



Transportation Science

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

Slack Induction by String Removals for Vehicle Routing Problems

Jan Christiaens, Greet Vanden Berghe

To cite this article:

Jan Christiaens, Greet Vanden Berghe (2020) Slack Induction by String Removals for Vehicle Routing Problems. Transportation Science

Published online in Articles in Advance 15 Jan 2020

. <https://doi.org/10.1287/trsc.2019.0914>

Full terms and conditions of use: <https://pubsonline.informs.org/Publications/Librarians-Portal/PubsOnLine-Terms-and-Conditions>

This article may be used only for the purposes of research, teaching, and/or private study. Commercial use or systematic downloading (by robots or other automatic processes) is prohibited without explicit Publisher approval, unless otherwise noted. For more information, contact permissions@informs.org.

The Publisher does not warrant or guarantee the article's accuracy, completeness, merchantability, fitness for a particular purpose, or non-infringement. Descriptions of, or references to, products or publications, or inclusion of an advertisement in this article, neither constitutes nor implies a guarantee, endorsement, or support of claims made of that product, publication, or service.

Copyright © 2020, INFORMS

Please scroll down for article—it is on subsequent pages




With 12,500 members from nearly 90 countries, INFORMS is the largest international association of operations research (O.R.) and analytics professionals and students. INFORMS provides unique networking and learning opportunities for individual professionals, and organizations of all types and sizes, to better understand and use O.R. and analytics tools and methods to transform strategic visions and achieve better outcomes.

For more information on INFORMS, its publications, membership, or meetings visit <http://www.informs.org>

Slack Induction by String Removals for Vehicle Routing Problems

Jan Christiaens,^a Greet Vanden Berghe^a

^a CODeS, imec, Department of Computer Science, KU Leuven, 9000 Gent, Belgium

Contact: jan.christiaens@cs.kuleuven.be,  <https://orcid.org/0000-0003-1204-904X> (JC); greet.vandenbergh@cs.kuleuven.be,

 <https://orcid.org/0000-0002-0275-5568> (GVB)

Received: June 20, 2017

Revised: May 7, 2018; December 21, 2018

Accepted: February 5, 2019

Published Online in Articles in Advance:
January 15, 2020

<https://doi.org/10.1287/trsc.2019.0914>

Copyright: © 2020 INFORMS

Abstract. Dedicated algorithm and modeling improvements continue to advance the state of the art with respect to vehicle routing problems (VRPs). Despite these academic achievements, solving large VRP instances sufficiently fast for real-life applicability remains challenging. By exploiting VRP solution characteristics in an effective manner, this paper arrives at a powerful and fast optimization heuristic. Its primary contributions are threefold: a ruin method, a recreate method, and a fleet minimization procedure. The ruin method functions via adjacent string removal, introducing with it a novel property regarding vehicle routing problems that we term *spatial slack*, whereas the recreate method is categorized as greedy insertion with blinks. Combining these results in slack induction by string removals (SISRs), a powerful ruin and recreate approach. The fleet minimization procedure, meanwhile, introduces an absences-based acceptance criterion that serves as a complementary optimization component for when minimizing the number of vehicles constitutes the primary VRP objective. Together these three elements provide a suite of simple, powerful, and easily reproducible algorithmic methods that are successfully applied not only to the capacitated VRP but also to a wide range of related problems such as pickup and delivery problems and others that include time windows. SISRs serves to strip back the layers of complexity and specialization synonymous with the trend of algorithmic development throughout recent decades. Moreover, such simplicity and reproducibility are shown to not necessarily come at the expense of solution quality, with SISRs consistently outperforming alternative general approaches as well as dedicated single-purpose methods. Finally, aside from performance-related criteria, SISRs also serves to showcase a fresh perspective with respect to VRPs more generally, introducing a range of new terminology and procedures that, it is hoped, will invigorate further research and innovation.

Funding: This work was partially supported by the Institute for the Promotion of Innovation through Science and Technology in Flanders (IWT-Vlaanderen) in cooperation with Conundra [Grant 130855].

Supplemental Material: The online appendix is available at <https://doi.org/10.1287/trsc.2019.0914>.

Keywords: vehicle routing • large-scale instances • string removal • spatial slack • capacity slack • fleet minimization • ruin and recreate

1. Introduction

The capacitated vehicle routing problem (CVRP) was introduced by Dantzig and Ramser (1959) as the truck dispatching problem, a generalization of the traveling salesman problem (TSP). Despite multiple decades of research (Laporte 2009), the CVRP remains a highly active field of research, and it continues to represent a significant computational challenge. Research concerning the problem mostly falls into one of two categories: exact algorithms or heuristics.

Developing exact algorithms is an interesting activity in itself, while also providing objective function bounds that enable accurate heuristic quality assessment. Toth and Vigo (2002) provide a comprehensive overview of exact algorithms to solve CVRPs with either symmetric or asymmetric distance matrices. They primarily focus on branch-and-bound algorithms and review a set of original powerful approaches. The

branch-and-cut-and-price algorithm by Fukasawa et al. (2006) improved on previous branch-and-cut algorithms. Several improvements such as that by Baldacci, Christofides, and Mingozzi (2008) have been combined with new elements (Pecin et al. 2014) and have proved capable of solving CVRPs of up to 350 customers and even 650 customers in some cases. Computation times may, however, be as long as five days, which makes such solvers impractical for many real-world purposes.

Heuristics, by contrast, are characterized as fast and adaptable methods. Absolute optimality is, however, not ensured—a necessary trade-off in terms of accommodating many real-world situations that rely on the presence of timely solutions. Heuristics generally employ specific neighborhoods within a metaheuristic to gradually improve candidate solutions. Prior to the latest improvements, classical neighborhoods such as 2-opt* (Potvin and Rousseau 1995) and CROSS exchange

(Taillard et al. 1997) have been explored in single-solution-based heuristics. Larger neighborhoods were employed by Shaw (1998) and Pisinger and Røpke (2007) in a ruin and recreate (R&R) framework. Prins (2004) contributed a hybrid genetic algorithm, the first population-based framework for vehicle routing problems (VRPs), which improved significantly on the prior approaches. This approach was refined by Vidal et al. (2015), thereby realizing the state-of-the-art VRP heuristic.

The vast majority of VRPs consider minimizing the total travel distance as their objective. However, a much smaller number of variants require a hierarchical objective to be handled, which is primarily the minimization of the number of vehicles and secondarily the minimization of the total distance. In such problems, solutions associated with smaller fleet sizes are preferred over solutions whose total travel distance is smaller but that require additional vehicles. Although this objective might appear artificial, it reflects the attitude of real-world transportation companies who far prefer to be able to accommodate all requests with their own fleet rather than having to outsource certain requests to other companies—the cost of which is prohibitively high. Solution methods that address this hierarchical objective are generally two-stage approaches wherein the number of vehicles is minimized in the first stage and the distance in the second stage, such as the methods by Pisinger and Røpke (2007) and Nagata, Bräysy, and Dullaert (2010). Hybrid methods, for example, by Vidal et al. (2013), are less common but also possible.

Although the initial ambition of heuristics is noble, they have gradually become increasingly complicated. This paper seeks to remedy this by introducing a low-level yet simultaneously powerful and fast heuristic that is sufficiently adaptable and may be easily incorporated into any current or future VRP approach. This ruin and recreate approach's improvements to the CVRP state of the art and its not only successful but also competitive application to a host of related VRPs will be demonstrated and documented.

Structurally, this paper begins by first offering a detailed problem definition of the CVRP, followed by an overview of the other VRPs addressed during experimentation. Next, Section 3 provides an introduction to ruin and recreate, whereas Section 4 provides a brief description of simulated annealing (SA), the metaheuristic employed for guiding the local search. Section 5 introduces slack induction by string removals (SISRs), this paper's novel R&R contribution. Following this, the new absences-based acceptance criterion for fleet minimization is detailed in Section 6. Parameters associated with this work are detailed in Section 7. Computational results for the CVRP and a variety of associated problems are provided in Section 8, and the

paper ultimately ends with a conclusion that summarizes the paper's contribution while also delineating the scope and possibilities for future research.

2. Problem Description

The CVRP primarily considered by this paper is defined as follows. Let $\mathcal{G} = \{\mathcal{V}, \mathcal{A}\}$ be a complete undirected graph in which \mathcal{V} is the set of vertices and \mathcal{A} the set of arcs connecting the vertices. The vertices $v_i \in \mathcal{V}$ for $i \in \{0, \dots, n\}$ represent locations in a two-dimensional space where v_0 corresponds to the *depot* and the other n vertices to *customers* having a demand q_i . Each arc (i, j) in $\mathcal{A} = \{(i, j) : i, j \in \mathcal{V}, i \neq j\}$ is associated with a *cost* c_{ij} . An unlimited homogeneous fleet of vehicles with capacity Q is situated at the depot. The CVRP consists of designing vehicle *tours* at minimum cost such that each customer is served exactly once by a single vehicle, and each tour starts at the depot v_0 , serves customers without exceeding the vehicle's capacity Q , and finally ends at the same depot v_0 .

Although the CVRP constitutes the primary problem addressed by this paper, the general applicability of SISRs is demonstrated by way of solving several other associated problems within the domain of vehicle routing. Key references for papers that provide descriptions of individual problems are supplied in Table 1.

The remainder of this paper refers to a *string* as a sequence of consecutive locations in a tour. Strings may contain at most all customers served by the tour. The number of customers included in a string is referred to as its *cardinality*, with tour t 's cardinality denoted by $|t|$.

3. Ruin and Recreate (R&R) Strategies

Classical neighborhoods for VRP heuristics—such as, 2-opt (Croes 1958); 2-opt* (Potvin and Rousseau 1995); k -opt (Lin 1965); Or-opt (Or 1976); exchange, cross, and relocate (Savelsbergh 1988); and CROSS exchange (Taillard et al. 1997)—are all defined via slight modifications to the incumbent solution. They therefore exhibit limited neighborhood size complexity, implying that a full neighborhood $\mathcal{N}(s)$ of a solution s may be evaluated within a reasonable amount of computational time. Very large neighborhoods often occur when a significant part of the solution is modified, making a full neighborhood exploration impractical. Therefore, neighbors may be selected from a reduced candidate set $\mathcal{C}(s) \subseteq \mathcal{N}(s)$, or a single neighbor may be generated by first ruining the current solution before recreating it into a feasible one. Such a method may be defined by a *ruin method* \mathcal{R}^- and *recreate method* \mathcal{R}^+ . This underlying concept is present in a variety of research papers. To our knowledge, it was established for the first time by Dees and Smith (1981) in their *rip-up and reroute* strategies for wiring point-to-point

Table 1. CVRP and Associated Problems

CVRP

The capacitated VRP, also known as the classical VRP, was introduced by Dantzig and Ramser (1959) as the truck dispatching problem. The algorithms by Subramanian, Uchoa, and Ochi (2013) and Vidal et al. (2014) are considered as the current best approaches.

Multidepot VRP

Vehicles are available at a fixed number of depots. Each vehicle is assumed to return to the depot where its tour starts. This problem was introduced by Gillett and Johnson (1976). The algorithms by Pisinger and Røpke (2007); Subramanian, Uchoa, and Ochi (2013); and Vidal et al. (2012) are considered as the current best approaches.

VRP with backhauls (VRPB)

The vehicle routing problem with backhauls, introduced by Goetschalckx and Jacobs-Blecha (1989), considers serving two sets of customers: delivery (line haul) and pickup (backhaul) customers. Each tour must serve at least one delivery customer, and all delivery customers must be served before any pickup customers are served. A tour's total delivery and pickup demand should not exceed its vehicle's capacity. The algorithms by Gajpal and Abad (2009), Zachariadis and Kiranoudis (2012), and Vidal et al. (2014) represent the best solution approaches for this problem.

VRP with mixed delivery and pickup (VRPMDP)

This problem, introduced by Min (1989), relaxes the VRPB by enabling any order of delivery and pickup customers within a tour. This implies that a vehicle's used capacity may increase and decrease during the tour, but should never exceed the vehicle's capacity. Best solutions are obtained by Subramanian, Uchoa, and Ochi (2013) and Vidal et al. (2014).

VRP with simultaneous delivery (VRPSDP)

Unlike the VRPMDP this problem, also introduced by Min (1989), does not require customers to be associated exclusively with either delivery or pickup demand. This implies that a single customer can have both delivery and pickup requests. Best solutions are obtained by the same authors as for the VRPMDP.

Multidepot VRP with mixed delivery and pickup

Min (1989) generalizes the VRPSDP by addition of multiple depots into the multidepot VRPMDP. Best approaches are those of Røpke and Pisinger (2006b) and Subramanian, Uchoa, and Ochi (2013).

Open VRP

In contrast to the CVRP, vehicles must not return to the depot after serving the last customer in the open VRP, introduced by Sariklis and Powell (2000). This reflects situations in which customers are assigned to external carriers. Current best approaches are those proposed by Subramanian, Uchoa, and Ochi (2013) and Vidal et al. (2014).

Cumulative CVRP

The cumulative CVRP (Ngueveu, Prins, and Wolfler Calvo 2010) requires, similar to "minavg routing" (Campbell, Vandenbussche, and Hermann 2008), the sum of arrival times at customers to be minimized rather than the total traveled distance. The approach of Vidal et al. (2014) obtained the current best known solutions for this problem.

VRPTW

The VRP with time windows, introduced sometime in the late 1970s or early 1980s, extends the CVRP with temporal constraints. Customers are associated with individual time windows within which they prefer to be served. Vehicles start their tour in the depot and are assumed to arrive back at the depot without exceeding the maximum tour duration as defined in Gehring and Homberger (1999). The current best approaches are those proposed by Repoussis, Tarantilis, and Ioannou (2009); Nagata, Bräysy, and Dullaert (2010); and Vidal et al. (2013).

PDPTW

Literature on the pickup and delivery problem also dates back to the same time frame as the VRPTW. As defined by Dumas, Desrosiers, and Soumis (1991), goods require transportation between pickup and delivery locations. Similar to the VRPTW, temporal constraints are present. The PDPTW should not be confused with delivery and pickup problems where goods are transported to or from the depot, rather than between the customers themselves. Røpke and Pisinger (2006a), Nagata and Kobayashi (2010), and Curtois et al. (2018) have contributed the current best solution approaches.

Dial-a-ride problem (DARP)

In the dial-a-ride problem, which was first examined by Wilson et al. (1971), transportation of customers between specified pickup and delivery locations are considered. The most recent definition of the DARP (Cordeau and Laporte 2003) is in line with the work of Jaw et al. (1986). The DARP restricts the PDPTW by imposing a maximum ride time in order to manage user inconvenience. Best approaches are those of Chassaing, Duhamel, and Lacomme (2016); Masmoudi et al. (2017); and Gschwind and Drexler (2019).

connections in electronic design automation. Shaw (1998) introduced *large neighborhood search* (LNS), wherein the ruin phase is implemented as the removal of related customers (related in terms of time, distance, and being served by the same vehicle). A branch-and-bound technique optimally inserts the removed customers in the recreate phase. Shaw (1998) applied LNS to both the CVRP and the vehicle routing problem with time windows (VRPTW). The term *ruin and recreate* was introduced by Schrimpf et al. (2000, p. 139), who applied the technique to a number of prominent problems, including the TSP and VRPTW. They ruined solutions by removing randomly selected customers, customers within a certain radius, or one single string. The solution is recreated by greedily reinserting the removed customers in a random order at minimum cost. Røpke and Pisinger (2006a) defined several \mathcal{R}^- and several \mathcal{R}^+ methods that compete to modify the solution in their adaptive large neighborhood search (ALNS) framework and applied their framework to the pickup and delivery problem with time windows (PDPTW). They implemented seven different strategies for selecting customers for removal: random, worst, related, cluster, time oriented, historical node pair, and historical request pair. The solution is recreated by employing either greedy or regret insertion (Potvin and Rousseau 1993) with or without a noise function. Pisinger and Røpke (2007) applied ALNS to the rich pickup and delivery problem with time windows.

Recent heuristic development based on the R&R principle builds on the general ALNS framework by enlarging the set of \mathcal{R}^- and \mathcal{R}^+ methods. Examples of such studies include, but are not limited to, the papers by Laporte, Musmanno, and Vocaturo (2010) and Ribeiro and Laporte (2012). In distinct contrast to the recent trend of introducing more and more \mathcal{R}^- and \mathcal{R}^+ methods, this paper introduces a streamlined R&R implementation: slack induction by string removals (SISRs; phonetically, “scissors”). This proposed approach employs a single \mathcal{R}^- method and a single \mathcal{R}^+ method: adjacent string removal and greedy insertion with blinks, respectively, with the significant additional benefit of being highly reproducible. An overview of these R&R methodological implementations is provided in Table 2.

4. Local Search Metaheuristic

The SISRs neighborhood search, detailed in Section 5, is guided by simulated annealing. This well-known metaheuristic, introduced by Kirkpatrick, Gelatt, and Vecchi (1983), is based on the statistical model concerning energy changes in annealing systems (Metropolis et al. 1953). In this study, the system’s initial temperature \mathcal{T}_0 is reduced by the exponential cooling schedule (Equation (1)) to its final temperature \mathcal{T}_f in f iterations. Cooling constant c is obtained by way of Equation (2):

$$\mathcal{T}_{k+1} = c \mathcal{T}_k \quad 0 \leq k < f, \quad (1)$$

$$\text{where} \quad c = \left(\frac{\mathcal{T}_f}{\mathcal{T}_0} \right)^{1/f} \quad \mathcal{T}_0 > \mathcal{T}_f > 0. \quad (2)$$

The probability h of accepting a neighbor solution s^* depends on the current temperature \mathcal{T} and the difference in energy $\Delta E = \text{dist}(s^*) - \text{dist}(s)$ relative to the current solution s (Equation (3)). Neighbor solution s^* is accepted if a random selected value from the continuous uniform distribution $U(0, 1)$ is less than the probability $h(\Delta E, \mathcal{T})$. Therefore, the acceptance criterion is expressed by way of Equation (4):

$$h(\Delta E, \mathcal{T}) = \exp(-\Delta E / \mathcal{T}), \quad (3)$$

$$\text{dist}(s^*) < \text{dist}(s) - \mathcal{T} \ln(U(0, 1)). \quad (4)$$

A solution s is represented by two sets $s = \{T, A\}$. Set T is the set of tours, each of which serves at least one customer, whereas set A is the set of absent customers, all of whom are not served by any tour $t \in T$. The initial solution contains an equal number of tours as there are customers defined in the problem; therefore, only one customer is served in each tour. This initial solution is passed to the local search procedure, detailed by way of Algorithm 1. First, the initial solution is saved as the best solution s^{best} found thus far (line 2). Temperature \mathcal{T} is set to the initial temperature \mathcal{T}_0 (Line 3). Next, f improvement iterations are performed (lines 4–10). During each iteration, a neighbor solution s^* is generated from current solution s by employing SISRs’ R&R procedure (line 5). This neighbor is accepted as the new current solution s if SA’s acceptance criterion is satisfied (lines 6 and 7) and saved if

Table 2. Overview of R&R Methodologies

Method	\mathcal{R}^-	\mathcal{R}^+
Shaw (1998) (LNS)	related removal	optimal insertion
Schrimpf et al. (2000)	random, radial, or string removal	greedy insertion
Pisinger and Røpke (2007) (ALNS)	random, worst, related, cluster, time-oriented, or historical removal	greedy or regret insertion with or without noise function
This study’s approach (SISRs)	adjacent string removal	greedy insertion with blinks

it improves on the best solution (lines 8 and 9). Finally, the system is cooled down (line 10) before proceeding onto the next iteration.

Algorithm 1 (Local Search Metaheuristic)

Input: Initial solution $s = \{T, A\}$, where T is the set of tours and A the set of absent customers

```

1: procedure LOCAL-SEARCH( $s$ )
2:    $s^{best} \leftarrow s$ 
3:    $\mathcal{T} \leftarrow \mathcal{T}_0$ 
4:   for  $f$  iterations do
5:      $s^* \leftarrow \text{SISRs-RUIN-RECREATE}(s)$ 
6:     if  $\text{dist}(s^*) < \text{dist}(s) - \mathcal{T} \ln(U(0, 1))$  then
7:        $s \leftarrow s^*$ 
8:     if  $\text{dist}(s^*) < \text{dist}(s^{best})$  then
9:        $s^{best} \leftarrow s^*$ 
10:     $\mathcal{T} \leftarrow c \mathcal{T}$ 
11: end procedure

```

5. Slack Induction by String Removals

This section provides a detailed description of the SISRs method. The concepts of *capacity slack* and *spatial slack* and three premises are introduced in Section 5.1, which lays out the rationale for the proposed method. Detailed descriptions of SISRs' \mathcal{R}^- and \mathcal{R}^+ methods may be found in Sections 5.2 and 5.3, respectively.

5.1. Rationale

When customers are removed from a tour's sequence, several types of slack may be introduced. First, the most apparent type of slack may be *capacity slack*, referring to the vehicle's capacity that is set free. Second, *spatial slack* may be identified if one imagines that the vehicle's travel distance is limited. Given this limited distance, the vehicle's reachable area may be defined. This area appears as an elliptical region around each edge, where each edge's end points are the focus points for its corresponding ellipse. This area increases when the locations of the removed customers are no longer visited, and we refer to this increase as *spatial slack*. Although the reduced travel distance contributes the most to gaining spatial slack, it is not a necessary condition. For example, removing the intermediate of three collinear locations does not reduce the travel distance, but the reachable area (spatial slack) does increase given the larger focal distance. Although the vehicle's travel distance is not limited within the CVRP definition, one may express spatial slack relative to the vehicle's original tour as the additional reachable area delimited by its original travel distance.

An effective \mathcal{R}^- method should introduce both types of slack into the solution such that the possibilities for improving the solution are greatly enhanced. The following premises serve as the basis for such an \mathcal{R}^- method. Each premise addresses the

potential failures of alternative \mathcal{R}^- methods within the context of slack induction.

Premise 1 (Remove a "Sufficient" Number of Customers). Removing a small number of customers (such as in Figure 1(a₁)) may fail to introduce sufficient capacity slack in the vehicles for serving customers of higher demand. Although it is visually clear (Figure 1(a₂)) by which vehicles the removed customers should be served, it may be impossible if the vehicles gained insufficient capacity slack.

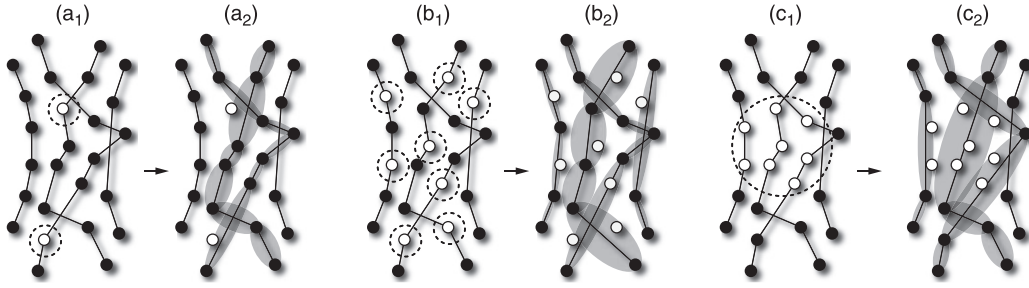
Premise 2 (Remove "Adjacent" Customers). Removing customers who are not geographically close (not adjacent) to each other (such as randomly selected customers in Figure 1(b₁)) generally introduces capacity and spatial slack scattered across many tours (Figure 1(b₂)). Such slack is unlikely to be beneficial when recreating the solution. The recreated solution may be comparable to the result of multiple R&R iterations wherein a small number of customers is removed each time.

Premise 3 (Remove Strings of Customers). Removing customers of which their selection is only based on adjacency (as in the case of radial removal, Figure 1(c₁)) may often fail to capture complete detours that remain present in the ruined state (Figure 1(c₂)). This may be remedied by increasing the radius; however, many more customers than strictly necessary may be removed as well. Therefore, removing strings may be much more effective.

If one can remove multiple strings that are geographically close to each other—removal of adjacent strings—a very effective yet simple of origin \mathcal{R}^- method may be obtained. An example of such adjacent string removals is depicted in Figure 2. Only two adjacent strings are selected for removal in Figure 2(a), and the ruined state of the solution is illustrated in Figure 2(b). Notice that (1) spatial slack is greatly introduced in only two tours, and (2) the spatial slacks of both overlap significantly. Ideally, the ruined state obtained by the \mathcal{R}^- method (in Figure 2(b)) is recreated by \mathcal{R}^+ in the solution illustrated in Figure 2(c).

5.2. Ruin Method—Adjacent String Removal

The proposed \mathcal{R}^- method is based on Premises 1 to 3 as follows. The number of removed customers is managed by means of the average number of removed customers \bar{c} . In addition, the \mathcal{R}^- method may be set for removing many strings of small cardinality or few strings of high cardinality by means of the maximum cardinality of the removed strings L^{max} . Let s be the solution being ruined. A valid number of removed strings k_s and cardinality l_t for every string (each of which are removed from a different tour t) is selected as follows. The maximum string cardinality l_s^{max} is initially set to the minimum of L^{max} and the solution's average tour cardinality $|\bar{t} \in \bar{T}|$ (Equation (5)) before

Figure 1. Examples of Ruin Methods That Introduce Insufficient Slack

Notes. The figure shows the removal of too few customers (panels (a₁) and (a₂)), randomly selected customers (panels (b₁) and (b₂)), and radially selected customers (panels (c₁) and (c₂)). Panels (a₁), (b₁), and (c₁) illustrate the partial solution, wherein the customers selected for removal are colored white and outlined by dashed lines. Panels (a₂), (b₂), and (c₂) illustrate the ruined state, wherein spatial slack relative to the original state is shaded gray.

the maximum number of strings k_s^{max} is derived (Equation (6)):

$$l_s^{max} = \min\{L^{max}, \lceil |t \in T| \rceil\} \quad l_s^{max} \in \mathbb{R}^+, \quad (5)$$

$$k_s^{max} = \frac{4\bar{c}}{1 + l_s^{max}} - 1 \quad k_s^{max} \in \mathbb{R}^+, \quad (6)$$

$$k_s = \lfloor U(1, k_s^{max} + 1) \rfloor \quad k_s \in \mathbb{N}^+. \quad (7)$$

The number of strings k_s to be removed is obtained by rounding down a randomly selected real value from the continuous uniform distribution $U(1, k_s^{max} + 1)$ by way of Equation (7). The maximum cardinality of each string is limited by the initial limit l_s^{max} and the tour's cardinality $|t|$ from which the string is to be removed (Equation (8)). Finally, the cardinality for the string to be removed from tour t is obtained by rounding down a real value selected from $U(l_t^{min}, l_t^{max} + 1)$ (Equation (9)):

$$l_t^{max} = \min\{|t|, l_s^{max}\} \quad l_t^{max} \in \mathbb{R}^+, \quad (8)$$

$$l_t = \lfloor U(1, l_t^{max} + 1) \rfloor \quad l_t \in \mathbb{N}^+. \quad (9)$$

Before the procedural steps of the ruin phase \mathcal{R}^- are detailed by way of Algorithm 2, two assumptions are

made. First, for each customer c , an adjacency list $adj(c)$ is assumed to be available for identifying adjacent strings. The list contains all customers ordered by increasing distance from c , which includes c as its first element. Second, each string being removed originates from a different tour such that spatial slack, as well as capacity slack, is spread across a number of tours, controlled by k_s . Therefore, a tour may be ruined at most once, and removing k_s strings implies ruining k_s tours.

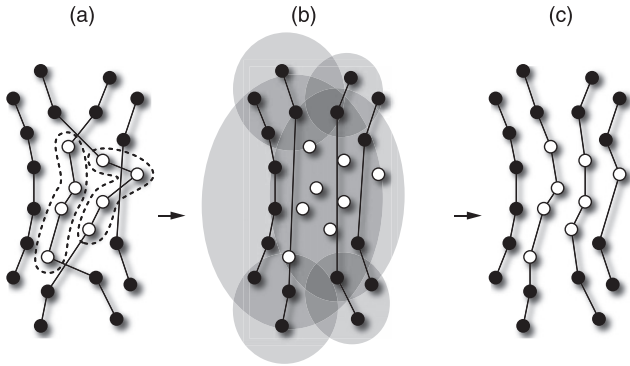
Algorithm 2 (SISRs' Ruin Method)

Input: Solution $s = \{T, A\}$, where T is the set of tours and A the set of absent customers

```

1: procedure RUIN( $s$ )
2:    $l_s^{max}, k_s^{max}, k_s \leftarrow \text{calculate}(\text{Equations}(5), (6), \text{and } (7)).$ 
3:    $c_s^{seed} \leftarrow \text{randomCustomer}(s)$ 
4:    $R \leftarrow \emptyset$ 
5:   for  $c \in adj(c_s^{seed})$  and  $|R| < k_s$  do
6:     if  $c \notin A$  and  $t \notin R$  then
7:        $c_t^* \leftarrow c$ 
8:        $l_t^{max}, l_t \leftarrow \text{calculate}(\text{Equations } (8) \text{ and } (9)).$ 
9:        $A \leftarrow A \cup \text{removeSelected}(t, l_t, c_t^*)$ 
10:       $R \leftarrow R \cup \{t\}$ 
11: end procedure

```

Figure 2. An Example of Adjacent String Removals

Note. The figure shows (a) selection of two adjacent strings, (b) a ruined state with significant spatial slack induced, and (c) an improved recreated state.

The procedural steps of the ruin phase \mathcal{R}^- are given by Algorithm 2. First, values for l_s^{max} , k_s^{max} , and k are derived by Equations (1)–(3) (line 2). A randomly selected customer serves as the seed customer, c_s^{seed} , from which string removals are initiated (line 3). The auxiliary set R , for storing ruined tours, is initially empty (line 4). The adjacency list from c_s^{seed} is iterated over from the first customer to the last until set R contains k_s ruined tours, indicating that k_s strings have been removed (lines 5–10). In each iteration, if the customer c has not been removed ($c \notin A$) and its serving tour t has not been ruined ($t \notin R$; line 6), a string is removed from this tour (lines 7–10). Such a customer is special in the sense that it is the closest customer to c_s^{seed} out of all customers served by t , denoted by c_t^* (line 7). The maximum cardinality l_t^{max}

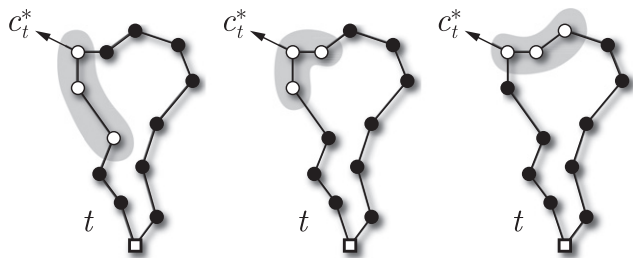
and final cardinality l_t of the string to be removed from tour t are obtained by way of Equations (4) and (5) (line 8). The string is removed via the “string” or “split string” procedure based on c_t^* and l_t . In either case, l_t customers are selected and removed from t before adding them to set A (line 9), and the ruined tour is added to set R (line 10).

The string procedure removes a randomly selected string of cardinality l_t in tour t , which includes customer c_t^* . By removing a string that includes c_t^* , other customers inside this string are also likely to be located close to c_{seed} . Therefore, every call to this procedure will result in a string removal of customers close to the seed customer or, simply, strings adjacent to each other. An example where $l_t = 3$ is illustrated in Figure 3. One of the three possible strings is randomly selected for removal.

The split string procedure begins much like the string procedure, by randomly selecting a string of cardinality $l + m$ that includes customer c_t^* (see Figure 4(a)). However, the ruin phase bypasses and preserves a random substring of m intervening customers, as shown in Figure 4(b). The number of preserved customers m is determined as follows. Initially $m = 1$, and its current value is maintained either if a random number $U(0, 1)$ is larger than β , which is referred to as the *split depth*, or once the maximum value for m is reached ($m = m^{max} = |t| - l$). If neither of these conditions is satisfied, m is incremented ($m = m + 1$), and the process repeats.

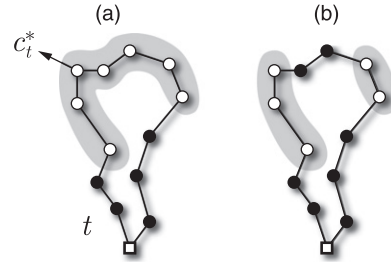
The probability of executing the split string is referred to as the *split rate* α . Examples are illustrated in Figure 5 for string (panel (a)) and split string (panels (b₁) and (b₂)) removals with small and large m -values. Customers being removed are colored white. The string procedure removes a string, introducing spatial slack (see Figure 5(a)). In the split string removal procedure, a small value for m is obtained with a very low probability. When this occurs, customers close to c_t^* are removed while remaining spatially bonded to the same regions as the preserved customers (see Figure 5(b₁)). Otherwise, $m = m^{max}$ is obtained with

Figure 3. Example of String Removal



Note. There are three possible string selections, which include c_t^* from tour t when $l_t = 3$.

Figure 4. Example of Split String Removal



Note. The figure shows (a) random selection of a string when $l = 5$ and $m = 2$ and (b) preserving a substring of $m = 2$ customers.

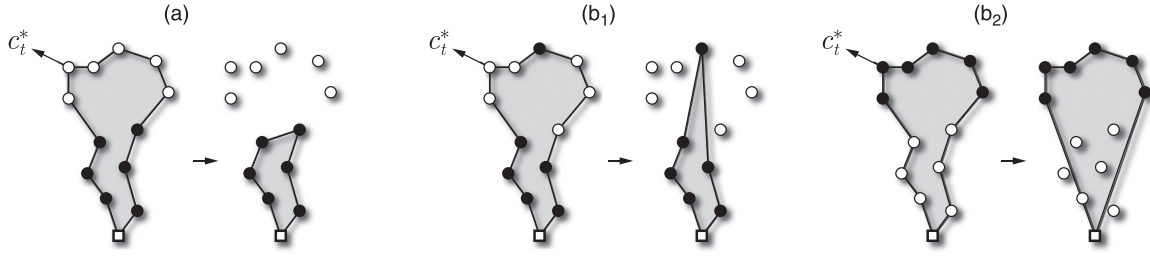
much greater probability, indicating that the customers removed are close to the depot (see Figure 5(b₂)).

Figure 6 presents an example of the ruin phase where two strings ($k_s = 2$) are removed from solution s . First, the seed customer c_s^{seed} is randomly selected (see Figure 6(a)). Following this, the list $adj(c_s^{seed})$ is iterated over. Because c_s^{seed} is always the first element in the list and no tours are ruined, $c_{t1}^* = c_s^{seed}$. A string of length $l_{t1} = 4$ is removed by the string procedure, which includes c_{t1}^* (see Figure 6, (b) and (c)). In Figure 6(d), $adj(c_s^{seed})$ is iterated over until the next customer (c_{t2}^*) is found in a not-yet-ruined tour. The second string is also removed by the string method, and l_{t2} is, coincidentally, again 4 (see Figure 6(e)). The final ruined state is illustrated in Figure 6(f).

5.3. Recreate Method—Greedy Insertion with Blinks

Although basic greedy methods insert each customer sequentially at the best position, one may deviate slightly from the best position. Pisinger and Røpke (2007) apply a *greedy insertion with noise function* by adding a randomized noise term to the insertion cost. This paper, by contrast, introduces *greedy insertion with blinks* (Algorithm 3).

First, the set of absent customers A is sorted according to one of the following orders: random, demand, far, or close (line 2). *Random* enables the set’s insertion without any ordering. *Demand* sorts customers by demand, placing those with the largest demand first. *Far* inserts the most distant customers from the depot first. Finally, *close* inserts customers closest to the depot first. Set A is sorted by random, demand, far, and close by weights equal to four, four, two, and one, respectively. Each customer $c \in A$ (line 3) is inserted into solution s at the “best” position P as follows. Initially, P is set to NULL, indicating that no position for c is selected (line 4). All current tours that are part of the solution ($t \in T$) are iterated over in a random order (line 5). When a tour t has sufficient capacity slack to serve c , all positions P_t inside this tour are iterated over (line 6). Each position is evaluated with a probability of $1 - \gamma$, otherwise skipping the position as if the algorithm blinks, and therefore, $\gamma = 0$ represents the blink rate

Figure 5. Examples of String and Split String Removals

Note. The figure shows (a) string removal and split string removal when m equals the (b₁) lowest or (b₂) highest possible value.

(line 7). If a position P_t is found for which the cost of inserting c is lower than the current best position P , P_t becomes the new best position (lines 8 and 9). If no position is found in existing tours, a new empty tour is created to serve c (lines 10–12). Finally, c is inserted at P and removed from the set of absent customers (lines 13 and 14). Blink rate $\gamma = 0$ denotes that customers are always inserted at the best position, a strategy proven to be suboptimal after experimentation. This may appear somewhat counterintuitive given that it is more logical to assume that consistently inserting each customer at the best possible position would result in the highest-quality solution.

Algorithm 3 (SISRs' Recreate Method)

Input: Ruined solution $s = \{T, A\}$ where T is the set of tours and A the set of absent customers

```

1: procedure RECREATE( $s$ )
2:    $sort(A)$ 
3:   for  $c \in A$  do
4:      $P \leftarrow \text{NULL}$ 
5:     for  $t \in T$  (which can serve  $c$ ) do
6:       for  $P_t$  in  $t$  do
7:         if  $U(0,1) < 1 - \gamma$  then
8:           if  $P = \text{NULL}$  or  $costAt(P_t) < costAt(P)$  then
9:              $P \leftarrow P_t$ 
10:    if  $P = \text{NULL}$  then
11:       $T \leftarrow T \cup \{\text{new tour } t\}$ 
12:       $P \leftarrow \text{position in } t$ 
13:    insert  $c$  at  $P$ 
14:     $A \leftarrow A \setminus \{c\}$ 
15: end procedure

```

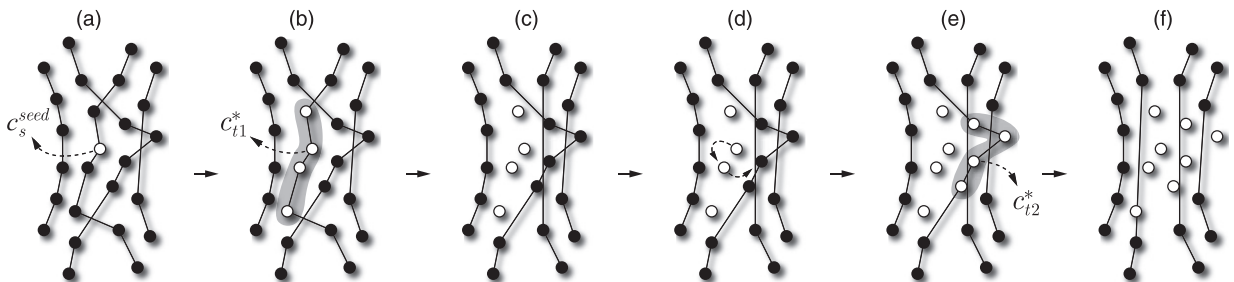
Of all feasible options for inserting a customer, one may make a random choice or use a heuristic to pick the best option (greedy insertion). The effect of blinking is reflected by the probability p of selecting a specific option based on its rank r . A blink rate of γ implies that the best option, which is ranked first ($r = 1$), is selected with probability $p(1) = (1 - \gamma)$. If the best-ranked option is blinked over, the probability of selecting the second-best option is $p(2) = (1 - \gamma)\gamma$, and the third-ranked option equals $p(3) = (1 - \gamma)\gamma^2$. The selection probability for each rank is expressed by the exponential function

$$p(r) = (1 - \gamma)\gamma^{(r-1)} \quad r \in \{1, \dots, \infty\}. \quad (10)$$

Notice how the blinking algorithm itself is unaware of each option's rank; it blinks only with probability γ while iterating over all options. Thus, blinking results in rank-based selection probabilities without requiring one to rank the options first, in contrast to *heuristic-biased stochastic sampling* (HBSS; Bresina 1996), a closely related selection algorithm. A comparison between HBSS and the blinking algorithm can be found in Online Appendix A.

6. Fleet Minimization

This paper introduces a two-stage approach to accommodate a hierarchical objective that is primarily the minimization of the number of vehicles. The newly developed absences-based neighbor acceptance criterion is employed during the vehicle minimization stage, detailed by way of Algorithm 4. The solution that serves all customers with the lowest number of vehicles is

Figure 6. An Example of SISRs' Ruin Method

represented by s^{best} (line 2). For each customer c , the number of solutions where c was not served by any tour (absent from the solution) is tracked by its associated absence counter abs_c . This counter is initially set to 0 for all customers (lines 3 and 4). In each iteration, the following steps are performed. First, a neighbor solution s^* is obtained by SISRs' \mathcal{R}^- and \mathcal{R}^+ procedures (line 6). This neighbor is accepted if (1) the number of absent customers decreased or (2) the sum of absence counters associated with the absent customers $sumAbs(A) = \sum_{c \in A} (abs_c)$ decreased (lines 7 and 8). If the current solution is feasible ($A^* = \emptyset$), a copy is saved as s^{best} . Afterward, the tour, whose served customers represent the lowest sum of absence counters, is removed from the current solution (lines 9–11). Finally, each absent customer in the neighbor solution has its associated absence counter incremented (lines 12 and 13).

Algorithm 4 (Fleet Minimization)

Input: Solution $s = \{T, A\}$ where T is the set of tours and A the set of absent customers

```

1: procedure FLEET-MINIMIZATION( $s$ )
2:    $s^{best} \leftarrow s$ 
3:   for  $c$  defined in the problem do
4:      $abs_c \leftarrow 0$ 
5:   while minimizing tours do
6:      $s^* \leftarrow$  SISRS-RUIN-RECREATE( $s$ )
7:     if  $|A^*| < |A|$  or  $sumAbs(A^*) < sumAbs(A)$  then
8:        $s \leftarrow s^*$ 
9:     if  $A^* = \emptyset$  then
10:       $s^{best} \leftarrow s^*$ 
11:     remove  $t \in T$  with the lowest  $sumAbs(t)$ 
12:     for  $c \in A^*$  do
13:        $abs_c \leftarrow abs_c + 1$ 
14: end procedure
    
```

In essence, neighbor s^* is accepted only if the number of absent customers $|A^*|$ decreased or if the sum of absence counters associated with the absent customers $sumAbs(A^*)$ decreased. This absences-based acceptance criterion for minimizing fleet size proved to be very efficient without complicating the overall method. In fact, it requires only the replacement of SA's distance-based acceptance criterion by this absences-based acceptance criterion during the fleet minimization phase.

7. Parameters

Parameter variations for both the \mathcal{R}^- and \mathcal{R}^+ methods are evaluated for the CVRP, VRPTW, and PDPTW. At most, two parameters are varied simultaneously to determine their influence on the algorithm's performance while the values of all other parameters remain fixed. These predefined values, presented in Table 3, were observed as high performing while developing

the proposed algorithm and are therefore employed throughout the experiments reported in Section 8.

7.1. Parameters for Distance Minimization

The benchmark sets employed throughout this analysis were introduced by Uchoa et al. (2017) for the CVRP, Gehring and Homberger (2001) for the VRPTW, and Li and Lim (2001) for the PDPTW. Whereas results for the CVRP can be directly compared by the solution's distance, results for the VRPTW and PDPTW are not directly comparable when the number of utilized vehicles differs. Solutions employing fewer vehicles usually result in much longer travel distances. Therefore, the number of utilized vehicles is fixed to the most frequently obtained number per instance, thereby ensuring a fair comparison. For each evaluated parameter setting, the complete benchmark set is solved once. Results for all settings are compared by way of their average ranks, presented throughout Tables 4, 5, and 6. For each benchmark instance, every parameter setting is ranked by its obtained result with respect to those obtained by all other settings. The average rank for each setting is the average of all its obtained ranks. Low average ranks represent the best-performing settings and are colored dark gray. Meanwhile, the highest average ranks correspond to the worst-performing settings and are colored light gray.

Because maximum string cardinality L^{max} and the average number of removed customers \bar{c} both influence the maximum number of strings K^{max} , the values of both these parameters are varied simultaneously in order to determine which combination performs best. Given that average computational run time is proportional to \bar{c} , changes to this value will result in varying run times. This is compensated for by adjusting the number of iterations inversely proportional to \bar{c} . This enables a comparison of different settings for \bar{c} while maintaining equal computational effort or calculation time. Average ranks are presented in Table 4. Note that not all combinations are feasible. Given that at least one string is removed in each iteration, it is impossible to obtain an average number of removed customers less than $(1 + L^{max})/2$. For example, when L^{max} is set to 5, \bar{c} should be equal to $(1 + 5)/2 = 3$ or greater. Therefore, smaller values for \bar{c} such as 2 have not been considered. It is observed that for both the CVRP and PDPTW, the best parameter combinations are identical ($\bar{c} = 10$, $L^{max} = 10$), whereas slightly larger values ($\bar{c} = 15$ and $L^{max} = 20$) perform better when solving the VRPTW.

Next, the split rate α and split depth β are varied simultaneously. The average rank obtained by each combination is presented in Table 5. In general, the best average ranks are obtained when split rate α is set within the range 0.25–0.75 in combination with high values for split depth β such as 0.9 or 0.99. Whereas this

Table 3. Parameter Settings for SISRs and SA

SISRs' \mathcal{R}^-	
$\bar{c} = 10$	An average of 10 removed customers has proved a good ruin parameter. When \bar{c} is equal to 10, it implies the removal of at least one and at most 19 customers. Decreasing this parameter, for example, to five, was observed to be insufficient for diversification. By contrast, increasing \bar{c} , for example, to 20, results in extensively ruined solutions. Such solutions cannot easily be recreated into improving solutions and most likely result in highly divergent neighbors, which disables intensification of potentially good solutions.
$L^{max} = 10$	The cardinality of removed strings is limited to 10 because removing longer strings fails to improve final solution quality. This confirms the observations made by Fahrion and Wrede (1990), whose chain-exchange procedure limits string cardinality to half the average tour length. By employing Equation (7) the number of removed strings k_s will range from 1 to 3 for solutions with large tours ($ t \in T \geq L^{max}$) and 1 to 19 for solutions whose tours all have a cardinality of 1 ($ t \in T = 1$). This situation almost always only occurs during the first iteration of the search (see Section 4).
$\alpha = 0.5$	By setting the split rate to 50% ($\alpha = 0.5$), both string and split string procedures are executed with equal probability.
$\beta = 0.01$	This setting results in values for the number of preserved customers $m = 1, 2, 3, \dots, m^{max}$ having respective probabilities $p = 1.00\%, 0.99\%, 0.9801\%, \dots, (100 - p_{m=1} - p_{m=2} - p_{m=3} - \dots - p_{m=m^{max}-1})\%$. As such, it is very likely that customers removed within a split string removal are those at the start or end in the tour's sequence.
SISRs' \mathcal{R}^+	
$\gamma = 0.01$	It was observed that a blink rate of 1% ($\gamma = 0.01$) was the most effective in terms of final solution quality.
Fleet minimization	
$iterations = 10\%$ of all iterations	Generally the minimum number of vehicles is found within three minutes. Therefore, dedicating 10% of all iterations to the fleet minimization is considered to be sufficient.
SA	
$\mathcal{T}_0 = 100, \mathcal{T}_f = 1$ $iterations = it(v)$	The search begins at an initial temperature $\mathcal{T}_0 = 100$ and ends at the final temperature $\mathcal{T}_f = 1$. The number of iterations it is determined as a function of problem size v by linear interpolation. The minimum and maximum problem sizes are 100 and 1,000, respectively, as per Uchoa et al. (2017). The present study set $it(100) = 3 \times 10^7$ and $it(1,000) = 3 \times 10^8$, thus enabling the direct comparison of SISRs' calculation times against those from the aforementioned paper.

well-performing region is less pronounced for the PDPTW, it is distinctive for both the CVRP and the VRPTW.

Finally, blink rate γ is varied while all other parameters remain fixed. Average ranks obtained for the CVRP exhibit two peaks. Further analysis of these results showed that the peak at $\gamma = 0$ is introduced by results on the small instances, whereas the second peak at $\gamma = 0.05$ emerges because of those for

the large instances. For both the VRPTW and PDPTW, the best value tends toward $\gamma = 0.05$.

7.2. Parameters for Fleet Minimization

The performance of Algorithm 4 is evaluated on the 1,000-customer benchmark set of Gehring and Homberger (2001) by taking the best of five runs per instance. The cumulative number of vehicles (CNV) for each set is compared against the (best of five) CNV obtained

Table 4. Maximum String Cardinality and Average Number of Removed Locations

L^{max}	CVRP					VRPTW					PDPTW				
	\bar{c}					\bar{c}					\bar{c}				
	2	5	10	15	30	2	5	10	15	30	2	5	10	15	30
2	12.2	8.1	8.9	11.0	12.6	13.4	10.9	11.1	11.7	12.7	10.1	8.6	9.2	9.3	10.8
5		4.7	4.7	5.4	8.9		7.2	6.3	6.1	8.2		5.3	5.5	6.3	7.6
10			4.1	5.5	7.8			5.0	4.1	5.7			4.8	5.2	6.6
20				5.3	7.0				3.6	4.6				6.1	6.6
50					8.0					4.9					7.1

Notes. The average ranks are shaded in proportion to their values. Low average ranks are shaded dark gray and correspond to the best performing settings. Highest average ranks are shaded light gray and correspond to the worst performing settings.

Table 5. Split String Parameters α and β

β	CVRP					VRPTW					PDPTW				
	α					α					α				
	0.0	0.25	0.50	0.75	1.0	0.0	0.25	0.50	0.75	1.0	0.0	0.25	0.50	0.75	1.0
0.00	14.2	12.3	12.5	13.8	13.4	12.7	12.1	12.3	12.1	13.6	11.0	10.5	11.1	11.3	11.3
0.50	14.3	13.2	12.2	12.8	11.7	12.8	12.4	11.2	12.5	12.6	10.6	10.6	9.8	11.4	11.1
0.90	14.5	10.5	10.4	8.6	9.7	13.0	10.5	10.8	10.3	12.3	12.0	10.0	9.2	10.2	9.9
0.99	14.1	9.4	9.0	10.7	12.5	13.6	10.1	10.3	10.9	14.2	11.2	10.6	10.6	11.3	13.2
1.00	13.2	11.2	10.0	10.1	13.7	12.9	10.1	11.6	12.4	17.2	11.5	11.0	10.9	12.5	18.0

Notes. The average ranks are shaded in proportion to their values. Low average ranks are shaded dark gray and correspond to the best performing settings. Highest average ranks are shaded light gray and correspond to the worst performing settings.

Table 6. Blink Rate Parameter γ

	CVRP					VRPTW					PDPTW				
	γ					α					α				
	0.0	0.001	0.01	0.05	0.10	0.0	0.001	0.01	0.05	0.10	0.0	0.001	0.01	0.05	0.10
	2.57	2.76	2.87	2.53	3.12	2.91	2.89	2.75	2.72	3.06	3.00	2.79	2.72	2.43	2.63

Notes. The average ranks are shaded in proportion to their values. Low average ranks are shaded dark gray and correspond to the best performing settings. Highest average ranks are shaded light gray and correspond to the worst performing settings.

by EAMA(100) (Nagata, Bräysy, and Dullaert 2010) and HGSADC (Vidal et al. 2013) in Table 7. In addition, results of SISRs with different L^{max} settings are reported. The best results are marked dark gray, whereas results of SISRs are marked light gray when better than the competing algorithms but not the best of the different L^{max} settings. The average calculation time T is specified in minutes, which is at most 10 minutes for EAMA(100) and unknown for HGSADC.

These results indicate the best settings may vary for different sets. For example, removing more strings of lower cardinality ($L^{max} = 2$) enables us to find the best CNV for set C2, whereas removing strings of intermediate cardinality ($L^{max} = 5$) obtains a better CNV than EAMA(100) and HGSADC for set C1. However, after conducting a parameter analysis similar to that performed in Section 7.1, no superior regions were

observed. Consequently, the parameter settings presented in Table 3 are employed during the fleet minimization phase when solving the VRPTW and PDPTW.

8. Computational Results

SISRs' computational results are documented in this section. First, SISRs' results for the most relevant CVRP benchmark set are compared against those obtained by current state-of-the-art methods in Section 8.1. Following this, although the CVRP constitutes this work's core focus, SISRs' general applicability is verified by way of solving numerous CVRP variants. The very minor adjustments necessary to enable SISRs to solve each of these problems are detailed in Section 8.2.

8.1. CVRP

Uchoa et al. (2017, p. 845) introduced a new benchmark set because "[t]he existing sets suffer from at least one of the following drawbacks: (i) became too easy for current algorithms; (ii) are too artificial; (iii) are too homogeneous, not covering the wide range of characteristics found in real applications." This recent benchmark set contains 100 instances whose sizes range from 100 to 1,000 customers and that cover a wide variety of characteristics. Uchoa et al. (2017) report the detailed results obtained by two state-of-the-art heuristics, both conducted on a Xeon central processing unit (CPU) at 3.07 GHz: iterated local search with set partitioning (ILS-SP; Subramanian, Uchoa, and Ochi 2013) and unified hybrid genetic search (UHGS; Vidal et al. 2014).

Table 7. Fleet Minimization with Different L^{max} Settings

Set	n	SISRs				
		EAMA(100)	HGSADC	$L^{max} = 10$	$L^{max} = 5$	$L^{max} = 2$
		CNV (T)	CNV	CNV (T)	CNV (T)	CNV (T)
R1	1,000	919 (≤ 10)	919	919 (0.2)	919 (0.2)	919 (0.1)
R2	1,000	190 (≤ 10)	190	190 (0.0)	190 (0.0)	190 (0.0)
C1	1,000	941 (≤ 10)	941	939 (2.8)	938 (2.4)	939 (2.7)
C2	1,000	291 (≤ 10)	288	291 (9.1)	292 (3.7)	288 (2.5)
RC1	1,000	900 (≤ 10)	900	900 (0.1)	900 (0.0)	900 (0.0)
RC2	1,000	183 (≤ 10)	182	182 (0.3)	182 (0.3)	188 (0.4)

Notes. Results are ranked first (best), second, and third by way of dark gray, light gray, and no shading, respectively. T, Average calculation time in minutes.

Table 8. Benchmark Subsets

	n	Instances	No. of instances
Small	100–250	X-n101-k25–X-n247-k47	32
Medium	250–500	X-n251-k28–X-n491-k59	36
Large	500–1,000	X-n502-k39–X-n1001-k43	32
All	100–1,000	X-n101-k25–X-n1001-k43	100

This study's experiments were performed on a Xeon E5-2650 v2 CPU at 2.60 GHz and were compared against the results of ILS-SP and UHGS, which were taken from Uchoa et al. (2017). No reimplementations of these methods were therefore required. The full benchmark set was divided into three sets for comparison based on problem size n : small, medium, and large (Table 8).

The complete results obtained by each heuristic are documented in Online Appendix B, where Tables B1, B2, and B3 report the results for the small, medium, and large instances, respectively. The average distance ("Avg"), the best distance ("Best"), and the average calculation time in minutes ("Time") obtained after 50 runs per instance are presented in these tables. Heuristics are ranked first (best), second, and third by way of dark gray, gray, and light gray, respectively, for the columns "Avg," "Best," and "Time." The distance and number of vehicles of the best known solution (BKS) are also reported. Throughout the entire experimental campaign, 40 BKSs were improved by SISRs and 40 solutions of equal quality were found,

which are highlighted in dark and light gray, respectively, in the "New BKS" columns.

To provide a clear overview, average calculation times and gaps to the best-known solution in percentage terms ($100(\text{result} - \text{BKS})/\text{BKS}$) for the average and best results are summarized by way of the minimum, maximum, average, and median values in Table 9. These values were taken directly from the bottom of Tables B1, B2, and B3 in Online Appendix B.

UHGS outperforms the other methods for small instances. For the medium set, UHGS and SISRs perform equally well insofar as obtaining the best solutions ("Best" in Table 9). Additionally, it was observed that SISRs was the most robust method. Meanwhile, for the large set, SISRs outperforms both heuristics on all aspects—it is the most robust method that finds the best solutions in the least amount of computation time.

The average solution quality ("Avg" columns in Tables B1, B2, and B3 in Online Appendix B) expressed as gaps has also been visualized by way of box plots for each heuristic and the problem set in Figure 7. These box plots demonstrate SISRs' efficiency and robustness by exhibiting both a small median and spread of average gaps.

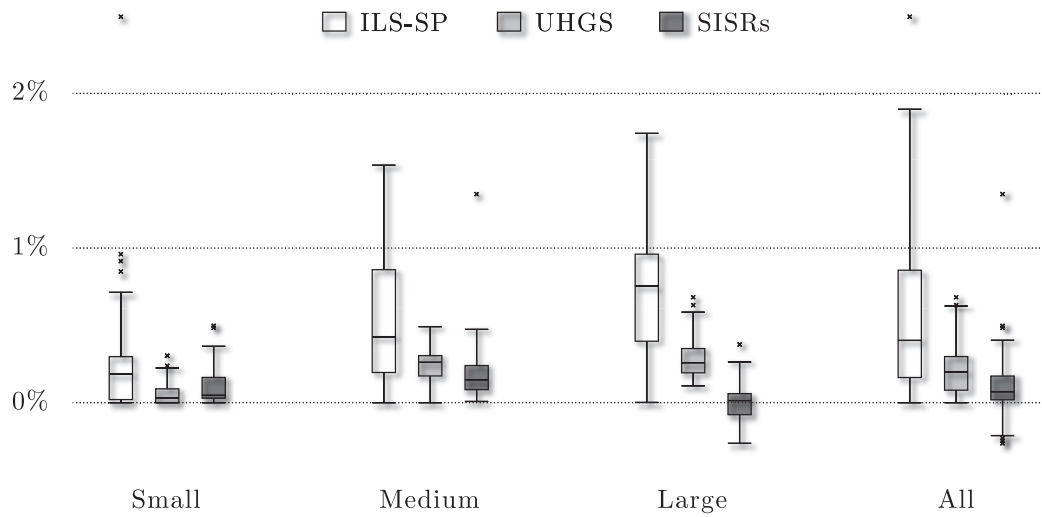
Finally, average solution quality is analyzed by conducting a one-tailed Wilcoxon signed-rank test (Wilcoxon 1945), which is a nonparametric test for comparing two independent groups and performed by the software environment R (R Core Team 2017). The only assumption required for this test is that observations are

Table 9. Summarized Computational Results

	ILS-SP			UHGS			SISRs		
	Avg	Best	Time	Avg	Best	Time	Avg	Best	Time
Small (100–250)									
Min	0.00	0.00	0.1	0.00	0.00	1.4	0.00	0.00	0.2
Max	2.50	1.19	17.8	0.30	0.06	20.4	0.50	0.16	18.4
Avg	0.31	0.12	2.4	0.07	0.00	6.0	0.11	0.01	5.2
Median	0.19	0.00	1.6	0.03	0.00	5.4	0.05	0.00	5.0
Medium (250–500)									
Min	0.00	0.00	2.0	0.00	0.00	6.5	0.01	–0.18	9.8
Max	1.54	0.73	60.6	0.49	0.10	86.7	1.35	0.28	58.4
Avg	0.54	0.17	23.1	0.24	0.02	30.3	0.20	0.02	27.0
Median	0.42	0.12	16.8	0.26	0.00	22.4	0.15	0.01	22.4
Large (500–1,000)									
Min	0.00	0.00	27.3	0.11	0.00	33.1	–0.26	–0.42	60.9
Max	1.74	1.45	792.8	0.68	0.33	560.8	0.38	0.15	412.7
Avg	0.76	0.49	195.7	0.29	0.04	268.6	–0.01	–0.13	152.0
Median	0.75	0.39	144.7	0.26	0.02	258.6	0.01	–0.14	144.8
All (100–1,000)									
Min	0.00	0.00	0.1	0.00	0.00	1.4	–0.26	–0.42	0.2
Max	2.50	1.45	792.8	0.68	0.33	560.8	1.35	0.28	412.7
Avg	0.53	0.26	71.7	0.20	0.02	98.8	0.10	–0.03	60.0
Median	0.40	0.11	17.7	0.20	0.00	22.4	0.07	0.00	22.4

Notes. Results are ranked first (best), second, and third place by use of the colors dark gray, gray, and light gray respectively. Time, average calculation time in minutes; Min, minimum; Max, maximum; Avg, average.

Figure 7. Gaps of the Average Results



Note. Gaps of the average results to the best-known solutions (Uchoa et al. 2017) are visualized by way of box plots.

independent. The following null hypothesis H_0 and alternative hypothesis H_1 are considered, where X represents ILS-SP or UHGS:

$$H_0: \text{Avg}(\text{SISRs}) = \text{Avg}(X);$$

$$H_1: \text{Avg}(\text{SISRs}) > \text{Avg}(X).$$

The average results obtained by SISRs are compared with the average results of ILS-SP and UHGS. Rejecting H_0 implies that the average results obtained by SISRs are different from those obtained by alternative heuristic X . Rejecting H_1 implies that the values of the average results obtained by SISRs are lower than those obtained by X or that SISRs' average solution quality is better than X 's. The obtained p -values are presented in Table 10.

Although the overall significance level α is set to 2.5%, Bonferroni correction (Dunn 1961) is applied in order to compensate for the family-wise error rate. Bonferroni correction sets the significance level for testing individual hypotheses to α/m , where m is the number of tests conducted using the same data. Because four tests are performed using the small, medium, and large data sets and four again using all data, a total of eight hypotheses are tested each time with the same data. Therefore p -values are assessed with respect to the Bonferroni-adjusted level $0.025/8 = 0.003125$, or 0.3125%. If the p -value is lower

than this level, the hypothesis is rejected, and the cell is colored gray. The null hypothesis H_0 and alternative hypothesis H_1 are both rejected when considering medium, large, and all benchmark sets, indicating that SISRs significantly differs from both other heuristics and that its average results are significantly better.

In addition to the performance analysis employing the benchmark set introduced by Uchoa et al. (2017), SISRs is applied to the classical benchmarks listed in Table 11. Detailed results obtained by SISRs with respect to these multiple benchmarks are documented in Online Appendix C. These experiments are performed only in the interests of completeness. Going forward, we highly recommend that further research use the most recent benchmark set of Uchoa et al. (2017). The reasons why immediately become clear on providing a brief comparative overview of the various benchmarks.

The composition of each set is summarized in Table 12: the number of instances (“#Inst”), a box plot that represents problem size distribution, the distribution of customers (“Cust”), and the cumulative distance (CD) obtained by SISRs. Customer distribution is categorized as random (R), clustered (C), real geographic coordinates (G), or artificially distributed along geometric shapes such as concentric circles or squares (A). All benchmark sets prior to R95 (Rochat and Taillard

Table 10. p -Values of the Wilcoxon Rank-Sum Test

	Small		Medium		Large		All	
	ILS-SP	UHGS	ILS-SP	UHGS	ILS-SP	UHGS	ILS-SP	UHGS
H_0	0.029239	0.022526	0.000013	0.002624	0.000000	0.000000	0.000000	0.000000
H_1	0.014620	0.989361	0.000007	0.001312	0.000000	0.000000	0.000000	0.000000

Note. p -Values are shaded gray when the corresponding hypothesis is rejected.

Table 11. CVRP Benchmark Sets

Acronym	Authors
C69	Christofides and Eilon (1969)
C79	Christofides, Mingozzi, and Toth (1979)
F94	Fisher (1994)
A95	Augerat et al. (1995)
R95	Rochat and Taillard (1995)
G98	Golden et al. (1998)
L05	Li, Golden, and Wasil (2005)
U17	Uchoa et al. (2017)

1995) contain very few instances and are of very small size. The first medium-sized instances were introduced in G98 (Golden et al. 1998), wherein customers are distributed along geometric shapes that are very unlikely to appear in real-world environments. L05 (Li, Golden, and Wasil 2005) contains large instances that are based on those of G98 with similarly unrealistic customer distributions. U17 (Uchoa et al. 2017) is the only benchmark set that exhibits a wide range of problem sizes with relevant customer distributions, vehicle capacities, and many other realistic characteristics. Interested readers are referred to Uchoa et al. (2017) for full details. SISRs' results for each benchmark are summarized by the average gap values ("Gap") in Table 12.

Average gaps obtained by SISRs are at most 0.13% for all benchmark sets except for G98 and L05, where average gaps were 0.44% and 0.82%, respectively. Note that initial calculation times for L05 were considerably longer than for other instances of comparable size. This is very likely due to the high number of customers per tour compared with all other

benchmarks. Therefore, the number of executed iterations was divided by 10 for this benchmark.

8.2. CVRP Variants

Various sets of benchmark instances from the literature were employed to evaluate SISRs' performance. Minimal adjustments were necessary to enable SISR to solve the CVRP variants addressed, and therefore, a structured overview for the adjustments related to each problem attribute is provided by way of Table 13. If more than one change was necessary, the changes are given in a numbered list. The computational results were then compared against state-of-the-art methods for each CVRP variant. A detailed breakdown of the results obtained by both SISRs and these state-of-the-art methods for each individual instance is presented in Online Appendix D. A causal consultation of these results immediately reveals SISRs' impressive dominance. Despite the various state-of-the-art methods it is compared against, being general frameworks or dedicated single-problem heuristics, SISRs proves highly competitive, very often matching or improving the best-known solutions across all variants. Even more impressively, for large-scale instances, which are more representative of real-world environments, SISRs produces many new best-known solutions. This is especially true for the PDPTW, for which SISRs improves more than half the considered benchmark set's instances (178 of 298, with 58 of these solutions using fewer vehicles). Full details concerning these improved solutions can be found in Table D11 in Online Appendix D, where improved solutions are colored dark gray and also marked with an asterisk

Table 12. Results for the Classical Benchmark Sets

Set	#Inst	Distribution of problem size n						Cust	Gap
		0	200	400	600	800	1000		
C69	13							R, G	0.00
C79 (M)	5							R, C	0.13
C79 (CMT)	14							R, C	0.05
F94	3							G	0.02
A95 (A)	27							R	0.00
A95 (B)	23							C	0.00
A95 (P)	23							R	0.00
R95	13							C, G	0.05
G98	20							A	0.44
L05	12							A	0.82
U17	100							R, C, RC	0.10

Note. Set, the acronym of the benchmark set; #Inst, the number of instances; Cust, the distribution of customers which can be categorized as random (R), clustered (C), real geographic coordinates (G), or artificially distributed (A); Gap, the average gap of 10 runs per instance obtained by SISRs.

Table 13. Adjustments Accommodating VRP Variants

Multidepot	
(1) (\mathcal{R}^+) While sorting customers by far from or close to the depot (see Section 5.3), the distance to the closest depot is employed. (2) (\mathcal{R}^+) When a new vehicle must be used to include customer c , the closest depot to c that has an unused vehicle is selected.	
Backhauls	
(1) (\mathcal{R}^-) If a tour contains only backhaul customers after a string is removed, all remaining customers are removed. (2) (SA) A solution associated with a higher number of tours that exclusively contain backhaul customers is never accepted.	
Mixed/simultaneous deliveries and pickups	
For each tour, location two additional values are maintained, which correspond to the maximum capacity consumption in all previous/following locations. This preprocessing requires $O(I)$ time and enables a capacity check in $O(1)$ time (\mathcal{R}^+).	
Open	
A dummy location to which all other locations have a distance equal to zero is introduced. This location represents the end depot in each tour.	
Cumulative	
(\mathcal{R}^+) The delta evaluation method, which is employed to select the best insertion position, is modified to represent the cumulative objective.	
Service time	
(\mathcal{R}^+) An additional ordering in terms of decreasing service time is employed when each customer has an individual service time.	
Time windows	
(1) Feasibility checks concerning time windows are based on the “push forward” global variables (Savelsbergh 1990), which require $O(I)$ time for preprocessing. These variables enable time window feasibility checks in $O(1)$ time. (2) (\mathcal{R}^+) In addition to the sorting orders introduced in Section 5.3 (random, demand, far, and close), customers are sorted with respect to increasing time window length, increasing time window start, and decreasing time window end.	
Maximum ride times	
The procedure for deriving the global variables concerning time windows is adapted for accommodating maximum ride times, ^a increasing its complexity to $O(I Q)$. In addition, this procedure is employed for deriving the maximum time span between one location and all successive location. Because it takes $ I $ executions for deriving the maximum time span between all pairs, preprocessing of a tour requires $O(I ^2Q)$ time, which again enables feasibility checks in $O(1)$. A similar approach was reported by Gschwind (2019).	
Pickup and deliveries	
(1) (\mathcal{R}^-) Strings are removed as described in Section 5.2. In addition, the associated pickup/delivery locations that were not included by the string are also removed. (2) (\mathcal{R}^+) Greedy insertion with blinks (Algorithm 3) is modified to iterate over feasible combinations for pickup and delivery insertion positions.	
Fleet minimization	
(1) Minimization of fleet size is performed by employing the absences-based acceptance criterion during the fleet minimization phase, which is executed during the first 10% of iterations. Algorithm 4 provides details concerning this procedure. (2) A solution with an increased number of tours is never accepted in the distance minimization phase.	
^a This corresponds to the procedure by Tang et al. (2010), who assume a worst-case complexity of $O(I ^2)$. Nevertheless, if $ I /2 > Q$, a worst-case complexity of $O(I Q)$ may be assumed because a service’s timing is adjusted at most Q times.	

when the number of vehicles has been reduced. Meanwhile, solutions of equal quality are colored light gray.

9. Conclusions

This paper introduced SISRs, a heuristic procedure capable of achieving highly competitive and new best results within short computation times for both the CVRP and a wide range of associated problems such as the VRP with time windows, the pickup and delivery problem, and the dial-a-ride problem. SISRs is composed of a ruin method that generates spatial slack, followed by a greedy recreate heuristic that leaves sufficient room for solution space exploration.

In addition to these components, this paper also contributes a fleet minimization approach to VRP variants focusing primarily on minimizing their number of vehicles employed.

In many respects, Laporte (2009, p. 415) provided much of the motivation behind this paper’s original contribution when writing of how “[t]here is ... a sense that several of the most successful metaheuristics are over-engineered and one should now attempt to produce simple and flexible algorithms capable of handling a larger variety of constraints, even if this were to translate into a small loss in accuracy.” Whereas SISRs exhibits the simplicity and flexibility Laporte (2009) recommends, it also serves as a counterargument that

this necessitates a loss in accuracy. As demonstrated by this paper, no such trade-off occurs.

Furthermore, performance with respect to large-scale academic instances was particularly impressive and exceeded expectations. Therefore, there is also room for further research in terms of even larger instances that are more representative of real-world conditions, the results for which should be broadly similar to those obtained for the largest instances addressed within the present study. Finally, all the newly introduced heuristic components demonstrate an impressive convergence speed which engenders an array of future real-world applications, providing a multitude of exciting avenues for future research.

Acknowledgments

The authors acknowledge the statistical advice provided by Wim Vancroonenburg (KU Leuven) and the editorial consultation provided by Luke Connolly (KU Leuven).

References

- Augerat P, Belenguer JM, Benavent E, Corberan A, Naddef D, Rinaldi G (1995) Computational results with a branch-and-cut code for the capacitated vehicle routing problem. Working paper, Université Joseph Fourier, Saint-Martin-d'Hères, France.
- Baldacci R, Christofides N, Mingozzi A (2008) An exact algorithm for the vehicle routing problem based on the set partitioning formulation with additional cuts. *Math. Programming* 115(2):351–385.
- Bresina JL (1996) Heuristic-biased stochastic sampling. *Proc. AAAI Conf. Artificial Intelligence*, Portland OR, 271–278.
- Campbell AM, Vandenbussche D, Hermann W (2008) Routing for relief efforts. *Transportation Sci.* 42(2):127–145.
- Chassaing M, Duhamel C, Lacomme P (2016) An ELS-based approach with dynamic probabilities management in local search for the dial-a-ride problem. *Engrg. Appl. Artificial Intelligence* 48(February):119–133.
- Christofides N, Eilon S (1969) An algorithm for the vehicle-dispatching problem. *J. Oper. Res. Soc.* 20(3):309–318.
- Christofides N, Mingozzi A, Toth P (1979) The vehicle routing problem. Christofides N, ed. *Combinatorial Optimization* (Wiley, Chichester), 315–338.
- Cordeau JF, Laporte G (2003) A tabu search heuristic for the static multi-vehicle dial-a-ride problem. *Transportation Res. Part B: Methodological* 37(6):579–594.
- Croes GA (1958) A method for solving traveling-salesman problems. *Oper. Res.* 6(6):791–812.
- Curtois T, Landa-Silva D, Qu Y, Laesanklang W (2018) Large neighbourhood search with adaptive guided ejection search for the pickup and delivery problem with time windows. *EURO J. Transportation Logist.* 7(2):151–192.
- Dantzig GB, Ramser JH (1959) The truck dispatching problem. *Management Sci.* 6(1):80–91.
- Dees WA, Smith RJ (1981) Performance of interconnection rip-up and reroute strategies. *18th Design Automation Conf.* (IEEE, Piscataway, NJ), 382–390.
- Dumas Y, Desrosiers J, Soumis F (1991) The pickup and delivery problem with time windows. *Eur. J. Oper. Res.* 54(1):7–22.
- Dunn OJ (1961) Multiple comparisons among means. *J. Amer. Statist. Assoc.* 56(293):52–64.
- Fahrion R, Wrede M (1990) On a principle of chain-exchange for vehicle-routing problems (1-vrp). *J. Oper. Res. Soc.* 41(9):821–827.
- Fisher ML (1994) Optimal solution of vehicle routing problems using minimum k-trees. *Oper. Res.* 42(4):626–642.
- Fukasawa R, Longo H, Lysgaard J, Poggi M, Reis M, Uchoa E, Werneck RF (2006) Robust branch-and-cut-and-price for the capacitated vehicle routing problem. *Math. Programming* 106(3):491–511.
- Gajpal Y, Abad P (2009) An ant colony system (ACS) for vehicle routing problem with simultaneous delivery and pickup. *Comput. Oper. Res.* 36(12):3215–3223.
- Gehring H, Homberger J (1999) A parallel hybrid evolutionary metaheuristic for the vehicle routing problem with time windows. Miettinen K, Mäkelä M, Toivanen J, eds. *Proc. EUROGEN99, Jyväskylä, Finland*, 57–64.
- Gehring H, Homberger J (2001) A parallel two-phase metaheuristic for routing problems with time windows. *Asia-Pacific J. Oper. Res.* 18(1):35–47.
- Gillett BE, Johnson JG (1976) Multi-terminal vehicle-dispatch algorithm. *Omega* 4(6):711–718.
- Goetschalckx M, Jacobs-Blecha C (1989) The vehicle routing problem with backhauls. *Eur. J. Oper. Res.* 42(1):39–51.
- Golden BL, Wasil EA, Kelly JP, Chao IM (1998) *The Impact of Metaheuristics on Solving the Vehicle Routing Problem: Algorithms, Problem Sets, and Computational Results* (Springer, Boston), 33–56.
- Gschwind T (2019) Route feasibility testing and forward time slack for the synchronized pickup and delivery problem. *OR Spectrum* 41(2):491–512.
- Gschwind T, Drexl M (2019) Adaptive large neighborhood search with a constant-time feasibility test for the dial-a-ride problem. *Transportation Sci.* 53(2):480–491.
- Jaw JJ, Odoni AR, Psaraftis HN, Wilson NHM (1986) A heuristic algorithm for the multi-vehicle advance request dial-a-ride problem with time windows. *Transportation Res. Part B: Methodological* 20(3):243–257.
- Kirkpatrick S, Gelatt CD, Vecchi MP (1983) Optimization by simulated annealing. *Science* 220(4598):671–680.
- Laporte G (2009) Fifty years of vehicle routing. *Transportation Sci.* 43(4):408–416.
- Laporte G, Musmanno R, Vucaturio F (2010) An adaptive large neighbourhood search heuristic for the capacitated arc-routing problem with stochastic demands. *Transportation Sci.* 44(1):125–135.
- Li F, Golden B, Wasil E (2005) Very large-scale vehicle routing: new test problems, algorithms, and results. *Comput. Oper. Res.* 32(5):1165–1179.
- Li H, Lim A (2001) A metaheuristic for the pickup and delivery problem with time windows. *Proc. 13th IEEE Internat. Conf. Tools Artificial Intelligence* (Institute of Electrical and Electronics Engineers, Piscataway, NJ), 160–167.
- Lin S (1965) Computer solutions of the traveling salesman problem. *Bell System Tech. J.* 44(10):2245–2269.
- Masmoudi MA, Braekers K, Masmoudi M, Dammak A (2017) A hybrid genetic algorithm for the heterogeneous dial-a-ride problem. *Comput. Oper. Res.* 81(May):1–13.
- Metropolis N, Rosenbluth AW, Rosenbluth MN, Teller AH, Teller E (1953) Equation of state calculations by fast computing machines. *J. Chemical Phys.* 21(6):1087–1092.
- Min H (1989) The multiple vehicle routing problem with simultaneous delivery and pick-up points. *Transportation Res. Part A: General* 23(5):377–386.
- Nagata Y, Kobayashi S (2010) A memetic algorithm for the pickup and delivery problem with time windows using selective route exchange crossover. Schaefer R, Cotta C, Kołodziej J, Rudolph G, eds. *Parallel Problem Solving from Nature, PPSN XI, Lecture Notes in Computer Science*, vol. 6238 (Springer, Berlin), 536–545.
- Nagata Y, Bräysy O, Dullaert W (2010) A penalty-based edge assembly memetic algorithm for the vehicle routing problem with time windows. *Comput. Oper. Res.* 37(4):724–737.

- Ngueveu SU, Prins C, Wolfler Calvo R (2010) An effective memetic algorithm for the cumulative capacitated vehicle routing problem. *Comput. Oper. Res.* 37(11):1877–1885.
- Or I (1976) Traveling salesman-type combinatorial problems and their relation to the logistics of regional blood banking. Thesis, Xerox University Microfilms, Ann Arbor, MI.
- Pecin D, Pessoa A, Poggi M, Uchoa E (2014) Improved branch-cut-and-price for capacitated vehicle routing. Lee J, Vygen J, eds. *Integer Programming and Combinatorial Optimization*, Lecture Notes in Computer Science, vol. 8494 (Springer International Publishing, Cham, Switzerland), 393–403.
- Pisinger D, Røpke S (2007) A general heuristic for vehicle routing problems. *Comput. Oper. Res.* 34(8):2403–2435.
- Potvin JY, Rousseau JM (1993) A parallel route building algorithm for the vehicle routing and scheduling problem with time windows. *Eur. J. Oper. Res.* 66(3):331–340.
- Potvin JY, Rousseau JM (1995) An exchange heuristic for routeing problems with time windows. *J. Oper. Res. Soc.* 46(12):1433–1446.
- Prins C (2004) A simple and effective evolutionary algorithm for the vehicle routing problem. *Comput. Oper. Res.* 31(12):1985–2002.
- R Core Team (2017) R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna.
- Repoussis PP, Tarantilis CD, Ioannou G (2009) Arc-guided evolutionary algorithm for the vehicle routing problem with time windows. *IEEE Trans. Evolutionary Comput.* 13(3):624–647.
- Ribeiro GM, Laporte G (2012) An adaptive large neighborhood search heuristic for the cumulative capacitated vehicle routing problem. *Comput. Oper. Res.* 39(3):728–735.
- Rochat Y, Taillard ED (1995) Probabilistic diversification and intensification in local search for vehicle routing. *J. Heuristics* 1(1):147–167.
- Røpke 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.
- Røpke 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.
- Sariklis D, Powell S (2000) A heuristic method for the open vehicle routing problem. *J. Oper. Res. Soc.* 51(5):564–573.
- Savelsbergh M (1990) An efficient implementation of local search algorithms for constrained routing problems. *Eur. J. Oper. Res.* 47(1):75–85.
- Savelsbergh MWP (1988) Computer aided routing. Doctoral dissertation, Centrum voor Wiskunde en Informatica, Amsterdam.
- Schrimpf G, Schneider J, Stamm-Wilbrandt H, Dueck G (2000) Record breaking optimization results using the ruin and recreate principle. *J. Comput. Phys.* 159(2):139–171.
- Shaw P (1998) Using constraint programming and local search methods to solve vehicle routing problems. *Proc. 4th Internat. Conf. Principles Practice Constraint Programming* (Springer-Verlag, 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.
- Taillard E, Badeau P, Gendreau M, Guertin F, Potvin JY (1997) A tabu search heuristic for the vehicle routing problem with soft time windows. *Transportation Sci.* 31(2):170–186.
- Tang J, Kong Y, Lau H, Ip AW (2010) A note on efficient feasibility testing for dial-a-ride problems. *Oper. Res. Lett.* 38(5):405–407.
- Toth P, Vigo D (2002) Models, relaxations and exact approaches for the capacitated vehicle routing problem. *Discrete Appl. Math.* 123(1):487–512.
- Uchoa E, Pecin D, Pessoa A, Poggi M, Vidal T, Subramanian A (2017) New benchmark instances for the capacitated vehicle routing problem. *Eur. J. Oper. Res.* 257(3):845–858.
- Vidal T, Crainic TG, Gendreau M, Prins C (2013) A hybrid genetic algorithm with adaptive diversity management for a large class of vehicle routing problems with time-windows. *Comput. Oper. Res.* 40(1):475–489.
- Vidal T, Crainic TG, Gendreau M, Prins C (2014) A unified solution framework for multi-attribute vehicle routing problems. *Eur. J. Oper. Res.* 234(3):658–673.
- Vidal T, Crainic TG, Gendreau M, Prins C (2015) Time-window relaxations in vehicle routing heuristics. *J. Heuristics* 21(3):329–358.
- Vidal T, Crainic TG, Gendreau M, Lahrichi N, Rei W (2012) A hybrid genetic algorithm for multidepot and periodic vehicle routing problems. *Oper. Res.* 60(3):611–624.
- Wilcoxon F (1945) Individual comparisons by ranking methods. *Biometrics Bull.* 1(6):80–83.
- Wilson NHM, Sussman JM, Wang HK, Higonnet BT (1971) Scheduling algorithms for a dial-a-ride system. Technical Report USL TR-70-13, Massachusetts Institute of Technology, Cambridge, MA.
- Zachariadis EE, Kiranoudis CT (2012) An effective local search approach for the vehicle routing problem with backhauls. *Expert Systems Appl.* 39(3):3174–3184.