup\{(i,j)|i,j\in\mathcal{N}_d\cap C : i\neq j\}$. Let us define the following two sets of variables:

- $x_{ijd}\in\{0,1\}$: Binary variable equal to one if arc $(i,j)\in\mathcal{A}_d(C)$ is used on day $d\in\mathcal{D}$ (zero otherwise),
- $b_{id}\in\mathbb{R}_+$: Continuous variable indicating the arrival time at vertex $i\in C$ on day $d\in\mathcal{D}_i$.

Then, the cost g_C of cluster C corresponds to the optimal value of the following MILP:

$$g_C = \min \sum_{d\in\mathcal{D}} \sum_{(i,j)\in\mathcal{A}_d(C)} \hat{t}_{ijd} x_{ijd} \tag{34}$$

$$\text{s.t.} \sum_{(0,j)\in\mathcal{A}_d(C)} x_{0jd} \leq 1 \qquad d\in\mathcal{D} \tag{35}$$

$$\sum_{(i,j)\in\mathcal{A}_d(C)} x_{ijd} = 1 \qquad j\in C \quad d\in\mathcal{D}_j \tag{36}$$

$$\sum_{(j,i)\in\mathcal{A}_d(C)} x_{jid} = 1 \qquad j\in C \quad d\in\mathcal{D}_j \tag{37}$$

$$b_{id} \leq T - (T - \hat{t}_{0id})x_{0id} \qquad d\in\mathcal{D} \quad i\in\mathcal{N}_d\cap C \tag{38}$$

$$b_{id} + (\hat{t}_{ijd} + T)x_{ijd} + (T - \hat{t}_{ijd})x_{jid} \leq b_{jd} + T \qquad d\in\mathcal{D} \quad i,j\in\mathcal{N}_d\cap C : i\neq j \tag{39}$$

$$b_{id} - b_{id'} \leq L \quad i\in C \quad d,d'\in\mathcal{D}_i : d\neq d' \tag{40}$$

$$x_{ijd}\in\{0,1\} \qquad d\in\mathcal{D} \quad (i,j)\in\mathcal{A}_d(C) \tag{41}$$

$$\hat{t}_{0id} \leq b_{id} \leq T - \hat{t}_{i0d} \qquad i\in C \quad d\in\mathcal{D}_i. \tag{42}$$

The objective function (34) aims at minimizing the total operating time to visit all customers of the set C. Constraints (35) ensure that the vehicle performs at most one route on each day of the planning horizon. Constraints (36) and (37) are in-degree and out-degree constraints, respectively. Constraints (38) along with constraints (42) properly set the arrival time at the first customer of each route of each day. Constraints (39) link variables $\boldsymbol{x}$ and $\boldsymbol{z}$ to update the arrival times at the customers, depending on the traversed arcs, and prevent subtours. Constraints (40) guarantee the ATC of the routes. Constraints (41) and (42) define the range of the decision variables.

We solve problems (34)–(42) by using a general purpose MILP solver. We also add, in a cutting-plane fashion, the well-known SECs defined as

$$\sum_{\substack{(i,j)\in\mathcal{A}_d(C):\\ i\in C, j\in\mathcal{V}(C)\setminus C}} x_{ijd} \geq 1 \qquad d\in\mathcal{D} \quad S\subseteq\mathcal{N}_d\cap C : |S|\geq 2. \tag{43}$$

In the instances studied in this paper, the number of customers in the clusters is usually limited to 10–15 customers. Therefore, it is possible to enumerate all SECs (43) a priori and let the MILP solver add them in a cutting-plane fashion. Alternatively, one could also separate them in polynomial time and only add the violated ones on the fly.

As mentioned in Section 4.2, problems (34)–(42) do not necessarily have feasible solutions.

5. Large Neighborhood Search for the ConVRP

We propose an LNS for the ConVRP that is used to obtain upper bounds within our exact method but that can also be used as a stand-alone metaheuristic approach. The LNS is enhanced by several components: (1) suitable penalty mechanisms to deal with infeasible solutions, (2) a repair procedure that is applied to improve the ATC, and (3) regularly solving a set-partitioning problem using the clusters previously found by the search to improve the solution quality.

In the following description, we represent a solution $\mathcal{S}$ as a set of routes $\{r_{kd}\,|\,k\in\mathcal{K}, d\in\mathcal{D}\}$. A route $r_{kd} = <v_0 = 0, v_1, \ldots, v_{n_{kd}}, v_{n_{kd}+1} = 0>$ is given as a sequence of vertices that starts and ends at the depot vertex zero and visits a set $\mathcal{N}(r_{kd})$ of n_{kd} customer vertices in between.

In the description of the algorithm, we directly report the utilized values of the algorithm parameters. The latter were determined in experimental fashion during the development of our algorithm, and no systematic fine-tuning was carried out. We found that our algorithm is quite stable with regards to changes in the parameter values as long as the new values stay within the magnitude of the values of the current setting.

An overview of the algorithm, which we call LNS with arrival-time consistency improvement (LNS-ATCI), is given in Figure 1. First, LNS-ATCI generates a feasible initial solution $\mathcal{S}_c$ with a savings algorithm that respects the consistency requirements of the ConVRP (see Section 5.1). Then, the initial solution is improved in 25,000 iterations of LNS, including our specialized component for improving the ATC (Section 5.3). Here, infeasible solutions are allowed and are evaluated with a generalized objective function (Section 5.2). The set-partitioning problem for feasible clusters is described in Section 5.4. Finally, every 250 iterations without improvement of $\mathcal{S}_{best}$, we reset $\mathcal{S}_c$ to $\mathcal{S}_{best}$.

5.1. Modified Savings Algorithm

To generate an initial solution, we adapt the savings algorithm of Clarke and Wright (1964) to handle the multiday horizon and the DC and ATC constraints of the ConVRP. Before the merge step of the savings algorithm is applied, the routes to be merged are assigned to two different vehicles. Consequently, for the ConVRP, merging two routes on a single day entails that the routes of the two respective vehicles are also merged on all other days.

Figure 1. Overview of the LNS-ATCI Algorithm

$\eta \leftarrow 1$ {set iteration counter}
$\mathcal{S}_c \leftarrow$ generateInitialSolution
while $\eta \leq 25000$ **do**
  {Large neighborhood search}
  $\delta \leftarrow$ drawNumberOfCustomersToRemove
  $\mathcal{S}_t \leftarrow$ insertCustomers(removeCustomers($\mathcal{S}_c$, δ))
  $\mathcal{S}_t \leftarrow$ applyATCImprovement ($\mathcal{S}_t$)
  updatePenalties($\mathcal{S}_t$)
  if acceptSA($\mathcal{S}_c, \mathcal{S}_t$) **then**
    $\mathcal{S}_c \leftarrow \mathcal{S}_t$
  end if
  if $\mathcal{S}_t$ improves $\mathcal{S}_{best}$ **then**
    $\mathcal{S}_{best} \leftarrow \mathcal{S}_t$
  end if
  {Set partitioning}
  $\Omega^{LNS} \leftarrow$ addClusters($\mathcal{S}_t$)
  if 5000 iterations have passed since last set partitioning **then**
    $\mathcal{S}_{best} \leftarrow$ solveSetPartitioning($\mathcal{S}_{best}, \Omega^{LNS}$)
  end if
  if solution has not improved for 250 iterations **then**
    $\mathcal{S}_c \leftarrow \mathcal{S}_{best}$
  end if
  $\eta \leftarrow \eta + 1$
end while
return $\mathcal{S}_{best}$

In detail, our procedure works as follows: At the beginning, each request is served by a dedicated route, and if a customer requests service on multiple days of the planning horizon, the corresponding routes are all assigned to the same vehicle. In the next step, we evaluate, for each pair of vehicles, how the solution changes if the routes of the two vehicles are merged on all days of the planning horizon. To this end, we sum up the individual savings of merging both routes on each day on which both vehicles provide service. We limit the evaluation to the two cases in which either (1) all routes of the first vehicle are served before all routes of the second vehicle or (2) vice versa, but we do not allow combinations hereof. This makes sense from a practical viewpoint because it entails that requests of a customer are served at about the same time on each day. The cases that we do not evaluate are likely to result in routes on which a customer is served early on one day and late on another day. Finally, we perform the merge that results in the largest total savings but only if it leads to a feasible solution; that is, no resulting route exceeds the vehicle capacity or maximum route duration, and the vehicle does not violate the ATC constraint. After each step, we remove one of the two vehicles whose routes were merged.

Figure 2 shows an example of four iterations of the modified savings algorithm for a two-day problem. In the beginning, each request is served by a dedicated route. Customer requests occurring on both days are depicted in black, and single-day requests are depicted in gray. In the first step, the routes serving requests of customers A and B are merged on both days (assuming that this generates the highest savings), and the vehicle serving customer B is removed. Arcs to be added are depicted as dashed lines, arcs to be removed as dotted lines. In the second step, the routes serving requests of customers G and D are merged on the second day (again assuming that this generates the highest savings), and we keep the route serving D on the first day (because G does not request service on the first day). The third merge operation affects both days, whereas the fourth merge operation is limited to the first day. The final solution uses three vehicles to serve all customer requests.

5.2. Generalized Objective Function and Penalty Calculation

We allow infeasible solutions during the LNS and evaluate a solution $\mathcal{S}$ using the following generalized objective function that penalizes constraint violations using an adaptive mechanism:

$$z_{gen}(\mathcal{S}) = z(\mathcal{S}) + \sigma_{cap} \cdot G_{cap}(\mathcal{S}) + \sigma_{dur} \cdot G_{dur}(\mathcal{S}) + \sigma_{atc} \cdot G_{atc}(\mathcal{S}),$$

where $z(\mathcal{S})$ denotes the objective value as defined in Equation (1), $G_{cap}(\mathcal{S})$ the capacity violation, $G_{dur}(\mathcal{S})$ the route-duration violation, and $G_{atc}(\mathcal{S})$ the ATC violation of solution $\mathcal{S}$, and σ_{cap}, σ_{dur}, and σ_{atc} are the respective penalty factors.

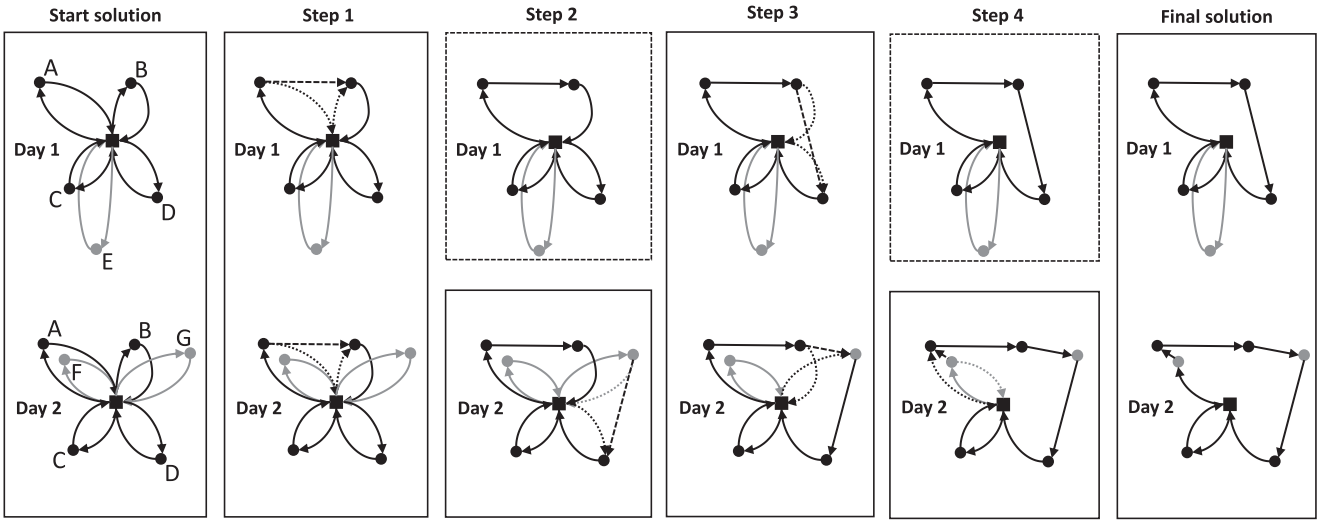
The constraint violations are determined as follows:

- Vehicle capacity violation: $G_{cap}(\mathcal{S}) = \sum_{k\in\mathcal{K}}\sum_{d\in\mathcal{D}} \cdot \max(0, \sum_{i\in\mathcal{N}(r_{kd})} q_{id} - Q)$,
- Route-duration violation: $G_{dur}(\mathcal{S}) = \sum_{k\in\mathcal{K}}\sum_{d\in\mathcal{D}} \cdot \max(0, \max_{i\in\mathcal{N}(r_{kd})}(b_{id} + \hat{t}_{i0d}) - T)$,
- ATC violation: $G_{atc}(\mathcal{S}) = \sum_{i\in\mathcal{N}}\sum_{d\in\mathcal{D}_i}\sum_{d'\in\mathcal{D}_i}\max(0, |b_{id} - b_{id'}| - L)$.

All penalty factors are initialized to a value of 10 and are restricted to the interval [0.01, 1,000]. In every iteration of LNS-ATCI, the penalty factors are multiplied or divided by a factor of 1.05 based on the following rules:

- Factor σ_{cap} is increased if $G_{cap}(\mathcal{S}) > 0$ and decreased otherwise.
- We link the behavior of the penalty factors σ_{dur} and σ_{atc} because $G_{atc}(\mathcal{S})$ and $G_{dur}(\mathcal{S})$ are interdependent in our algorithm. This is due to our procedure for improving the ATC (Section 5.3.2), which often reduces violations of the ATC at the expense of generating longer routes that are likely to violate the route-duration constraint. Therefore, σ_{dur} is increased if $G_{dur}(\mathcal{S}) > 0 \wedge G_{atc}(\mathcal{S}) = 0$, decreased if $G_{dur}(\mathcal{S}) = 0 \wedge G_{atc}(\mathcal{S}) = 0$, and kept at its current value otherwise. Analogously, σ_{atc} is increased if $G_{atc}(\mathcal{S}) > 0 \wedge G_{dur}(\mathcal{S}) = 0$, decreased if $G_{atc}(\mathcal{S}) = 0 \wedge G_{dur}(\mathcal{S}) = 0$, and kept fixed otherwise. We also studied the effect of (1) adjusting the penalty

Figure 2. Four Steps of Our Modified Savings Algorithm on a Two-Day Example Problem



factors independently of each other, that is, also increasing if $G_{dur}(\mathcal{S}) > 0 \wedge G_{atc}(\mathcal{S}) > 0$ and decreasing if $G_{dur}(\mathcal{S}) = 0 \wedge G_{atc}(\mathcal{S}) > 0$ (and analogously for $G_{atc}(\mathcal{S})$) and (2) increasing if $G_{dur}(\mathcal{S}) > 0 \wedge G_{atc}(\mathcal{S}) > 0$ and keeping at the current value if $G_{dur}(\mathcal{S}) = 0 \wedge G_{atc}(\mathcal{S}) > 0$ (and again analogously for $G_{atc}(\mathcal{S})$) in preliminary studies. Both of these variants have a slight detrimental effect on solution quality.

5.3. Large Neighborhood Search Component

LNS, originally introduced by Shaw (1998), is a metaheuristic principle that aims at iteratively improving an initial solution by first removing a larger part of the solution (using a set of so-called removal operators) and then reinserting the removed solution components (using so-called insertion operators). In recent years, LNS has successfully been applied to many variants of the VRP (see, e.g., Ropke and Pisinger 2006a; Masson, Lehuédé, and Péton 2013; and Adulyasak, Cordeau, and Jans 2014).

In each iteration of LNS-ATCI, the number of customers to be removed is randomly drawn from the interval $\delta = \text{rand}([0.05, 0.2]) \cdot \min(150, |\mathcal{N}|)$. Removal, insertion, and subsequent ATC improvement (see Sections 5.3.1 and 5.3.2) create a tentative solution $\mathcal{S}_t$, which may be infeasible because LNS-ATCI always generates a complete solution and does not leave customer requests in a so-called request bank as is often done in LNS. The decision whether to accept $\mathcal{S}_t$ or to keep the current solution $\mathcal{S}_c$ is based on a simulated annealing (SA) criterion (Section 5.3.3).

5.3.1. Removal and Insertion Operators. In each iteration, LNS-ATCI randomly selects one of the removal and one of the insertion operators with uniform probability. Removal/insertion of a customer implies the removal/insertion of all service requests of this customer on all days of the planning horizon. The following removal operators are used:

Random removal removes δ arbitrarily selected customers.

Worst removal was introduced by Ropke and Pisinger (2006b) to remove vertices that are served at undesirable positions in the routes. We propose a modified version of the operator that is (1) not randomized and (2) adapted to the ConVRP. Let $\mathcal{S}^{-i}$ denote a solution in which customer i is removed on all days. We define the following measure κ_i to determine which customers should be removed from the solution:

$$\kappa_i = (z(\mathcal{S}^{-i}) - z(\mathcal{S}))/|\mathcal{D}_i| + \sigma_{cap} \cdot (G_{cap}(\mathcal{S}^{-i}) - G_{cap}(\mathcal{S})) + \sigma_{dur} \cdot ((G_{dur}(\mathcal{S}^{-i}) - G_{dur}(\mathcal{S})).$$

Note that we divide the reduction in total operating time by the number of days on which customer i requires service; that is, we base the decision on the average operating time reduction per request. Otherwise, the selection would be biased toward customers with a higher number of service requests. The positive effect of dividing by the number of service days is supported by the results of preliminary experiments. Finally, all customers are sorted in ascending order of κ_i, and we select the first δ customers for removal.

Proximity removal removes close customers. Let $t_{max} = \max_{i \in \mathcal{N}, j \in \mathcal{N}} t_{ij}$ be the maximum travel time between any pair of customers. The first customer i to remove is randomly selected. It serves as a center point for the subsequent removals of customers j that are randomly selected if $t_{ij} \leq 0.2 \cdot t_{max}$ until δ customers are removed. If the number of customers within $0.2 \cdot t_{max}$ is less than the number of customers to be removed, we randomly select a customer that is served by the same vehicle as the last removed customer to be the next center point.

Vehicle removal selects customers for removal that are served by the same vehicle. We start with a randomly

selected vehicle and remove all customers served by the vehicle. If at least δ customers have been removed, we terminate the procedure. Otherwise, each remaining vehicle k is selected as the next vehicle for removal with a probability $p_k = \chi_{kk'} / \sum_{k'' \in \mathcal{K}} \chi_{k'k''}$ that is proportional to the inverse distance $\chi_{kk'}$ between vehicle k and the previously selected vehicle k'. The distance between vehicles is the Euclidean distance between their centers of gravity, which is determined as the mean of the weighted coordinates of the customers served by the vehicle. The coordinates of a customer i are weighted with the factor $|\mathcal{D}_i|/|\mathcal{D}|$.

We use the following insertion operators:

Greedy insertion iteratively performs the best possible insertion in a myopic manner. Computational experience shows that the ATC violations caused by the insertion of customers into partial solutions are not representative for the ATC violations of the final completed solution. In addition, calculating these violations is computationally expensive. Therefore, we do not consider the direct effect of an insertion on the ATC violation by means of G_{atc} and instead use a learning-based penalty component P_{atc} that aims at indirectly improving the ATC. Not directly considering ATC violations allows the determination of the best insertion position separately on each individual day because violations of route duration and capacity are not linked over the days.

Let $r_{kd}^{+(i,p)}$ be the current route of vehicle k on day d with customer i inserted after position p. Then, for each still-unassigned customer i and each vehicle k, we compute the cost increase

$$\begin{aligned}\Delta\hat{z}_{ik} = \sum_{d \in \mathcal{D}_i} \min_{p=0,\dots,n_{kd}} & ((z(r_{kd}^{+(i,p)}) - z(r_{kd})) + \sigma_{cap} \\ & \cdot (G_{cap}(r_{kd}^{+(i,p)}) - G_{cap}(r_{kd})) \\ & + \sigma_{dur} \cdot ((G_{dur}(r_{kd}^{+(i,p)}) - G_{dur}(r_{kd})) + P_{atc}(r_{kd}^{+(i,p)})),\end{aligned}$$

and perform the cheapest insertion according to $\Delta\hat{z}_{ik}$. The procedure is iterated until all customers are inserted.

The aim of the penalty P_{atc} is to identify solution components that are critical with regards to the ATC constraints. For every arc $(h,j) \in \mathcal{A}_d$ and for every day $d \in \mathcal{D}$, we store a penalty value μ_{hjd} that is initially set to zero. After each complete LNS iteration, including the ATC repair step (see Section 5.3.2 for details), the penalty values of a subset of the arcs contained in the newly generated solution $\mathcal{S}$ are updated based on the ATC of the vehicle k traveling the arc:

- If vehicle k does not violate the ATC, we set $\mu_{hjd} := \max(0, \mu_{hjd} - \Delta\mu_{hjd})$ with $\Delta\mu_{hjd} = 0.25 \cdot t_{hj}$ for all arcs traveled by vehicle k.
- If vehicle k violates the ATC, we first draw a randomly selected subset $\overline{\mathcal{D}}$ of the days of the planning horizon (each day is drawn with a probability of 0.5) on which the arcs contained in the solution shall be penalized; penalizing on all of the days on which the vehicle is used does not help our algorithm to explore new solution components. Then, we increase on every day $d \in \overline{\mathcal{D}}$ the penalty values μ_{hjd} of all arcs $(h,j) \in \mathcal{A}_d$ that are traveled by the vehicle by $\Delta\mu_{hjd}$.

Thus, we determine the penalty $P_{atc}(r_{kd}^{+(i,p)})$ for inserting customer i after position p into route r_{kd}, that is, between vertices $v = r_{kd}(p)$ and $w = r_{kd}(p+1)$ as $P_{atc}(r_{kd}^{+(i,p)}) = \mu_{vid} + \mu_{iwd}$. A large penalty μ_{hjd} might prevent the corresponding arc from being included in a solution, and consequently, with the rules described, this penalty value would never be reduced again. To counteract this undesired behavior, we discount in every iteration all penalty values by a constant factor as $\mu_{hjd} := \mu_{hjd}/1.5$.

Regret insertion tries to anticipate and avoid the negative future consequences of greedy insertion (Ropke and Pisinger 2006b). We calculate the two-regret value of each customer i as the difference between the insertion cost $\Delta\hat{z}_{ik}$ of assigning customer i to the best vehicle k and the cost $\Delta\hat{z}_{ik'}$ of assigning it to the second-best vehicle k'. The customer with the largest absolute two-regret value is selected for insertion, and the procedure is iterated until all customers are inserted.

We implement two additional variants of greedy and regret insertion that add a continuous diversification penalty $P_{div}(i,k)$ for assigning customer i to vehicle k (see, e.g., Cordeau, Laporte, and Mercier 2001):

$$P_{div}(i,k) = \text{rand}([0.5, 1.0]) \cdot \frac{\sqrt{z(\mathcal{S}) \cdot \zeta_{i,k}}}{\sum_{i \in \mathcal{N}} |\mathcal{D}_i|},$$

where $\zeta_{i,k}$ is the frequency with which customer i was assigned to vehicle k by an LNS insertion operator, and the randomization is introduced to prevent cycling of the algorithm. The goal is to encourage the experimental exploration of different solutions. Note that it might happen that identical routes are associated with different vehicles during the course of the algorithm, and therefore, the frequencies $\zeta_{i,k}$ for a certain customer set may be reset from time to time. In our case, this does not cause any numerical problems. The same diversification penalty has also been used in several other papers (see, e.g., Cordeau and Laporte 2003 and Goeke and Schneider 2015).

5.3.2. ATC Improvement. We find that the generated solutions that violate ATC constraints often contain routes that serve the same set of customers in almost reversed order on different days; that is, they strongly disregard the precedence principle of Groër, Golden, and Wasil (2009) described in Section 1. For each vehicle violating ATC constraints, we try to improve the ATC with the following two-stage procedure:

1. Inversion of a subset of the routes to generate similar orders of the customer visits on all days,
2. Customer relocation to reduce ATC violations.

To determine which routes should be selected for inversion in step 1, we require (1) a measure for the difference between routes with regard to the order of customer visits and (2) a mechanism to decide which subset of day routes should be inverted based on the pairwise difference of the routes with respect to the measure defined in (1).

As a difference measure between two routes, we use the number of customer pairs that occur in reverse order in the two routes. More precisely, we first define a function $p(r_{kd}, v)$ that returns the position of vertex v in route r_{kd}. Then, we determine $\mathcal{R}^{<}_{kd} = \{(v,w) \mid v, w \in \mathcal{N}(r_{kd}), p(r_{kd},v) < p(r_{kd},w)\}$ as the set of all pairs of customers (v,w), where customer v is served before customer w in route r_{kd}. Further, we define a function $\gamma(v,w)$ that returns one if $v = w$ and zero otherwise. With this, we measure the difference $\rho(r_{kd}, r_{kd'})$ between routes r_{kd} and $r_{kd'}$ on two days d and d' as

$$\rho(r_{kd}, r_{kd'}) = \sum_{(v,w)\in\mathcal{R}^{<}_{kd}} \sum_{(v',w')\in\mathcal{R}^{<}_{kd'}} \gamma(v,w') \cdot \gamma(w,v').$$

We illustrate the calculation with the following example: routes $r_{kd} = <0,4,1,2,0>$ and $r_{kd'} = <0,1,2,4,0>$ have the corresponding sets $\mathcal{R}^{<}_{kd} = \{(4,1),(4,2),(1,2)\}$ and $\mathcal{R}^{<}_{kd'} = \{(1,2),(1,4),(2,4)\}$, respectively. The difference $\rho(r_{kd}, r_{kd'})$ is equal to two because two pairs occur in reverse order in both routes, namely (4,1) and (1,4) and (4,2) and (2,4).

To determine the subset of routes to be inverted, we separate the day routes of each vehicle into two groups using average linkage clustering (see, e.g., Sarstedt and Mooi 2014) based on the distance measure ρ: first, we create one cluster $\mathcal{H} = \{r_{kd}\}$ for each day $d \in \mathcal{D}$; then, we iteratively merge the pair of clusters $\mathcal{H}$ and $\mathcal{H}'$ that minimizes $\sum_{r_{kd}\in\mathcal{H}} \sum_{r_{kd'}\in\mathcal{H}'} \rho(r_{kd}, r_{kd'})/(|\mathcal{H}| \cdot |\mathcal{H}'|)$ until only two clusters remain. We first generate a solution by inverting all routes of the first group and then a second solution by inverting all routes of the second group. We calculate the number of customers that violate the ATC constraint in each of the two resulting solutions and in the original solution, and we hand the solution with the lower number over to the second stage. To save computational effort spent on evaluating unpromising steps, we skip the clustering and the subsequent route inversion if the routes of a vehicle k are too similar; that is, if $\sum_{d\in\mathcal{D}} \left(\sum_{d'\in\mathcal{D}, d\neq d'} \rho(r_{kd}, r_{kd'})/|\mathcal{N}(r_{kd})|\right)$ is below a threshold value that we set to 0.01.

The idea of the second stage is to use customer relocations to improve the ATC. We first determine the customer with the largest ATC violation and inspect all requests of this customer. We call every request late (early) if it causes a violation of the maximum allowed time difference assuming that the request served earliest (latest) is fixed. Then, we investigate two options for improving ATC of customer i: (1) to serve customer i later on the days with early service and (2) to serve customer i earlier on the days with late service. For both cases, we determine a time window within which the customer should be served on the violating days.

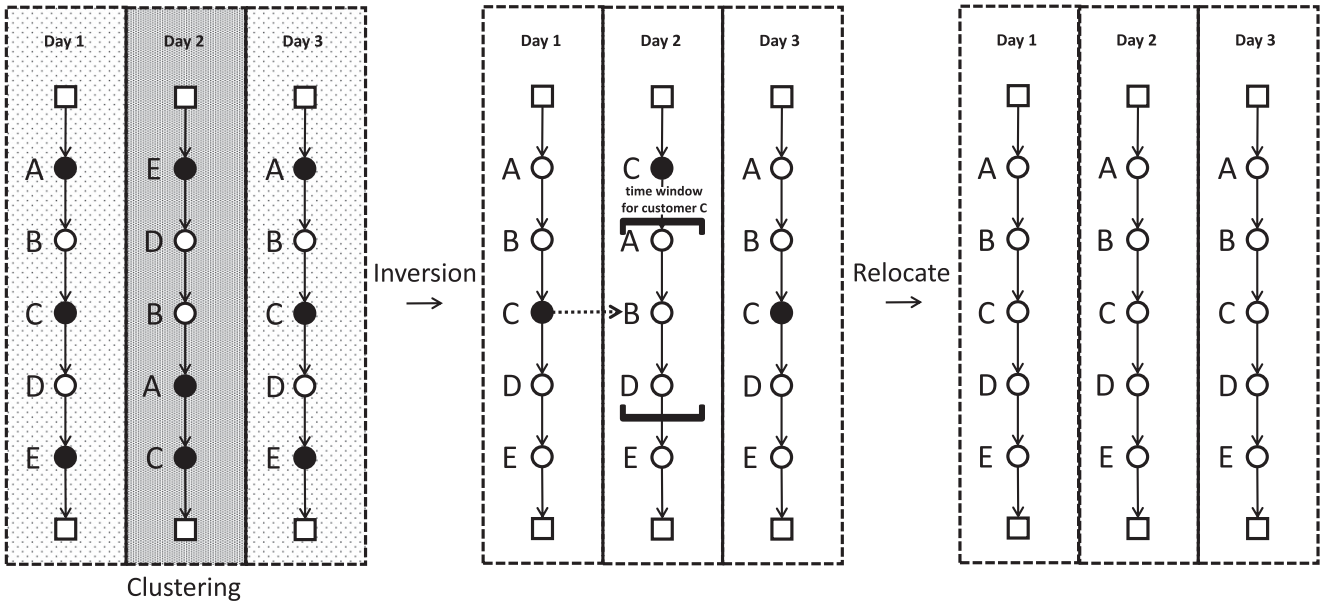
In case (1), we separate the planning horizon into a set of early days $\Psi_i = \{d | d \in \mathcal{D}_i \wedge b_{id} < (\max_{d\in\mathcal{D}_i} b_{id}) - L\}$ and a set of feasible days $\Theta_i = \mathcal{D}_i \backslash \Psi_i$. Now, we determine a time window $[\max_{d\in\Theta_i} b_{id} - L, \min_{d\in\Theta_i} b_{id} + L]$. Then, for each violating day $d \in \Psi_i$, we relocate customer i such that the new arrival time lies within the time window and the increase in operating time $z(S)$ is minimal. The procedure for case (2) works analogously. Note that the determined time window does not guarantee ATC of the resulting solution (because ATC at other customers is not taken into account) but aims to guide the algorithm toward better ATC.

We tentatively perform the relocations for the selected customer and both cases. Because the relocation of requests may lead to violations of the maximum allowed time difference at other customers, we either keep the original solution or the solution related to case (1) or case (2), depending on which solution has the lowest number of violating customers. Then, ATC violations are recalculated, and we continue with the customer that now has the largest ATC violation until all customers are served consistently or each customer has been tried once.

Figure 3 shows an example application of our ATC improvement procedure. Vertices depicted in black violate the ATC constraint. In the left part, we illustrate the clustering algorithm and the inversion of routes. First, the difference between every pair of days is calculated: days 1 and 3 are identical and have a distance of zero, days 1 and 2 and days 2 and 3 both have a distance of eight. The clustering assigns days 1 and 3 to the first cluster and day 2 to the second cluster. The best solution is obtained by inverting the route on the second day (the inversion of the routes of the first cluster is not shown). In the center part of the figure, the resulting solution is shown: only customer C exhibits an ATC violation. Now, we evaluate both options: to serve customer C later on day 2 or earlier on days 1 and 3 (only the first case is shown). Based on the fixed arrival times on days 1 and 3, we determine the time window within which customer C has to be served on day 2. Customer C is relocated between customers B and D because the increase in operating time is minimal. The final solution, which respects the ATC constraints, is shown in the right part of the figure.

5.3.3. Simulated Annealing–Based Acceptance. Our LNS-ATCI always accepts improving solutions, and a deteriorating solution $\mathcal{S}_t$ is accepted with a probability that depends on the difference between the objective

Figure 3. Example Application of the ATC Improvement Procedure



function values Δz_{rel} and a temperature θ (see, e.g., Kirkpatrick, Gelatt, and Vecchi 1983):

$$p(\mathcal{S}_t, \mathcal{S}_c, \theta) = e^{\frac{-\Delta z_{rel}(\mathcal{S}_t, \mathcal{S}_c)}{\theta}}.$$

To avoid the undesired effect that differences between objective function values also depend on the values of the penalty factors, we use the relative difference between objective function values to calculate the acceptance probabilities (see Goeke and Schneider 2015):

$$\Delta z_{rel}(\mathcal{S}_t, \mathcal{S}_c) = \frac{z_{gen}(\mathcal{S}_t) - z_{gen}(\mathcal{S}_c)}{z_{gen}(\mathcal{S}_c)}.$$

The temperature follows a predefined cooling schedule defined by an initial temperature and a cooling rate. The initial temperature is such that a solution that deteriorates the initial solution by 50% is accepted with a probability of 50%. We decrease the temperature in every iteration by multiplying it with the cooling rate, and we set the cooling rate such that the temperature is below $\theta_{min} = 0.0001$ in the last 20% of iterations.

5.4. Set Partitioning

Every 5,000 iterations, we try to improve the best solution found so far by solving the set-partitioning formulation (15)–(18) for a pool Ω^{LNS} of heuristically determined clusters with a commercial solver. If we find a new best solution, we replace the previous best solution $\mathcal{S}_{best}$. To speed up the solution process, we use the current best solution as the initial solution. To generate Ω^{LNS}, we add all feasible clusters that we find during the search, and we store the associated objective value g_C and the routing solutions of the individual days. Whenever we encounter a cluster that is already present in Ω^{LNS}, we update the objective value and the routing if the new objective value is better.

Because we never remove any cluster from Ω^{LNS}, the problem size is strictly increasing, and we use the following three approaches to reduce runtime:

Restrict the Number of Routes. Subramanian, Uchoa, and Ochi (2013) found that restricting the number of available routes can speed up the solution of the set-partitioning problem for a wide range of VRPs. We add the following constraint to restrict the number of selected clusters:

$$|\mathcal{K}'| - 1 \leq \sum_{C \in \Omega^{LNS}} \xi_C \leq |\mathcal{K}'| + 1,$$

with $\mathcal{K}'$ the set of vehicles in the current $\mathcal{S}_{best}$ that serve at least one customer.

Only Solve Promising Problems. We observed that it is unlikely to find a new best solution if the lower bound LB given by the linear relaxation of the current set-partitioning does not improve compared with the LB of the last set-partitioning solved. Therefore, before we solve the set-partition problem including the integrality constraints (18), we relax the latter to quickly obtain an LB. Now, we only add the integrality constraints if this LB improves the previous LB by more than 0.4%.

Limit the Runtime of the Solver. We adjust the time limit of the commercial solver depending on the initial optimality gap, that is, the difference between the objective value $\mathcal{S}_{best}$ and LB. In detail, we calculate the runtime in seconds as $10 + \min(20, 20 \cdot (z(\mathcal{S}_{best}) - \text{LB})/0.04)$. Consequently, the maximum runtime of

30 seconds is used if the optimality gap is at least 4%. Furthermore, the last problem solved during LNS-ATCI is always performed with the maximum time limit of 30 seconds.

6. Numerical Studies

In this section, we present our numerical studies to assess the performance of CCG and LNS-ATCI. In Section 6.1, we describe the benchmark instances available from the literature and the generation of new medium-sized instances. In the first experiment (Section 6.2), we solve the compact formulation of Groër, Golden, and Wasil (2009) and our improved formulation using CPLEX and compare it to CCG on small-sized instances from the literature. In the second experiment (Section 6.3), we study the performance of CCG on the newly generated medium-sized instances and investigate how the performance is related to parameters of the problem. In the third experiment (Section 6.4), we study the effect of allowing flexible departure times at the depot and waiting at customers on the performance of CCG and on the total operating time of the resulting solutions. Finally, we compare LNS-ATCI as a stand-alone metaheuristic with other state-of-the-art metaheuristics on benchmark instances from the literature (Section 6.5).

We performed all tests on a desktop computer with an AMD FX-6300 processor at 3.5 GHz with 8 GB of RAM and running Windows 10 Pro. We used CPLEX 12.6.3 as a MILP solver to solve problems (29)–(33) in Step 1, problems (19)–(22) in Steps 2 and 3, problems (34)–(42) in Step 4, and the set-partitioning problem detailed in Section 5.4. CCG is implemented in C and LNS-ATCI in Java. Both codes and CPLEX were executed using a single core. For CCG, we set a time limit of two hours for all tests. All computing times are reported in seconds.

6.1. Benchmark Instances

Several benchmark sets for the ConVRP are available in the literature. Two sets are introduced in Groër, Golden, and Wasil (2009) and differ with regard to instance size (small and large) and the way the instances are generated: Data set A contains five instances with 10 customers and five instances with 12 customers; the planning horizon spans three days; customers have a 70% service frequency, that is, the probability that a customer requires service on a given day is 0.7. Data set B contains 12 instances with 50 to 199 customers. The instances are derived from instances of the well-known benchmark set for the distance-constrained capacitated VRP presented in Christofides, Mingozzi, and Toth (1979). Five of these instances have route-duration constraints; the planning horizon spans five days; customers have a 70% service frequency. Note that Groër, Golden, and Wasil (2009) do not restrict the maximum allowed time difference L on these instances but report the maximum value that they obtain for each instance. Because later works (Tarantilis, Stavropoulou, and Repoussis 2012; Kovacs, Parragh, and Hartl 2014; Kovacs et al. 2015) reported this maximum value as a limit on L, we do the same to have comparable results.

Data set C contains 144 instances and was introduced by Kovacs, Parragh, and Hartl (2014) to investigate the impact of varying the maximum allowed time difference and the service frequency. In addition to the instances from data set B with 70% service frequency, the authors generate instances with 50% and 90% service frequency. Then, for each service frequency, they vary the maximum allowed time difference and generate four instances: the first has an unbounded value of L (these instances are referred to as L_∞ in the following), and the other three are obtained by setting $L_x = x \cdot L_{max}$, where L_{max} is the maximum arrival time difference obtained by solving the instance L_∞ and $x = 0.4, 0.6, 0.8$ (these instances are referred to as $L_{0.4}$, $L_{0.6}$, and $L_{0.8}$).

Data sets A–C are used to assess the performance of LNS-ATCI, but only data set A is suitable to assess the performance of CCG because most of the instances from the other sets are too large. Therefore, we create an additional data set D with 144 instances with 20 to 30 customers by adapting the instances from data set C. We leave the planning horizon of five days unchanged because a weekly plan seems a reasonable setting. From every subset representing one combination of service frequency and maximum allowed time difference, we select the instances labeled 6–8 from instances 1–12 because they have both route-duration and capacity constraints and create two new instances: the first containing the first 20 customers and the second the first 30 customers. In addition, we duplicate every new instance by removing the route duration to study the influence that this parameter has on the performance of our approach. We round all distances to the second decimal place to make our results practically independent of the internal precision of the hardware and software used. We name the instances according to the following exemplary format: 6_19_0.5_0.4 means that the instance is based on instance 6 from data set C, contains 19 customers that require service (some instances contain customers that do not require service on any day, and for clarity, we remove these customers from the instance), has a service frequency of 50%, and has a maximum allowed time difference of 40% of the maximum arrival time difference obtained for any of the customers in the unbounded instance in Kovacs, Parragh, and Hartl (2014).

6.2. Comparison Between Compact Formulations and CCG on Data Set A

In this section, we compare the computational performance of the original compact formulation of Groër, Golden, and Wasil (2009) (hereafter called GGW), our

improved compact formulation (1)–(12) (called GRS), and CCG on data set A.

Table 1 reports the instance name (Inst) and the optimal solution cost (z(Opt)). For both GGW and GRS, we provide the lower bound obtained by the corresponding linear relaxation in percentage of the optimal solution (Δ_{LB}) and the total computing time (t). For CCG, we give the number of routes ($|\Phi|$ with $\Phi = \cup_{d\in\mathcal{D}}\Phi_d$) and the initial number of clusters generated in Step 2 ($|\hat{\Omega}'|$), lower bounds $x \in \{z(LP_1), z(LP_2), z(\overline{SP})\}$ in percentage of the optimal solution (Δ_x), the number of clusters left after executing steps 1–3 ($|\hat{\Omega}''|$), the number of times step 4 is executed ($|\overline{\Omega}|$), and the total computing time of CCG without considering the time to compute UB (t_{noUB}) and including the time to compute UB (t_{tot}). Detailed results for CCG can be found in Table EC.1 of the e-companion of this paper.

As the computing times show, all three formulations could solve all instances to optimality. However, GRS outperforms GGW in terms of both lower bound provided by the linear relaxation, which is on average 7.8% higher, and the total computing time (18.5 versus 573.1 seconds). CCG is significantly faster on average than GRS even when taking into account the time to compute UB. We also observe that the lower bounds computed by CCG are of very good quality. In particular, note that the average optimality gaps of LP_1 and LP_2, which do not consider ATC, are quite small (1.3% and 0.4%, respectively). Moreover, the number of clusters for which ATC had to be included a posteriori is very low (see column $|\overline{\Omega}|$). This suggests that DC in many instances already implies ATC.

Finally, we assessed the impact of using GSECs by removing them from GGW and GRS (not reported in the table): without GSECs, we are still able to solve all instances to optimality within the time limit, but the average computing time increases significantly, that is, for GGW from 573.1 to 1,378.3 seconds and for GRS from 18.5 to 272.3 seconds.

6.3. Computational Results of CCG on Data Set D

In this section, we investigate the computational performance of CCG on the 144 new instances of data set D. In Table 2, panels A and B, the results are aggregated according to the number of customers ($|\mathcal{N}|$) and the presence/absence of route-duration constraints (T = yes/no). Table 2, panel A reports results based on different values of maximum allowed time difference ($L_{0.4}$, $L_{0.6}$, $L_{0.8}$, L_∞), and Table 2, panel B is based on different values of service frequency ($\mathcal{D}_{0.5}$, $\mathcal{D}_{0.7}$, and $\mathcal{D}_{0.9}$). The values reported in each row of Table 2, panels A and B are averages over the corresponding nine and 12 instances, respectively. Column Opt reports the number of instances solved to optimality out of the total number of instances in the group. Detailed results can be found in Tables EC.2–EC.5 of the e-companion of this paper.

The discussion of the results follows the order of the columns in Table 2, panels A and B. We observe that CCG solves 128 of the 144 instances to optimality. Instances with 30 customers (60 solved) are obviously more difficult than instances with 20 customers (68 solved). The remaining 16 instances could not be solved because either the time limit was reached (eight instances) or CCG ran out of memory in step 2 while generating the set of all routes Φ (eight instances). Instances without a route-duration constraint are harder to solve: 15 out of the 16 unsolved instances have unlimited route duration, and all cases of insufficient memory occur for this type of instance. As Table 2, panel A shows, instances that have a lower maximum allowed time difference are more difficult: 10 of the 16 open instances belong to the group $L_{0.4}$, and the average computing times increase significantly if the maximum allowed time difference decreases. The latter effect is due to the lower quality of the bounds provided by CCG. Table 2, panel B shows that no similar effect can be observed for the service frequency: six of the open instances are in group $\mathcal{D}_{0.5}$, two in $\mathcal{D}_{0.7}$, and eight in $\mathcal{D}_{0.9}$.

Table 1. Computational Performance of GGW, GRS, and CCG on Data Set A

		GGW		GRS		CCG								
Inst.	z(Opt)	Δ_{LB}	t	Δ_{LB}	t	$\lvert\Phi\rvert$	$\lvert\hat{\Omega}'\rvert$	$\Delta_{z(LP_1)}$	$\Delta_{z(LP_2)}$	$\lvert\hat{\Omega}''\rvert$	$\Delta_{z(\overline{SP})}$	$\lvert\overline{\Omega}\rvert$	t_{noUB}	t_{tot}
1_10	142.03	61.0	9.4	66.5	3.0	305	744	100.0	100.0	5	100.0	2	0.1	5.1
2_10	121.07	57.3	2.0	69.4	0.9	760	951	97.6	100.0	9	100.0	2	0.1	2.6
3_10	149.41	53.5	10.4	61.8	6.3	1,221	774	100.0	100.0	10	100.0	2	0.1	2.7
4_10	150.89	58.5	13.0	62.7	2.1	753	801	100.0	100.0	9	100.0	2	0.1	2.5
5_10	132.31	63.9	636.3	71.2	13.1	718	810	95.9	98.8	14	100.0	4	0.2	3.6
1_12	171.02	57.2	1,524.1	66.4	70.5	852	2,150	98.9	99.9	19	100.0	2	0.1	3.3
2_12	111.54	64.5	4.5	72.1	3.9	657	3,743	99.5	99.5	11	100.0	2	0.1	3.2
3_12	145.69	51.4	179.1	59.2	31.2	1,573	3,666	99.7	99.7	10	100.0	4	0.2	3.0
4_12	166.37	51.7	3,286.3	60.1	38.7	928	2,317	97.1	98.6	47	100.0	13	0.5	3.9
5_12	140.42	52.8	66.0	60.5	15.2	1,188	3,412	98.5	99.5	11	100.0	5	0.2	3.3
Average		57.2	573.1	65.0	18.5	896	1,937	98.7	99.6	15	100.0	4	0.2	3.3

Table 2. Overview of Results on Newly Generated Medium-Sized Instances of Data Set D

$\lvert\mathcal{N}\rvert$	T	Opt	$\lvert\Phi\rvert$	$\lvert\hat{\Omega}'\rvert$	$\Delta_{z(\mathrm{LP}_1)}$	$\Delta_{z(\mathrm{LP}_2)}$	$\lvert\hat{\Omega}''\rvert$	$\Delta_{z(\overline{\mathrm{SP}})}$	$\lvert\overline{\Omega}\rvert$	t_{noUB}	t_{tot}
Panel A: Aggregated for different values of maximum allowed time difference (L_x)											
$L_{0.4}$											
20	Yes	9/9	51,143	183,386	97.6	97.8	379	100.0	30	25.0	164.0
20	No	5/9	211,817	561,468	95.8	96.1	39,780	97.8	41	3,253.1	3,410.8
30	Yes	8/9	870,272	17,395,266	97.6	98.0	5,575	99.6	30	905.1	1,159.0
30	No	4/9	10,924,570	202,276,014	97.6	97.9	6,322	99.2	67	3,241.5	3,482.1
$L_{0.6}$											
20	Yes	9/9	51,143	183,386	99.4	99.5	47	100.0	6	0.7	105.8
20	No	9/9	211,817	561,468	98.6	98.7	100	100.0	10	485.0	633.5
30	Yes	9/9	870,272	17,395,266	99.3	99.5	200	100.0	6	55.2	290.7
30	No	7/9	10,924,570	202,276,014	99.1	99.2	104	100.0	15	324.8	541.0
$L_{0.8}$											
20	Yes	9/9	51,143	183,386	99.6	99.8	34	100.0	4	0.5	86.0
20	No	9/9	211,817	561,468	99.2	99.3	37	100.0	4	59.2	194.8
30	Yes	9/9	870,272	17,395,266	99.4	99.6	148	100.0	5	53.9	266.0
30	No	7/9	10,924,570	202,276,014	99.9	99.9	16	100.0	4	280.0	466.0
L_{∞}											
20	Yes	9/9	51,143	183,386	99.9	99.9	22	100.0	3	0.4	59.4
20	No	9/9	211,817	561,468	100.0	100.0	10	100.0	2	6.9	81.9
30	Yes	9/9	870,272	17,395,266	99.5	99.7	132	100.0	4	53.6	187.5
30	No	7/9	10,924,570	202,276,014	100.0	100.0	12	100.0	3	280.2	406.0
Panel B: Aggregated for different service frequencies ($\mathcal{D}_x$)											
$\mathcal{D}_{0.5}$											
20	Yes	12/12	3,831	372,678	99.3	99.4	47	100.0	12	2.0	76.3
20	No	11/12	6,239	699,534	97.8	97.8	27,938	99.4	29	882.5	970.5
30	Yes	12/12	43,382	47,459,139	99.5	99.6	141	100.0	15	105.4	260.0
30	No	7/12	93,653	205,949,404	98.9	98.9	2,059	99.9	49	1,077.5	1,208.1
$\mathcal{D}_{0.7}$											
20	Yes	12/12	30,451	112,843	98.6	98.7	210	100.0	13	12.0	108.1
20	No	11/12	85,691	513,249	98.7	98.8	989	99.5	10	654.2	793.4
30	Yes	12/12	519,274	3,438,466	98.7	99.3	782	100.0	11	45.5	272.5
30	No	11/12	9,137,334	316,518,019	99.3	99.4	1,624	99.8	16	994.0	1,221.0
$\mathcal{D}_{0.9}$											
20	Yes	12/12	119,146	64,637	99.4	99.6	104	100.0	7	5.8	127.0
20	No	10/12	543,521	471,622	98.7	98.9	1,019	99.4	4	1,316.4	1,476.9
30	Yes	11/12	2,048,161	1,288,192	98.5	98.7	3,619	99.7	8	650.0	894.9
30	No	7/12	24,436,342	27,239,617	99.2	99.4	1,152	99.7	6	1,042.3	1,243.6

The number of routes $|\Phi|$ and the initial number of clusters $|\hat{\Omega}'|$ increases with the number of customers and with unlimited route duration. As Table 2, panel A shows, both values are independent of the group L_x because the maximum allowed time difference is not considered in this step. From Table 2, panel B, we observe that the number of routes increases drastically with the service frequency because more customers request service on any of the days. The number of clusters $|\hat{\Omega}'|$ decreases with rising service frequency because there are fewer ways to combine the routes to clusters that respect the DC. Note that the values $|\hat{\Omega}'|$ provided for $|\mathcal{N}| = 30$ and $T =$ no of groups $\mathcal{D}_{0.5}$ and $\mathcal{D}_{0.9}$ can be compared with each other because, for both groups, we could not generate the initial routes for the same set of instances because of insufficient memory. This does not hold for the comparison with $\mathcal{D}_{0.7}$.

CCG provides high-quality lower bounds: $\Delta_{z(\mathrm{LP}_1)}$ is 98.9% averaged over all instances of data set D (computed from the results reported in Tables EC.2–EC.5 of the e-companion), $\Delta_{z(\mathrm{LP}_2)}$ is 99.0%, the lowest value for any instance is 91.3% for both bounds, and for none of the groups considered in Table 2, panels A and B the bounds lie below 96.1%. The quality of the bounds decreases for lower values of L_x, but there is no clear relationship between their quality and the service frequency $\mathcal{D}_x$, the number of customers, or the existence of a limit on the route duration. Analogous effects can be observed for the final number of clusters $|\hat{\Omega}''|$. It is noteworthy how strongly the number of clusters can be reduced because of step 3 when bounds are tight.

The final lower bounds $\Delta_{z(\mathrm{SP})}$ show that, for groups with a maximum allowed time difference larger than $L_{0.4}$, we are able to solve all instances to optimality

whenever we have enough memory to generate the initial clusters. For a maximum allowed time difference of $L_{0.4}$, we obtain aggregated final bounds of at least 97.8%. In addition, we find that for large values of L_x or $\mathcal{D}_x$, the number of clusters for which ATC had to be included a posteriori (see column $|\overline{\Omega}|$) is typically very low; that is, the set-partitioning often quickly identifies the optimal clusters in step 4. As already discussed, removing the route duration or decreasing the maximum allowed time difference makes the instances more difficult, and thus, the total computing time increases. On the other hand, there is no clear relationship between the service frequency and the total computing time.

Finally, we note that the performance of CCG strongly depends on the number of clusters that need to be generated, that is, how restrictive vehicle capacity and route duration constraints are. If they are not binding, the number of possible clusters for an instance with 30 customers exceeds one billion (2^{30}), which translates to about 8 GB of RAM. The number of clusters in larger instances can only be enumerated if capacity or route-duration constraints are restrictive.

6.4. Effect of Flexible Departure Times at the Depot and Waiting at Customers

The compact formulation GRS, the exact method CCG, and the heuristic method LNS-ATCI address the ConVRP as defined in Groër, Golden, and Wasil (2009). Therefore, the departure of a vehicle cannot be shifted, but all vehicles are required to start their route at time zero, and no waiting is allowed between customer visits. These two constraints may be too restrictive in some practical settings, and we are, therefore, interested in the effect of relaxing them. In this section, we describe how GRS and CCG can be adapted to handle the variants of the ConVRP in which

- Only the departure time at the depot is flexible (Section 6.4.1). This variant was introduced by Kovacs, Parragh, and Hartl (2014) as ConVRP with shiftable starting times (ConVRP-SST).
- The departure time at the depot is flexible and waiting at customers is allowed (Section 6.4.2). We denote this variant as ConVRP-SST and waiting at customers (ConVRP-SSTW).

In Section 6.4.3, we study the performance of CCG on these variants, and we discuss the benefits of adding such flexibility.

6.4.1. ConVRP with Shiftable Starting Times.

In GRS, constraints (7) are removed to allow shifts of the departure time from the depot. In CCG, constraints (38) are removed from problems (34)–(42) when computing the cost g_C of a given cluster C in step 4.

6.4.2. ConVRP with Shiftable Starting Times and Waiting at Customers.

To allow both shifts of the departure time from the depot and waiting at customers, GRS is changed as follows:

- An additional set of variables $w_{id} \in \mathbb{R}_+$ that represent the waiting time on day $d \in \mathcal{D}$ at customer $i \in \mathcal{N}_d$ of the vehicle serving customer i is added.
- The objective function (1) must take the total waiting time into account:

$$z = \min \sum_{d\in\mathcal{D}} \sum_{k\in\mathcal{K}} \sum_{(i,j)\in\mathcal{A}_d} \hat{t}_{ijd} x_{ijkd} + \sum_{d\in\mathcal{D}} \sum_{i\in\mathcal{N}_d} w_{id}.$$

- Constraints (7) are removed to allow vehicles to depart from the depot at any time.
- To take into account the waiting time while computing the arrival time at a customer, constraints (8) are changed as follows:

$$b_{id} + (\hat{t}_{ijd} + T)\sum_{k\in\mathcal{K}} x_{ijkd} + w_{id} \le b_{jd} + T \quad d \in \mathcal{D} \quad i,j \in \mathcal{N}_d : i \ne j.$$

- To determine the waiting times w_{id}, the following set of constraints is added:

$$w_{id} \ge b_{jd} - b_{id} - \hat{t}_{ijd} - T + T\sum_{k\in\mathcal{K}} x_{ijkd} \quad d \in \mathcal{D} \quad i,j \in \mathcal{N}_d : i \ne j.$$

Similarly, in CCG, problems (34)–(42) are changed as follows:

- An additional set of variables $w_{id} \in \mathbb{R}_+$ that represent the waiting time on day $d \in \mathcal{D}$ at customer $i \in \mathcal{N}_d \cap C$ of the vehicle serving customer i is added.
- The objective function (34) must take the waiting times at customers into account:

$$g_C = \min \sum_{d\in\mathcal{D}} \sum_{(i,j)\in\mathcal{A}_d(C)} \hat{t}_{ijd} x_{ijd} + \sum_{i\in C} \sum_{d\in\mathcal{D}_i} w_{id}.$$

- Constraints (38) are removed.
- Constraints (39) are changed as follows:

$$b_{id} + (\hat{t}_{ijd} + T)x_{ijd} + w_{id} \le b_{jd} + T \quad d \in \mathcal{D} \quad i,j \in \mathcal{N}_d \cap C : i \ne j.$$

- The following set of constraints is added:

$$w_{id} \ge b_{jd} - b_{id} - \hat{t}_{ijd} - T + Tx_{ijd} \quad d \in \mathcal{D} \quad i,j \in \mathcal{N}_d \cap C : i \ne j.$$

6.4.3. Computational Results.

To study the impact of the described relaxations on the performance of CCG and on solution costs, Table 3 compares the results of CCG on the instances of data set D interpreted as instances of the ConVRP, the ConVRP-SST, and the ConVRP-SSTW, respectively. The results are again

aggregated according to the number of customers ($|\mathcal{N}|$) and the presence/absence of route-duration constraints (T = yes/no). For each group, we report the number of instances solved to optimality (Opt) and average values (considering only the instances that are solved to optimality for every problem variant) for $|\overline{\Omega}|$ and t_{noUB}. In addition, for the ConVRP-SST and the ConVRP-SSTW, we report the average gap between the final upper bound obtained and the upper bound for the ConVRP (Δz). Because we use the same initial upper bound for all variants, the values $|\Phi|$, $|\hat{\Omega}'|$, and $|\hat{\Omega}''|$ reported in Table 1 are the same in the three variants considered and, therefore, are not reported.

For both ConVRP-SST and ConVRP-SSTW, the number of instances solved to optimality increases: only one instance with 20 customers and route-duration constraints and five instances with 30 customers and route-duration constraints remain unsolved. Furthermore, $|\overline{\Omega}|$ is reduced by a factor of two to five. Allowing flexible departure times at the depot leads to cost savings between roughly –0.3% and –0.9%; the largest savings are achieved when the route duration is not restricted. Waiting at customers provides only minor benefits for one of the groups. Detailed results can be found in Tables EC.6 and EC.7 of the e-companion of this paper.

6.5. Computational Results of LNS-ATCI

In this section, we investigate the performance of LNS-ATCI as a stand-alone method. First, we show how the components of LNS-ATCI contribute to solution quality and runtimes. In Table 4, we report the results obtained when solving data set B with (1) the described method using a total of η_{total} = 25,000 iterations (LNS-ATCI–25k), (2) LNS-ATCI–25k without using the set-partitioning formulation (w/o SP), and (3) LNS-ATCI–25k without the ATC improvement procedure (w/o ATCI). For each variant, we report the average percentage gap of the best solution found in 10 runs to the previous best-known solution (BKS) as Δz_b, the average gap of the average solution value of the 10 runs to the BKS (Δz_a), and the average computation time in seconds (t). We find that the ATC improvement procedure has a strong positive impact on the solution quality and robustness of our method. In comparison, the set-partitioning component only slightly improves the two measures. In the following experiments, we put maximum emphasis on solution quality and robustness and, thus, keep the ATC and SP components. Clearly, in other applications, it could be more beneficial to remove the SP component to achieve faster runtime.

Second, we compare LNS-ATCI to the approaches from the literature on data set B; for the sake of conciseness, we limit the comparison with the two best-performing approaches, that is, the template-based ALNS of Kovacs, Parragh, and Hartl (2014) (denoted as KPH) and the LNS method of Kovacs et al. (2015) (denoted as KGHP). Third, we study the performance of LNS-ATCI in comparison with KPH on data set C (no results are available for KGHP) and assess the influence of different service frequencies $\mathcal{D}_x$ and maximum allowed time differences L_x on the comparison.

Table 5 shows the results for data set B. Two different versions of LNS-ATCI are studied: LNS-ATCI–25k using a total of η_{total} = iterations, and LNS-ATCI–5k using a reduced number of iterations η_{total} = 5,000. For each instance, we report the name and the previous BKS. For each solution method, we report Δz_b, Δz_a, and t as introduced. In addition, the best solution that we found during the overall testing of our method and its gap to the BKS are reported in columns LNS–ATCI. For each instance, the best solution found by any of the tested methods is marked in bold.

The performance of LNS-ATCI on data set B is very convincing. LNS-ATCI–25k improves the previous BKS on 10 out of the 12 instances (for two of the instances, the improvement is above 2%) and matches it on the remaining two. The average improvement based on the best run is nearly 1%, and even the average of the runs shows a negative gap of –0.3% to the previous BKS. Concerning the comparison of the runtimes of the different solution methods, we think that a relatively fair comparison is possible because all algorithms were tested on modern desktop computers with processors of similar speed (Intel Xeon X5550 at 2.67 GHz for KPH and KGHP). The runtime of LNS-ATCI–25k is approximately five times the runtimes of the comparison methods; however, the runtime stays below eight minutes for all instances, which we deem very reasonable for a multiperiod problem with up to 199 customers and five periods from a practical perspective.

The fast variant of our algorithm, LNS-ATCI–5k, has roughly the same runtimes as the comparison methods

Table 3. Comparison of Results on the ConVRP, the ConVRP-SST, and the ConVRP-SSTW on Data Set D

		ConVRP			ConVRP-SST				ConVRP-SSTW			
$\lvert\mathcal{N}\rvert$	T	Opt	$\lvert\overline{\Omega}\rvert$	t_{noUB}	Opt	Δz	$\lvert\overline{\Omega}\rvert$	t_{noUB}	Opt	Δz	$\lvert\overline{\Omega}\rvert$	t_{noUB}
20	Yes	36/36	11	6.6	36/36	–0.56	3	0.6	36/36	–0.56	3	0.6
20	No	33/36	15	382.9	35/36	–0.89	3	124.5	35/36	–0.92	3	26.3
30	Yes	35/36	11	76.5	36/36	–0.29	5	63.8	36/36	–0.29	5	64.3
30	No	26/36	15	317.8	31/36	–0.62	4	269.3	31/36	–0.62	4	275.3

Table 4. Contribution of Heuristic Components Shown on Data Set B

	LNS-ATCI-25k			Without SP			Without ATCI		
	Δz_b	Δz_a	t	Δz_b	Δz_a	t	Δz_b	Δz_a	t
Average	−0.9	−0.3	217.3	−0.8	0.2	129.5	0.3	2.3	151.3

but is able to improve on the previous BKS for nine of the 12 instances, matches it on two, and yields a gap of 0.1% on one instance. On average, LNS-ATCI–5k still shows a negative gap to the previous BKS of −0.6%. Finally, during the overall testing, we find new BKS for all instances with an average gap to the previous BKS of −1.0%. On three instances, we obtain significant improvements of the solution quality with gaps above −2.3%.

Table 6 shows the results for data set C. Only averages over 10 runs are reported by KGHP, so we perform the same number of runs and conduct the comparison based on averages: Δz_a reports the percentage gap between the average objective value of LNS-ATCI–25k and that of KPH; that is, $\Delta z_a = (z_a(\text{LNS-ATCI–25k}) - z_a(\text{KPH}))/z_a(\text{KPH})$. Moreover, we report the percentage gap of the average maximum arrival-time difference between any two visits to a customer in column $\Delta b_a^{\max}$; that is, for every run, we memorize the maximum arrival time difference that occurs for any of the customers in the best solution obtained during that run; then, we average these values over all runs and calculate the gap in percentage to the corresponding value reported for KPH. To provide comparison values for future researchers, we additionally provide the best objective function value obtained during 10 runs in column z_b.

We find that the solution quality of LNS-ATCI–25k is clearly superior to that of KPH: the average gap is negative for 133 of 144 instances with an average improvement of −12.4%. The largest gaps are obtained for small values of L_x; depending on the service frequency, the gaps lie between −44.2% and −32.2% for $L_{0.4}$. With regards to the maximum arrival-time difference $b_a^{\max}$, we observe a nonnegative gap between LNS-ATCI and KPH for all groups. This suggests that our method is able to better utilize the maximum allowed time difference to find high-quality solutions.

Summarizing, the results indicate that, contrary to the template-based approach of KPH, LNS-ATCI is also suitable for low values of L. As can be expected, the difference between the two methods is smaller for a high service frequency of $\mathcal{D}_x$ because instances in which days resemble each other with regards to the customers that have to be served are beneficial for the template concept.

7. Summary and Conclusion

In this paper, we address the ConVRP and present the first exact solution method and a heuristic that represent the new state-of-the-art solution methods to solve the problem.

Unlike most of the state-of-the-art exact methods for VRPs that rely on route-based formulations, the proposed exact method is based on a formulation in which variables represent a set of customers (called a cluster) assigned to the same vehicle over the planning horizon. We first generate the entire set of clusters and then eliminate those clusters that cannot belong to any optimal ConVRP solution by computing gradually stronger lower bounds to the problem. The main idea of our algorithm is that the DC consistency is implied by the definition of the clusters, and the ATC is iteratively imposed on a small number of clusters only when necessary. The computational experiments show that, because of the strength of the computed lower bounds,

Table 5. Comparison of LNS-ATCI to the Best-Performing Approaches from the Literature: KPH (Kovacs, Parragh, and Hartl 2014) and KGHP (Kovacs et al. 2015) on Data Set B

		KPH			KGHP			LNS-ATCI–25k			LNS-ATCI–5k			$\overline{\text{LNS–ATCI}}$	
Inst.	BKS	Δz_b	Δz_a	t	Δz_b	Δz_a	t	Δz_b	Δz_a	t	Δz_b	Δz_a	t	z	Δz
1_50_0.7	2,124.21	0.0	3.3	5.5	0.0	0.4	15.1	0.0	0.2	40.5	0.0	0.9	9.2	**2,121.84**	**−0.1**
2_75_0.7	3,540.80	1.7	1.8	14.7	0.0	1.4	18.8	−1.3	−0.9	86.9	−1.0	0.2	16.9	**3,481.72**	**−1.7**
3_100_0.7	3,280.47	1.4	1.8	25.6	0.0	0.9	40.2	**−0.1**	0.5	195.3	0.1	1.5	33.6	**3,278.36**	**−0.1**
4_149_0,7	4,473.31	1.9	2.8	84.3	0.0	1.9	62.7	−1.4	−0.2	369.7	−1.4	1.3	80.1	**4,355.47**	**−2.6**
5_199_0.7	5,632.22	0.6	0.9	122.2	0.0	0.8	87.3	−2.6	−0.9	477.6	−0.3	0.7	97.7	**5,480.00**	**−2.7**
6_49_0.7	**4,051.48**	**0.0**	0.0	6.6	0.5	0.6	14.6	**0.0**	0.0	31.7	**0.0**	0.1	6.9	**4,051.48**	**0.0**
7_75_0.7	6,673.61	1.5	2.0	18.3	0.0	0.6	19.7	**−0.4**	−0.4	73.6	**−0.4**	0.5	16.4	**6,645.05**	**−0.4**
8_100_0.7	7,126.29	0.0	0.9	32.2	0.0	1.0	31.3	**−0.5**	−0.1	145.5	−0.4	0.4	30.1	**7,094.05**	**−0.5**
9_150_0.7	10,381.90	0.0	0.7	97.4	0.1	0.6	50.2	−0.5	−0.1	367.6	−0.6	0.1	66.8	**10,318.99**	**−0.6**
10_198_0.7	12,955.10	1.1	2.2	146.3	0.0	0.7	78.7	**−0.9**	0.1	467.0	−0.2	0.3	83.9	**12,839.78**	**−0.9**
11_119_0.7	4,471.22	0.3	0.3	36.0	0.0	2.6	83.6	−0.5	0.5	227.1	−0.3	3.7	119.1	**4,447.45**	**−0.5**
12_100_0.7	3,497.93	0.0	0.0	25.6	0.7	2.5	27.4	**−2.3**	−2.0	125.9	−2.1	0.3	27.2	**3,416.08**	**−2.3**
Average		0.7	1.4	51.2	0.1	1.2	44.1	−0.9	−0.3	217.3	−0.6	0.8	49.0		−1.0

Note. Best solutions are indicated in bold.

Table 6. Comparison of LNS-ATCI–25k to KPH (Kovacs, Parragh, and Hartl 2014) on Data Set C

	L_∞			$L_{0.8}$			$L_{0.6}$			$L_{0.4}$		
Inst.	z_b	Δz_a	$\Delta b_a^{\max}$	z_b	Δz_a	$\Delta b_a^{\max}$	z_b	Δz_a	$\Delta b_a^{\max}$	z_b	Δz_a	$\Delta b_a^{\max}$
$\mathcal{D}_{0.5}$												
1_50_0.5	1,616.37	−2.0	56.7	1,628.76	−2.4	−4.0	1,641.51	−6.0	4.0	1,696.33	−7.6	1.5
2_75_0.5	2,554.83	−0.7	66.7	2,554.94	−1.2	9.4	2,563.74	−1.1	−2.2	2,590.12	−27.5	6.1
3_100_0.5	2,632.43	−1.7	158.6	2,632.96	−1.8	6.4	2,658.19	−1.5	0.9	2,716.13	−36.7	2.1
4_149_0.5	3,317.49	−1.7	103.2	3,333.52	−1.3	2.1	3,337.12	−9.5	−3.3	3,366.54	−75.7	7.6
5_199_0.5	3,986.56	−0.2	164.0	3,988.21	−1.5	1.9	3,994.09	−2.8	−0.6	4,098.44	−79.7	27.4
6_49_0.5	2,863.55	−0.2	65.8	2,872.94	−0.4	3.5	2,889.42	−1.8	2.3	2,943.77	−22.8	5.5
7_75_0.5	4,632.31	−1.3	42.8	4,637.52	−1.2	1.9	4,642.50	−3.4	−3.1	4,662.84	−44.1	16.2
8_100_0.5	5,332.55	−0.3	109.0	5,335.32	−0.2	0.8	5,342.04	−0.5	−1.1	5,384.56	−45.4	3.4
9_150_0.5	7,347.40	−1.1	70.5	7,352.24	−1.5	2.8	7,354.43	−4.5	0.7	7,402.12	−52.7	4.9
10_198_0.5	9,267.06	−0.5	99.7	9,238.96	−0.8	−0.3	9,363.47	−52.9	1.8	9,576.16	−60.2	2.3
11_119_0.5	3,245.08	−1.3	208.2	3,253.59	−3.5	36.2	3,256.72	−3.6	−1.7	3,258.40	−7.0	−1.4
12_100_0.5	2,835.65	−1.4	339.4	2,845.27	−2.0	8.4	2,847.79	−2.0	2.8	2,895.90	−71.5	27.6
Average		−1.0	123.7		−1.5	5.8		−7.5	0.0		−44.2	8.6
$\mathcal{D}_{0.7}$												
1_50_0.7	2,105.39	−0.5	183.2	2,110.59	−0.5	−9.2	2,118.97	−1.0	2.2	2,137.65	−29.0	6.1
2_75_0.7	3,481.82	−2.4	113.9	3,481.72	−1.8	−1.3	3,513.81	−3.4	−0.3	3,543.00	−55.2	2.0
3_100_0.7	3,266.77	−1.0	220.7	3,272.50	−1.1	2.0	3,283.23	−0.6	0.9	3,325.92	−66.8	7.1
4_149_0,7	4,346.38	−2.5	249.8	4,408.77	−4.1	0.1	4,479.40	−26.1	1.8	4,640.36	−81.9	−22.0
5_199_0.7	5,464.52	−2.8	217.0	5,488.70	−2.2	9.9	5,525.68	−2.7	5.6	5,571.35	−13.9	0.6
6_49_0.7	4,048.96	−0.1	129.0	4,051.48	0.0	−2.7	4,062.70	0.0	0.1	4,102.95	−5.9	−1.5
7_75_0.7	6,645.05	−2.1	77.3	6,645.95	−2.6	5.1	6,658.68	−3.7	−4.1	6,676.41	−35.4	2.1
8_100_0.7	7,092.22	−1.4	195.4	7,097.27	−1.4	3.6	7,125.73	−7.6	0.4	7,321.12	−55.3	2.5
9_150_0.7	10,316.71	−1.5	58.4	10,327.97	−1.4	0.4	10,339.81	−3.0	−1.3	10,373.95	−47.7	1.4
10_198_0.7	12,827.08	−2.1	117.4	12,909.74	−2.0	−1.8	12,912.85	−17.8	−0.4	13,200.08	−60.3	4.0
11_119_0.7	4,443.76	−0.7	843.5	4,450.56	3.3	−2.2	4,458.83	−10.6	0.9	4,950.48	−86.9	39.2
12_100_0.7	3,408.55	−3.0	325.5	3,416.08	−2.9	2.6	3,418.03	−.5	0.4	3,489.39	−80.7	10.4
Average		−1.7	227.6		−1.4	0.6		−6.6	0.5		−51.6	4.3
$\mathcal{D}_{0.9}$												
1_50_0.9	2,478.84	−0.3	128.9	2,488.27	−0.1	−8.8	2,493.14	−0.1	4.1	2,507.37	−21.1	−4.5
2_75_0.9	4,001.08	−0.9	232.2	4,003.68	−0.6	−2.4	4,007.29	−1.8	10.2	4,044.71	−8.9	7.7
3_100_0.9	3,974.74	−0.4	282.9	3,988.08	0.1	−0.4	4,001.64	−0.3	−3.1	4,039.43	−5.8	2.6
4_149_0.9	4,942.23	−0.8	373.6	4,971.81	−0.1	1.7	4,929.50	−0.2	3.8	5,108.54	−79.8	20.1
5_199_0.9	6,376.09	−0.6	371.0	6,399.05	−3.5	10.1	6,397.04	−7.0	−7.3	6,453.19	−76.1	45.0
6_49_0.9	4,751.79	−0.2	111.8	4,761.17	0.0	7.2	4,768.31	−1.0	0.6	4,877.14	−18.9	−0.8
7_75_0.9	7,705.73	−0.5	53.5	7,706.18	−0.4	5.2	7,706.18	−0.3	−6.2	7,718.94	−2.8	1.3
8_100_0.9	8,733.72	0.0	196.3	8,673.73	0.0	−3.2	8,776.89	−0.6	2.1	8,835.97	−29.3	3.9
9_150_0.9	12,377.60	−0.5	203.5	12,391.43	−0.4	−1.1	12,442.23	−7.4	0.2	12,618.92	−58.6	5.5
10_198_0.9	15,820.63	−0.4	321.9	15,824.39	−0.4	0.9	15,828.35	−11.0	0.8	16,212.92	−65.6	2.5
11_119_0.9	4,986.96	7.7	866.3	4,975.01	9.0	−7.2	5,452.57	9.9	−6.0	5,496.98	−16.3	−1.3
12_100_0.9	4,011.73	0.9	524.4	4,013.50	0.8	−2.3	4,014.73	0.3	35.8	4,024.41	−3.4	3.5
Average		0.3	305.5		0.4	0.0		−1.6	2.9		−32.2	7.1

the proposed exact method can solve, within reasonable amounts of computing times, instances with up to 30 customers and a five-day planning horizon. The performance of the method is not affected by the service frequency in the instances, and the method is particularly effective if ATC constraints are not extremely tight. Furthermore, we describe how the method can be adapted to solve problem variants that allow flexible departure times at the depot and waiting at customers.

In addition, we present an LNS that is used in combination with the exact method to find optimal ConVRP solutions and as a stand-alone heuristic to find high-quality solutions in short runtimes. The method embeds (1) a suitable penalty mechanism to deal with infeasible solutions, (2) a repair procedure to improve the ATC, and (3) the solution of a set-partitioning problem to enhance solution quality. The computational experiments show that our LNS is able to clearly improve the solution quality compared with previously published heuristics on benchmark instances from the literature, especially if ATC constraints are tight.

Acknowledgments

The authors are grateful to the associate editor and the anonymous referees for their insightful comments that contributed to improve the quality of the paper.

References

Adulyasak Y, Cordeau JF, Jans R (2014) Optimization-based adaptive large neighborhood search for the production routing problem. *Transportation Sci.* 48(1):20–45.

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

Christofides N, Mingozzi A, Toth P (1979) The vehicle routing problem. Christofides N, Mingozzi A, Toth P, Sandi C, eds. *Combinatorial Optimization* (Wiley, Chichester, UK), 315–338.

Clarke G, Wright J (1964) Scheduling of vehicles from a central depot to a number of delivery points. *Oper. Res.* 12(4):568–581.

Contardo C, Martinelli R (2014) A new exact algorithm for the multi-depot vehicle routing problem under capacity and route length constraints. *Discrete Optim.* 12:129–146.

Cordeau JF, Laporte G (2003) A tabu search heuristic for the static multi-vehicle dial-a-ride problem. *Transportation Res. Methodology* 37(6):579–594.

Cordeau JF, Laporte G, Mercier A (2001) A unified tabu search heuristic for vehicle routing problems with time windows. *J. Oper. Res. Soc.* 52(8):928–936.

Dabia S, Ropke S, van Woensel T, Kok TD (2013) Branch and price for the time-dependent vehicle routing problem with time windows. *Transportation Sci.* 47(3):380–396.

Dayarian I, Crainic TG, Gendreau M, Rei W (2015) A branch-and-price approach for a multi-period vehicle routing problem. *Comput. Oper. Res.* 55:167–184.

Dayarian I, Crainic TG, Gendreau M, Rei W (2016) An adaptive large-neighborhood search heuristic for a multi-period vehicle routing problem. *Transportation Res. Logist. Transportation Rev.* 95:95–123.

Feillet D, Garaix T, Lehuédé F, Péton O, Quadri D (2014) A new consistent vehicle routing problem for the transportation of people with disabilities. *Networks* 63(3):211–224.

Goeke D, Schneider M (2015) Routing a mixed fleet of electric and conventional vehicles. *Eur. J. Oper. Res.* 245(1):81–99.

Groër C, Golden B, Wasil E (2009) The consistent vehicle routing problem. *Manufacturing Service Oper. Management* 11(4):630–643.

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

Kirkpatrick S, Gelatt CD, Vecchi MP (1983) Optimization by simulated annealing. *Science* 220(4598):671–680.

Kovacs AA, Parragh SN, Hartl RF (2014) A template-based adaptive large neighborhood search for the consistent vehicle routing problem. *Networks* 63(1):60–81.

Kovacs AA, Parragh SN, Hartl RF (2015) The multi-objective generalized consistent vehicle routing problem. *Eur. J. Oper. Res.* 247(2):441–458.

Kovacs AA, Golden BL, Hartl RF, Parragh SN (2015) The generalized consistent vehicle routing problem. *Transportation Sci.* 49(4):796–816.

Kovacs AA, Hartl RF, Parragh SN, Golden BL (2014) Vehicle routing problems in which consistency considerations are important: A survey. *Networks* 64(3):192–213.

Lian K, Milburn AB, Rardin RL (2016) An improved multi-directional local search algorithm for the multi-objective consistent vehicle routing problem. *IIE Trans.* 48(10):975–992.

Masson R, Lehuédé F, Péton O (2013) An adaptive large neighborhood search for the pickup and delivery problem with transfers. *Transportation Sci.* 47(3):344–355.

Pecin D, Contardo C, Desaulniers G, Uchoa E (2017a) New enhancements for the exact solution of the vehicle routing problem with time windows. *INFORMS J. Comput.* 29(3):489–502.

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

Ropke S, Pisinger D (2006a) An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Sci.* 40(4):455–472.

Ropke S, Pisinger D (2006b) A unified heuristic for a large class of vehicle routing problems with backhauls. *Eur. J. Oper. Res.* 171(3):750–775.

Sarstedt M, Mooi E (2014) Cluster analysis. *A Concise Guide to Market Research: The Process, Data, and Methods Using IBM SPSS Statistics* (Springer, Berlin, Heidelberg), 273–324.

Shaw P (1998) Using constraint programming and local search methods to solve vehicle routing problems. Maher M, Puget JF, eds. *Principles and Practice of Constraint Programming*, Lecture Notes in Computer Science, vol. 1520 (Springer, London), 417–431.

Subramanian A, Uchoa E, Ochi LS (2013) A hybrid algorithm for a class of vehicle routing problems. *Comput. Oper. Res.* 40(10):2519–2531.

Subramanyam A, Gounaris CE (2016) A branch-and-cut framework for the consistent traveling salesman problem. *Eur. J. Oper. Res.* 248(2):384–395.

Subramanyam A, Gounaris CE (2017) A decomposition algorithm for the consistent traveling salesman problem with vehicle idling. *Transportation Sci.* 52(2):386–401.

Sungur I, Ren Y, Ordóñez F, Dessouky M, Zhong H (2010) A model and algorithm for the courier delivery problem with uncertainty. *Transportation Sci.* 44(2):193–205.

Tarantilis C, Stavropoulou F, Repoussis P (2012) A template-based tabu search algorithm for the consistent vehicle routing problem. *Expert Systems Appl.* 39(4):4233–4239.