CIRRELT

CIRRELT-2023-28

A Metaheuristic for a Time-Dependent Vehicle Routing Problem with Time Windows,Two Vehicle Fleets and Synchronization on a Road Network

Fernando Obed Guillen Reyes
Michel Gendreau
Jean-Yves Potvin

September 2023

Bureau de Montréal
Université de Montréal
C.P. 6128, succ. Centre-Ville
Montréal (Québec) H3C 3J7
Tél : 1-514-343-7575
Télécopie : 1-514-343-7121

Bureau de Québec
Université Laval,
2325, rue de la Terrasse
Pavillon Palasis-Prince, local 2415
Québec (Québec) G1V 0A6
Tél : 1-418-656-2073
Télécopie : 1-418-656-2624

A Metaheuristic for a Time-Dependent Vehicle Routing Problem with Time Windows, Two Vehicle Fleets and Synchronization on a Road Network

Fernando Obed Guillen Reyes[1,2], Michel Gendreau[1,3], Jean-Yves Potvin[1,2,*]

1. Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT)
2. Département d'informatique et de recherche opérationnelle, Université de Montréal, Canada
3. Département de mathématiques et de génie industriel, Polytechnique Montréal, Canada

Abstract. In this work, we extend the time-dependent vehicle routing problem with time windows on a road network by considering two types of vehicles, large and small, to serve customers. Motivated from city logistics applications, large vehicles are forbidden from the downtown area. Accordingly, goods must be transferred from large to small vehicles to serve downtown customers. This leads to synchronization issues at transfer points, which are special locations without storage capacity. The problem is not a pure two-echelon vehicle routing problem, since customers located outside of the downtown area can be served directly by large vehicles. The problem is further compounded by the presence of time-dependent travel times that are defined on the arcs of the road network and are used to model congestion periods. To solve this complex problem, we propose an adaptation of the Slack Induction by String Removals metaheuristic, which is state-of-the-art for the classical capacitated vehicle routing problem. Computational results on a set of test instances with different characteristics empirically demonstrate the optimization capabilities of this new metaheuristic on a problem which is much more complex than the capacitated vehicle routing problem.

Keywords: Time-dependent vehicle routing, time windows, road network, transfer points, synchronization, metaheuristic.

Acknowledgements. The work reported in this paper was financially supported by the Natural Sciences and Engineering Research Council of Canada (NSERC). This support is gratefully acknowledged.

---

* Corresponding author: potvin@iro.umontreal.ca

Dépôt légal – Bibliothèque et Archives nationales du Québec
Bibliothèque et Archives Canada, 2023

1 Introduction

Although the vehicle routing problem (VRP) has been widely studied for a long time, time-dependent variants have spurred the interest of researchers only recently. Time-dependency is an important issue, since the time to travel from one point to another in a network often depends on the departure time (c.f., rush hours). Furthermore, not only does the time to travel along a path between two customers may change depending on the departure time, but even the best path to use may also change. Thus, recent studies have exploited the additional information available in a road network to account for multiple possible paths between two customers, which is often referred to as the time-dependent vehicle routing problem with time windows on a road network ($TDVRPTW_{RN}$). In this paper, we consider an extension of this problem where both large (black) and small (green) vehicles are involved and where some parts of the road network are forbidden to one type of vehicles or the other. For example, the downtown area is not accessible to large vehicles, whereas areas far from downtown are not accessible to small vehicles (e.g., bicycles). Since the goods to be delivered are initially loaded in large vehicles, a customer located in an area not accessible to them can only be served through a transfer of its demand from a large to a small vehicle. This transfer takes place at special locations with no storage capacity, known as transfer points (TPs). This also leads to synchronization issues between the two types of vehicles at transfer points. In the following, this complex delivery problem will be referred to as the $TDVRPTW_{RN}$ with transfer points or $TDVRPTWTP_{RN}$.

Our problem needs to be distinguished from problems with intermediate facilities, with or without storage capacity, since there is no facility as such to transfer goods. It must also be distinguished from two- or multi-echelon VRPs where vehicles are organized into a strict hierarchical structure to deliver goods to customers. In our problem, black vehicles can very well serve customers directly, as long as they do not belong to forbidden areas. Our contribution lies in the adaptation of a state-of-the-art metaheuristic for the capacitated VRP (CVRP) for a much more complex problem that involves two types of vehicles, three types of customers, time-dependent travel times and synchronization issues between the two types of vehicles to transfer loads at transfer points. As far as we know, this problem has never been addressed in the literature.

In the following, Section 2 first reviews problems related to ours, namely time-dependent VRPs and VRPs with intermediate facilities. Section 3 then precisely describes our problem. The original implementation of the metaheuristic Slack Induction by String Removals (SISR) for solving the CVRP is described in Section 4. Then, Section 5 introduces time issues that arise in the $TDVRPTWTP_{RN}$, in particular calculation of time bounds to check insertion feasibility in constant time and synchronization between black and green vehicles at transfer points. Specific modifications to the original SISR implementation that are required to address the much more complex $TDVRPTWTP_{RN}$ are reported in Section 6. Then, computational results obtained on test instances derived from a benchmark for the $TDVRPTW_{RN}$ are reported. Finally, a conclusion follows.

2 Literature review

Two main features of our problem are time-dependent travel times and the presence of intermediate points to transfer loads from one type of vehicles to another. Problems with these characteristics are briefly reviewed in the following.

2.1 Time-dependent VRPs

In the first studies about time-dependent VRPs, a customer-based graph was used, where nodes correspond to customers plus the depot and arcs stand for a particular path between two customers in the underlying road network (e.g., shortest path in distance). Since a fixed path is used to travel between two customers, it is only possible to account for different travel times at different departure times along that path. However, it is not possible to consider different paths between two customers at different departure times. To the best of our knowledge, the first work that addressed time-dependency (although without time windows at customers) on a customer-based graph is found in [1]. In this work, the time horizon is divided into periods with a different travel time matrix for each period. This is equivalent to defining a step function to model the travel time at different periods between any given pair of nodes. A similar approach is proposed in [2], although for a problem with time windows. Since the travel time is constant within a period, but may abruptly change from one period to the next, the two previous models do not satisfy the First-In-First-Out (FIFO) property, where it is required that a vehicle traveling earlier on an arc must arrive at the destination node earlier than any other vehicle traveling later on the same arc. A different model is proposed in [3], where each node is assigned a speed at a given period of time, which can be interpreted as the average speed around the node. Then, the travel time on a given arc between two nodes is based on the average speed around these two nodes. Once again, since the travel time on a given arc is constant within a period, the FIFO property is not satisfied neither. A model that satisfies the FIFO property was finally proposed in [4]. Here, the authors use a step function to model speed (rather than travel time) at different time periods. That is, the speed is constant within a given time period, but may change from one period to the next. The travel time along an arc is then computed by taking into account the new speed when the time boundary between two periods is crossed. Thus, every vehicle that travels along the same arc within the same period has the same speed and the speed of every vehicle changes similarly when the boundary between two given periods is crossed. This way to model time-dependency has been largely adopted in the following years to solve TDVRPTWs using exact methods and metaheuristics [5, 6, 7, 8]. In a few cases, continuous functions with special characteristics to satisfy the FIFO property have also been used to model time-dependent travel times [9, 10]. For a detailed literature review on time-dependent VRPs using customer-based graphs (up to 2015), the reader is referred to [11].

Realistic objective functions are often based on travel times and the fastest path between two customers may well change depending on the departure time. To account for this, multi-graph representations have been proposed [12, 13, 14]. In these graphs, parallel arcs between any given pair of customer nodes stand for different least-cost paths in the

underlying road network associated with different departure times from the origin customer node (or, more generally, for different non-dominated paths in multi-objective optimization). In [13, 14], the authors empirically demonstrate, using a branch-and-price algorithm and an adaptive large neighborhood search (ALNS), that a multigraph representation for a bi-objective VRP with time windows (VRPTW) that accounts for both travel time and travel cost leads to considerably better solutions when compared to a standard customer-based graph with a single arc (path) between two customers. However, the authors note the considerable computation times needed to compute the multigraph. Accordingly, the authors in [15], propose to work directly on the road network. In [16], a comparison between a branch-and-price algorithm applied to a road network and to a multigraph representation for a bi-objective VRPTW (travel time, travel cost) shows that both approaches are competitive, but with a slight advantage for multigraphs on the more realistic instances. On the other hand, working directly on a road network is more natural and straightforward. Thus, recent works that addressed TDVRPTWs typically use road networks [17, 18].

2.2 VRPs with intermediate facilities

Due to the presence of transfer points in our problem, we provide an overview of the literature on VRPs with intermediate facilities, which are referred to as satellites, hubs, transshipment points or cross-docks. They all represent intermediate points where goods can be transferred while they move from their origin to their destination. In [19], the authors provide a survey about intermediate facilities in freight transportation, while a survey dedicated to cross-docking is found in [20]. In [19], the problems are divided into two classes, that is, two-echelon VRPs (2E-VRPs) and pickup and delivery problems with cross-docks (PDPCDs). With regard to 2E-VRPs, the intermediate facilities are called satellites and have some storage capacity. At the first-level or echelon, vehicles carry goods from a depot to satellites, while at the second-level goods are transported by other vehicles from satellites to customers. Typically, a strict hierarchy is observed, that is, direct deliveries from the depot to customers is forbidden. In this survey, no work requires synchronization between vehicles at satellites. In the case of the surveyed PDPCDs, however, cross-docks have no or little capacity and synchronization is required. Two works are worth mentioning, since there is no real intermediate facility, only transfer points (like in our work). In [21], transfers can take place at arbitrary locations and the vehicle that arrives first at the transfer point waits as long as necessary to transfer goods to the other vehicle, although a waiting penalty is incurred. In [22], transshipment points are predetermined and synchronization is achieved by setting a time window at these transfer points.

In [23], a 2E-VRP is proposed in the context of city logistics, where it is called a two-tier city logistics system. City Distribution Centers (CDCs), located at the outskirts of the city, form the first tier of the system where freight is sorted and consolidated. The second tier of the system is made of satellites located close to or within the city-center area. Different vehicle fleets are used to transport freight from CDCs to satellites and from satellites to customers. In particular, vehicles of the second tier must be adapted for utilization in dense city zones. Since it is assumed that satellites operate according to a cross-dock trans-

shipment operational model, vehicle synchronization is required. That is, vehicles of the first and second tier must meet at satellites at a given time, with very short waiting time permitted. This work proposes only a modeling framework and no algorithmic solution is developed. Different exact and heuristic algorithms were later proposed in [24, 25, 26] to solve variants of the initial model (e.g., no synchronization, storage capacity at satellites). A recent work in [27] addresses a two-commodity 2E-VRP with synchronization at satellites. Two types of vehicles are considered at the first level, one for each commodity, as opposed to the second level where only one type of vehicles is considered. Synchronization is only established between the two types of first-level vehicles, which have to meet at satellites to favor efficiency at the second level. The problem is solved with ALNS where, at each iteration, destroy and a repair operators are applied to the second-level tours, and a reconstruction procedure is then applied to the first-level tours in case of infeasibility, followed by an improvement procedure. The 2E-VRP reported in [28] is particularly interesting because is shares similarities with our problem. In this work, the first-level vehicles are called vans and the second-level vehicles are called bicycles. Similarly, there are two classes of customers depending on their location: customers located at the city center are called bike-customers, while customers outside of the center are called van-customers. Transfers take place at satellites, with no storage capacity. These satellites are located at the boundary of the city center (a van can cross the city center, but a penalty is incurred). In the proposed heuristic methodology based on GRASP and path-relinking, the second-level tours are constructed before the first level tours. In this way, information about the arrival times of bikes at satellites can be used to construct the first-level tours and account for synchronization. The authors in [29] address a similar problem called the two-echelon multi-trip VRP with satellite synchronization. The proposed methodology also constructs second-level tours before first-level tours to produce the initial solution. Then, an ALNS is applied with features aimed at improving synchronization, like a destroy operator that removes trips with the worst synchronization. In [30], the authors describe a crowdsourced system for urban parcel deliveries, where truck-carriers visit intermediate facilities called relay points and where parcels are transferred to pedestrians or cyclists that are close to the end customers. However, if no pedestrian or cyclist is available, the truck can perform the deliveries itself. In the proposed system, the delivery tasks, as well as candidate relay points, are broadcast on-line. Then, pedestrians and cyclists bid for these delivery tasks. Thus, a bid selection problem must be solved in addition to the routing problem. This is done in both cases with a tabu search. Another similar crowdsourced system is described In [31], where goods can be dropped at transfer points to be picked up later by other vehicles (i.e., transfer points have storage capacity).

3 Problem Definition

This paper addresses the time-dependent vehicle routing problem with time windows and transfer points on a road network or $TDVRPTWTP_{RN}$. As previously mentioned, two types of vehicles with different capacities are considered: black (large) and green (small) vehicles. The two sets of vehicles are denoted K^B and K^G, respectively. There are also three types of customers: black customers that can be served by black vehicles only; green

customers that can be served by green vehicles only; and neutral customers that can be served by both types of vehicles.

A road network in this context is a directed graph $G = (V, A)$, where V is the set of nodes of cardinality n and A the set of arcs or road segments. The set of nodes is then partitioned as follow:

- $D = \{d^b, d^g\}$ is the set of depots with d^b the depot for black vehicles and d^g the depot for green vehicles;
- C^B is the set of black customers of cardinality n_B ;
- C^G is the set of green customers of cardinality n_G ;
- C^E is the set of neutral customers of cardinality n_E;
- TP is the set of transfer points of cardinality n_{TP} ;
- RJ is the set of road junctions (i.e., any node that is not a depot, a customer or a transfer point).

The $TDVRPTWTP_{RN}$ can be characterized as follow:

- Each customer i has a demand d_i and a service (or dwell) time st_i;
- Each customer i has a time window $[\alpha_i, \beta_i]$ to constrain the service start time. If a vehicle arrives at customer i before α_i, then it must wait until α_i to start the service. On the other hand, a vehicle cannot arrive after β_i;
- A green vehicle can serve only one customer at a time, while a black vehicle has a capacity Q^b that allows it to serve many customers;
- The demand of all customers is assumed to be loaded into black vehicles at the start;
- Each black vehicle performs a single route that starts and ends at the black depot; each green vehicle performs a single route that starts and ends at the green depot;
- The black and green depots have a time window $[0, T]$, where T is the end of the time horizon; all vehicles must be back at their depot before or at time T;
- Black and green vehicles have different speeds. Thus, each arc $(i, j) \in A$ is associated with two time-dependent travel speed functions $v^b_{i,j}(t)$ and $v^g_{i,j}(t)$ for black and green vehicles, respectively.
- Transfer points are fixed locations without storage capacity, where a black vehicle can transfer loads to one or more green vehicles. We assume, without loss of generality, that the time to transfer a load is null. A black vehicle can visit the same transfer point multiple times along its route; the same is true of green vehicles. Each visit to transfer point $tp \in TP$ in a route is represented by a copy which is unambiguously denoted tp^k_j, where k is a vehicle and j is the copy (or visit) index. That is, copy tp^k_j corresponds to the j^{th} visit of transfer point tp in the route of vehicle k;

- Each black customer is served directly and exactly once by a black vehicle; each green customer is served exactly once by a green vehicle after its demand has been transferred from a black vehicle at a transfer point; each neutral customer is served exactly once, either directly by a black vehicle or by a green vehicle after its demand has been transferred from a black vehicle at a transfer point;
- The objective is to determine routes of minimal total duration such that all customers are served and all constraints are satisfied.

Figure 1 shows a typical solution, with one black route starting from the black depot (square). This route is identified by arcs (1) to (10). At the first transfer point tp_1 (triangle), there is a connection with the green route with arcs identified with broken lines. This route starts at the green depot (square), gets a load from the black vehicle at tp_1, delivers the load to neutral customer nc_1 (gray node), gets another load from the same black vehicle at the second transfer point tp_2, delivers the load to green customer gc_1 and returns to the green depot. The arcs of the second green route are identified with dotted lines. This small route starts at the green depot, gets a load from the black vehicle at tp_2, delivers the load to green customer gc_2 and returns to the green depot. It should be noted that the black vehicle transfers two loads, one for each green vehicle, at transfer point tp_2.

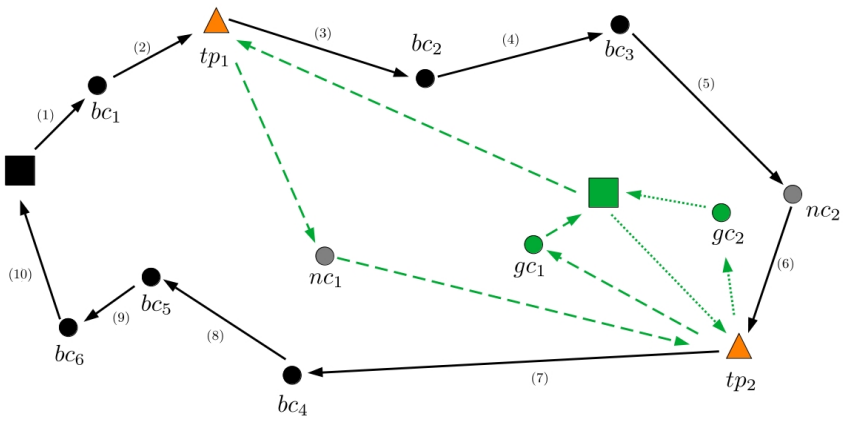

Figure 1: An example of a solution to the $TDVRPTWTP_{RN}$

4 SISR for the CVRP

The proposed methodology for solving our problem is the Slack Induction by String Removals (SISR) metaheuristic [32], which is state-of-the-art for the CVRP. This metaheuristic is based on the ruin-and-recreate principle where, at each iteration, a number of nodes are first removed from the routes of the current solution (ruin) and reinserted (recreate) to produce a new solution. A simulated annealing-based criterion is then applied to decide if the new solution should be accepted or not as the current solution. In the following, we precisely describe the *SISR* metaheuristic, as initially proposed for the *CVRP*.

The basic idea of SISR is to remove strings of consecutive customers from a solution, with at most one string removed from any given route. Algorithm 1 shows the pseudo-code of SISR for the CVRP. First, two parameter values are set : L^{max}, which is used to determine the maximum length of a string, and $\overline{c}$, which corresponds to the average number of customers to be removed from a solution. By appropriately setting these values, many strings of small length or only a few strings of large length can be removed. Given that simulated annealing principles guide the search through an exponential cooling schedule, the starting temperature τ_0, final temperature τ_f, $\tau_0 > \tau_f > 0$, and number of iterations f are defined, with the current temperature τ initially set to τ_0, see statements 2 and 3. Then, the cooling factor ρ is defined in statement 4 in such a way that f ruin-and-recreate iterations are performed.

An adjacency list $adj(i)$ is then created for each customer i in statement 5. This list contains all customers ordered from closest to farthest in distance from i, with i as its first element. This adjacency list is used to favor the removal of strings that are relatively close to each other, even if they come from different routes. Before proceeding with the main loop, an initial solution is created in step 6 in a straightforward way, by creating an individual route for each customer. This initial solution becomes the current solution s as well as the best solution known to date s_{best}.

The main loop corresponds to statements 8 to 18. At each iteration, a ruin operator and a recreate operator are applied to a copy $\overline{s}$ of current solution s, see statements 9 and 10. Note that the set A^- is used to store the removed customers. The resulting solution of the ruin-and-recreate process $\overline{s}$ is accepted as the new current solution s if it satisfies the simulated annealing-based criterion in statement 11 (see [32]). It also replaces s_{best} if it is the best solution found thus far. The current temperature τ is then updated before the next iteration starts. After f iterations of the main loop, the whole procedure stops and returns the best solution found.

4.1 Ruin

In the ruin procedure described in Algorithm 2, the maximum length of a string to be removed l_s^{max} is first set to the minimum of L^{max} and the average number of nodes in a route of the current solution $AvgRouteNodes(s)$, see statement 1. Then, in statement 2, the maximum number of removed strings n_s^{max} is calculated using l_s^{max} and $\overline{c}$, see the exact formula in [32]. The actual number of removed strings n_s is chosen from a continuous uniform distribution defined between 1 and $n_s^{max} + 1$, as indicated in statement 3. The set of ruined routes R^- and the set of removed customers A^- are then initialized with the empty set. After creating a copy $\overline{s}$ of the current solution s, a random seed customer i^{seed} is chosen and its adjacency list is processed (from closest to farthest customers) until all customers have been considered or the number of ruined routes is reached, see the main loop in statements 8-20. Note that the number of ruined routes is the same as the number of removed strings n_s, since each string is removed from a different route. If the current customer i in the adjacency list of i^{seed} has not been previously removed and if the route r that serves i has not been previously ruined (statement 10) then the ruin operator is

Algorithm 1 SISR for CVRP

Set L^{max} and $\overline{c}$
Set τ_0, τ_f and f
$\tau \leftarrow \tau_0$
$\rho \leftarrow \left(\frac{\tau_f}{\tau_0}\right)^{1/f}$
Generate adjacency list $adj(i)$ for each customer i
Generate initial solution s (with set of routes R_s)
$s_{best} \leftarrow s$
for f iterations **do**
    $\overline{s}, A^- \leftarrow Ruin(s)$
    $\overline{s} \leftarrow Recreate(\overline{s}, A^-)$
    if $Cost(\overline{s}) < (Cost(s) - \tau \ln(U(0,1)))$ **then**
        $s \leftarrow \overline{s}$;
    end if
    if $Cost(\overline{s}) < Cost(s_{best})$ **then**
        $s_{best} \leftarrow \overline{s}$
    end if
    $\tau \leftarrow \rho\tau$
end for
Return s_{best}

Algorithm 2 Ruin(s)

$l_s^{max} \leftarrow \min\{L^{max}, AvgNodesInRoutes(s)\}$
Calculate n_s^{max} with l_s^{max} and $\overline{c}$
$n_s \leftarrow \lfloor U(1, n_s^{\max} + 1) \rfloor$
$R^- \leftarrow \emptyset$
$A^- \leftarrow \emptyset$
$\overline{s} \leftarrow s \quad R_{\overline{s}} \leftarrow R_s$
Select randomly a seed customer i^{seed} in $\overline{s}$
for $i \in adj(i^{seed})$ and $|R^-| < n_s$ **do**
    $r \leftarrow$ route of customer i
    if $i \notin A^-$ and $r \notin R^-$ **then**
        $l_r^{\max} \leftarrow \min\{l_s^{\max}, |r|\}$
        $l_r \leftarrow \lfloor U(1, l_r^{\max} + 1) \rfloor$
        $RuinOp \leftarrow Random(String, Split\text{-}String)$
        $A^- \leftarrow A^- \cup RuinOp(\overline{s}, r, l_r, i)$
        $R^- \leftarrow R^- \cup \{r\}$
        if r is empty **then**
            $R_{\overline{s}} \leftarrow R_{\overline{s}} \backslash \{r\}$
        end if
    end if
end for
Return $\overline{s}$, A^-

applied to route r. In statements 11 and 12, the actual length of the removed string l_r is chosen from a continuous uniform distribution defined between 1 and $l_r^{\max} + 1$, where $l_r^{\max}$ is the minimum of $l_s^{\max}$ and cardinality of r (since the length of the removed string cannot exceed the number of nodes in r).

Then, a random choice between two ruin operators takes place in statement 13. These operators are:

- *String*: A random string of length l_r that contains the current customer i in the adjacency list of i^{seed} is removed from route r. This is illustrated in Figure 2(a) for a string of length four with the gray node i_3 as current customer i;
- *Split-String*: A random string of length $l_r + m$ that contains the current customer i in the adjacency list of i^{seed} is chosen in route r (where the procedure to select a value for m is precisely described in [32]). Then, a random substring of m consecutive customers within the chosen string is kept in the route, so that only l_r customers are removed. The substring of length m cuts the string of length $l_r + m$ in two parts, unless the substring is at the very beginning or very end of the string of length $l_r + m$. An example is provided in Figure 2(b) for a string of length five with the gray node i_3 as current customer i. In this example, $m = 2$, so that only three customers are removed from the route.

The removed customers are then added to A^- and the ruined route to R^- in statements 14 and 15. If route r becomes empty, then it is deleted from the set of routes in the solution, as indicated in statements 16 and 17. At the end, the ruined solution $\overline{s}$ and the set of removed customers are returned.

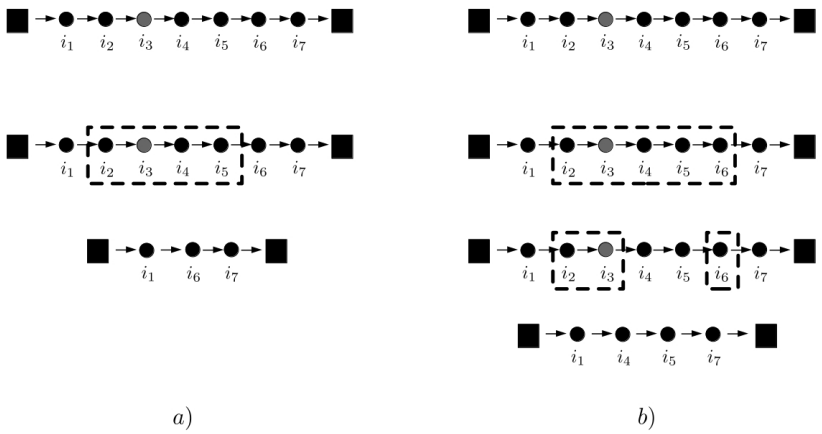

Figure 2: Examples of the two SISR ruin operators : (a) String (b) Split-String.

4.2 Recreate

A new complete solution is then produced with the recreate operator by reinserting the removed customers. This operator is described in Algorithm 3. In statement 1, the removed customers in A^- are first sorted using a sorting criterion chosen with a particular distribution probability among: random, decreasing demand, increasing distance from the depot, decreasing distance from the depot. Based on the chosen order, the customers in A^- are considered one by one and reinserted in the set of routes $R_{\overline{s}}$ of solution $\overline{s}$, see the main loop in statements 2-19. Each insertion place in each route that can accommodate the demand of customer i is considered and the best encountered insertion place p_{best} is identified. It should be noted, however, that the chosen insertion place is not necessarily the best one among all feasible insertion places, due to blinks that correspond to a small probability γ of skipping a position, see statement 6. In particular, if the best position is skipped then only the second best position can be chosen (as long as this position is not skipped too). Statements 14-17 cover the situation when no feasible insertion place is found for customer i. In this case, a new route is created for that customer. At the end, the recreated solution $\overline{s}$ is returned.

Algorithm 3 Recreate($\overline{s}$, A^-)

1: Sort(A^-) ▷ Recreate
2: **for** $i \in A^-$ **do**
3: $p_{best} \leftarrow NULL$; $CostInsert_{best} \leftarrow \infty$
4: **for** $r \in R_{\overline{s}}$ and r feasible with insertion of i **do**
5: **for** p_r in r **do**
6: **if** $U(0,1) < 1 - \gamma$ **then**
7: **if** $p_{best} = NULL$ or $CostInsert(i, p_r) < CostInsert_{best}$ **then**
8: $p_{best} \leftarrow p_r$
9: $CostInsert_{best} \leftarrow CostInsert(i, p_r)$
10: **end if**
11: **end if**
12: **end for**
13: **end for**
14: **if** $p_{best} = NULL$ **then**
15: $R_{\overline{s}} \leftarrow R_{\overline{s}} \cup \{\text{new empty route } r\}$
16: $p_{best} \leftarrow$ first position in r
17: **end if**
18: Insert i in position p_{best}
19: **end for**
20: Return $\overline{s}$

5 Time dependency

The SISR metaheuristic for the CVRP, as described in the previous section, needs to be considerably modified to address the much more complex $TDVRPTWTP_{RN}$. In particular, the time dimension must now be taken into account; furthermore, routes are not indepen-

dent anymore since they interact through transfer points. In this section, we introduce the basics of our time-dependent travel time model and explain how time bounds can be derived at each node along the routes of black and green vehicles.

5.1 Time-dependent travel times

The IGP model proposed in [4] is used to model time dependency. In this model, the time horizon $[0, T]$ is partitioned into a number l of time periods $[0, t_1)$,$[t_1, t_2)$,...., $[t_{l-2}, t_{l-1})$, $[t_{l-1}, T]$, where t_1, t_2,..., t_{l-1} are time boundaries between two periods. For any given arc, a travel speed is associated with each period and a speed change occurs when a vehicle crosses a time boundary. The algorithmic procedure to compute the travel time along an arc for a given departure time based on this model is provided in [4]. Although speed is modeled as a step function of time, the corresponding travel time function is a piecewise linear function. Figure 3 shows an example of a travel speed function on a given arc (i, j) and the corresponding travel time function, assuming that the arc is of length 4.

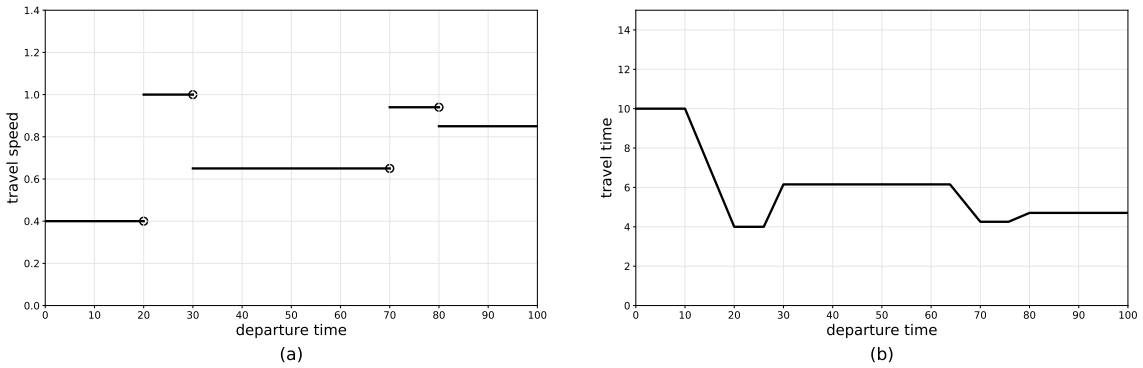

Figure 3: (a) Travel speed function of arc (i, j) (b) Corresponding travel time function assuming that arc (i, j) is of length 4.

5.2 Dominant shortest-path structure

The dominant shortest-path structure (DSPS), as described in [18], is useful to quickly identify the fastest path between any given pair of nodes (either customers, depots or transfer points) in the road network for any given departure time. First, a number of good paths between two given nodes i and j are identified by applying a time-dependent Dijkstra's algorithm [18] using different departure times from i, like time boundaries between two periods. The travel time function of each one of those paths is obtained by combining the travel time functions of all arcs along that path (which also produces a piecewise linear function). Figure 4 shows an example of a DSPS based on three different fastest paths between two nodes. In this figure, the arrival time is represented as a function of the departure time, so that the corresponding travel time is simply the difference between arrival and departure times. Since the IGP model satisfies the FIFO property, this piecewise linear function is non decreasing. By overlapping the three paths, it is possible to

identify the fastest among the three paths for any given departure time. It is worth noting that the DSPS is exact only if the time-dependent Dijkstra's algorithm is applied with a sufficiently large number of departure times to cover all fastest paths between two nodes, which is rarely the case in practice. But better accuracy is obtained with more departure times.

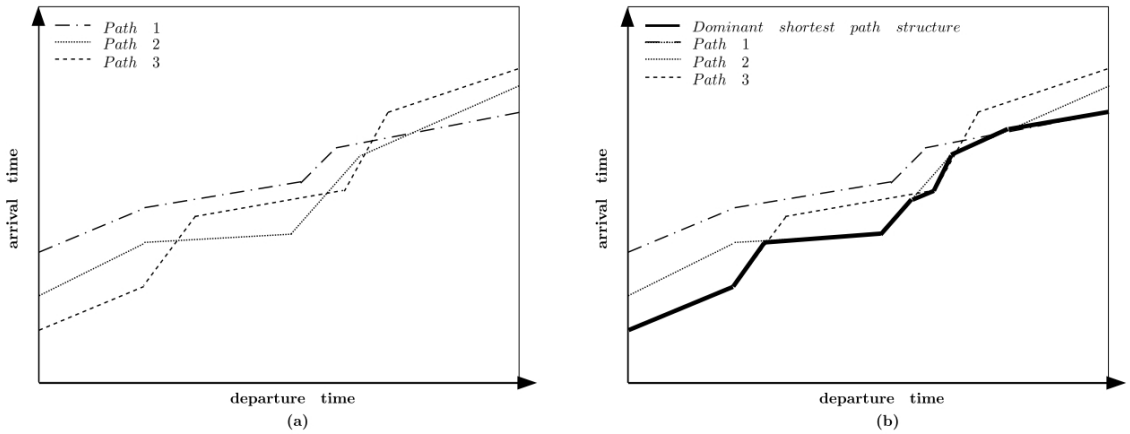

Figure 4: (a) Three different fastest paths between two nodes obtained at different time points using a time-dependent Dijkstra's algorithm (b) Corresponding dominant shortest path structure

Two different travel time functions are associated with each arc, depending if a black or a green vehicle follows that arc, because they do not have the same speed. This leads to two different DSPSs between each pair of nodes made of either customers, depots or transfer points. Accordingly, the following notation will be used:

- $AT^b(i, j, dt)$ is the arrival time at j when a black vehicle departs from i at time dt and follows the fastest path to reach j, as determined by the DSPS of nodes i and j for a black vehicle;
- $DT^b(i, j, at)$ is the inverse of $AT_b(i, j, dt)$ and is the departure time at i that allows a black vehicle to arrive at j at time at;
- $AT^g(i, j, dt)$ is the arrival time at j when a green vehicle departs from i at time dt and follows the fastest path to reach j, as determined by the DSPS of the pair of nodes i and j for a green vehicle;
- $DT^g(i, j, at)$ is the inverse of $AT^g(i, j, dt)$ and is the departure time at i that allows a green vehicle to arrive at j at time at.

5.3 Synchronization at a transfer point

Let us consider $tp_{\cdot}^{k^b}$ a copy of transfer point $tp \in TP$ in the route of black vehicle $k^b \in K^B$, where the subscript $\cdot$ corresponds to a particular copy (visit) index of transfer point tp in

the route of vehicle k^b. Let us also consider $tp_{\cdot}^{k_1^g}$, $tp_{\cdot}^{k_2^g}$, ..., $tp_{\cdot}^{k_h^g}$, h copies of transfer point tp in the routes of green vehicles k_1^g, k_2^g, ..., $k_h^g \in K^G$. We assume that black vehicle k^b needs to be synchronized with green vehicles k_1^g, k_2^g, ..., k_h^g at these copies of transfer point tp.

We first account for the arrival time of the last green vehicle:

$$at^{max} = \max_{l=1,...,h} \{at_{tp_{\cdot}^{k_l^g}}\} \tag{1}$$

Then, the departure time of black vehicle k^b from $tp_{\cdot}^{k^b}$ is :

$$dt_{tp_{\cdot}^{k^b}} = \max\{at_{tp_{\cdot}^{k^b}}, at^{max}\} \tag{2}$$

That is, if all green vehicles arrive at the transfer point before black vehicle k^b, then the latter can depart immediately (given that the time to transfer loads from the black vehicle to green vehicles is null) with $dt_{tp_{\cdot}^{k^b}} = at_{tp_{\cdot}^{k^b}}$. Otherwise, vehicle k^b will depart at the arrival time at^{max} of the last green vehicle.

The departure of each green vehicle k_l^g from $tp_{\cdot}^{k_l^g}$, $l = 1, ..., h$, is:

$$dt_{tp_{\cdot}^{k_l^g}} = \max\{at_{tp_{\cdot}^{k^b}}, at_{tp_{\cdot}^{k_l^g}}\} \quad l = 1, ..., h \tag{3}$$

That is, if green vehicle k_l^g arrives at the transfer point before black vehicle k^b, it must wait for the arrival of vehicle k^b before it can depart from $tp_{\cdot}^{k_l^g}$. Otherwise, it can depart immediately with $dt_{tp_{\cdot}^{k_l^g}} = at_{tp_{\cdot}^{k_l^g}}$.

5.4 Time bounds

In the following, we define earliest and latest time bounds for the arrival at and departure from each node in the route of a black or green vehicle, where a node can be a customer, a transfer point or a depot. That is, a vehicle must arrive at (depart from) a node before its latest arrival (departure) time to guarantee that the rest of the route satisfies the time constraints. For simplifications purposes, the forward and backward propagation procedures described below focus on a single black or green route and do not account for possible complex interactions among routes (see subsection 5.5) .

5.4.1 Green route

Here, we explain how to propagate the earliest and latest arrival and departure times in a green route. For this purpose, let us consider the route of green vehicle $k^g \in K^G$ which is made of (1) a copy d_0^g of the green depot to start the route, (2) a sequence of copies of one or more transfer points $tp_{l\cdot}^{k^g}$, $l = 1, ..., p$, each followed by a green (or neutral) customer $i_l \in C^G \cup C^E$, $l = 1, ..., p$ and (3) a copy d_{p+1}^g of the green depot to end the route. That is, the route of green vehicle k^g is d_0^g, $tp_{1\cdot}^{k^g}$, i_1, $tp_{2\cdot}^{k^g}$, i_2, ..., $tp_{p\cdot}^{k^g}$, i_p, d_{p+1}^g.

Earliest arrival and departure times

First, the earliest departure time from d_0^g is set equal to 0. Then, we go forward by first computing the earliest arrival time *eat* at transfer point $tp_{1.}^{k^g}$, using function AT^g:

$$eat_{tp_{1.}^{k^g}} = AT^g(d_0^g, tp_{1.}^{k^g}, 0) \tag{4}$$

Now, to determine the earliest departure time *edt*, we need to account for the corresponding black vehicle k^b from which green vehicle k^g should receive a load. Accordingly, if $eat_{tp_{1.}^{k^g}} < eat_{tp_{1.}^{k^b}}$, then $edt_{tp_{1.}^{k^g}} = eat_{tp_{1.}^{k^b}}$, since green vehicle k^g cannot depart from the transfer point before the earliest arrival time of black vehicle k^b. Otherwise, $edt_{tp_{1.}^{k^g}} = eat_{tp_{1.}^{k^g}}$, given that the time to transfer a load is null.

Still going forward, we now consider customer i_i and compute its earliest departure time as :

$$eat_{i_1} = AT^g(tp_{1.}^{k^g}, i_1, edt_{tp_{1.}^{k^g}}) \tag{5}$$

Now, if $eat_{i_1} < \alpha_{i_1}$, then the earliest departure time $edt_{i_1} = \alpha_{i_1} + st_{i_1}$, otherwise $edt_{i_1} = eat_{i_1} + st_{i_1}$.

This forward procedure is repeated until the end depot d_{p+1}^g is reached and its earliest arrival time is determined.

Latest arrival and departure times

We start by setting the latest arrival time *lat* at the end depot d_{p+1}^g to be the end of time horizon T, that is $lat_{d_{p+1}^g} = T$. Then, we go backward by first computing the latest departure time *ldt* at customer i_p that allows vehicle k^g to arrive at d_{p+1}^g at time $lat_{d_{p+1}^g}$, using function DT^g:

$$ldt_{i_p} = DT^g(i_p, d_{p+1}^g, lat_{d_{p+1}^g}) \tag{6}$$

Now, if $ldt_{i_p} > \beta_{i_p} + st_{i_p}$, then ldt_{i_p} is reset to $\beta_{i_p} + st_{i_p}$, because vehicle k^g cannot depart from i_p later than $\beta_{i_p} + st_{i_p}$ without violating the time window constraint (i.e., the arrival time cannot exceed β_{i_p}). Then, the latest arrival time at customer i_p is simply computed as $lat_{i_p} = ldt_{i_p} - st_{i_p}$.

Still going backward, we now consider the transfer point $tp_{p.}^{k^g}$ and compute its latest departure time as :

$$ldt_{tp_{p.}^{k^g}} = DT^g(tp_{p.}^{k^g}, i_p, lat_{i_p}) \tag{7}$$

To determine the latest arrival time of the green vehicle k^g, we must account for the corresponding black vehicle k^b that transfers a load to vehicle k^g. That is, the green vehicle cannot arrive after the latest departure time of black vehicle k^b through the following formula :

$$lat_{tp^{k^g}_{p^\cdot}} = \min\{ldt_{tp^{k^g}_{p^\cdot}}, ldt_{tp^{k^b}_{p^\cdot}}\} \tag{8}$$

This backward procedure is applied until the the starting depot d^g_0 is reached and its latest departure time is determined. It should be noted that a forward propagation starting from the latest departure time at d^g_0, until d^g_{p+1} is reached, would produce the latest feasible schedule (i.e., latest possible arrival and departure times at each node along the green route).

5.4.2 Black route

Here, we explain how to propagate the earliest and latest arrival and departure times in a black route. For this purpose, let us consider the route of black vehicle $k^b \in K^B$ which is made of (1) a copy d^b_0 of the black depot to start the route, (2) an arbitrary sequence of length p of black (or neutral) customers and copies of one or more transfer points and (3) a copy d^b_{p+1} of the black depot to end the route.

Earliest arrival and departure times

The procedure to compute the earliest arrival and departure times in a black route is similar to the one described for the green route, but two differences are noteworthy: (1) the function AT^b is used to compute the arrival time at a given node from the earliest departure time of the previous node and (2) the earliest departure time at a copy of a transfer point is computed differently, because green vehicles that visit the same transfer point to get a load from the black vehicle must be accounted for.

Considering case (2), let us suppose that black vehicle k^b visits copy $tp^{k^b}_\cdot$ of tranfer point $tp \in TP$ and that h green vehicles $k^g_1, k^g_2, ..., k^g_h$ visit copies $tp^{k^g_1}_\cdot, tp^{k^g_2}_\cdot, ..., tp^{k^g_h}_\cdot$ of the same transfer point and that synchronization is required (i.e., black vehicle k^b must transfer a load to each green vehicle). To compute the earliest departure time of vehicle k_b at the transfer point, we first consider the maximum earliest arrival time over all green vehicles, that is:

$$eat^{max} = \max_{l=1,...,h}\{eat_{tp^{k^g_l}_\cdot}\} \tag{9}$$

Then, the earliest departure time of vehicle k^b at $tp^{k^b}_\cdot$ can be computed from its earliest arrival time as follow:

$$edt_{tp^{k^b}_\cdot} = \max\{eat^{max}, eat_{tp^{k^b}_\cdot}\} \tag{10}$$

That is, black vehicle k^b cannot depart earlier than the earliest arrival time of the last green vehicle, otherwise one or more green vehicles will not get their load.

Latest arrival and departure times

The procedure to compute the latest arrival and departure times of a black vehicle is similar to the one described for a green vehicle, although two differences are noteworthy:

(1) the function DT^b is used to compute the departure time from a given node to reach the next node at its latest arrival time and (2) the latest arrival time at a copy of a transfer point is computed differently, because green vehicles that visit the same transfer point to get a load from the black vehicle must be accounted for.

Considering case (2), let us suppose that black vehicle k^b visits copy $tp_{\cdot}^{k^b}$ of tranfer point $tp \in TP$ and that h green vehicles $k_1^g, k_2^g, ..., k_h^g$ visit copies $tp_{\cdot}^{k_1^g}, tp_{\cdot}^{k_2^g}, ..., tp_{\cdot}^{k_h^g}$ of the same transfer point and that synchronization is required (i.e., black vehicle k^b must transfer a load to each green vehicle). To compute the latest arrival time of vehicle k^b at the transfer point, we first consider the minimum latest departure time over all green vehicles, that is:

$$ldt^{min} = \min_{l=1,...,h} \{ldt_{tp_{\cdot}^{k_l^g}}\} \tag{11}$$

Then, the latest arrival time of vehicle k^b at $tp_{\cdot}^{k^b}$ can be computed from its latest departure time as follow:

$$lat_{tp_{\cdot}^{k^b}} = \min\{ldt^{min}, ldt_{tp_{\cdot}^{k^b}}\} \tag{12}$$

That is, black vehicle k^b cannot arrive at the transfer point later than the minimum latest departure time over all green vehicles that require synchronization, otherwise one or more green vehicles will not get their load.

5.5 Interaction among routes

In the previous section, our description of forward and backward propagation procedures to derive time bounds has focused on a single black or green route. However, complex interactions may occur when multiple black and green routes are involved.

Figure 5 shows an example where customer i is inserted between nodes *prev* and *next* in black route k_2^b (as it occurs during the recreate procedure of SISR). Forward propagation is illustrated in Figure 5(a). First, the ealiest arrival and departure times of the newly inserted customer i are calculated from the earliest departure time at transfer point *prev*. Then, forward propagation is triggered along the black route. However, a green route that connects the three black routes is encountered at the transfer point just after customer *next*. Thus, another forward propagation is triggered at this transfer point along the green route which, in turn, leads to a transfer point that connects the green route to black route k_3^b, thus triggering another forward propagation along that black route. It should be noted that this illustration is a worst case, because forward propagation along a route terminates as soon as the earliest departure time from a node does not change.

Backward propagation is illustrated in Figure 5(b). First, latest arrival and departure times at the newly inserted customer i are calculated from the latest arrival time at customer *next*. Then, backward propagation is triggered along black route k_2^b. Since node *prev* is a transfer point, it triggers another backward propagation along the corresponding green route. Still going backward along the black route, another transfer point is met that

triggers backward propagation along another green route. Finally, both green routes connect to black route k_1^bat the same transfer point, thus triggering a backward propagation along that black route also. Once again, this illustration is a worst case, because backward propagation along a route stops as soon as the latest arrival time at a node does not change.

To summarize, we had to implement forward and backward propagation procedures that account for the whole solution.

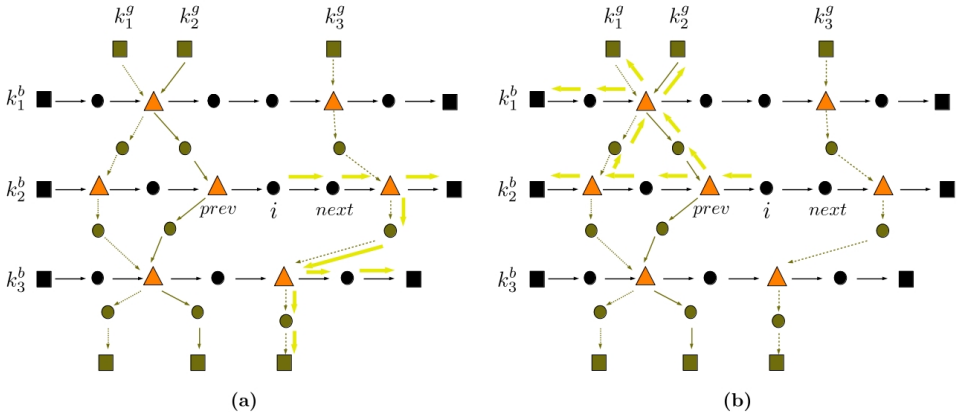

Figure 5: Example of forward and backward propagations when customer i is inserted in a route: (a) forward propagation (b) backward propagation.

6 SISR for the $TDVRPTWTP_{RN}$

The previous section has focused on time issues that arise when the $TDVRPTWTP_{RN}$ is considered. In the following, we revisit the SISR metaheuristic described in Section 4 given that our problem is much more complex than the CVRP.

Although the general algorithmic framework of SISR remains quite the same, there are important differences that can be observed in Algorithm 4. In the initialization phase, the dominant shortest path structures obtained from the underlying road network must first be created to account for the time-dependent travel times, see statements 5 and 6. More precisely, the DSPS of every pair of nodes that involves a black customer, a neutral customer, a transfer point or the black depot is generated, assuming that a black vehicle is used to travel between the two nodes. Then, the DSPS of every pair of nodes that involves a green customer, a neutral customer, a transfer point or the green depot is generated, assuming that a green vehicle is used to travel between the two nodes.

In statements 7-10, the adjacency list of each black, green and neutral customer i is generated. In the adjacency list $adj^b(i)$ of a black customer, only black and neutral customers are considered. In the adjacency list adj^g of a green customer, only green and neutral customers are considered. In the case of a neutral customer, since they can be

Algorithm 4 SISR for $TDVRPTWTP_{RN}$

1: Set $L^{\max}$ and $\overline{c}$
2: Set τ_0, τ_f and f
3: $\tau \leftarrow \tau_0$
4: $\rho \leftarrow \left(\frac{\tau_f}{\tau_0}\right)^{1/f}$
5: Generate DSPS^b for every admissible pair of nodes, assuming that a black vehicle is used
6: Generate DSPS^g for every admissible pair of nodes, assuming that a green vehicle is used
7: Generate $adj^b(i)$ of each black customer $i \in C^B$
8: Generate $adj^g(i)$ of each green customer $i \in C^G$
9: Generate $adj^b(i)$ and $adj^g(i)$ of each neutral customer $i \in C^E$
10: Generate $adj^{tp}(i)$ of each green and neutral customer $i \in C^G \cup C^E$
11: $s \leftarrow Initial\_Solution(k)$
12: **for** f iterations **do**
13: Calculate $\overline{c}_s^b$ and $\overline{c}_s^g$
14: $\overline{s}, A^b \leftarrow RuinBlack(s)$
15: $\overline{s}, A^g \leftarrow RuinGreen(\overline{s})$
16: $A^- \leftarrow A^b \cup A^g$
17: $\overline{s} \leftarrow Recreate(\overline{s}, A^-)$
18: **if** $Cost(\overline{s}) < (Cost(s) - \tau \ln(U(0,1)))$ **then**
19: $s \leftarrow \overline{s}$;
20: **end if**
21: **if** $Cost(\overline{s}) < Cost(s_{best})$ **then**
22: $s_{best} \leftarrow \overline{s}$
23: **end if**
24: $\tau \leftarrow \rho\tau$
25: **end for**
26: Return s_{best}

visited by both types of vehicles, two adjacency lists $adj^b(i)$ and $adj^g(i)$ are generated: one that contains only black and neutral customers (in case the neutral customer is in a black route) and the other that contains only green and neutral customers (in case the neutral customer is in a green route). Also, for each green and neutral customer, an adjacency list $adj^{tp}(i)$ made of all feasible transfer points, from closest to farthest from i, is created. A transfer point is feasible for a green or neutral customer, if it is possible for a green vehicle to receive the corresponding load from a black vehicle and serve the customer, while satisfying all constraints.

6.1 Initial solution

Another difference with the original SISR implementation is how the initial solution is generated in statement 11. A solution is constructed with a greedy insertion heuristic where, at each iteration, a customer is randomly selected and then inserted at its best place in the current partial solution. This is repeated until all customers are served. This insertion procedure is the same as the one used in the recreate operator, except that all feasible insertion places are considered (i.e., there is no blink so that no insertion place is skipped). Since this is a randomized heuristic, different runs typically produce different solutions. Accordingly, in the computational results, the greedy insertion heuristic was run 100 times

on each instance and the best solution obtained was chosen as the intial solution (preliminary experiments have shown that no significant improvement is observed beyond 100 runs).

At each iteration of the main loop in statements 12-25, *RuinBlack* and *RuinGreen* are applied in sequence to ruin black and green routes, respectively. In the original algorithm, parameter $\overline{c}$ determines the average number of customers that are removed from the current solution by the ruin operator. Since we have two types of routes, we define $\overline{c}_s^b$ and $\overline{c}_s^g$ in statement 13 to control the average number of customers that are removed from the black and green routes, respectively, of solution s with $\overline{c}_s^b + \overline{c}_s^g = \overline{c}$. The values of $\overline{c}_s^b$ and $\overline{c}_s^g$ are dynamically set at each iteration depending on the number of customers in black routes n_s^{BR} and number of customers in green routes n_s^{GR} in the current solution, using the formula :

$$\overline{c}_s^b = \left\lceil \left(\frac{n_s^{BR}}{n_s^{BR} + n_s^{GR}} \right) \cdot \overline{c} \right\rceil \tag{13}$$

with $\overline{c}_s^g = \overline{c} - \overline{c}_s^b$. This dynamic setting is required because the number of customers in the routes of black and green vehicles cannot be known a priori, due to the presence of neutral customers that can be served by both types of vehicles. After collecting in set A^- the two sets of removed customers A^b and A^g from the black and green routes, respectively (see statements 14 and 15), the *Recreate* operator is called in statement 17 with the ruined solution and the removed customers. When the new recreated solution $\overline{s}$ is returned, this new solution is compared with the current solution s in the same way as in the original algorithm, see statements 18-23.

In the following, we will now focus on the *Ruin* and *Recreate* procedures.

6.2 Ruin

Given that black and green routes do not have the same structure, a different *Ruin* operator has been designed for each type of route. It should first be noted that removing customers has no impact on solution feasibility (i.e., a feasible solution will remain feasible). Thus, the earliest and latest arrival and departure times at each node can be recalculated only once after the two ruin operators for black and green routes have been applied. The same applies to the solution value, which is of no interest while customers are removed from the solution.

6.2.1 Ruining black routes

The pseudo-code of *RuinBlack* is shown in Algorithm 5. The *RuinBlack* procedure is very similar to the *Ruin* procedure, except that the focus is only on black routes. Thus, the maximum length of a string l_s^{max} is taken as the minimum between L^{max} and the average number of customers in black routes *AvgCustInBlackRoutes(s)*, see statement 1, while the maximum number of strings n_s is calculated using l_s^{max} and $\overline{c}_s^b$ in statement 2. The seed customer must also be chosen in a black route, see statement 7. As in the original implementation, the ruin operator is randomly chosen between $String^b$ and $Split - String^b$,

which are adaptations of *String* and *Split-String*, see statement 14. In these two new operators, the transfer points in a black route are always preserved and only customers are removed. This is because a transfer point in a black route connects to one or more green routes, so that the green or neutral customer that follows the removed transfer point in each green route would have to be removed too.

Algorithm 5 RuinBlack(s)

1: $l_s^{max} \leftarrow \min\{L^{max}, AvgCustInBlackRoutes(s)\}$
2: Calculate n_s^{max} with l_s^{max} and $\overline{c}_s^b$
3: $n_s \leftarrow \lfloor U(1, n_s^{max} + 1) \rfloor$
4: $R^b \leftarrow \emptyset$
5: $A^b \leftarrow \emptyset$
6: $\overline{s} \leftarrow s \quad R_{\overline{s}} \leftarrow R_s$
7: Select randomly a seed customer i_{seed}^b over all black and neutral customers in black routes
8: **for** $i \in adj^b(i_{seed}^b)$ and $|R^b| < n_s$ **do**
9: **if** $(i \in C^B)$ or $(i \in C^E$ and i is in a black route) **then**
10: $r \leftarrow$ route of customer i
11: **if** $i \notin A^b$ and $r \notin R^b$ **then**
12: $l_r^{max} \leftarrow \min\{l_s^{max}, |r|\}$
13: $l_r \leftarrow \lfloor U(1, l_r^{max} + 1) \rfloor$
14: $RuinOp \leftarrow Random(String^b, Split - String^b)$
15: $A^b \leftarrow A^b \cup RuinOp(\overline{s}, r, l_r, i)$
16: $R^b \leftarrow R^b \cup \{r\}$
17: **if** r is empty **then**
18: $R_{\overline{s}} \leftarrow R_{\overline{s}} \backslash \{r\}$
19: **end if**
20: **end if**
21: **end if**
22: **end for**
23: Return $\overline{s}$, A^b

Figure 6 illustrates operator $String^b$, assuming that four customers must be removed. When two or more consecutive copies of the same transfer point are visited by the black vehicle after the string removal, then these copies are merged into a single copy. This is illustrated in the figure where two consecutive copies of the same transfer point tp_3 are merged together. Furthermore, the green routes that were connected to the original copies are collected and are all connected to the new single copy. The operator $Split - String^b$, works similarly, except that m customers in the string are preserved.

6.2.2 Ruining green routes

Ruining a green route also shares similarity with the original *Ruin* operator, except that the focus is on green routes. One particularity of *RuinGreen* is that the seed customer is the closest from the seed customer chosen in *RuinBlack*, over all green and neutral customers in green routes. The idea is to ruin green routes that are close to the previously ruined black routes. Another particularity is that each time a green or neutral customer is removed from a green route, the copy of the transfer point where the load is transferred from a

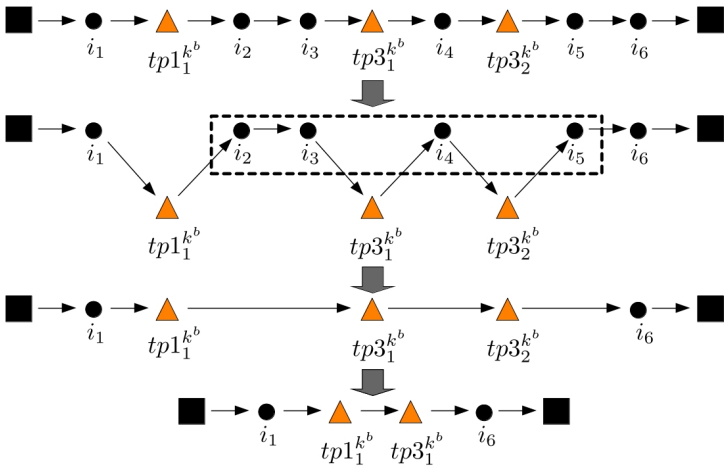

Figure 6: Example of $String^b$ where four customers are removed. First, the transfer points are set apart, then a string of cardinality four is chosen and the corresponding customers are removed.

black route to the green route must also be removed. That is, a pair $(tp_{\cdot}^{k^g}, i)$ is removed, where $tp_{\cdot}^{k^g}$ is the copy of transfer point $tp \in TP$ in the green route and i is the immediate successor customer, that is, the one who receives the load. The pseudo-code of *RuinGreen* is shown in Algorithm 6.

Algorithm 6 RuinGreen(s)

$l_s^{max} \leftarrow \min\{L^{max}, AvgCustInGreenRoutes(s)\}$
Calculate n_s^{max} with l_s^{max} and $\overline{c}_s^g$
$n_s \leftarrow \lfloor U(1, n_s^{\max} + 1) \rfloor$
$R^g \leftarrow \emptyset$
$A^g \leftarrow \emptyset$
Select i_{seed}^g the closest customer from i_{seed}^b over all green and neutral customers in green routes
for $i \in adj^g(i_{seed}^g)$ and $|R^g| < n_s$ **do**
    if $(i \in C^G)$ or $(i \in C^E$ and i is in a green route) **then**
        $r \leftarrow$ route of customer i
        if $i \notin A^g$ and $r \notin R^g$ **then**
            $l_r^{max} \leftarrow \min\{l_s^{max}, |r|\}$
            $l_r \leftarrow \lfloor U(1, l_r^{max} + 1) \rfloor$
            $RuinOp \leftarrow Random(String^g, Split - String^g)$
            $A^g \leftarrow A^g \cup RuinOp(\overline{s}, r, l_r, i)$
            $R^g \leftarrow R^g \cup \{r\}$
            if r is empty **then**
                $R_{\overline{s}} \leftarrow R_{\overline{s}} \backslash \{r\}$
            end if
        end if
    end if
end for
Return $\overline{s}$, A^g

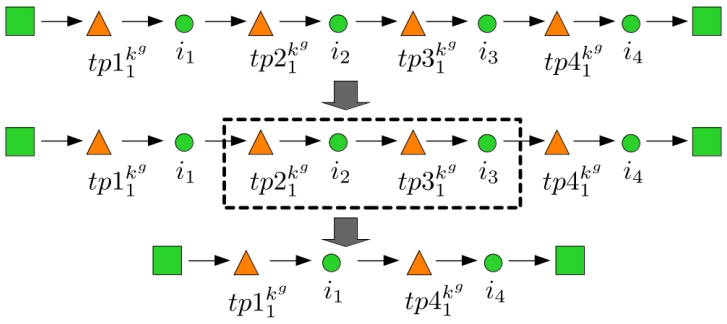

Figure 7: Example of $String^g$ where two customers are removed.

Figure 7 shows an example of $String^g$ where two customers are removed. It should be noted that removing a copy of a transfer point in a green route may have an impact on the corresponding black route if the green route is the only one that connects to the black route there. In this case, the corresponding copy of the transfer point in the black route must also be removed. Also, if this removal leads to the visit of two or more consecutive copies of another transfer point in the black route, then these copies are merged into a simple copy, as previously explained in subsection 6.2.1.

6.3 Recreate

Given that we have two different types of routes and three different types of customers, reinserting the removed customers is more complex than in the original implementation of SISR for the CVRP. As shown in Algorithm 7, the removed customers in set A^- are first sorted randomly or by decreasing demand, with equal probability (the two other sorting criteria proposed in [32], increasing and decreasing distance from the depot are not considered due to the ambiguity for neutral customers who can appear in black and green routes). Then, the customers are considered one by one in the sorted list. If the current customer is black then the method *InsertionBlack* for insertion in a black route is called. If the current customer is green then the method *InsertionGreen* for insertion in a green route is called. Finally, if the current customer is neutral then both methods are called and the best of the two proposed insertions is chosen. The procedure for inserting a customer in black routes is similar to the one presented in Algorithm 3 for the CVRP, except that the focus is on black routes only. However, the procedure for inserting a customer in a green route is more complex, as it is explained below.

It should also be noted that the *Recreate* method assumes that the vehicles visit nodes as soon as possible, that is, they follow a schedule based on the earliest arrival and departure times at each node. This earliest schedule induces slack time (waiting time) in the routes that can be exploited to feasibly insert new customers. But since we aim at minimizing the total duration of routes, the latest schedule is then used to get exact solution costs.

Algorithm 7 Recreate($\overline{s}$, A^-)

```
Sort(A^-)
for i ∈ A^- do
    if i is a black customer then
        s̄ ← InsertionBlack(i, s̄)
    end if
    if i is a green customer then
        s̄ ← InsertionGreen(i, s̄)
    end if
    if i is a neutral customer then
        s̄ ← Best of InsertionBlack(i, s̄) and InsertionGreen(i, s̄)
    end if
end for
Return s̄
```

6.3.1 Insertion in black routes

As in the original implementation for the CVRP, *InsertionBlack* considers all possible insertions of the current customer i between two consecutive nodes in black routes with enough residual capacity to accommodate the demand of i, except for blinks (i.e., a position may be skipped with a small probability γ).

It is worth noting that the feasibility and (approximate) evaluation of inserting customer i between two consecutive nodes j and l in a black route are done in constant time. First, the arrival and departure times at i are calculated from the departure time at j to check if the time window at i is satisfied. If i is feasible, then the new arrival time at l is calculated from the departure time at i. If the new arrival time at l does not exceed its latest arrival time lat_l, then the insertion is feasible. Assuming feasibility, the additional cost induced by this insertion is then evaluated. To maintain a constant time evaluation, an approximation is used. That is, the approximate or local additional cost corresponds to the arrival time delay at node l due to the insertion of customer i (even if l is the end depot). This delay corresponds to the difference between the arrival time at l after the insertion of i minus the arrival time at l before the insertion of i. To get an exact evaluation of the additional cost, the delay at l would need to be propagated along the route, until either it vanishes (due to waiting times) or the end depot is reached. It may also lead to forward propagation along other connecting routes (see Section 5.5). To alleviate the impact of using only approximate additional costs, the n_{pos} best insertion places of customer i, based on the approximation, are kept. Each one of these n_{pos} alternative insertion places are then evaluated exactly through propagation. The best insertion place, based on the exact evaluation of the additional cost, is finally chosen. It is worth noting that, after the insertion of customer i, the latest arrival and departure times need to be recomputed through backward propagation from i to the starting depot. This may also lead to backpropagation along other connecting routes (see Section 5.5). If no feasible insertion place is found for customer i then a new route is created for this customer.

6.3.2 Insertion in green routes

When a customer is inserted in a green route, a copy of a transfer point needs to be coupled with it. Accordingly, the search for possible insertion places is divided into four different phases, as it is explained below. It should be noted that the insertion procedure of each phase accounts for blinks. Furthermore, as for black routes, the feasibility and (approximate) evaluation of all possible insertion places of the current customer in green routes are done in constant time.

Phase I. Existing copy of a transfer point in a black route; existing green route.

If i is a green or neutral customer, we consider every black vehicle k^b with enough residual capacity to accommodate the demand of i. Then, we consider the route of every green vehicle k^g and every copy $tp_{\cdot}^{k^b}$ of a transfer point in the route of black vehicle k^b that satisfy the two following conditions: (1) the route of black vehicle k^b does not already connect with the route of green vehicle k^g through $tp_{\cdot}^{k^b}$ and (2) the transfer point is in set $adj^{tp}(i)$. Then, we create a corresponding copy $tp_{\cdot}^{k^g}$ for the green vehicle k^g and we try to insert the pair $(tp_{\cdot}^{k^g}, i)$ after every customer in the green route. An example is provided in Figure 8 where the two corresponding copies of a transfer point in the black and green routes are represented as a single node tp. A further refinement allows a reduction in the number of insertion places to be considered. Let us denote $\widetilde{tp}_L$ the last transfer point before tp that connects vehicles k^b and k^g (if any) and $\widetilde{tp}_R$ the first transfer point after tp that connects vehicles k^b and k^g (if any), see Figure 8. Then, there is no need to consider insertion places in the green route after customers that are visited before $\widetilde{tp}_L$. In such a case, $\widetilde{tp}_L$ would be visited before tp in the black route while $\widetilde{tp}_L$ would be visited after tp in the green route, thus no synchronization of the black and green vehicles is possible. Similarly, insertion places in the green route after customers that are visited after $\widetilde{tp}_R$ are discarded, since in such case $\widetilde{tp}_R$ would be visited after tp in the black route, but tp would be visited before $\widetilde{tp}_R$ in the green route.

It is important to note that the insertion of customer i impacts both the black and green routes. With regard to feasibility of the black route, the departure time of vehicle k^b at tp may be delayed since a load must now be transferred to green vehicle k^g (vehicle k^b may have to wait for vehicle k^g). However, it the new departure time does not exceed the latest departure time of k^b at tp, then the insertion is feasible in the black route. With regard to feasibility of the green route, we start with customer j^g after which the transfer point and customer i are inserted. That is, time is forward propagated from the latest arrival time at j^g to the transfer point, customer i and the following transfer point (or end depot), denoted l^g in the figure. If the time constraints are satisfied at the transfer point and at customer i, and if the new arrival time of the green vehicle at l^g does not exceed its latest arrival time then the insertion is feasible in the green route. The approximate or local cost of this insertion corresponds to the arrival time delay of black vehicle k^b at l^b plus the arrival time delay of green vehicle k^b at l^g.

Phase II. Existing copy of a transfer point in a black route; new green route.

This is similar to Phase I, except that a new green route for the current green or neutral customer i is created. This is illustrated in Figure 9. Feasibility can be checked similarly to Phase I. The approximate or local cost corresponds here to the duration of the new green route plus the arrival time delay of black vehicle k^b at l^b, given the new green route.

Phase III. New copy of a transfer point in a black route; existing green route.

Due to the additional computational complexity of this phase, we only consider the most interesting insertion places through a subset of $adj^{tp}(i)$ that contains the n_{ftp} nearest feasible transfer points of current customer i. For each such transfer point, we insert a copy of the transfer point at each possible insertion place in the route of each black vehicle with enough residual capacity to accommodate the demand of i. Then, for each such insertion place of this copy in a black route, we insert a corresponding copy of the transfer point followed by customer i after each green customer in each green route. This is illustrated in Figure 10 where the copies of the transfer point in the black and green routes are represented as a single node denoted *tp*. The transfer points denoted as $\widetilde{tp}_L$ and $\widetilde{tp}_R$ have the same meaning than in Phase I and are also used to reduce the number of insertion places that must be considered. In addition, there is no need to consider the insertion of the new copy before or after a copy of the same transfer point in the black route. These two consecutive copies of the same transfer point could then be merged, which would lead to a case already considered in Phase I (c.f., existing copy of a transfer point in a black route). Feasibility is checked as in Phase I and the approximate cost is obtained by summing the arrival time delay of the black vehicle at customer l^b and arrival time delay of the green vehicle at l^g.

Phase IV. New copy of a transfer point in a black route; new green route.

This is similar to Phase III, except that a new green route for the current green or neutral customer i is created, as illustrated in Figure 11. The approximate cost is obtained here by summing the duration of the new green route, plus the arrival time delay of the black vehicle at l^b, given the new green route.

The n_{pos} best insertion places according to the approximate cost, as identified during Phases I to IV, are then evaluated exactly through propagation. For evaluation purposes, the latest schedule is used to reduce as much as possible the waiting time. The best insertion place, based on the exact evaluation of the additional cost, is finally chosen. After the insertion of customer i at the chosen place, the latest arrival and departure times must be recomputed through backward propagation from customer i.

If no feasible insertion place is found for customer i, then two new routes are created to serve this customer, one for a black vehicle and one for a green vehicle. This is shown in Figure 12, where the nearest feasible transfer point from i is used to connect the two routes.

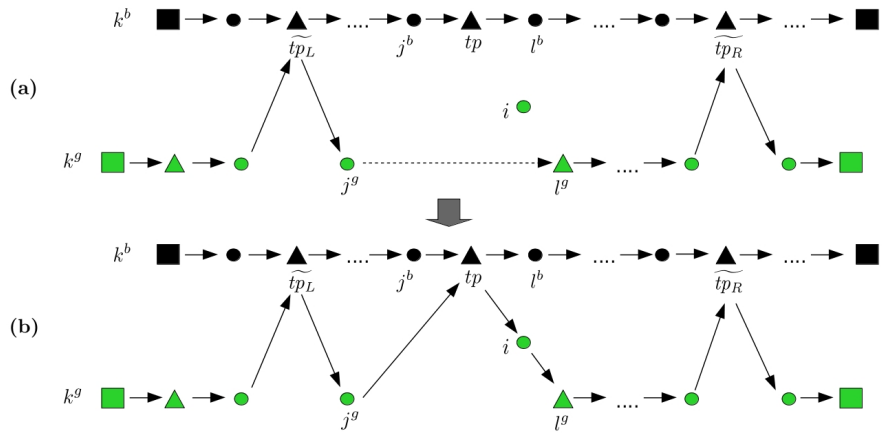

Figure 8: Phase I : a transfer point tp (already present in the black route) followed by customer i are inserted after customer j^g in the green route; (a) and (b) show the black and green routes before and after the insertion, respectively.

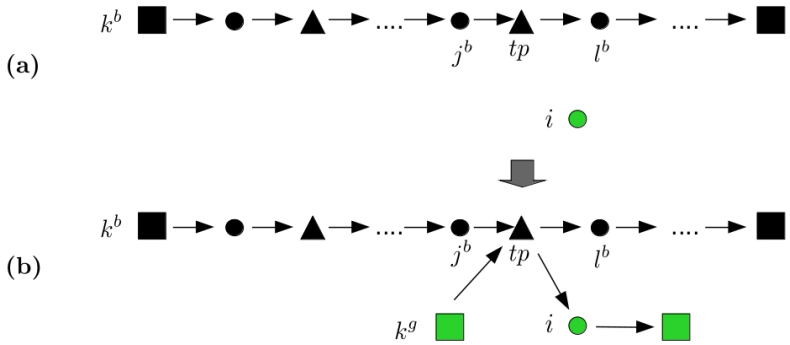

Figure 9: Phase II: a transfer point tp (already present in the black route) followed by customer i are inserted in a new green route; (a) and (b) show the black and green routes before and after the insertion, respectively.

7 Computational experiments

In the following sections, we first describe how the test instances were designed. Then, we explain the parameter tuning process of our SISR. This is followed by the results obtained on our test instances. Finally, we analyze the synchronization efficiency at transfer points and study the impact of customer distributions, scenarios, time windows and number of transfer points on the solutions obtained. We note that the results reported in this section were obtained with a processor Intel Gold 6148 Skylake 2.4 GHz, with 6GB of RAM.

7.1 Test instances

Our benchmark was produced by extending the well-known *NEWLET* instances for the $TDVRPTW_{RN}$ [33]. These Euclidean instances are defined on road networks generated by a procedure reported in [15]. Node coordinates in these networks are randomly generated

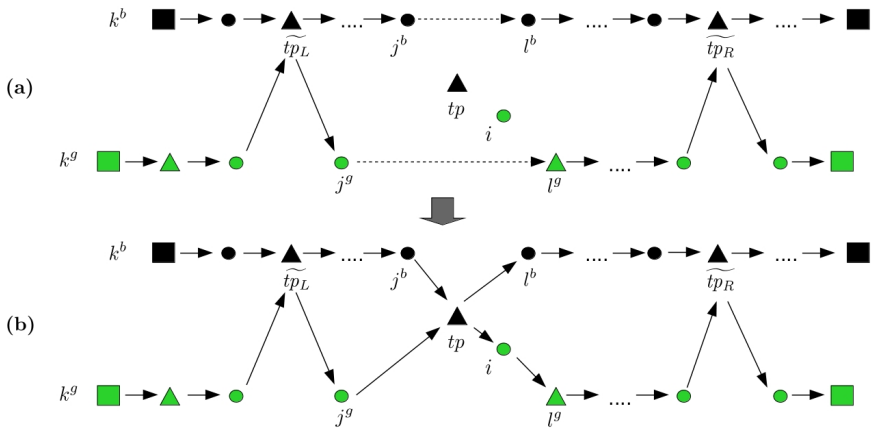

Figure 10: Phase III: a new transfer point tp is inserted in the black route; this transfer point followed by customer i are inserted after customer j^g in the green route; (a) and (b) show the black and green routes before and after the insertion, respectively.

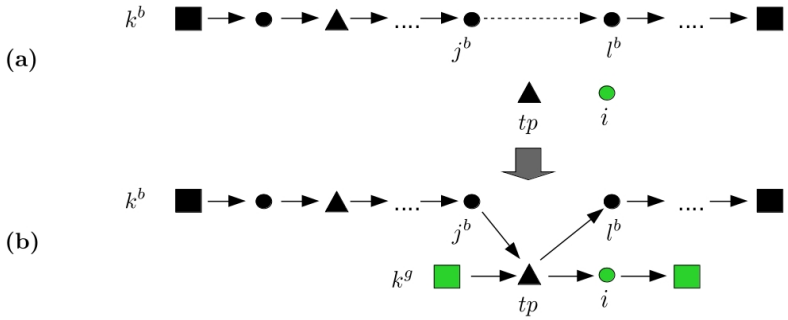

Figure 11: Phase IV: a new transfer point tp is inserted in the black route; this transfer point followed by customer i are inserted in a new green route; (a) and (b) show the black and green routes before and after the insertion, respectively.

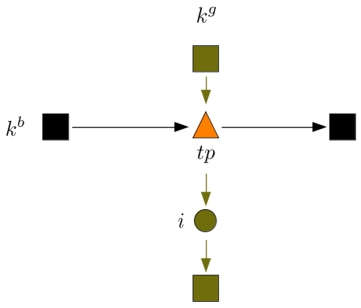

Figure 12: A new black and a new green route are created for a single green or neutral customer

in the interval $[-10\sqrt{n}, 10\sqrt{n}]$, where n is the number of nodes. The arc set E is obtained by considering all possible arcs and by adding an arc to set E when two conditions are satisfied: (1) the new arc does not cross any arc already included in E and (2) if the new arc has in common one endpoint with an arc already included in E, then the angle between these two arcs must be greater than or equal to 60 degrees.

In the $NEWLET$ instances, there are different networks of different sizes. For each road network of each size, three different ways to set the fixed or nominal travel time on every arc, depending on the correlation level with its length, are proposed through the formula $t_{ij} = \nu \cdot d_{ij} + \mu \cdot \gamma_{ij} \cdot \bar{d}$, where t_{ij} and d_{ij} are the travel time and length of arc (i, j), respectively. In the formula, $\bar{d}$ denotes the maximum arc length in the road network, ν and μ are correlation parameters selected in interval $[0, 1]$ and γ_{ij} is a randomly generated number in interval $[0, 1]$ for each arc (i, j). Thus, for each road network of each size, three different instances are obtained, which are denoted NC (i.e., no correlation, with $\nu = 0$ and $\mu = 1$), WC (i.e., weak correlation, with $\nu = 0.5$ and $\mu = 0.5$) and SC (i.e., strong correlation, with $\nu = 0.9$ and $\mu = 0.1$). The nominal travel times are then used to derive nominal travel speeds. Then, each one of these instances is duplicated by considering instances with narrow time windows (NTW) and wide time windows (WTW), where the length of narrow time windows is randomly selected from $\{3, 4\}$, and the length of wide time windows is randomly selected from $\{10, ..., 15\}$. To account for time-dependency, the time horizon $[0, 100]$ is partitioned into five periods $\tau_1 = [0, 20)$, $\tau_2 = [20, 30)$, $\tau_3 = [30, 70)$, $\tau_4 = [70, 80)$ and $\tau_5 = [80, 100]$. Based on this partition, one of three different time-dependent speed profiles is randomly associated with each arc, where a profile corresponds to a set of five speed multipliers, one for each time period. Finally, a service time is randomly selected from $\{1, 2\}$ for each customer.

For the computational tests, we used four SC road networks with 500 nodes, which is the largest network size. They are called RN_1, RN_2, RN_3 and RN_4 in the following. Since there is only one type of vehicles and one type of customers in the $NEWLET$ instances, we had to extend these instances to fit our purposes. The new characteristics of our test instances with regard to the original $NEWLET$ instances are the following:

- A number of $n_c = 50$, 100 and 200 customers are randomly selected from the 500 nodes of a given network.
- For a given road network and n_c value, four customer distributions are considered:
  - D_1: 60% black customers, 20% green customers and 20% neutral customers.
  - D_2: 20% black customers, 60% green customers and 20% neutral customers.
  - D_3: 20% black customers, 20% green customers and 60% neutral customers.
  - D_4: 40% black customers, 40% green customers and 20% neutral customers.
- Each road network is divided into three regions: downtown, boundary (or frontier) and outside, as illustrated in Figure 13. In this figure, the light gray area represents downtown, while the dark gray area represents the boundary region. Orange triangles

are transfer points, while the black and green squares are the black and green depot, respectively. Also, the black, green and gray nodes stand for black, green and neutral customers, respectively. The downtown area is defined by expanding a central and rectangular area until it can contain 80% of the maximum possible number of green customers (see distribution D_2), the green depot and additional road junctions. Then, the surrounding boundary or frontier region is expanded until it can contain the maximum possible number of neutral customers (see distribution D_3), 20% of the maximum possible number of black and green customers (see distributions D_1 and D_2), 10 transfer points and additional road junctions. The outside region contains the rest of the nodes.

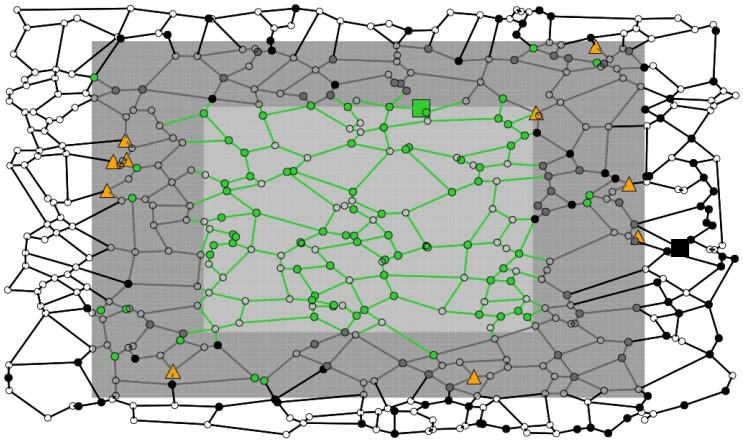

Figure 13: Example of an instance generated with road network RN_3.

- For all instances derived from a given road network, the three regions are the same as well as the location of the depots and transfer points.
- Black vehicles can perform deliveries to customers in the outside and boundary regions only. That is, downtown is forbidden to them. Conversely, the smaller green vehicles can perform deliveries to customers in the downtown and boundary regions only. For this reason, black customers are located in the outside and boundary regions, while green customers are located in the downtown and boundary regions. Neutral customers and transfer points are only found in the boundary region, since they can be visited by both black and green vehicles.
- The black depot is randomly located in the outside region, while the green depot is randomly located in the downtown region. There are also 10 transfer points that are randomly located in the boundary region.
- Arcs are characterized by the regions where they are found. Let (i, j) be an arc in the network. If both i and j are in the boundary (or frontier) region, then the arc is of type F (and is accessible to both black and green vehicles). If both i and j are in the downtown region, or one is in the downtown region and the other in the

boundary region, then the arc is of type D (and is accessible only to green vehicles). If both i and j are in the outside region, or if one is in the outside region and the other is in the boundary region, then the arc is of type O (and is accessible only to black vehicles). It should be noted that no arcs connect the downtown and outside regions. Given that the regions are the same for a given network, then the arc type also stays the same for all instances generated from a given road network.

- The test instances are duplicated by considering two different sets of travel speed multipliers (scenarios), where a speed multiplier depends on the time period, vehicle (black or green) and arc type (D, F, O), see Tables 1 and 2. In the two scenarios S_I and S_{II}, the second and fourth periods correspond to rush hours. The speed multipliers of green vehicles are fixed at 1 everywhere, which means that they are not affected by congestion since they are small (e.g., bicycles). Black vehicles are faster than green vehicles when there is no congestion. However, they are slower than green vehicles in the boundary region during rush hours in scenario S_I while they have the same speed than green vehicles in scenario S_{II}. The second scenario is aimed at evaluating the impact of increasing the speed of black vehicles when compared to green vehicles.
- The capacity of black vehicles is set to 40.
- The demand of each customer is randomly selected from $\{1, ..., 5\}$.

Overall, there are 4 road networks $\times$ 3 numbers of customers (n_c) $\times$ 4 customer distributions $\times$ 2 types of time windows $\times$ 2 scenarios for a total of 192 instances, that is, 96 instances for each scenario.

	Time period				
Arc type	$\tau_1 = [0, 20)$	$\tau_2 = [20, 30)$	$\tau_3 = [30, 70)$	$\tau_4 = [70, 80)$	$\tau_5 = [80, 100)$
Type F	1.2	0.8	1.2	0.8	1.2
Type O	1.5	1.0	1.5	1.0	1.5

(a) Black vehicles

	Time period				
Arc type	$\tau_1 = [0, 20)$	$\tau_2 = [20, 30)$	$\tau_3 = [30, 70)$	$\tau_4 = [70, 80)$	$\tau_5 = [80, 100)$
Type D	1.0	1.0	1.0	1.0	1.0
Type F	1.0	1.0	1.0	1.0	1.0

(b) Green vehicles

Table 1: Speed multipliers for (a) black vehicles and (b) green vehicles under scenario S_I

	Time period				
Arc type	$\tau_1 = [0, 20)$	$\tau_2 = [20, 30)$	$\tau_3 = [30, 70)$	$\tau_4 = [70, 80)$	$\tau_5 = [80, 100)$
Type F	1.5	1.0	1.5	1.0	1.5
Type O	2.0	1.5	2.0	1.5	2.0

(a) Black vehicles

	Time period				
Arc type	$\tau_1 = [0, 20)$	$\tau_2 = [20, 30)$	$\tau_3 = [30, 70)$	$\tau_4 = [70, 80)$	$\tau_5 = [80, 100)$
Type D	1.0	1.0	1.0	1.0	1.0
Type F	1.0	1.0	1.0	1.0	1.0

(b) Green vehicles

Table 2: Speed multipliers for (a) black vehicles and (b) green vehicles under scenario S_{II}

7.2 Parameter tuning

Four parameters have a significant impact on the performance of our SISR, namely, $\overline{c}$ (average number of removed customers), n_{pos} (number of best insertion positions, based on the approximation), n_{ftp} (number of nearest feasible transfer points) and L^{max} (maximum length of removed strings). To adjust their values, we selected a subset of 16 tuning instances with 100 customers by randomly selecting only one of the four networks, for each possible configuration of customer distribution (D_1, D_2, D_3, D_4), time window (NTW, WTW) and scenario (S_1, S_2). Given that solution quality tends to improve with increasing values of n_{pos} and n_{ftp}, at the expense of computation time, these two parameters were first set to high values, that is, $n_{pos} = 7$ and $n_{ftp} = 10$ (the latter value cannot be larger, since there are only 10 transfer points in each instance). In other words, we did not care at this point about computation time.

Then, we focused on parameters $\overline{c}$ and L^{max} and tuned them with the IRACE software [34], using $\overline{c} = \{5, ..., 17\}$ and $L^{max} = \{3, ..., 13\}$. The best values returned by IRACE were $\overline{c} = 15$ and $L^{max} = 8$. Based on the default configuration $\overline{c} = 15$, $n_{pos} = 7$, $n_{ftp} = 10$ and $L^{max} = 8$, we then modified the value of one parameter at a time, keeping the other parameters at their default value. The values considered for each parameter were: $\overline{c} = \{9, 11, 13, 15, 17, 19\}$, $n_{pos} = \{1, ..., 10\}$, $n_{ftp} = \{1, ..., 10\}$ and $L^{max} = \{1, ..., 10\}$. Since our algorithm is non deterministic, we show the average results (solution quality, computation time in hours) obtained over 10 runs on each tuning instance in Table 3.

As expected, increasing the values of parameters n_{pos} and n_{ftp} leads to an increase in computation time, although the impact is more significant in the case of n_{ftp}, since it increases the number of possible insertions of green and neutral customers in *Phase III* and *Phase IV* of the *Recreate* method (these insertions are quite complex). The computation times increase even more with increasing values of parameter $\overline{c}$ because more removed

		Parameter values									
	$\bar{c}$ =	**9**	**11**	**13**	**15**	**17**	**19**				
Avg. cost		2552.1	2532.7	2523.3	2520.3	2520.1	2520.0				
Avg. time		1.29	1.54	1.81	2.10	2.36	2.56				
	$\mathbf{n_{pos}}$ =	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Avg. cost		2537.9	2524.6	2521.4	2521.9	2520.3	2521.0	2520.3	2520.2	2519.6	2520.6
Avg. time		1.76	1.92	1.93	1.99	2.00	2.05	2.1	2.16	2.16	2.21
	$\mathbf{n_{ftp}}$ =	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Avg. cost		2531.2	2525.5	2522.0	2520.2	2520.6	2518.9	2520.1	2520.3	2520.4	2520.3
Avg. time		0.83	1.15	1.41	1.61	1.75	1.88	1.97	2.02	2.06	2.10
	$\mathbf{L^{max}}$ =	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Avg. cost		2542.9	2522.1	2520.2	2519.7	2520.4	2520.0	2520.1	2520.3	2519.8	2520.5
Avg. time		2.02	2.14	2.09	2.09	2.08	2.10	2.11	2.10	2.08	2.08

Table 3: Impact of parameter values on solution quality and computation times

customers simply mean more customers to be reinserted. On the other hand, parameter L^{max} has no impact on computation time. With regard to solution quality, we observe an improvement in solution quality for the first values of each parameter, but then some kind of stagnation is observed. Accordingly, the parameter setting $\bar{c}$ =15, $n_{pos} = 3$, $n_{ftp} = 4$ and $L^{max} = 4$ was chosen for the experiments reported in the following sections. We also checked that this particular combination of parameter values led to good solutions on the tuning instances, which turned to be true with an average solution cost of 2522.2 and average computation time of 1.47 hours.

Some experiments were also performed with regard to the number of iterations. We observed that convergence is obtained, even on the largest instances with 200 customers, after a maximum of 300,000 iterations. That is, a plateau is reached and no further significant improvement in solution quality is observed. Figure 14 shows an example of convergence curves for the best solutions found on instances with 200 customers generated with road network RN_4, using the four customer distributions, and both narrow and wide time windows, under scenario S_I. In this figure, black, green, gray and blue curves are associated with customer distributions D_1, D_2, D_3 and D_4, respectively, while full lines and broken lines are associated with instances with narrow and wide time windows, respectively. Based on the results obtained, the number of iterations was set to 300,000 for all instances (which is admittedly too much for instances with 50 and 100 customers).

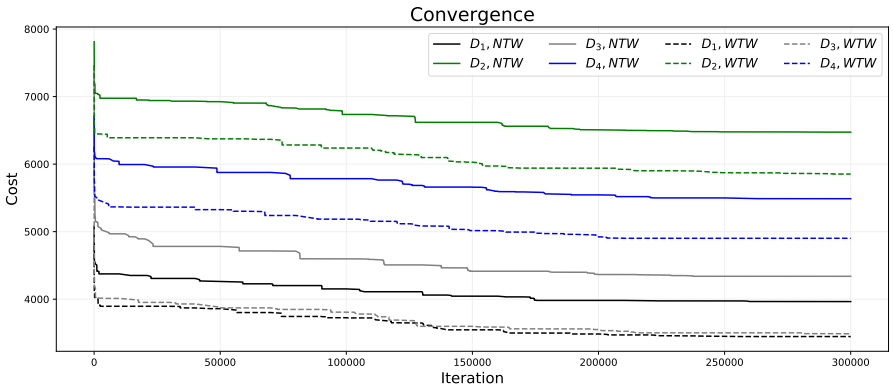

Figure 14: Convergence curves observed for instances with 200 customers generated with road network RN_4 under scenario S_I.

7.3 Results on test intances

We report in this section the results produced by our algorithm on the whole set of test instances, based on 10 different runs on each instance. Table 8 in the Appendix reports the best and average costs, as well as the average computation times in hours for each instance. Each line of this table corresponds to a particular type of instance using the notation RNx_ny_Dz, where x is the road network index, y is the number of customers and z is the customer distribution index. For each type, we show the results obtained on instances associated with narrow time windows (NTW) and wide time windows (WTW) under scenarios S_I and S_{II}.

Table 4 in this section is a reduced version, where averages are taken over the four road networks. That is, ny_Dz in Table 4 encompasses $RN1\_ny\_Dz$, $RN2\_ny\_Dz$, $RN3\_ny\_Dz$ and $RN4\_ny\_Dz$, so that the numbers in Table 4 correspond to the bold Avg. lines in the full table in the Appendix.

Since there are no similar instances in the literature that we could refer to for comparison purposes, Table 5 reports both the average best improvements and average improvements provided by our algorithm over the initial solutions, to measure its optimization power. Denoting s_i and s_f the initial and final solutions produced by our algorithm on a given instance, with $cost(s_i)$ and $cost(s_f)$ their respective cost, the percentage of improvement of the final solution over the initial one is calculated as follow:

$$Impr = 100 \left(\frac{cost(s_i) - cost(s_f)}{cost(s_i)} \right) \tag{14}$$

Tables 4 and 5 will be referred to in the following sections when we analyze in more detail the behavior of our algorithm for different characteristics of the test instances. For now, we can observe the obvious increase in solution cost and computation time with instance size in Table 4.

	NTW						WTW					
Instances	**Avg. Best Cost**		**Avg. Cost**		**Avg. Time**		**Avg. Best Cost**		**Avg. Cost**		**Avg. Time**	
	$\mathbf{S_I}$	$\mathbf{S_{II}}$	$\mathbf{S_I}$	$\mathbf{S_{II}}$	$\mathbf{S_I}$	$\mathbf{S_{II}}$	$\mathbf{S_I}$	$\mathbf{S_{II}}$	$\mathbf{S_I}$	$\mathbf{S_{II}}$	$\mathbf{S_I}$	$\mathbf{S_{II}}$
n50_D1	1397.8	1125.3	1398.9	1126.6	0.27	0.28	1219.0	929.5	1220.5	930.0	0.26	0.26
n50_D2	2002.5	1841.0	2004.5	1844.5	0.77	0.87	1801.4	1608.4	1805.2	1615.5	0.76	0.81
n50_D3	1368.4	1112.9	1374.8	1118.0	0.66	0.69	1117.0	899.2	1117.0	901.8	0.59	0.54
n50_D4	1748.9	1517.1	1752.0	1520.4	0.53	0.56	1542.4	1279.1	1543.8	1280.3	0.51	0.51
n100_D1	2498.1	2030.9	2505.7	2034.1	0.64	0.70	2128.9	1702.6	2140.4	1710.0	0.64	0.65
n100_D2	3684.7	3298.5	3696.0	3308.6	2.18	2.37	3314.9	2909.0	3326.5	2922.0	2.06	2.13
n100_D3	2368.0	1976.9	2372.9	1979.9	1.86	1.92	1916.3	1564.2	1921.6	1572.9	1.42	1.39
n100_D4	3138.3	2734.4	3148.2	2756.3	1.34	1.55	2743.1	2303.3	2754.8	2316.5	1.35	1.34
n200_D1	4186.9	3461.1	4221.5	3489.7	1.71	1.88	3582.6	2822.7	3610.5	2855.8	1.42	1.57
n200_D2	6980.9	6356.7	7028.4	6408.2	5.76	6.47	6321.5	5691.5	6380.1	5758.7	5.47	5.69
n200_D3	4399.9	3638.8	4439.4	3672.7	4.54	5.03	3636.4	2918.6	3663.4	2941.5	3.44	3.52
n200_D4	5613.9	4894.5	5642.2	4930.7	3.70	4.21	4927.8	4255.0	4967.9	4294.4	3.41	3.57
Overall Avg.	**3282.4**	**2832.3**	**3298.7**	**2849.1**	**2.00**	**2.21**	**2854.3**	**2407.0**	**2871.0**	**2425.0**	**1.78**	**1.83**

Table 4: Solution cost and computation time in hours for each subset of instances

	NTW				WTW			
Instances	**Avg. Best Impr.**		**Avg. Impr.**		**Avg. Best Impr.**		**Avg. Impr.**	
	$\mathbf{S_I}$	$\mathbf{S_{II}}$	$\mathbf{S_I}$	$\mathbf{S_{II}}$	$\mathbf{S_I}$	$\mathbf{S_{II}}$	$\mathbf{S_I}$	$\mathbf{S_{II}}$
n50_D1	13.02	14.14	10.48	11.19	16.03	18.38	13.48	15.03
n50_D2	13.28	15.33	11.21	13.15	16.75	19.36	14.28	16.51
n50_D3	33.53	41.27	30.95	38.51	35.42	43.14	32.02	39.05
n50_D4	11.20	14.87	9.49	11.64	15.06	18.21	12.62	15.52
n100_D1	17.59	20.28	15.96	18.46	22.32	24.56	20.33	22.70
n100_D2	17.55	19.68	15.34	17.67	19.48	22.70	17.63	20.68
n100_D3	41.76	46.49	39.68	44.93	44.31	48.55	42.14	45.87
n100_D4	14.67	17.62	13.27	15.28	20.16	22.18	17.98	20.56
n200_D1	25.57	27.73	24.00	25.46	29.57	31.24	27.73	29.54
n200_D2	18.76	22.39	17.32	21.05	21.01	23.97	19.50	22.20
n200_D3	46.53	52.75	45.27	51.05	48.17	54.24	46.51	52.07
n200_D4	19.75	22.61	18.46	20.89	23.95	25.71	21.95	23.81
Overall Avg.	**22.77**	**26.26**	**20.95**	**24.11**	**26.02**	**29.35**	**23.85**	**26.96**

Table 5: Average best improvements and average improvements for each subset of instances.

We also note the overall average improvements over the initial solutions in Table 5 that stand between 20% and 30%, which is substantial. Clearly, the greedy insertion heuristic is limited with regard to solution quality on such a complex problem, but still, these percentages of improvement show that our algorithm can take advantage of optimization opportunities.

7.4 Synchronization

In this section, we examine if synchronization between black and green vehicles at transfer points is efficient. For this purpose, we define the percentage of solution cost (duration) that corresponds to the total time that black and green vehicles spend at transfer points. For a given solution s, this percentage is denoted as ρ_s. In Equation (15), this percentage is defined using ΔTP_s^b and ΔTP_s^g, which are the total time spent at transfer points by black and green vehicles, respectively.

$$\rho_s = 100 \left(\frac{\Delta TP_s^b + \Delta TP_s^g}{cost(s)} \right) \tag{15}$$

Table 6 reports the minimum, maximum and average values of ρ_s on different subsets of instances. The format of this table is reduced when compared to Tables 4 and 5 by also averaging over the two types of time windows and the two scenarios. We observe that the ρ_s values are smaller for customer distributions D_1 and D_3. In fact, if we compute the average ρ_s values for D_1, D_2, D_3 and D_4, we obtain 0.97%, 1.97%, 0.94% and 1.57%, respectively. The fact that D_1 and D_3 lead to better synchronization than D_2 and D_4 can be explained ty their small percentage of green customers (20%), since they are the only ones for which synchronization at a transfer point is mandatory. In any case, the differences observed are small in absolute terms. The fact is that only 1.36% (overall average) of solution cost is due to synchronization, which indicates that synchronization is well achieved.

Instances	Min	Max	Avg.
n50_D1	0.12	2.18	0.77
n50_D2	1.05	3.71	2.08
n50_D3	0.03	2.55	1.02
n50_D4	0.18	3.23	1.60
Avg.	**0.34**	**2.92**	**1.37**
n100_D1	0.01	3.03	0.90
n100_D2	0.82	3.11	1.91
n100_D3	0.19	1.97	0.95
n100_D4	0.45	3.13	1.48
Avg.	**0.37**	**2.81**	**1.31**
n200_D1	0.45	2.45	1.25
n200_D2	0.84	3.37	1.91
n200_D3	0.25	2.20	0.86
n200_D4	0.53	2.85	1.63
Avg.	**0.52**	**2.71**	**1.41**
Overall Avg.	**0.01**	**3.71**	**1.36**

Table 6: Minimum, maximum and average values of ρ_s for each subset of instances

7.5 Customer distributions

Table 4 shows that customer distributions D_1 and D_3 are the best for solution cost, while D_1 outperforms the three other distributions for computation time. Both D_1 and D_3 have

only 20% of green customers, which is beneficial for solution cost because these customers require a detour at a transfer point for both a black and a green vehicle. Furthermore, D_1 has a large percentage of 60% of black customers (versus only 20% of neutral customers), which is helpful for computation time because only simple insertions in black routes need to be considered for black customers. Although D_3 is competitive with D_1 for solution cost, this is not the case for computation time. Distribution D_3 has 60% of neutral customers (versus only 20% of black customers) and their potential insertion in both black and green routes need to be considered. As opposed to black routes, insertion in green routes is quite complex and computationally expensive. With the largest percentage of 60% of green customers, distribution D_2 is consequently the worst for solution cost and computation time.

When considering improvements over initial solutions in Table 5, the largest improvements are associated with distribution D_3. This distribution has the largest percentage of neutral customers (60%) and these customers offer more flexibility for optimization because they can be inserted either in black or green routes. In this case, the average percentage of neutral customers belonging to black routes in the initial solutions ranges between 51.1% and 54.4%. However, in the final solutions, these percentages drastically increase between 94.6% and 98.6%. That is, the optimization algorithm finds ways to move a large proportion of neutral customers in black routes, which decreases solution cost. Accordingly, more neutral customers means more opportunities for improvement.

7.6 Time windows

Tables 4 and 5 show that better solution costs and larger improvements over initial solutions are associated with instances with wide time windows. Clearly, these time windows offer more feasible insertion places for customers and, consequently, greater flexibility for the optimization procedure to move them around. For example, the percentage of neutral customers in black routes for the instances with wide time windows is 98.3%, as compared with 92.8% for instances with narrow time windows. We also observed a reduced number of routes in the solutions obtained on instances with wide time windows, when compared with narrow time windows, namely 25.2% less black routes and 15.1% less green routes. Finally, black vehicles visit 2% more transfer points on average in the presence of wide time windows.

7.7 Scenarios

Better solution costs are observed in Table 4 under scenario S_{II}, when compared with scenario S_I. Since black vehicles now travel faster, the time windows at black and neutral customers become easier to satisfy. We also observed that, on average, a larger percentage of neutral customers is served by black vehicles in the final solutions under scenario S_{II} (97.4%) when compared to scenario S_I (93.7%). It is generally less costly to serve neutral customers with black vehicles and the latter can get a larger share of neutral customers when their speed increases. In other words, they win more often the "battle" for neutral customers against green vehicles in the boundary region. We also observed a reduction of 14.7% and 6.2% in the number of black and green routes, respectively, under scenario S_{II}, with corresponding increases of 20.5% and 3.5% in the average number of customers in

black and green routes, respectively. Finaly, black vehicles visit 4.4% more transfer points on average, when compared to scenario S_I.

7.8 Number of transfer points

We propose here an experiment where we gradually reduce the number of transfer points to see the impact on the solutions obtained. To this end, some transfer points are randomly selected and transformed into simple nodes (road junctions). In some cases, infeasible instances may be created (e.g., it may not be possible for any green vehicle to visit a transfer point and serve a green customer before the upper bound of its time window).

Here, we focus on the test instances with 100 customers. First, two transfer points are randomly chosen and removed from the 10 original ones to obtain instances with 8 transfer points. From these 8 transfer points, the procedure is repeated to get instances with 6 transfer points. Finally, two additional transfer points are removed to get instances with 4 transfer points. Ten runs were performed on each feasible instance with a reduced number of transfer points. Table 7 shows the average gaps between the average cost of solutions obtained with 10 transfer points and with $k = 8, 6, 4$ transfer points, based on Equation (16). Each line in this table is the average over four instances, considering that there are two types of time windows and two scenarios.

$$Gap = 100 \left(\frac{ave_{cost}^{|TP|=k} - ave_{cost}^{|TP|=10}}{ave_{cost}^{|TP|=10}} \right) \tag{16}$$

We observed no infeasible instance with $k = 8$ transfer points, one infeasible instance with $k = 6$ transfer points and 19 infeasible instances (out of 64 instances) with $k = 4$. Thus, some averages are computed with less than four gaps. When there is no value, the four instances are infeasible. A few small negative values appear in the table, which means that slightly better solutions are obtained with fewer transfer points. This situation can occur, due to the randomized nature of our algorithm, when the removed transfer points are not or seldom used in the original instances, thus producing no or little impact on solution quality.

Obviously, removing transfer points generally lead to worse solutions and this trend is more pronounced when more transfer points are removed. Customer distribution D_2 is more affected than the other distributions due to its large percentage of green customers (which must use transfer points). We also see that instances associated with road network RN_1 are greatly affected when 4 or 6 transfer points are removed. In the solutions obtained with 10 transfer points, we observed that some transfer points are much more exploited than others. Clearly, such critical transfer points are more likely to disappear when more transfer points are removed, which in turn greatly impact solution quality.

Instances	Avg. Gap		
	$k = 8$	$k = 6$	$k = 4$
RN1_n100_D1	-0.04	6.02	– – –
RN2_n100_D1	-0.10	-0.11	0.47
RN3_n100_D1	-0.02	-0.04	0.51
RN4_n100_D1	0.51	1.74	1.76
RN1_n100_D2	2.06	10.95	27.45
RN2_n100_D2	1.56	1.58	5.86
RN3_n100_D2	0.62	1.56	9.08
RN4_n100_D2	1.81	7.09	7.30
RN1_n100_D3	0.48	7.32	16.08
RN2_n100_D3	-0.05	-0.09	0.48
RN3_n100_D3	1.12	2.34	6.55
RN4_n100_D3	1.24	1.79	1.82
RN1_n100_D4	1.00	9.72	– – –
RN2_n100_D4	-0.86	-0.75	-0.02
RN3_n100_D4	0.19	1.21	4.45
RN4_n100_D4	0.86	1.47	1.68

Table 7: Average gaps between average solution costs with 10 transfer points and $k = 8$, 6, 4 transfer points

8 Conclusion

In this work, we have proposed a new SISR metaheuristic for the $TDVRPTWTP_{RN}$. To the best of our knowledge, this challenging problem where customers can be served either directly by black vehicles or indirectly by green vehicles through transfer points has not been previously addressed in the literature. Computational results on test instances with different characteristics show that our algorithm performs well, in particular by finding ways to transfer more neutral customers into black routes, which lead to solutions of better quality. Furthermore, we observed that the time spent by vehicles at transfer points is very low, thus indicating that good synchronization is achieved.

For the future, new ruin operators could be developed to enhance the performance of our algorithm and solve other complex variants of vehicle routing problems. It would also be interesting to consider the integration of learning into our algorithm, for example to identify the most promising transfer points, based on the topology of the network and distribution of customers.

Acknowledgements. The work reported in this paper was financially supported by the Natural Sciences and Engineering Research Council (NSERC) of Canada. This support is gratefully acknowledged.

Appendix

Instances	NTW Best Cost (10 runs)		NTW Avg. Cost (10 runs)		NTW Avg. Time (hours)		WTW Best Cost (10 runs)		WTW Avg. Cost (10 runs)		WTW Avg. Time (hours)	
	S_I	S_{II}	S_I	S_{II}	S_I	S_{II}	S_I	S_{II}	S_I	S_{II}	S_I	S_{II}
RN1_n50_D1	1563.47	1217.90	1563.91	1217.90	0.30	0.29	1331.90	1030.23	1332.03	1030.23	0.26	0.29
RN2_n50_D1	1365.48	1096.91	1365.48	1096.91	0.25	0.23	1159.71	898.09	1165.04	898.09	0.26	0.24
RN3_n50_D1	1430.09	1159.05	1432.17	1159.65	0.25	0.27	1292.61	950.84	1292.81	950.85	0.25	0.24
RN4_n50_D1	1232.26	1027.51	1233.85	1031.95	0.28	0.34	1091.71	839.01	1092.13	840.98	0.27	0.28
Avg.	**1397.83**	**1125.34**	**1398.85**	**1126.60**	**0.27**	**0.28**	**1218.98**	**929.54**	**1220.50**	**930.04**	**0.26**	**0.26**
RN1_n50_D2	2295.75	2067.62	2295.83	2069.44	0.81	0.90	2018.20	1737.63	2018.72	1742.08	0.82	0.89
RN2_n50_D2	1929.28	1752.89	1932.68	1753.13	0.74	0.81	1664.44	1476.99	1664.84	1478.37	0.71	0.77
RN3_n50_D2	1996.08	1864.24	1996.14	1875.20	0.69	0.71	1843.62	1696.06	1855.69	1713.47	0.63	0.71
RN4_n50_D2	1788.88	1679.38	1793.19	1680.13	0.83	1.06	1679.18	1523.08	1681.66	1527.97	0.88	0.88
Avg.	**2002.50**	**1841.03**	**2004.46**	**1844.48**	**0.77**	**0.87**	**1801.36**	**1608.44**	**1805.23**	**1615.47**	**0.76**	**0.81**
RN1_n50_D3	1440.17	1183.37	1449.07	1184.69	0.65	0.68	1234.84	983.12	1234.86	985.14	0.61	0.53
RN2_n50_D3	1365.98	1087.79	1378.58	1105.24	0.72	0.66	1059.19	845.14	1059.19	853.41	0.57	0.53
RN3_n50_D3	1353.17	1120.44	1356.42	1121.83	0.57	0.61	1094.61	931.62	1094.61	931.62	0.59	0.53
RN4_n50_D3	1314.31	1059.94	1314.97	1060.22	0.70	0.78	1079.52	836.87	1079.52	837.14	0.60	0.58
Avg.	**1368.41**	**1112.88**	**1374.76**	**1118.00**	**0.66**	**0.69**	**1117.04**	**899.19**	**1117.05**	**901.83**	**0.59**	**0.54**
RN1_n50_D4	1786.66	1587.00	1786.66	1587.00	0.49	0.53	1679.56	1365.57	1679.56	1365.71	0.56	0.47
RN2_n50_D4	1944.47	1629.15	1947.70	1630.98	0.54	0.51	1622.76	1328.29	1622.76	1330.34	0.57	0.61
RN3_n50_D4	1535.58	1376.85	1537.07	1386.52	0.48	0.47	1355.06	1195.54	1355.06	1197.90	0.41	0.45
RN4_n50_D4	1728.69	1475.37	1736.48	1477.26	0.61	0.72	1512.20	1227.06	1517.89	1227.06	0.49	0.51
Avg.	**1748.85**	**1517.09**	**1751.98**	**1520.44**	**0.53**	**0.56**	**1542.39**	**1279.12**	**1543.82**	**1280.25**	**0.51**	**0.51**
RN1_n100_D1	2526.91	2085.89	2528.87	2088.53	0.64	0.70	2189.86	1662.22	2198.75	1675.03	0.65	0.56
RN2_n100_D1	2559.89	2067.83	2570.32	2071.26	0.53	0.65	2161.96	1726.10	2177.80	1727.87	0.56	0.59
RN3_n100_D1	2540.37	2113.93	2540.48	2115.17	0.64	0.65	2200.53	1889.07	2202.66	1891.56	0.70	0.78
RN4_n100_D1	2365.21	1856.08	2383.25	1861.46	0.73	0.81	1963.41	1533.17	1982.42	1545.58	0.63	0.69
Avg.	**2498.09**	**2030.93**	**2505.73**	**2034.11**	**0.64**	**0.70**	**2128.94**	**1702.64**	**2140.41**	**1710.01**	**0.64**	**0.65**
RN1_n100_D2	3804.11	3452.62	3812.33	3461.66	2.07	2.22	3426.28	2988.77	3434.23	3006.73	2.03	1.99
RN2_n100_D2	3537.95	3237.98	3543.50	3246.02	1.84	2.37	3256.69	2866.68	3265.31	2873.14	2.05	2.03
RN3_n100_D2	3919.34	3520.71	3936.62	3534.99	2.12	2.18	3538.40	3158.21	3551.07	3168.52	1.79	1.98
RN4_n100_D2	3477.32	2982.62	3491.33	2991.65	2.69	2.72	3038.18	2622.38	3055.51	2639.41	2.36	2.53
Avg.	**3684.68**	**3298.48**	**3695.95**	**3308.58**	**2.18**	**2.37**	**3314.89**	**2909.01**	**3326.53**	**2921.95**	**2.06**	**2.13**
RN1_n100_D3	2480.34	1996.41	2488.51	1997.98	1.83	1.70	2124.82	1635.38	2127.03	1645.98	1.58	1.29
RN2_n100_D3	2315.24	1990.81	2319.78	1996.14	1.70	1.71	1923.52	1612.65	1930.14	1618.16	1.37	1.44
RN3_n100_D3	2435.62	2051.60	2440.28	2054.78	2.09	2.03	1851.77	1569.09	1856.39	1573.41	1.32	1.36
RN4_n100_D3	2240.82	1868.86	2243.17	1870.65	1.82	2.23	1764.91	1439.53	1772.65	1454.09	1.40	1.47
Avg.	**2368.00**	**1976.92**	**2372.94**	**1979.89**	**1.86**	**1.92**	**1916.26**	**1564.16**	**1921.55**	**1572.91**	**1.42**	**1.39**
RN1_n100_D4	3534.04	2929.31	3538.32	2940.50	1.56	1.65	3044.64	2398.28	3048.74	2415.50	1.43	1.31
RN2_n100_D4	2807.13	2505.56	2827.05	2573.06	1.09	1.27	2444.41	2141.93	2456.91	2151.48	1.07	1.16
RN3_n100_D4	3161.54	2876.90	3168.41	2882.75	1.28	1.64	2742.29	2428.80	2765.12	2448.75	1.48	1.50
RN4_n100_D4	3050.61	2625.61	3058.81	2628.78	1.43	1.65	2741.18	2244.28	2748.61	2250.31	1.41	1.38
Avg.	**3138.33**	**2734.35**	**3148.15**	**2756.27**	**1.34**	**1.55**	**2743.13**	**2303.32**	**2754.84**	**2316.51**	**1.35**	**1.34**
RN1_n200_D1	4586.99	3673.59	4607.01	3698.37	1.97	1.84	3890.26	2964.06	3911.65	3002.76	1.59	1.63
RN2_n200_D1	4006.30	3338.27	4037.61	3380.21	1.41	1.76	3420.59	2774.14	3444.88	2805.14	1.32	1.62
RN3_n200_D1	4235.06	3605.87	4287.61	3632.18	1.83	1.97	3632.22	3011.05	3649.54	3032.49	1.27	1.43
RN4_n200_D1	3919.20	3226.55	3953.88	3247.98	1.62	1.95	3387.47	2541.72	3436.05	2582.96	1.52	1.59
Avg.	**4186.89**	**3461.07**	**4221.53**	**3489.68**	**1.71**	**1.88**	**3582.63**	**2822.74**	**3610.53**	**2855.84**	**1.42**	**1.57**
RN1_n200_D2	7428.48	6656.70	7468.67	6667.52	5.64	6.09	6650.45	5893.97	6692.08	5936.45	5.23	5.60
RN2_n200_D2	6883.99	6320.66	6938.59	6428.07	5.29	6.62	6220.31	5650.36	6267.01	5712.54	5.40	5.85
RN3_n200_D2	7143.03	6595.11	7201.80	6636.35	6.35	6.69	6615.83	6020.32	6681.27	6079.96	5.38	5.24
RN4_n200_D2	6468.03	5854.25	6504.63	5900.71	5.76	6.48	5799.34	5201.53	5879.89	5306.00	5.86	6.07
Avg.	**6980.88**	**6356.68**	**7028.42**	**6408.16**	**5.76**	**6.47**	**6321.48**	**5691.54**	**6380.06**	**5758.74**	**5.47**	**5.69**
RN1_n200_D3	4333.66	3631.80	4385.00	3671.93	4.12	4.57	3647.47	2941.04	3684.66	2960.16	2.97	2.99
RN2_n200_D3	4336.39	3638.83	4385.59	3681.60	4.46	5.27	3622.43	2916.56	3644.45	2943.40	3.56	4.12
RN3_n200_D3	4600.69	3735.66	4628.11	3767.51	4.49	4.84	3818.68	3096.20	3835.78	3112.43	3.53	3.35
RN4_n200_D3	4328.94	3548.88	4358.82	3569.77	5.06	5.45	3457.12	2720.75	3488.66	2749.87	3.69	3.63
Avg.	**4399.92**	**3638.79**	**4439.38**	**3672.70**	**4.54**	**5.03**	**3636.43**	**2918.64**	**3663.39**	**2941.46**	**3.44**	**3.52**
RN1_n200_D4	5863.31	5023.25	5872.71	5048.40	4.19	4.35	5056.61	4284.61	5081.22	4323.29	3.37	3.26
RN2_n200_D4	5302.13	4717.90	5314.75	4770.30	2.82	3.42	4661.26	4074.33	4685.65	4125.05	2.93	3.15
RN3_n200_D4	5842.90	5220.16	5902.48	5258.46	4.38	4.93	5141.24	4588.17	5213.82	4629.61	3.86	3.95
RN4_n200_D4	5447.38	4616.55	5478.77	4645.50	3.42	4.17	4851.89	4073.05	4890.97	4099.56	3.50	3.93
Avg.	**5613.93**	**4894.47**	**5642.18**	**4930.67**	**3.70**	**4.21**	**4927.75**	**4255.04**	**4967.92**	**4294.38**	**3.41**	**3.57**
Overall avg.	**3282.36**	**2832.34**	**3298.69**	**2849.13**	**2.00**	**2.21**	**2854.27**	**2406.95**	**2870.99**	**2424.95**	**1.78**	**1.83**

Table 8: Solution cost and computation time in hours for each instance.

References

[1] J. Beasley, Adapting the savings algorithm for varying inter-customer travel times, Omega 9 (1981) 658–659.

[2] C. Malandraki, M. Daskin, Time dependent vehicle routing problems: Formulations, properties and heuristic algorithms, Transportation Science 26 (1992) 185–200.

[3] A. V. Hill, W. C. Benton, Modelling intra-city time-dependent travel speeds for vehicle scheduling problems, Journal of the Operational Research Society 43 (1992) 343–351.

[4] S. Ichoua, M. Gendreau, J.-Y. Potvin, Vehicle dispatching with time-dependent travel times, European Journal of Operational Research 144 (2003) 379–396.

[5] A. V. Donati, R. Montemanni, N. Casagrande, A. E. Rizzoli, L. M. Gambardella, Time dependent vehicle routing problem with a multi ant colony system, European Journal of Operational Research 185 (2008) 1174–1191.

[6] S. Dabia, S. Röpke, T. Van Woensel, T. de Kok, Branch and price for the time-dependent vehicle routing problem with time windows, Transportation Science 47 (2013) 380–391.

[7] B. Pan, Z. Zhang, A. Lim, A hybrid algorithm for time-dependent vehicle routing problem with time windows, Computers & Operations Research 128 (2021) 105193.

[8] B. Pan, Z. Zhang, A. Lim, Multi-trip time-dependent vehicle routing problem with time windows, European Journal of Operational Research 291 (2021) 218–231.

[9] A. Haghani, S. Jung, A dynamic vehicle routing problem with time-dependent travel times, Computers & Operations Research 32 (2005) 2959–2986.

[10] S. Balseiro, I. Loiseau, J. Ramonet, An ant colony algorithm hybridized with insertion heuristics for the time dependent vehicle routing problem with time windows, Computers & Operations Research 38 (2011) 954–966.

[11] M. Gendreau, G. Ghiani, E. Guerriero, Time-dependent routing problems: A review, Computers & Operations Research 64 (2015) 189–197.

[12] T. Garaix, C. Artigues, D. Feillet, D. Josselin, Vehicle routing problems with alternative paths: An application to on-demand transportation, European Journal of Operational Research 204 (2010) 62–75.

[13] H. Ben Ticha, N. Absi, D. Feillet, A. Quilliot, Empirical analysis for the VRPTW with a multigraph representation for the road network, Computers & Operations Research 88 (2017) 103–116.

[14] H. Ben Ticha, N. Absi, D. Feillet, A. Quilliot, Multigraph modeling and adaptive large neighborhood search for the vehicle routing problem with time windows, Computers & Operations Research 104 (2019) 113–126.

[15] A. N. Letchford, S. D. Nasiri, A. Oukil, Pricing routines for vehicle routing with time windows on road networks, Computers & Operations Research 51 (2014) 331–337.

[16] H. Ben Ticha, N. Absi, D. Feillet, A. Quilliot, T. Van Woensel, A branch-and-price algorithm for the vehicle routing problem with time windows on a road network, Networks 73 (2019) 401–417.

[17] H. Ben Ticha, N. Absi, D. Feillet, A. Quilliot, T. Van Woensel, The time-dependent vehicle routing problem with time windows and road-network information, SN Operations Research Forum 2 (2021) 4.

[18] M. Gmira, M. Gendreau, A. Lodi, J.-Y. Potvin, Tabu search for the time-dependent vehicle routing problem with time windows on a road network, European Journal of Operational Research 288 (2021) 129–140.

[19] M. Speranza, G. Guastaroba, D. Vigo, Intermediate facilities in freight transportation planning: A survey, Transportation Science 50 (2016) 763–789.

[20] J. Van Belle, P. Valckenaers, D. Cattrysse, Cross-docking: State of the art, Omega 40 (2012) 827–846.

[21] P. Bouros, D. Sacharidis, T. Dalamagas, T. Sellis, Dynamic pickup and delivery with transfers, in: M. Gertz, M. Renz, X. Zhou, E. Hoel, W.-S. Ku, A. Voisard, C. Zhang, H. Chen, L. Tang, Y. Huang, C.-T. Lu, S. Ravada (Eds.), Advances in Spatial and Temporal Databases, Lecture Notes in Computer Science 10411, 2011, pp. 112–129.

[22] S. Minic, G. Laporte, The pickup and delivery problem with time windows and transshipment, INFOR 44 (2006) 217–227.

[23] T. G. Crainic, N. Ricciardi, G. Storchi, Models for evaluating and planning city logistics systems, Transportation Science 43 (2009) 432–454.

[24] T. G. Crainic, S. Mancini, G. Perboli, R. Tadei, Multi-start heuristics for the two-echelon vehicle routing problem, in: P. Merz, J.-K. Hao (Eds.), Evolutionary Computation in Combinatorial Optimization, Lecture Notes in Computer Science 6622, 2011, pp. 179–190.

[25] G. Perboli, R. Tadei, R. Tadei, New families of valid inequalities for the two-echelon vehicle routing problem, Electronic Notes in Discrete Mathematics 36 (2010) 639–646.

[26] G. Perboli, R. Tadei, D. Vigo, The two-echelon capacitated vehicle routing problem: Models and math-based heuristics, Transportation Science 45 (2009) 364–380.

[27] S. Jia, L. Deng, Q. Zhao, Y. Chen, An adaptive large neighborhood search heuristic for multi-commodity two-echelon vehicle routing problem with satellite synchronization, Journal of Industrial & Management Optimization 19 (2022) 1187–1210.

[28] A. Anderluh, V. Hemmelmayr, P. Nolz, Synchronizing vans and cargo bikes in a city distribution network, Central European Journal of Operations Research 25 (2017) 345–376.

[29] P. Grangier, M. Gendreau, F. Lehuédé, L.-M. Rousseau, An adaptive large neighborhood search for the two-echelon multiple-trip vehicle routing problem with satellite synchronization, European Journal of Operational Research 254 (2016) 80–91.

[30] N. Kafle, B. Zou, J. Lin, Design and modeling of a crowdsource-enabled system for urban parcel relay and delivery, Transportation Research Part B: Methodological 99 (2017) 62–82.

[31] A. Sampaio, M. Savelsbergh, L. Veelenturf, T. Van Woensel, Delivery systems with crowd-sourced drivers: A pickup and delivery problem with transfers, Networks 76 (2020) 232–255.

[32] J. Christiaens, G. Vanden Berghe, Slack induction by string removals for vehicle routing problems, Transportation Science 54 (2020) 417–433.

[33] H. Ben Ticha, Vehicle routing problems with road-network information, Ph.D. thesis, Université Clermont Auvergne (2017).

[34] M. López-Ibáñez, J. Dubois-Lacoste, L. Pérez Cáceres, T. Stützle, M. Birattari, The IRACE package: Iterated racing for automatic algorithm configuration, Operations Research Perspectives 3 (2016) 43–58.