A Hybrid Metaheuristic for Single Truck and Trailer Routing Problems

Submitted to *Transportation Science*
manuscript (Please, provide the manuscript number!)

A Hybrid Metaheuristic for Single Truck and Trailer Routing Problems

Luca Accorsi, Daniele Vigo
Department of Electrical, Electronic and Information Engineering "G. Marconi", University of Bologna,
Viale del Risorgimento 2, 40136 - Bologna, Italy, {luca.accorsi4, daniele.vigo}@unibo.it

In this paper, we propose a general solution approach for a broad class of vehicle routing problems that all use a single vehicle, composed of a truck and a detachable trailer, to serve a set of customers with known demand and accessibility constraints. A more general problem, called the Extended Single Truck and Trailer Routing Problem (XSTTRP), is used as a common baseline to describe and model this class of problems. In particular, the XSTTRP contains, all together, a variety of vertex types previously only considered separately: truck customers, vehicle customers with and without parking facilities, and parking-only locations. To solve XSTTRP we developed a fast and effective hybrid metaheuristic, consisting of an iterative core part, in which routes that define high-quality solutions are stored in a pool. Eventually, a set-partitioning based post-optimization selects the best combination of routes that forms a feasible solution from the pool. The algorithm is tested on extensively studied literature problems such as the Multiple Depot Vehicle Routing Problem, the Location Routing Problem, the Single Truck and Trailer Routing Problem with Satellite Depots, and the Single Truck and Trailer Routing Problem. Finally, computational results and a thorough analysis of the main algorithm's components on newly designed XSTTRP instances are provided. The obtained results show that the proposed hybrid metaheuristic is highly competitive with previous approaches designed to solve specific specialized problems, both in terms of computing time and solution quality.

Key words: vehicle routing; truck and trailer; metaheuristics

1. Introduction

Truck and trailer routing problems constitute a very well-studied class of vehicle routing problems (VRPs) in which vehicle capacities may be augmented with trailers. This class of problems was introduced by Chao (2002) as an extension of the basic VRPs to better model real-world applications arising mainly in freight distribution and city logistics. The literature contains several variations on the basic settings for these problems that add specific constraints modeling specific

scenarios. In Section 2 we give a brief overview of the literature, and we refer the interested reader to Cuda, Guastaroba, and Speranza (2015) for a general survey of related VRPs, including truck and trailer problems.

In this paper, we study the Extended Single Truck and Trailer Routing Problem (XSTTRP) as a more general variant which allows us to derive a unified solution approach for a class of single truck and trailer routing problems. In particular, the XSTTRP contains, all together, a variety of vertex types previously only considered separately, namely: truck customers, vehicle customers with and without parking facilities, and parking-only locations. The XSTTRP calls for servicing a set of customers with known demand using a single vehicle, composed of a capacitated truck and a non-autonomous, detachable trailer, which is initially located at a (main) depot. In the following, the term "truck" refers to the vehicle when the trailer is detached and parked at some parking location, while "complete vehicle" denotes the vehicle when the trailer is attached to the truck. The customers are partitioned into two different sets: *truck customers* and *vehicle customers*. The *accessibility constraints* impose that truck customers must be visited by the truck only, while vehicle customers can be visited either by the complete vehicle or by the truck. Vehicle customers are, in turn, split into vehicle customers *with parking facilities* and vehicle customers *without parking facilities*. The problem also contains a set of *satellite depots* (or just satellites), i.e., locations (which are not customers) where the trailer may be parked whenever necessary.

An XSTTRP solution is made up of a main route, in the following referred to as *main-route*, traveled by the complete vehicle, which starts from the main depot, visits a subset of vehicle customers and satellites, and returns to the depot. When the vehicle visits a *parking location* (i.e., either a satellite depot or a vehicle customer with parking facilities) it can detach its trailer, serve a subset of customers with the truck, and then return to pick up the trailer. We call this a *sub-route*, and the place where the trailer has been decoupled is the *root* of the sub-route. The objective is to find a solution which serves all customers while minimizing the total traveling cost and respecting both the truck capacity along the sub-routes and the accessibility constraints.

We developed a comprehensive, yet effective, heuristic solution approach to the XSTTRP. The resulting metaheuristic has been extensively tested on many instances, including special cases involving well-known problems such as the Multiple Depot Vehicle Routing Problem (MDVRP; see e.g., Cordeau, Gendreau, and Laporte (1997)), the Location Routing Problem (LRP; see Schneider and Drexl (2017)), the Single Truck and Trailer Routing Problem with Satellite Depots (STTRPSD; see Villegas et al. (2010)) and the Single Truck and Trailer Routing Problem (STTRP; see Bartolini and Schneider (2018)). The results show that the proposed method is highly successful in tackling the studied problems, producing good quality solutions in short computing time. The XSTTRP was originally introduced in a preliminary work by Accorsi (2017).

This paper is structured as follows. In Section 2 we review the literature relating to the XST-TRP in more depth and discuss related problems. In Section 3 we describe the XSTTRP more comprehensively and compare it with existing related problems. Section 4 describes the details of our solution approach, and experimental results are provided in Section 5. Section 6 offers an experimental analysis of the algorithm components. Finally, the paper ends with possible future research directions and concluding remarks in Section 7.

2. Literature review

The first comprehensive reference for the class of truck and trailer problems is the work by Chao (2002) who introduced the original Truck and Trailer Routing Problem (TTRP). Some earlier papers, such as that written by Semet and Taillard (1993), introduced similar variants which will be examined later in this section. The TTRP identifies a class of vehicle routing problems in which a fleet of capacitated vehicles, each composed of a truck and possibly a trailer, is used to serve a set of customers with a known demand. The objective is to minimize the total traveling cost without violating the capacity and accessibility constraints. The customers are split into *truck customers* and *vehicle customers*. Truck customers can only be visited by the truck without the trailer, so prior to visiting them, the vehicle has to park its trailer at an appropriate vehicle customer. The truck must then return to the same customer to pick up the trailer before continuing the journey. Vehicle customers can instead be served either by the truck or by the complete vehicle. Chao (2002) associated three possible types of route with each vehicle: a *pure truck route* which starts from the depot with the truck only, visits a subset of customers and returns to the depot; a *pure vehicle route* which starts from the depot with the complete vehicle, visits a subset of vehicle customers and returns to the depot; and a *complete vehicle route* consisting of a pure vehicle route and a set of pure truck routes starting from customer locations where the trailer could be parked. A feasible solution for the TTRP is thus composed of a set of routes, at most one for each vehicle, satisfying capacity constraints and belonging to one of the three types.

The current state-of-the-art algorithm for the TTRP is a matheuristic developed by Villegas et al. (2013). They populated a pool of high-quality solutions using a GRASP $\times$ ILS method (a greedy randomized adaptive search procedure combined with an iterated local search), and then they selected the best combination of routes from the pool by solving a set-partitioning formulation. The core of their approach was the construction of a giant tour that used a randomized nearest-neighbor heuristic to visit all the customers; the tour was further optimized by the interleaved application of a variable neighborhood descent (VND) and a perturbation phase. Each local optimum of the VND phase is stored in the set-partitioning solutions pool that is used to provide the final solution. The authors dealt with the standard TTRP as well as the Relaxed TTRP (see Lin, Yu, and Chou

(2010)) which has an unlimited fleet. A previous method based on a GRASP with evolutionary path relinking was presented by Villegas et al. (2011).

Villegas et al. (2010) introduced the Single Truck and Trailer Routing Problem with Satellite Depots (STTRPSD). In this problem, a set of truck customers is served by a single capacitated vehicle composed of a truck and a trailer. The problem does not contain any vehicle customer; instead, it contains a set of parking locations called *satellite depots* where the trailer can be parked. Satellites do not have an associated demand and thus can be left unvisited if not needed. The authors did not consider the depot to be a parking location.

The presence of satellites introduces an interesting variation to the problem, relating it to the well-known class of Location Routing Problems (LRPs). For a comprehensive survey of LRPs refer to Schneider and Drexl (2017). In fact, the STTRPSD generalizes the 2-echelon LRP with zero opening costs, uncapacitated depots and capacitated vehicles (see e.g., Tuzun and Burke (1999)). The authors proposed four heuristic methods for solving the STTRPSD, which they also tested on the MDVRP (a special case of the LRP). The best-performing of the four was a multi-start evolutionary local search.

Recently, Belenguer et al. (2016) proposed an exact branch-and-cut algorithm for the STTRPSD, based on an arc-flow formulation, which was able to consistently handle instances with up to sixty vertices.

A paper that is important for its work towards unifying the TTRP variants is the Generalized Truck and Trailer Routing Problem (GTTRP) and its extension, the Vehicle Routing Problem with Trailers and Transshipment (VRPTT) proposed by Drexl (2011, 2014). In the GTTRP, vehicles can leave the trailer either at vehicle customers or at transshipment locations (similar to satellites in STTRPSD); both may have associated time windows. The VRPTT extends the GTTRP by dropping the fixed assignment of a truck to a trailer; that is, any trailer may be pulled by any compatible truck for all or part of its itinerary. Moreover, the VRPTT adds the possibility that trucks may transfer, all or part of their load to any trailer at any transshipment location (with load-dependent transfer times). Note that, different vehicles may have different utilization (fixed and distance-dependent) costs. Drexl (2011) addressed the GTTRP with a branch-and-price algorithm and several heuristic variants, whereas in a different paper Drexl (2014) solved the VRPTT using five different branch-and-cut algorithms.

As previously mentioned, real-world TTRP-like applications were documented long before the actual problem was defined by Chao (2002). For example, in the Site-Dependent Vehicle Routing Problem considered by Semet and Taillard (1993), a fleet of heterogeneous vehicles composed of trucks and trailers serves a number of grocery stores in Switzerland cantons, taking into account vehicle-location incompatibilities, time windows for deliveries and vehicle-dependent utilization

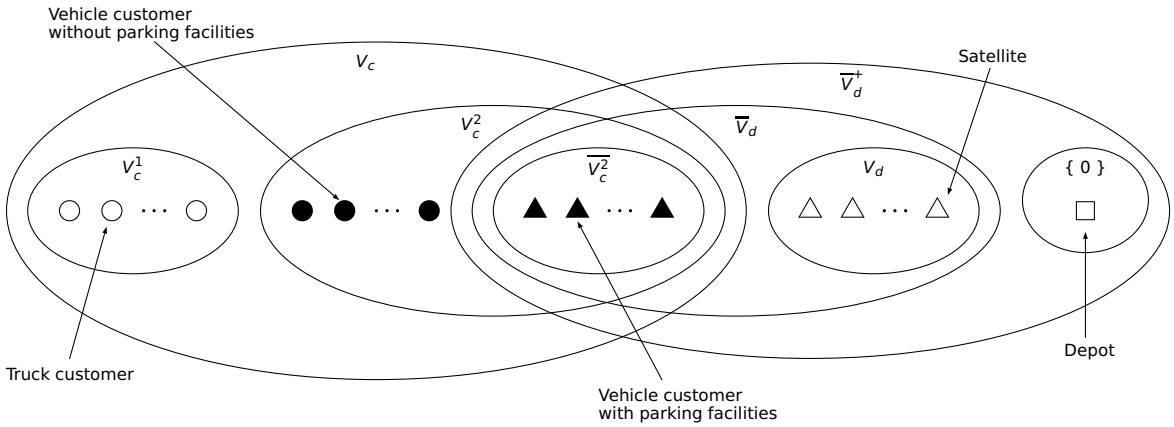


Figure 1 Relations between vertex types.

costs. The Partial Accessibility Constraint Vehicle Routing Problem analyzed in Semet (1995) consists of defining an appropriate number of trailers to increase the capacity of some trucks. Gerdessen (1996) considered the Vehicle Routing Problem with Trailers, which has been used to model two real-world scenarios: a Dutch dairy industry that has to serve customers located in crowded cities, where maneuvering a complete vehicle is difficult or forbidden; and the distribution of animal feed among farmers, some of whom are only reachable traversing narrow roads or small bridges. In both cases the trailer is often parked in appropriate locations before the truck performs the service. Further real-world TTRP applications include milk collection from farms (see Caramia and Guerriero (2010)), fuel oil delivery to private households (see Drexl (2011)), postal delivery (see Bodin and Levy (2000)) and container movement (see Tan, Chew, and Lee (2006)).

3. Problem description

The XSTTRP can be formulated by using an undirected graph $G = (V, E)$, where V is the vertex set and E is the edge set. The vertex set V is partitioned into $V = \{0\} \cup V_c \cup V_d$, where 0 is the depot, V_c is the set of customers and V_d is the set of satellite depots. V_c is, in turn, decomposed into $V_c = V_c^1 \cup V_c^2$, where V_c^1 represents the truck customers and V_c^2 the vehicle customers. Finally, we assume that a subset $\overline{V_c^2} \subseteq V_c^2$ of vehicle customers has parking facilities. From now on, $\overline{V}_d = V_d \cup \overline{V_c^2}$ denotes the parking locations where the vehicle is allowed to detach its trailer, namely the satellite depots and the vehicle customers with parking facilities, and $\overline{V}_d^+ = \overline{V}_d \cup \{0\}$. Figure 1 provides a graphic summary of the vertex sets and their relations. A cost c_{ij} is associated with each edge $(i, j) \in E$, and we assume that the cost matrix $\boldsymbol{c}$ satisfies the triangle inequality. Each customer $i \in V_c$ requires an integer quantity $q_i > 0$ from the depot and we assume that $q_i = 0, i \in \{0\} \cup V_d$. A vehicle with capacity $Q = Q_1 + Q_2$ is located at the depot, where Q_1 is the truck capacity and Q_2 is the trailer capacity. We assume that the vehicle is able to serve all the customers; that is, $\sum_{i \in V_c} q_i \leq Q$. We further suppose that, at parking locations, the goods required to satisfy the

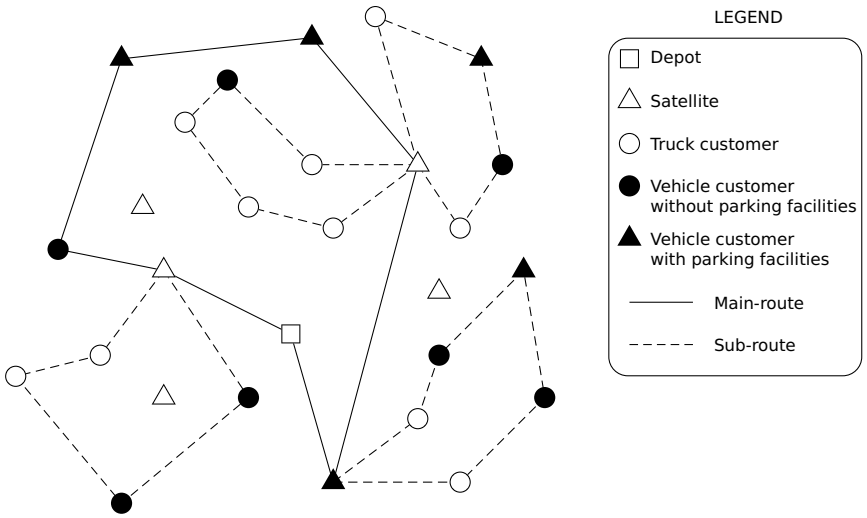


Figure 2 An example of a feasible solution to an XSTTRP instance.

demand of the subsequent sub-routes are instantaneously transferred from the trailer to the truck. Therefore, all customers can be served by just one vehicle. Truck customers $i \in V_c^1$ can be visited by the truck only, i.e., without the trailer, while vehicle customers $i \in V_c^2$ can be visited either by the truck or by the complete vehicle. A *feasible solution* requires that the trailer is detached (either at an appropriate satellite depot $k \in V_d$ or at a vehicle customer with parking facilities $j \in \overline{V_c^2}$) before visiting any truck customer $i \in V_c^1$ and, possibly, some vehicle customer $i \in V_c^2$. Then, the truck returns to the parking location where the trailer has been detached, to pick the trailer up before moving to the next vehicle customer, satellite, or even to the main depot. In other words, reminding that a Hamiltonian circuit is a closed cycle that visits a set of vertices exactly once, a feasible XSTTRP solution is composed of:

- a Hamiltonian circuit, called *main-route*, that starts from the depot, visits a subset of vehicle customers $i \in V_c^2$ and satellites $k \in V_d$ and eventually returns to the depot;
- a number of Hamiltonian circuits, called *sub-routes*, each of which starts from a parking location $k \in \overline{V}_d$ visited by the main-route, visits one or more customers $i \in V_c$ and ends at the starting parking location k. It is allowed to have more than one sub-route rooted at the same parking location.

Each customer must be visited exactly once, while satellite depots may remain unvisited if not necessary. Moreover, in order to be compatible with the STTRPSD definition, sub-routes directly starting from the depot are not allowed. This limit can be easily overcome by creating a satellite coincident with the main depot location. Figure 2 shows a possible solution for an XSTTRP instance where the empty square denotes the main depot, the empty triangles represent satellite depots, the empty circles denote truck customers, the full circles represent vehicle customers without parking facilities and the full triangles are vehicle customers with parking facilities. The

main-route is defined by a continuous line and the sub-routes by dashed lines.

3.1. Comparative analysis of related problems

The XSTTRP is inspired by the combination of STTRPSD and STTRP. Both problems are direct specializations of the XSTTRP, and as such include just a subset of its vertex types. However, the XSTTRP generalizes many other vehicle routing problem variants. To illustrate this, we focus on the MDVRP (with capacitated vehicles) and the LRP (with uncapacitated depots and capacitated vehicles), because both of these problems require two levels of decisions (assignment of customers to depots and routing) so they naturally fit into the XSTTRP model. We note that the Capacitated VRP (CVRP), being a special case of the MDVRP, is also generalized by the XSTTRP, which is therefore NP-Hard in the strong sense. The MDVRP, LRP and CVRP could be directly encoded as XSTTRP by: *(i)* mapping depots and customers to satellites and truck customers respectively, *(ii)* adding an additional dummy main depot, *(iii)* setting to 0 all costs associated with the edges between satellites and those between satellites and depot, and *(iv)* setting the truck capacity equal to the original problem vehicle capacity and the trailer capacity equal to, at least, the sum of customer demands. The MDVRP usually imposes a maximum on the number of vehicles available in each depot, while the LRP defines a fixed cost associated with each vehicle and depot used. Both features are not directly modeled by the XSTTRP; thus they must either be handled by the solution procedure or indirectly encoded in the instance definition, e.g., by changing the cost of some arcs.

Figure 3 shows a possible hierarchical structure where several related problem variants are linked together by generalization/specialization relationships. From the figure we deduce that the natural generalization of the XSTTRP is the Extended TTRP (XTTRP); several, possibly heterogeneous, capacitated vehicles (possibly equipped with a trailer) are available to serve the customers. We also note that the STTRPSD is in fact a special case of the 2 Echelon-VRP (2E-VRP). In the latter, satellites are capacitated and might be visited multiple times by different vehicles. Finally, the GTTRP defined by Drexl (2011) is a complex and interesting unified model for vehicle routing problems with trailers. It generalizes the TTRP by including transshipment locations and several complex side-constraints. The XSTTRP and the XTTRP are much more basic problems, including just capacity and accessibility constraints. Moreover, the XSTTRP differs structurally from the GTTRP because it includes vehicle customers without parking facilities. Table 1 classifies the previously introduced problems according to the vertex types and the available number of vehicles they contain. In our computational testing we focused on the variants, enclosed in the dashed line of Figure 3, that preserve the multiple-depot structure of XSTTRP in which satellites are present.

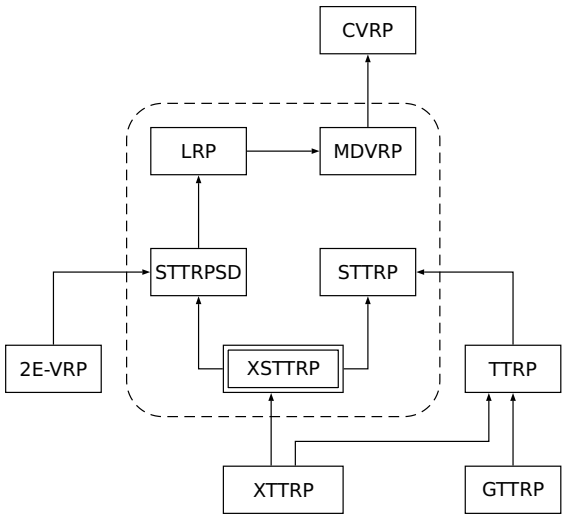


Figure 3 A possible topology (for a subset) of VRPs.

Note. The relation $X \to Y$ means X is a generalization of Y. The problem variants studied in this paper are those enclosed by the dashed line.

Table 1 A problem classification based on the vertex types and number of vehicles #veh.

	Depot	Satellite	Vehicle customer with p. f.	Vehicle customer without p. f.	Truck customer	
Problem	□	△	▲	●	○	**#veh**
CVRP	✓	✗	✗	✓	✗	≥ 1
MDVRP	✓	✗	✗	✓	✗	≥ 1
LRP	✓	✗	✗	✓	✗	≥ 1
2E-VRP	✓	✗	✓	✗	✓	≥ 1
STTRPSD	✓	✓	✗	✗	✓	1
STTRP	✓	✗	✓	✗	✓	1
TTRP	✓	✗	✓	✗	✓	≥ 1
XSTTRP	✓	✓	✓	✓	✓	1
XTTRP	✓	✓	✓	✓	✓	≥ 1
GTTRP	✓	✓	✓	✗	✓	≥ 1

The symbol ✓ means that the problem contains the vertex type and ✗ that it does not. The abbreviation p. f. stands for parking facilities.

4. Solution approach

In this section we describe our metaheuristic for the XSTTRP, resulting in the algorithm AVXS. The algorithm consists of a core part followed by a post-optimization phase. The core part iterates through the following three *phases*:

1. The *assignment* phase associates each vertex $i \in V$ with an appropriate vertex $j \in V$, e.g., customers are assigned to parking locations or to the depot;
2. The *construction* phase builds an initial routing solution which is complete and feasible, starting from the given assignment;
3. The *improvement* phase may further optimize the given solution.

These phases are executed for a fixed number of iterations, called *restarts*, during which a limited set of routes composing high-quality solutions are stored in a pool $\mathcal{P}$. The set-partitioning model F, described in Section 4.4, is then populated with the routes from $\mathcal{P}$. A concluding post-optimization *polishing* phase then selects the best combination of routes as the final solution. Pseudo-code for AVXS is given in Algorithm 1, while the following paragraphs provide a detailed description of the four phases. Finally, the section ends with a discussion on how to handle some special requirements arising in problem variants within AVXS, such as the MDVRP and the LRP.

Algorithm 1 AVXS algorithm

procedure AVXS($instance, seed$)
  $\mathcal{R} \leftarrow$ randomEngine($seed$) ▷ Initialize the pseudo-random number generator
  $\mathcal{P} \leftarrow \emptyset$ ▷ Initialize the pool that will eventually contain high-quality routes
  $S^* \leftarrow$ emptySolution($instance$) ▷ Initialize the best known solution
  for $r \leftarrow 1$ to Δ **do**
    $S \leftarrow$ emptySolution($instance$) ▷ Initialize the current restart best solution
    $\mathcal{A} \leftarrow$ assignment($instance, \mathcal{R}$)
    $S \leftarrow$ construction($S, \mathcal{A}$)
    $S \leftarrow$ improvement($S, \mathcal{P}, \mathcal{R}$)
    if cost(S) < cost(S^*) **then** $S^* \leftarrow S$ ▷ Update the best known solution
  end for
  $F \leftarrow$ buildModel($\mathcal{P}$) ▷ Populate the set-partitioning model F using the routes in $\mathcal{P}$
  $S^* \leftarrow$ polishing(S^*, F)
  return S^*
end procedure

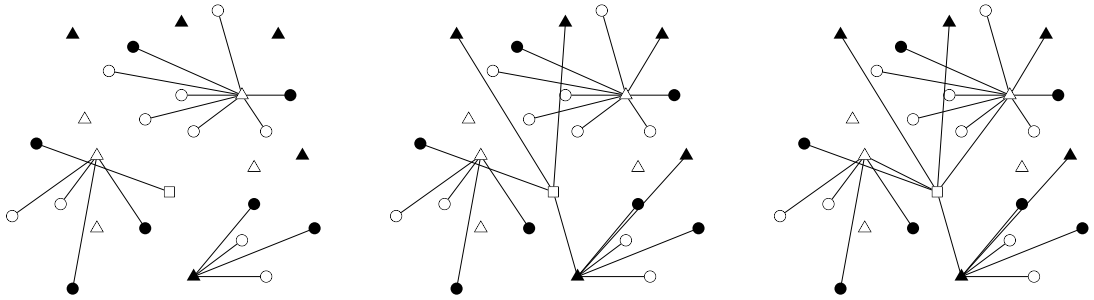
4.1. Assignment

In this phase we gradually build a feasible *assignment tree* by iteratively assigning customers to parking locations and the depot. To this end we define the *assignment cost* $\hat{c}_{ij}$ as the estimated cost of serving vertex $i \in V$ from vertex $j \in V$. In our implementation, we define the assignment costs as

$$\hat{c}_{ij} = \begin{cases} c_{ij} & (i \in V_c^1 \wedge j \in \overline{V}_d) \vee (i \in V_c^2 \wedge j \in \overline{V}_d^+ \setminus \{i\}) \vee (i \in V_d \wedge j = 0) \\ +\infty & \text{otherwise} \end{cases}$$

As an alternative, one could use a more sophisticated estimate, such as the reduced costs obtained by a combinatorial optimization bound (see e.g., Toth and Vigo (2002)). However, our preliminary computational experiments showed that the additional effort of using reduced costs for the assignments did not provide substantial improvement compared to the original costs.

We define a parking location $k \in \overline{V}_d$ as *open* if it has at least one customer assigned to it, and *closed* otherwise. The assignment phase performs the following steps in sequence:



(a) (b) (c)

Figure 4 Construction of an assignment tree starting from the instance depicted in Figure 2. Each separate picture represents the result of each step of the assignment phase.

(a) Truck customers and vehicle customers without parking facilities $i \in V_c^1 \cup (V_c^2 \setminus \overline{V_c^2})$ are probabilistically assigned to vertices $k_i \in \overline{V}_d^+$ through a roulette-wheel selection using an assignment fitness function f^i. Once the step is completed some vehicle customers with parking facilities will be open and others will be closed. See Figure 4(a) for an illustration.

(b) Closed vehicle customers with parking facilities $i \in \overline{V_c^2}$ are assigned to open parking locations and the depot $j \in \overline{V}_d \cup \{0\}$, using the same strategy as in step (a). Open vehicle customers with parking facilities $k \in \overline{V_c^2}$ are assigned to the depot, as depicted in Figure 4(b).

(c) Open satellites $k \in V_d$ are assigned to the depot and the closed ones are ignored, as in Figure 4(c).

Observe that the assignment resulting from this phase will always respect the accessibility constraints defined by the XSTTRP whenever the instance is feasible. The definition of the assignment fitness function f^i for each vertex $i \in V$ obviously influences the outcome of this phase. In our implementation, we used the d-NEAR assignment fitness function with d equal to 25. The d-NEAR is a restricted GRASP-based function defined as

$$f^i(j) = \begin{cases} 1 & j \in \mathcal{H}_i^d(\hat{\boldsymbol{c}}) \\ 0 & \text{otherwise} \end{cases}$$

where $\mathcal{H}_i^d(\hat{\boldsymbol{c}})$ denotes the d nearest vertices to i according to the assignment costs $\hat{\boldsymbol{c}}$. For a comprehensive analysis of the behavior of the d-NEAR function we refer to Section 6.1. Finally, the probability p_{ij} of assigning vertex $i \in V$ to vertex $j \in V$ is defined as $p_{ij} = f^i(j) / \sum_{k \in V} f^i(k)$. It is worth noting that, due to the roulette-wheel selection procedure, subsequent calls of the assignment phase might produce different assignment trees.

4.2. Construction

Given an assignment tree, an initial feasible solution to XSTTRP is defined as follows. First, the main-route is determined by solving a Traveling Salesman Problem (TSP) visiting the depot, the open parking locations, and any vehicle customer without parking facilities assigned to the depot.

Then, for each open parking location $k \in \overline{V}_d$, the sub-routes are obtained by solving a specific CVRP with the customers belonging to the sub-tree rooted in k. Note that there is no limit on the number of sub-routes rooted in a given parking location. As extensively discussed in Section 6.1, the quality of the initial solution is not crucial for the algorithm's overall performance. Therefore, we solved both the CVRP and the TSP using an adaptation of the well-known savings algorithm by Clarke and Wright (1964).

4.3. Improvement

The initial solution defined in the construction phase is improved using a specific procedure based on the Iterated Local Search paradigm (ILS, see Lourenço, Martin, and Stützle (2003)) in which the local search is performed using a Randomized Variable Neighborhood Descent (RVND, see e.g., Subramanian, Uchoa, and Ochi (2013)). In addition, to speed up the local search execution, we adopted a granular approach (see, Toth and Vigo (2003) and Schneider, Schwahn, and Vigo (2017)). Our ILS intensifies the search around the current best-known solution. The diversification is naturally provided throughout the procedure itself, strengthened by a global restart mechanism which provides a different starting assignment tree. The randomization in the RVND is used for selecting the order of the neighborhoods and that of the moves within each neighborhood, the latter are searched according to a first-improvement strategy. This improves the likelihood that different search pathways are explored, thus diversifying while improving the solution. We considered the following neighborhoods:

- 1-0 exchange (RELOCATE) in which each customer is removed from its current position and re-inserted in the position that minimizes the insertion cost in the current solution;
- 1-1 exchange (SWAP) in which a pair of customers is exchanged;
- intra-route & intra-park 2-opt (TWOPT) in which an intra-route 2-opt procedure is applied to each route and an inter-route 2-opt procedure is applied to each VRP, defined by an open parking location and the vertices assigned to it;
- sub-routes segments-swap (SEGSWAP) in which contiguous, possible empty, paths within sub-routes including from two up to five vertices are swapped. SEGSWAP is a simple adaptation of the CROSS-exchange operator introduced by Taillard et al. (1997). Note that, for efficiency purposes, our definition of SEGSWAP works on already open satellites and sub-routes only. Therefore, it does not include RELOCATE or SWAP moves that can work on main- and sub-routes. In fact, from our experience, swapping a contiguous path of vertices between the main-route and a sub-route seldom results in a feasible exchange;
- sub-route root-refine (ROOTREF) (see Chao (2002)), consists of replacing the root of a sub-route with another one. We extend the operator by also considering the possibility of leaving

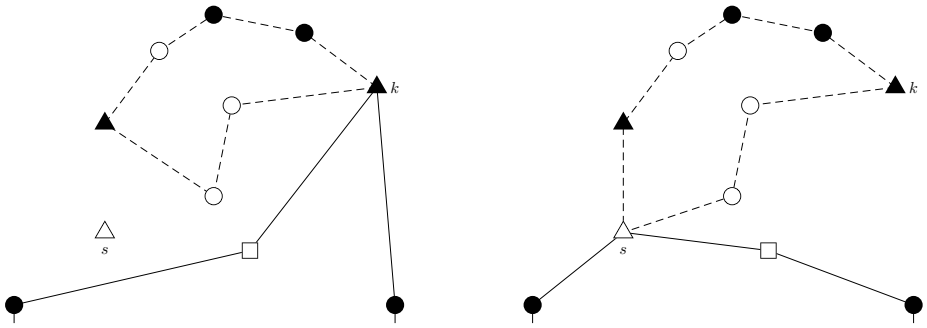


Figure 5 A special case of the rootref move.

Note. In (a) we show the initial solution with the sub-route rooted in vertex k. In (b) the root is changed to s, and k is removed from the main-route and inserted into the sub-route.

the original root, corresponding to a vehicle customer with parking facilities, in the sub-route (as shown in Figure 5).

At this point, we have five local search operators, all but one of which, the TWOPT, can change the initial assignment tree. As will be analyzed in Section 6.1, the local search step is capable of greatly improving the quality of the solutions. In the RVND we examine each granular neighborhood exhaustively before moving to the next one. If an improving move is found, we apply the move and restart the search using the current neighborhood. The local search terminates once a complete RVND is executed without finding an improving solution.

Once the local search is stopped at a local optimum, we apply a *shake* procedure to the solution in a *ruin-and-recreate* style (see Schrimpf et al. (2000)). The shake is always applied to the best known solution found during the current restart. More precisely, we designed three different removal procedures to be used in the *ruin* step. The first one, called *main-vertex-removal* (MREM), operates by removing, with probability η, each vehicle customer served by the main-route that is not an open parking location (as discussed in Section 5.2, we used $\eta = 0.5$). The other removal procedures work on sub-routes. The *sub-routes-removal* procedure (SREM) aims to improve the assignment by destroying a number of long routes from a set of randomly chosen, open parking locations. For each selected parking location k, we remove the r longest sub-routes, where r is randomly chosen between 0 and the total number of sub-routes rooted in k. Note that if the parking location is a satellite, then all the sub-routes rooted in that satellite are removed and the satellite is closed. We define this special case as SREM.1. On the other hand, the *sub-routes unloading* procedure (SUNLOAD), which is intended to improve the routing solution, consists of removing random customers from each sub-route until its load is less than a threshold $\zeta \times load_{avg}$, where ζ is a coefficient that is uniformly randomly generated on the interval $[0.3, 0.9]$ (at each shake application) and $load_{avg}$ is the average load of all sub-routes. At each shake application we always execute MREM, and we

alternate the executions of SREM and SUNLOAD. The *recreate* step inserts each removed customer in the position of the current solution that minimizes the insertion cost. The order in which customers are considered affects the final result. Therefore, at every recreate step we examine the customers according to an ordering randomly chosen among five possibilities: ascending or descending values of their x or y coordinates, respectively, or a random permutation. Pseudo-code for the improvement phase is provided in Algorithm 2.

Algorithm 2 Improvement phase

procedure IMPROVEMENT($S, \mathcal{P}, \mathcal{R}$)
  $\pi \leftarrow \pi_{base}$ ▷ Initialize the sparsification factor
  $i \leftarrow 0, g \leftarrow 0$ ▷ Initialize ILS and granular counters
  $S' \leftarrow S$ ▷ Initialize the current restart best solution
  while true **do**
    $S \leftarrow$ RVND($S, \mathcal{R}$) ▷ Improve the current solution using the randomized VND
    if COST(S) < COST(S') **then** ▷ Update the current restart best solution, if necessary
      $i \leftarrow 0, g \leftarrow 0, S' \leftarrow S, \pi \leftarrow \pi_{base}$ ▷ Reset some variables
      $\mathcal{P} \leftarrow \mathcal{P} \cup$ ROUTESOF(S) ▷ Add routes to the route pool
    end if
    $i \leftarrow i + 1, g \leftarrow g + 1$ ▷ Increment ILS and granular counters
    if $i \geq \delta$ **then break** ▷ Terminate the improvement phase
    if $g \geq \phi \cdot \delta$ **then** $\pi \leftarrow \lambda \cdot \pi, g \leftarrow 0$ ▷ Update the move generators set
    $S \leftarrow S'$
    $S \leftarrow$ SHAKE(S)
  end while
  return S'
end procedure

Finally, as to the algorithm's efficiency, we observe that both RELOCATE and ROOTREF might evaluate moves which require the opening of closed satellites. To identify the insertion position for the satellite which minimizes the insertion cost in the main-route, we adopted a lazy caching strategy. In particular, we use an additional cache data structure that for each closed satellite stores its best insertion position. At the beginning, the cache is empty (or invalid). When a move needs to evaluate the opening of a closed satellite, we check the cache for a valid result. If the result is valid, we return the stored position, otherwise we scan the current main-route, find the current best insertion position, and update the corresponding cached value. The whole cache is invalidated each time the main-route is modified.

4.3.1. Granular neighborhoods. Granular neighborhoods have been successfully applied to several Vehicle Routing Problems to speed up local search procedures while preserving solution quality compared to complete neighborhood exploration (see Toth and Vigo (2003)). Following the guidelines presented by Toth and Vigo (2003) and Schneider, Schwahn, and Vigo (2017) a granular neighborhood is completely defined by a set $T \in E$ of appropriately chosen arcs, called *move generators*. Each arc in T is used in the neighborhood search to generate a uniquely defined single move, thus reducing the search time to $O(|T|)$. Typically, the set T of move generators for a specific neighborhood is defined according to problem-related criteria, often called *sparsification rules*. For example, according to Toth and Vigo (2003), arcs are chosen if their cost is below a given threshold, whereas Schneider, Schwahn, and Vigo (2017) used the reduced cost coming from a simple relaxation for the same purpose. In our algorithm all the neighborhoods used in the RVND are defined as granular and we adopted a simple cost-based sparsification rule.

The multi-level nature of a problem with a main-route and several sub-routes calls for additional care in the definition of the set of move generators. More precisely, by examining Figure 6, which reports a sub-optimal solution for instance 25 of Villegas et al. (2010), one may easily observe that the arcs in the main-route are considerably longer than those of the sub-routes. Therefore, if we use a relatively small cost threshold to define the sparsification rule, very few arcs between satellites are likely to be inserted in T, possibly causing severe harm to the solution quality. As a consequence, we used two different sparsification rules for the arcs of the main-route $\{(i,j) : i,j \in V_c^2 \cup V_d \cup \{0\}\}$, called *main-route arcs*, and for those of the sub-routes $\{(i,j) : i,j \in V_c \cup V_d\}$, called *sub-route arcs*. Note that some arcs, e.g., those between vehicle customers, may belong to both sets because such customers may be served either in the main-route or in the sub-routes. Given a sparsification factor π, we define T as the union of two sets T_M and T_S corresponding to main- and sub-route arcs. The set T_M includes the $\pi \cdot (|V_c^2| + |V_d| + 1)$ shortest main-route arcs, while the set T_S includes the $\pi \cdot (|V_c| + |V_d|)$ shortest sub-route arcs.

Finally, following Schneider, Schwahn, and Vigo (2017), we adopted a dynamic updating of the sparsification level. We initially set $\pi = \pi_{base}$, and whenever $\phi \cdot \delta$ non-improving iterations are performed, $\pi = \pi \cdot \lambda$. The value of π is reset to π_{base}, however, whenever an improving solution is found.

4.4. Polishing

We store every route that constitutes an improving solution produced by the RVND procedure in a pool $\mathcal{P}$ of high-quality routes (see line 9 of Algorithm 2). Note that we only store local minima which are improving with respect to the current restart best solution and not all the generated ones. We found that this allows us to limit the size of the pool without a significant worsening of

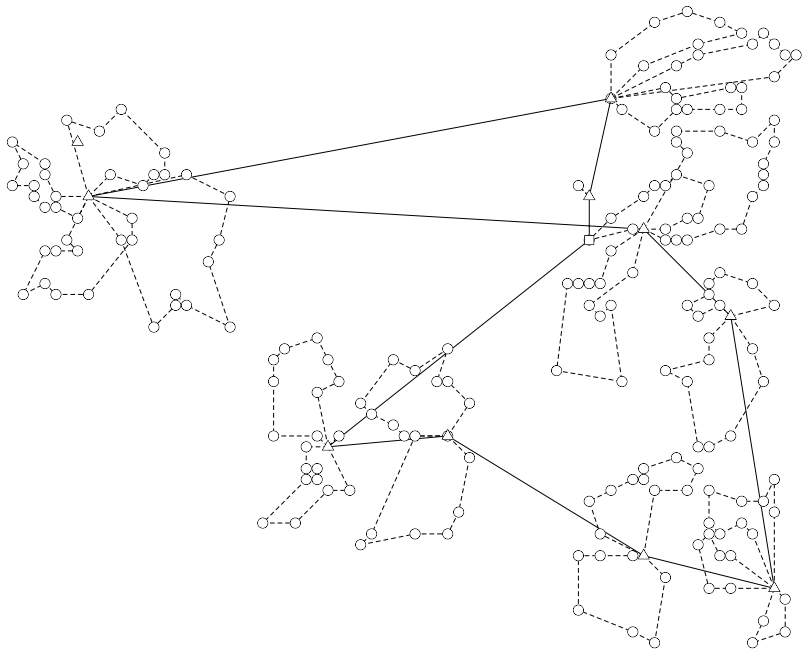


Figure 6 A solution for instance 25 by Villegas et al. (2010), a clustered instance with 200 truck customers and ten satellite depots. The main-route is clearly sub-optimal.

the solution quality. In storing the routes we discard duplicates and, possibly, update the existing ones with improved versions. We effectively handle routes retrieval by using an additional hash-table data structure that, given a set of nodes, is able to efficiently check whether a route passing through the given set of nodes exists or not. Moreover, routes in $\mathcal{P}$ are stored as set of vertices with an associated cost label which depends on the sequence of visits of the vertices. Given two routes passing through the same set of vertices we only maintain the one with the lowest cost label. The final result of our algorithm is the best solution found by a set-partitioning post-optimization performed on the routes from $\mathcal{P}$.

More precisely, let $\mathcal{M}$ be the subset of all the main-routes in $\mathcal{P}$ starting from the depot 0 and passing through a subset of vehicle customers V_c^2 and, possibly, through one or more satellites in V_d. Let $\mathcal{M}_k \subset \mathcal{M}$ be the subset of main-routes passing through $k \in V_c^2 \cup V_d$. Note that $\mathcal{M}_k = \emptyset$ for each truck customer $k \in V_c^1$. Given a main-route $r \in \mathcal{M}$, we denote by g_r its cost and by R_r the subset of vehicle customers and satellite depots visited by r. Let $\mathcal{R}_k$ be the subset of all the sub-routes in $\mathcal{P}$ rooted in parking location $k \in \overline{V}_d$. We denote by $\mathcal{R}_{ki} \subset \mathcal{R}_k$ the subset of sub-routes rooted in $k \in \overline{V}_d$ that pass through customer $i \in V_c$. Let $\ell \in \mathcal{R}_k$ be a sub-route rooted in k; we denote by $d_{k\ell}$ its cost and by R_ℓ^k the subset of customers visited by ℓ and we assume that $k \notin R_\ell^k$. Let y_r be a binary variable that is equal to 1 if and only if the main-route $r \in \mathcal{M}$ is in the solution, and 0 otherwise. Furthermore, let $x_{k\ell}$ be a binary variable that is equal to 1 if and only if the

sub-route $\ell \in \mathcal{R}_k, k \in \overline{V}_d$ is in the solution, and 0 otherwise. The XSTTRP route-based formulation F is defined as follows.

$$(F) \quad \min \quad \sum_{k \in \overline{V}_d} \sum_{\ell \in \mathcal{R}_k} d_{k\ell} x_{k\ell} + \sum_{r \in \mathcal{M}} g_r y_r \tag{1}$$

$$\text{s.t.} \quad \sum_{k \in \overline{V}_d} \sum_{\ell \in \mathcal{R}_{ki}} x_{k\ell} = 1, \qquad i \in V_c^1 \tag{2}$$

$$\sum_{k \in \overline{V}_d} \sum_{\ell \in \mathcal{R}_{ki}} x_{k\ell} + \sum_{r \in \mathcal{M}_i} y_r = 1, \qquad i \in V_c^2 \tag{3}$$

$$\sum_{\ell \in \mathcal{R}_{ki}} x_{k\ell} - \sum_{r \in \mathcal{M}_k} y_r \leq 0, \qquad k \in \overline{V}_d, i \in V_c \setminus \{k\} \tag{4}$$

$$\sum_{r \in \mathcal{M}} y_r = 1 \tag{5}$$

$$x_{k\ell} \in \{0,1\}, \quad \ell \in \mathcal{R}_k, k \in \overline{V}_d \tag{6}$$

$$y_r \in \{0,1\}, \quad r \in \mathcal{M} \tag{7}$$

where the objective function (1) imposes the selection of the best combination of main-route and sub-routes. Equations (2) and (3) require that each customer $i \in V_c$ be visited exactly once by a compatible route. Inequalities (4) specify that a customer $i \in V_c$ can be visited by a sub-route rooted in a parking location $k \in \overline{V}_d$ only if that location is visited by the main-route. Constraint (5) dictates that just one main-route must be in the solution. Finally, constraints (6) and (7) define the variables as binary. The mathematical formulation was introduced in a preliminary work by Accorsi (2017). The polishing phase brings together the otherwise unrelated restart results. Similar methods have proved effective in other vehicle routing algorithms (see e.g., the works by Subramanian, Uchoa, and Ochi (2013) and Villegas et al. (2013)). In doing the post-optimization, we supply the best found solution as a warm start; we stop the model resolution after a predetermined computer-independent number of ticks, see Section 5.1.

4.5. Extensions to handle specific sub-problems

The conversion from MDVRP and LRP instances to XSTTRP ones is straightforward, although both problems include special requirements not directly definable in XSTTRP terms. In particular, the MDVRP imposes a maximum on the number of vehicles available in each depot and the LRP defines a fixed cost on the use of vehicles and on the opening of depots. As discussed in Section 3.1, these problems are encoded as XSTTRP by: *(i)* mapping depots and customers to satellites and truck customers respectively, *(ii)* adding an additional dummy main depot, *(iii)* setting to 0 all costs associated with the edges between satellites and those between satellites and depot, and *(iv)* setting the truck capacity equal to the original problem vehicle capacity and the trailer capacity equal to, at least, the sum of customer demands.

In the MDVRP, we build a zero-cost main-route through the main depot and all the satellites (that is, we assume all satellites have been visited). Next, in order to handle a maximum number of available vehicles (say n_v) in each satellite, we simply introduce a penalty; the solution cost is incremented by a very large value for each additional vehicle used in the solution. Although this alone does not guarantee the discovery of feasible solutions, it has been proved effective experimentally in solving the instances we consider. Furthermore, this same constraint must be considered in model F by adding

$$\sum_{\ell \in \mathcal{R}_k} x_{k\ell} \leq n_v, \quad k \in V_d$$

Finally, we relax the shaking procedure SREM by discarding the special case SREM.1 defined in Section 4.3, which removes all routes rooted in a satellite chosen by the SREM shaking procedure. Since the main-route does not contribute to the objective function in the MDVRP, any empty visited satellite can remain open. Thus, instead of removing all sub-routes from a satellite, we remove a random number of them.

To handle the LRP, we simply incorporate the fixed costs of using a vehicle $\mathcal{F}_v$ and opening an LRP depot $\mathcal{F}_d$ into the cost matrix as follows.

$$c'_{ij} = \begin{cases} \dfrac{\mathcal{F}_d}{2} & (i \in V_d \wedge j = 0) \vee (i = 0 \wedge j \in V_d) \\ \mathcal{F}_d & i \in V_d \wedge j \in V_d \wedge i \neq j \\ \dfrac{\mathcal{F}_v}{2} & (i \in V_c \wedge j \in V_d) \vee (i \in V_d \wedge j \in V_c) \\ c_{ij} & \text{otherwise} \end{cases}$$

We use this new cost matrix $\boldsymbol{c'}$ to execute the optimization. It is important to note that, when opening a satellite has high fixed cost, we cannot rely on a probabilistic assignment phase, because a wrong initial assignment is very difficult to modify when the fixed cost is greater than the routing contribution. In other words, the attractiveness of already open satellites might be very high with respect to the opening of closed ones. In particular, our local search procedures do not contain any specialized technique which is able to completely move several routes from one satellite to another one in a single evaluation. The SREM ruin procedure might cause a satellite to be completely closed but, again, if opening a satellite has high fixed cost compared to the routing cost, the recreate step will prefer to insert the removed customers in routes of already open satellites instead of opening new ones. Hence, in this situation, a meaningful selection of initial satellites during the assignment phase can be very important. To handle this, we use the d-NEAR assignment fitness function with d equal to 1, which deterministically assigns a vertex $i \in V$ to its nearest neighbor $j \in V$ according to the assignment cost $\hat{\boldsymbol{c}}$.

5. Computational results

The computational testing had two objectives. First of all we wanted to assess the success of AVXS. To this end, we used the algorithm to solve instances of four special cases of the XSTTRP which are extensively studied in the literature, namely the MDVRP, the LRP, the STTRPSD and the STTRP. We also performed a preliminary testing on the CVRP instances of the X benchmark by Uchoa et al. (2017). Unfortunately, our results with the algorithm's parameters tuned as described in Section 5.2 were not state-of-the-art on those large scale instances, therefore we do not describe them in this section. The second objective was to analyze the impact of the algorithm's main components on its performance in detail, by testing AVXS on new instances of the XSTTRP which have not been previously studied in the literature.

5.1. Implementation and experimental environment

The proposed algorithm was implemented in C++ and compiled using g++ 6.3.0. To solve the set-partitioning model F we used CPLEX 12.8 (CPLEX (2017), C callable library) with default settings and a prefixed number of ticks as a computer-independent measure to prematurely terminate the model resolution. In particular, assuming our machine performs an average number of 1900 ticks per second, we assigned CPLEX a maximum of $60 \cdot 1900 = 114000$ ticks. The experiments were performed on a 64-bit desktop computer with an Intel Core i7-8700K CPU, running at 3.7 GHz with 32 GB of RAM on a GNU/Linux Debian operating system. Both the algorithm and CPLEX were executed using a single thread. All the instances used in our testing with the corresponding graphic solutions, as well as the source code, could be downloaded from `https://acco93.github.io/avxs/`. Detailed instructions are given as an aid to others wishing to accurately reproduce our results. Because our algorithm contains randomized elements, for every experiment we performed a set of ten runs for each instance and we defined the seed of the pseudo-random engine, the Mersenne Twister (Matsumoto and Nishimura (1998)), equal to the run counter minus one. To facilitate comparisons with other algorithms run on computers with different CPUs, we used the single-thread rating defined by PassMark Software (2018), which assigns a score of 2704 to our CPU. Moreover, to mitigate the impact of small time-variations due to overhead of the operating system we intentionally used a clock function that reports running times with a precision set to one second as the minimum recordable time. Therefore, when the algorithm took less than one second, we considered the elapsed time to be one. Finally, if not stated otherwise, we assume that the arc costs c_{ij} were computed as $c_{ij} = Euc(i,j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$, where x_i, x_j, y_i and y_j represent the x and y coordinates of vertices $i, j \in V$, respectively.

5.2. Parameter definitions

Algorithm AVXS requires that initial values be defined for a number of parameters. We set these values during preliminary testing, performed on a training which included the XSTTRP instances with 200 total vertices. The resulting parameter values, which were used on all problem classes, are the following:

- number of restarts, $\Delta = 100$;
- maximum number of non-improving ILS iterations, $\delta = 100$;
- probability of removing a customer from the main-route in the MREM procedure, $\eta = 0.5$;
- unloading range used in the SUNLOAD procedure, $\zeta = [0.3, 0.9]$;
- initial sparsification value used to define the granular neighborhood, $\pi_{base} = 1.25$;
- factor, $\phi = 0.2$, used to define the fraction of non-improving iterations to be performed before incrementing the sparsification parameter π;
- factor, $\lambda = 2$, used to increment the value of π when $\phi \cdot \delta$ non-improving iterations are performed.

The parameters tuning followed a simple sequential strategy. At the beginning, we identified initial reasonable values for all the parameters by performing a preliminary limited trial-and-error testing. Then, we assessed the performance of the algorithm when changing the value of one parameter at a time. We kept a new value when it allowed good quality solutions while keeping the computational time low.

In particular, we noted that few iterations in the improvement phase are typically sufficient to converge to good local optima and few restarts are generally able to produce a moderate-sized and diversified pool of high-quality routes. In fact, the results obtained by setting $\delta = 1000$ or $\Delta = 1000$ were, on average, at most 0.2% better while requiring computing times up to 10 times larger. Moreover, we noticed that the algorithm is not very sensitive to reasonable values of η and ζ. Indeed, whenever values of those parameters that cause extremely disruptive effects or no changes at all are discarded, the algorithm is able to produce comparable results in similar computing time. On the contrary, π_{base}, ϕ and λ heavily affects the algorithm performance. The value of π_{base} was set following the guidelines defined in Toth and Vigo (2003) who reported that the best results are typically obtained with a sparsification factor between 1.0 and 2.5 and used a sparsification value equal to 1.25. From our experience, starting with a very aggressive sparsification level and reducing it relatively often resulted to be the most effective and efficient strategy.

5.3. Testing on MDVRP instances

We tested our algorithm on the subset of MDVRP instances with capacitated vehicles proposed by Cordeau, Gendreau, and Laporte (1997). The set contains 11 instances with 50 to 360 customers

Table 2 Computations on MDVRP instances.

						HGSADC		AVXS				
Id	$\lvert V_c^1\rvert$	$\lvert V_d\rvert$	Q_1	*Type*	*BKS*	*Avg*	t_{10}	*Best*	*Avg*	*Worst*	t_{10}	$\hat{t}_{10}$
p01	50	4	80	rd	**576.87**	0.00	138	0.00	0.00	0.00	10	39.47
p02	50	4	160	rd	**473.53**	0.00	126	0.00	0.00	0.00	10	39.47
p03	75	2	140	rd	641.19	0.00	258	0.00	0.00	0.00	20	78.95
p04	100	2	100	rd	1001.04	0.02	1164	0.00	0.15	0.31	178	702.65
p05	100	2	200	rd	750.03	0.00	636	0.00	0.00	0.00	56	221.06
p06	100	3	100	rd	**876.50**	0.00	684	0.00	0.00	0.00	75	296.06
p07	100	4	100	rd	**881.97**	0.28	930	0.00	0.06	0.30	86	339.48
p12	80	2	60	g	**1318.95**	0.00	312	0.00	0.00	0.00	10	39.47
p15	160	4	60	g	2505.42	0.00	1152	0.00	0.00	0.00	89	351.32
p18	240	6	60	g	3702.85	0.00	2712	0.00	0.00	0.00	273	1077.65
p21	360	9	60	g	5474.84	0.03	6000	0.00	0.00	0.00	774	3055.32
Mean						0.03	1282.91	0.00	0.02	0.06	143.73	567.35

Optimality has been proved for the solutions in boldface by Baldacci and Mingozzi (2008).

and up to nine depots. The first seven instances have randomly distributed customers, while in the last four they are arranged in geometrical patterns. We compared AVXS with the HGSADC (Hybrid Genetic Search with Advanced Diversity Control) algorithm proposed by Vidal et al. (2012), which is currently one of the most effective MDVRP solution approaches capable of handling the complete MDVRP instance set and several other VRP variants. In our computations, we omitted MDVRP instances with constraints on route length. Table 2 summarizes the computational results, organized in the following columns: the instance identifier (*Id*), the number of truck customers ($|V_c^1|$), the number of satellites ($|V_d|$), the truck capacity (Q_1), the instance type (*Type*) (equal to "rd" for randomly distributed and "g" for geometrical instances), and the best known solution value (*BKS*) reported in Vidal et al. (2012). For each algorithm we report (when available) the best (*Best*), the worst (*Worst*) and the average (*Avg*) gap of the solution found by the algorithm with respect to *BKS*, the computational time in seconds for 10 runs (t_{10}) and the corresponding scaled time ($\hat{t}_{10}$) computed as $\hat{t}_{10} = t_{10}(P_A/P_B)$ where P_A is our CPU single-thread rating and P_B is the CPU rating of the competing method. The gap is computed as $100 \cdot (z - BKS)/BKS$, where z is the solution value.

Because Vidal et al. (2012) reported only the average computing time of a single run, the values of t_{10} for HGSADC from the original paper have been multiplied by ten. Vidal et al. (2012) tested their algorithm on an AMD Opteron 250 CPU at 2.4 GHz. We were not able to find the exact CPU values in PassMark Software (2018) so we found a similar model, the AMD Opteron 275 at 2.2 GHz, which has a single-thread rating of 685; thus our CPU is roughly 3.95 times faster. As can be seen from the table, our algorithm was able to find a solution with the same value of the best known solution for all instances, and the average solution quality was comparable with that of

HGSADC. Moreover, as our computing times, scaled to be comparable with those of the competing method, were much shorter, we can conclude that AVXS was able to reach state-of-the-art results on MDVRP instances within a very limited amount of time.

5.4. Testing on LRP instances

We tested AVXS on the set of 36 LRP instances proposed by Tuzun and Burke (1999), which are characterized by capacitated vehicles, uncapacitated depots, and fixed costs for using vehicles and depots. Each instance has between 100 and 200 customers, and between 10 and 20 candidate depots. Customers are uniformly randomly distributed or arranged in either three or five clusters. The vehicle capacity is 150. Table 3 shows the details of our computational results; the columns are the same as for Table 2. Type "c (h)" denotes clustered instances with h clusters. We observe that AVXS is able to find solutions close to the best known ones in relatively short computing times and even detected a new best solution for instance P123222. A general comparison of our method with other state-of-the-art methods is reported in Table 4, where the details of previous methods are obtained from the recent literature survey by Schneider and Drexl (2017). The table contains the following columns: the algorithm acronym (*Algorithm*), the best (*Best*) and, when available, the average (*Avg*) percentage gap with respect to the best known solution, the average raw run-time of a single run solving the considered LRP instances in seconds (t), the PassMark Software (2018) score as reported in the survey paper (P_{score}), and the scaled running time defined as the normalized single run-time $\times$ the number of conducted runs ($\hat{t}$). Run-times are normalized according to the processor used by Lopes, Ferreira, and Santos (2016). Finally, a graphic comparison of the algorithm performance is given in Figure 7.

We conclude that our AVXS favourably compares with the best existing methods, achieving an excellent compromise between solution quality and running time, as shown by its presence on the Pareto frontier of Figure 7.

5.5. Testing on STTRPSD instances

The STTRPSD instances were proposed by Villegas et al. (2010); they are a set of 32 randomly generated Euclidean instances defined in a square grid of 100 $\times$ 100 units. The set contains randomly distributed and clustered instances including only truck customers and satellite depots. The size of the instances vary from 25 to 200 truck customers and from 5 to 20 satellites. The truck capacity is either 1000 or 2000, and the customer demand is uniformly distributed in the interval [1, 200]. The algorithm of Villegas et al. (2010) was implemented in Java. The experiments were run on a computer equipped with an Intel Pentium D 945 processor running at 3.4 GHz with 1024 MB of RAM. We were not able to find the exact CPU in PassMark Software (2018), so we referred to a similar model, namely the Intel Pentium D 940 working at 3.20 GHz, which has a single-thread

Table 3 Computations on LRP instances.

Id	$\|V_c^1\|$	$\|V_d\|$	*Type*	*BKS*	*Best*	*Avg*	*Worst*	t_{10}
P111112	100	10	rd	**1467.68**	0.00	0.00	0.04	110
P111122	100	10	rd	1448.37	0.00	0.03	0.06	183
P111212	100	10	rd	**1394.80**	0.00	0.00	0.01	100
P111222	100	10	rd	1432.29	0.00	0.21	2.08	198
P112112	100	10	c (3)	**1167.16**	0.00	0.12	0.33	588
P112122	100	10	c (3)	1102.24	0.00	0.00	0.01	584
P112212	100	10	c (3)	**791.66**	0.00	0.01	0.15	124
P112222	100	10	c (3)	728.30	0.00	0.00	0.00	131
P113112	100	10	c (5)	1238.24	0.02	0.05	0.15	542
P113122	100	10	c (5)	1245.30	0.00	0.00	0.01	438
P113212	100	10	c (5)	**902.26**	0.00	0.00	0.00	433
P113222	100	10	c (5)	1018.29	0.00	0.00	0.00	499
P131112	150	10	rd	1892.17	0.56	1.08	2.08	704
P131122	150	10	rd	1819.68	0.25	1.06	2.14	770
P131212	150	10	rd	1960.02	0.24	0.27	0.38	420
P131222	150	10	rd	1792.77	0.00	0.26	0.82	688
P132112	150	10	c (3)	**1443.32**	0.00	0.22	0.31	808
P132122	150	10	c (3)	1429.30	0.86	1.24	1.50	875
P132212	150	10	c (3)	1204.42	0.10	0.20	0.46	699
P132222	150	10	c (3)	924.68	0.70	0.79	0.89	751
P133112	150	10	c (5)	1694.18	0.08	0.58	1.52	705
P133122	150	10	c (5)	1392.01	0.37	0.60	0.79	840
P133212	150	10	c (5)	1197.95	0.00	0.01	0.02	420
P133222	150	10	c (5)	1151.37	0.07	0.15	0.25	725
P121112	200	10	rd	2237.73	0.03	0.66	1.18	863
P121122	200	10	rd	2137.45	0.15	1.02	3.66	855
P121212	200	10	rd	2195.17	0.00	0.66	1.33	883
P121222	200	10	rd	2214.86	0.51	1.44	2.43	981
P122112	200	10	c (3)	2070.43	0.61	1.31	1.94	939
P122122	200	10	c (3)	1685.52	0.86	1.76	2.90	1039
P122212	200	10	c (3)	1449.93	1.20	1.44	1.56	936
P122222	200	10	c (3)	1082.46	0.02	0.07	0.25	937
P123112	200	10	c (5)	1942.23	1.38	1.64	1.76	965
P123122	200	10	c (5)	1910.08	0.09	0.53	1.79	1064
P123212	200	10	c (5)	1761.11	0.34	0.73	1.21	1081
P123222	200	10	c (5)	1390.86	-0.01	0.01	0.04	736
Mean					0.23	0.50	0.95	655.94

Optimality has been proved for the solutions in boldface by Baldacci, Mingozzi, and Wolfler Calvo (2011).
New best solutions (instance identifier, value): (P123222, 1390.74)

rating of 758. Our CPU is thus 3.57 times faster. The authors presented four metaheuristics with various characteristics and levels of performance. In Table 5 we compare our algorithm with the algorithm MS-ILS of Villegas et al. (2010), which is the best-performing one with respect to the objective function (and just slightly more time-consuming than the second-best).

Table 5 includes several columns similar to those of previous result tables. Since Villegas et al. (2010) reported only the average computing time needed to perform a single run of their algorithm,

Table 4 Comparison with state-of-the-art LRP algorithms.

Algorithm	*Best*	*Avg*	t	P_{score}	$\hat{t}$
DLPP	1.40[a]		606.64	1200	317.55 × 5
YLLT	1.59[a]		826.47	1135	409.62 × 1
HCC (500K)	0.53	0.99	165.93	685	49.63 × 5
HCC (5000K)	0.39	0.60	1620.60	685	484.76 × 5
CCG	0.27	0.66	2589.53	1219	1378.44 × 10
ELT	1.24	1.24	392.33	776	132.94 × 1
TC	1.33[a]		202.08	546	48.18 × 10
ELBT	0.86	0.86	201.22	776	68.19 × 1
SL (speed)	0.31	0.57	65.75	1652	47.43 × 5
SL (quality)	0.04	0.15	1004.65	1652	724.75 × 5
LFS	0.96	1.55	86.02	2290	70.02 × 10
LFS+	0.80	1.19	363.61	2290	256.66 × 10
A&S	0.30	0.30	171.93	1878	141.00 × 1
AVXS	0.23	0.50	65.59	2704	77.45 × 10

[a] Data on average gaps not available.
The algorithm acronyms are: DLPP (Duhamel et al. (2010)), YLLT (Yu et al. (2010)), HCC (Hemmelmayr, Cordeau, and Crainic (2012)), CCG (Contardo, Cordeau, and Gendron (2014)), ELT (Escobar, Linfati, and Toth (2013)), TC (Ting and Chen (2013)), ELBT (Escobar et al. (2014)), SL (Schneider and Lffler (2019)), LFS (Lopes, Ferreira, and Santos (2016)), A&S (Florian (2018))

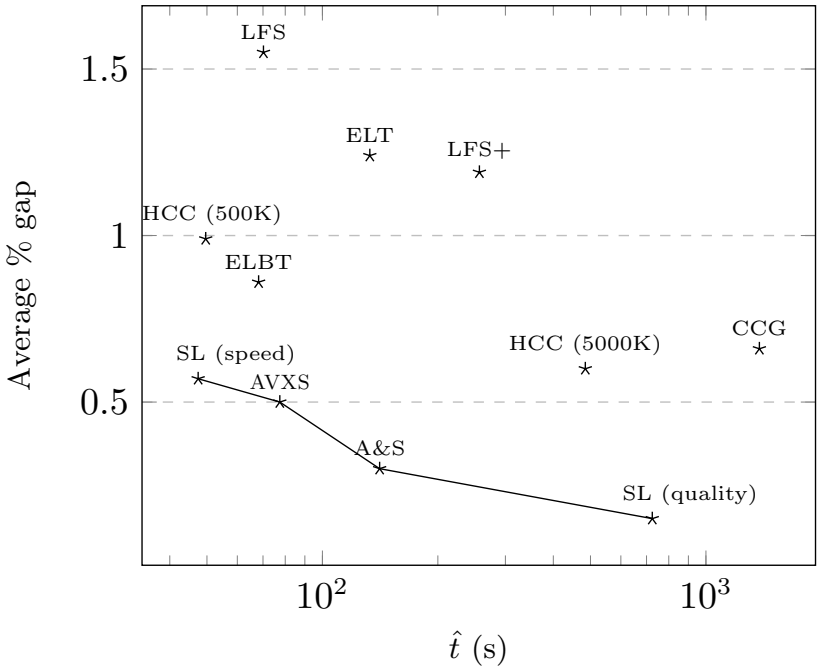


Figure 7 LRP algorithm performance comparison over the instances of Tuzun and Burke (1999). We excluded algorithms DLPP, YLLT and TC for which the average gaps are not available.

to obtain the values of t_{10} we multiplied their times by ten. As can be seen from Table 5, AVXS was able to find seven new best solutions for the largest instances. Moreover, we note that instances 1-18, 21 and 22 have already been proved to be optimal by Belenguer et al. (2016); thus we improved

Table 5 Computations on STTRPSD instances.

						MS-ILS				AVXS				
Id	$\|V_c^1\|$	$\|V_d\|$	Q_1	*Type*	*BKS*	*Best*	*Avg*	*Worst*	t_{10}	*Best*	*Avg*	*Worst*	t_{10}	$\hat{t}_{10}$
1	25	5	1000	c	**405.46**	0.00	0.00	0.00	114	0.00	0.00	0.00	10	35.67
2	25	5	2000	c	**374.79**	0.00	0.00	0.00	114	0.00	0.00	0.00	10	35.67
3	25	5	1000	rd	**584.03**	0.00	0.00	0.00	126	0.00	0.00	0.00	10	35.67
4	25	5	2000	rd	**508.48**	0.00	0.00	0.00	102	0.00	0.00	0.00	10	35.67
5	25	10	1000	c	**386.45**	0.00	0.00	0.00	132	0.00	0.00	0.00	10	35.67
6	25	10	2000	c	**380.86**	0.00	0.00	0.00	126	0.00	0.00	0.00	10	35.67
7	25	10	1000	rd	**573.96**	0.00	0.00	0.00	138	0.00	0.00	0.00	10	35.67
8	25	10	2000	rd	**506.37**	0.00	0.00	0.00	126	0.00	0.00	0.00	10	35.67
9	50	5	1000	c	**583.07**	0.00	0.00	0.00	708	0.00	0.00	0.00	10	35.67
10	50	5	2000	c	**516.98**	0.00	0.00	0.00	570	0.00	0.00	0.00	10	35.67
11	50	5	1000	rd	**870.51**	0.00	0.00	0.00	612	0.00	0.00	0.00	10	35.67
12	50	5	2000	rd	**766.03**	0.00	0.00	0.00	588	0.00	0.00	0.00	10	35.67
13	50	10	1000	c	**387.83**	0.00	0.00	0.00	822	0.00	0.00	0.00	10	35.67
14	50	10	2000	c	**367.01**	0.00	0.00	0.00	750	0.00	0.00	0.00	10	35.67
15	50	10	1000	rd	**811.28**	0.00	0.00	0.00	720	0.00	0.00	0.00	10	35.67
16	50	10	2000	rd	**731.53**	0.00	0.00	0.00	630	0.00	0.00	0.00	10	35.67
17	100	10	1000	c	**614.02**	0.00	0.06	0.21	3480	0.00	0.00	0.00	58	206.90
18	100	10	2000	c	**547.44**	0.00	0.02	0.12	3894	0.00	0.00	0.00	30	107.02
19	100	10	1000	rd	1275.76	0.33	0.55	0.81	2976	-0.31	-0.31	-0.31	72	256.84
20	100	10	2000	rd	1097.28	0.00	0.06	0.56	2604	0.00	0.00	0.00	52	185.5
21	100	20	1000	c	**642.61**	0.00	0.00	0.00	2892	0.00	0.00	0.00	65	231.87
22	100	20	2000	c	**581.56**	0.00	0.11	0.37	3600	0.00	0.00	0.00	40	142.69
23	100	20	1000	rd	1143.10	0.00	0.31	0.63	3012	0.00	0.00	0.00	58	206.9
24	100	20	2000	rd	1060.75	0.00	0.20	0.40	2946	0.01	0.24	0.27	91	324.62
25	200	10	1000	c	822.52	0.00	0.71	1.52	11544	-0.31	-0.31	-0.31	311	1109.42
26	200	10	2000	c	714.33	0.00	0.79	1.63	11682	-0.51	-0.44	-0.29	284	1013.11
27	200	10	1000	rd	1761.10	0.12	1.26	2.53	10314	-0.32	-0.32	-0.32	418	1491.12
28	200	10	2000	rd	1445.94	0.00	0.84	1.71	9288	0.00	0.00	0.00	255	909.66
29	200	20	1000	c	909.46	0.00	0.45	1.51	13746	-0.25	-0.25	-0.25	319	1137.96
30	200	20	2000	c	815.51	0.63	0.82	1.14	14646	-0.13	-0.13	-0.09	227	809.77
31	200	20	1000	rd	1614.18	0.00	1.08	2.16	12282	-0.22	-0.22	-0.22	341	1216.44
32	200	20	2000	rd	1413.32	0.00	0.81	1.81	12084	0.00	0.00	0.00	307	1095.16
Mean						0.03	0.25	0.54	3980.25	-0.06	-0.05	-0.05	96.50	344.24

Optimality has been proved for the solutions in boldface by Belenguer et al. (2016).
New best solutions (instance identifier, value): (19, 1271.78) (25, 819.96) (26, 710.70) (27, 1755.44) (29, 907.17) (30, 814.42) (31, 1610.62)

7 out of 12 of the remaining ones. The computational results show the effectiveness of our method, in terms of both solution quality and processing time (which is more than ten times shorter than the competing algorithm).

5.6. Testing on STTRP instances

In order to assess the performances of AVXS on the STTRP, we first solved the small-scale instances proposed by Bartolini and Schneider (2018) and used in a branch-and-cut algorithm. They had selected subsets of customers with sizes of 30 and 40 from the TTRP instances proposed by Chao (2002) and Lin, Yu, and Chou (2010), deriving the set of 36 instances that we used for initial

testing. Moreover, to comply with the multiple-vehicle versions of the original TTRP in which each vehicle can perform at most one route type among the ones defined by Chao (2002) (see Section 2), Bartolini and Schneider (2018) imposed the constraint of not having any sub-route directly starting from the depot even in the single-vehicle version. Since the XSTTRP does not allow sub-routes directly starting from the depot, we could capture the STTRP considered by Bartolini and Schneider (2018) as a special case of our problem. Table 6 contains the computational results of those instances. We computed the arc costs $c_{ij} = (\lfloor 10000 \cdot Euc(i,j) \rfloor)/10000$ as in the the original paper. By analyzing the computational results of the set of instances shown in Table 6, we can conclude that AVXS is able to solve small-scaled instances of this XSTTRP sub-problem effectively. In particular, the algorithm improved five best known solutions.

After successfully solving the small-sized STTRP instances, we derived larger test instances from the TTRP ones proposed by Chao (2002) by setting the trailer capacity Q_2 equal to the sum of customer demands and adding a dummy satellite at the same location as the main depot, to allow an unlimited number of sub-routes starting from the depot location. These new instances have either randomly distributed or clustered customers, who number between 50 and 200, and the capacity of the truck Q_1 is either 100 or 150. Table 7 shows our computational results with columns similar to previous tables. The number of vehicle customers with parking facilities is denoted as $|\overline{V_c^2}|$, and *BKS* identifies the best known solution value found during the experiments.

From the table we note that AVXS provides results with very small differences between the best and the worst average gap, and the computing times to perform ten runs never exceed 13 minutes for the largest instances and are, on average, less than three minutes.

5.7. Testing on XSTTRP instances

We derived a set of XSTTRP instances from the TTRP ones proposed by Chao (2002) as follows. For each TTRP instance, for n_c the number of TTRP customers, we turned n_s randomly chosen customers into satellites. From the remaining $n'_c = n_c - n_s$ customers, we randomly chose $t \cdot n'_c$ to turn into vehicle customers without parking facilities. The possible values for n_s depend on the original TTRP instance size: $n_s = 5$ if $n_c \leq 99$, $n_s \in \{5, 10\}$ when $100 \leq n_c \leq 199$ and $n_s \in \{5, 10, 20\}$ if $200 \leq n_c$. The values for t are $t \in \{0.2, 0.8\}$. Moreover, every instance contains an additional satellite placed at the depot location.

Table 8 shows our computational results; in addition to the columns already defined for previous sub-problems we include a column for the number of vehicle customers without parking facilities ($|V_c^2 \setminus \overline{V_c^2}|$).

The results show that, as for the STTRP instances, the average difference between the best and the worst average gap remains small. This experimentally indicates that AVXS is able to provide stable results for XSTTRP instances. Moreover, the average computing time to perform a single run is always less than 80 seconds and on average slightly more that 19 seconds.

Table 6 Computations on STTRP instances proposed by Bartolini and Schneider (2018).

Id	$\lvert V_c^1\rvert$	$\lvert\overline{V_c^2}\rvert$	Q_1	*Type*	*BKS*	*Best*	*Avg*	*Worst*	t_{10}
chao1.30	7	23	150	rd	**350.6**	0.00	0.00	0.00	10
chao2.30	15	15	150	rd	**371.6**	0.00	0.00	0.00	10
chao3.30	23	7	150	rd	**383.6**	0.00	0.00	0.00	10
chao4.30	7	23	150	rd	**351.7**	0.00	0.00	0.00	10
chao5.30	15	15	150	rd	**383.8**	0.00	0.00	0.00	10
chao6.30	22	8	150	rd	**435.8**	0.00	0.00	0.00	10
chao7.30	8	22	200	rd	**365.1**	0.00	0.00	0.00	10
chao8.30	15	15	100	rd	**419.5**	0.00	0.00	0.00	10
chao9.30	22	8	200	rd	**408.0**	0.00	0.00	0.00	10
chao10.30	8	22	100	rd	**411.5**	0.00	0.00	0.00	10
chao11.30	15	15	100	rd	**398.8**	0.00	0.00	0.00	10
chao12.30	22	8	100	rd	**471.8**	0.00	0.00	0.00	10
tai1	7	23	750	c	**508.9**	0.02	0.02	0.02	10
tai2	15	15	750	c	**711.3**	0.00	0.00	0.00	10
tai3	22	8	750	c	**757.4**	0.00	0.00	0.00	10
tai4	7	23	850	c	**310.6**	0.00	0.00	0.00	10
tai5	15	15	850	c	**333.9**	0.00	0.00	0.00	10
tai6	22	8	850	c	**359.1**	0.00	0.00	0.00	10
tai7	7	23	600	c	**491.1**	0.00	0.00	0.00	10
tai8	15	15	600	c	**538.5**	0.00	0.00	0.00	10
tai9	23	7	600	c	**585.4**	0.00	0.00	0.00	10
tai10	7	23	850	c	**608.8**	0.00	0.00	0.00	10
tai11	15	15	850	c	**667.6**	0.00	0.00	0.00	10
tai12	23	7	850	c	**719.1**	0.00	0.00	0.00	10
chao1.40	10	30	150	rd	399.2	0.03	0.03	0.03	10
chao2.40	20	20	150	rd	452.5	-2.48	-2.48	-2.48	10
chao3.40	30	10	150	rd	472.3	0.00	0.00	0.00	10
chao4.40	10	30	150	rd	415.2	-0.17	-0.17	-0.17	10
chao5.40	20	20	150	rd	455.5	0.00	0.00	0.00	10
chao6.40	30	10	150	rd	501.0	0.00	0.00	0.00	10
chao7.40	10	30	200	rd	423.1	-0.21	-0.21	-0.21	10
chao8.40	20	20	100	rd	461.5	0.00	0.00	0.00	10
chao9.40	30	10	200	rd	449.0	0.00	0.00	0.00	10
chao10.40	10	30	100	rd	502.2	-3.33	-3.33	-3.33	10
chao11.40	20	20	100	rd	500.7	0.00	0.00	0.00	10
chao12.40	30	10	100	rd	576.6	-0.36	-0.36	-0.36	10
Mean						-0.18	-0.18	-0.18	10

Optimality has been proved for the solutions in boldface by Bartolini and Schneider (2018), for the remaining ones *BKS* reports the upper-bound provided by Bartolini and Schneider (2018).
New best solutions (instance identifier, value): (chao2.40, 441.3) (chao4.40, 414.5) (chao7.40, 422.2) (chao10.40, 485.5) (chao12.40, 574.5)
The objective values from which we computed the above gaps were rounded to the first decimal place as in the original paper.

6. Algorithm components analysis

This section examines the most relevant design choices by providing a detailed analysis of the different components of AVXS, namely: *(i)* the construction procedure to build an initial feasible

Table 7 Computations on STTRP instances.

Id	$\lvert V_c^1\rvert$	$\lvert\overline{V_c^2}\rvert$	Q_1	*Type*	*BKS*	*Best*	*Avg*	*Worst*	t_{10}
1	12	38	100	rd	486.07	0.00	0.00	0.00	19
2	25	25	100	rd	548.14	0.00	0.00	0.00	20
3	37	13	100	rd	583.32	0.00	0.00	0.00	10
4	18	57	100	rd	617.13	0.00	0.01	0.09	41
5	37	38	100	rd	676.42	0.00	0.00	0.00	44
6	56	19	100	rd	769.88	0.00	0.00	0.00	30
7	25	75	100	rd	687.64	0.00	0.16	0.28	92
8	50	50	150	rd	745.19	0.00	0.00	0.00	102
9	75	25	150	rd	821.31	0.00	0.17	0.36	78
10	37	113	150	rd	790.54	0.00	0.13	0.23	377
11	75	75	150	rd	857.27	0.00	0.02	0.16	285
12	112	38	150	rd	936.00	0.06	0.21	0.39	249
13	49	150	150	rd	875.85	0.15	0.34	0.71	769
14	99	100	150	rd	950.80	0.19	0.63	1.15	620
15	149	50	150	rd	1049.97	0.18	0.26	0.43	476
16	30	90	150	c	579.29	0.00	0.08	0.18	140
17	60	60	150	c	611.30	0.00	0.00	0.00	136
18	90	30	150	c	698.57	0.00	0.00	0.00	73
19	25	75	150	c	541.87	0.00	0.00	0.00	65
20	50	50	150	c	582.62	0.00	0.00	0.00	75
21	75	25	150	c	676.13	0.00	0.00	0.00	61
Mean						0.03	0.10	0.19	179.14

solution, *(ii)* the different neighborhoods explored in the RVND and *(iii)* the shaking procedures used in the improvement phase, *(iv)* the granular speedup strategy, and *(v)* the set-partitioning-based post-optimization phase. The results presented here refer primarily to the XSTTRP variant, the most general of those we studied. However, similar conclusions apply to the other variants considered in this paper.

Table 8 Computations on XSTTRP instances

Id	$\lvert V_c^1\rvert$	$\lvert V_c^2 \setminus \overline{V_c^2}\rvert$	$\lvert\overline{V_c^2}\rvert$	$\lvert V_d\rvert$	Q_1	*Type*	*BKS*	*Best*	*Avg*	*Worst*	t_{10}
1	73	17	5	6	150	c	753.43	0.00	0.00	0.00	61
2	66	4	20	11	150	c	602.99	0.00	0.00	0.00	40
3	67	18	5	11	150	c	696.03	0.00	0.00	0.00	50
4	22	4	19	6	100	rd	505.23	0.00	0.00	0.00	10
5	21	19	5	6	100	rd	517.86	0.00	0.00	0.00	10
6	18	10	42	6	100	rd	586.87	0.00	0.00	0.00	40
7	18	41	11	6	100	rd	624.39	0.00	0.00	0.00	30
8	23	14	58	6	150	c	538.40	0.00	0.00	0.00	60
9	25	56	14	6	150	c	582.27	0.00	0.00	0.00	40
10	21	13	56	11	150	c	522.54	0.00	0.00	0.00	61
11	22	54	14	11	150	c	556.61	0.00	0.00	0.00	43
12	108	7	30	6	150	rd	916.39	0.00	0.10	0.19	224
13	108	29	8	6	150	rd	935.35	0.00	0.00	0.00	184
14	106	6	28	11	150	rd	906.58	0.02	0.12	0.29	365
15	103	29	8	11	150	rd	896.46	0.02	0.26	0.50	196

Table 8 Computations on XSTTRP instances

Id	$\lvert V_c^1\rvert$	$\lvert V_c^2 \setminus \overline{V_c^2}\rvert$	$\lvert\overline{V_c^2}\rvert$	$\lvert V_d\rvert$	Q_1	*Type*	*BKS*	*Best*	*Avg*	*Worst*	t_{10}
16	36	21	88	6	150	rd	776.73	0.00	0.12	0.36	315
17	35	88	22	6	150	rd	792.22	0.00	0.27	0.77	207
18	36	20	84	11	150	rd	766.28	0.03	0.12	0.31	308
19	35	84	21	11	150	rd	773.97	0.00	0.03	0.09	181
20	48	9	38	6	150	rd	735.26	0.00	0.00	0.01	90
21	48	37	10	6	150	rd	743.71	0.00	0.00	0.00	73
22	45	9	36	11	150	rd	700.54	0.00	0.00	0.00	95
23	45	36	9	11	150	rd	715.08	0.00	0.01	0.10	81
24	52	3	15	6	100	rd	716.31	0.00	0.00	0.00	30
25	52	14	4	6	100	rd	802.23	0.00	0.00	0.00	40
26	34	7	29	6	100	rd	652.27	0.00	0.00	0.00	41
27	34	28	8	6	100	rd	661.35	0.00	0.00	0.00	30
28	71	14	60	6	150	rd	849.31	0.05	0.16	0.40	290
29	73	57	15	6	150	rd	887.13	0.00	0.23	0.43	229
30	73	13	54	11	150	rd	820.05	0.02	0.13	0.24	264
31	71	55	14	11	150	rd	858.46	0.01	0.10	0.41	207
32	25	14	56	6	150	rd	667.99	0.00	0.03	0.05	80
33	24	56	15	6	150	rd	671.63	0.00	0.04	0.19	56
34	22	13	55	11	150	rd	627.53	0.00	0.01	0.04	90
35	23	53	14	11	150	rd	651.17	0.04	0.09	0.12	62
36	98	19	77	6	150	rd	948.90	0.43	0.54	0.89	582
37	98	76	20	6	150	rd	980.57	0.10	0.23	0.47	418
38	95	18	76	11	150	rd	931.48	0.00	0.24	0.60	566
39	93	76	20	11	150	rd	963.67	0.48	0.75	1.06	439
40	92	17	70	21	150	rd	900.43	0.02	0.21	0.39	525
41	88	72	19	21	150	rd	929.58	0.34	0.72	1.42	454
42	71	4	20	6	150	rd	787.71	0.00	0.03	0.27	90
43	72	18	5	6	150	rd	784.69	0.00	0.26	0.51	61
44	68	4	18	11	150	rd	758.79	0.01	0.05	0.12	92
45	67	18	5	11	150	rd	773.22	0.00	0.00	0.00	72
46	144	10	40	6	150	rd	1040.68	0.00	0.03	0.12	509
47	145	39	10	6	150	rd	1085.86	0.17	0.29	0.50	460
48	140	9	40	11	150	rd	1009.72	0.02	0.05	0.07	403
49	143	36	10	11	150	rd	1069.28	0.06	0.31	0.67	576
50	132	9	38	21	150	rd	971.67	0.05	0.28	0.70	477
51	135	35	9	21	150	rd	997.43	0.00	0.12	0.35	371
52	57	11	47	6	150	c	603.08	0.00	0.00	0.00	90
53	58	45	12	6	150	c	666.36	0.00	0.00	0.00	73
54	53	11	46	11	150	c	589.21	0.00	0.00	0.03	111
55	54	44	12	11	150	c	607.69	0.00	0.02	0.05	80
56	29	17	69	6	150	c	570.53	0.00	0.01	0.13	121
57	29	68	18	6	150	c	580.00	0.00	0.00	0.00	79
58	26	16	68	11	150	c	561.61	0.08	0.14	0.27	122
59	28	65	17	11	150	c	560.75	0.00	0.12	0.41	70
60	87	5	23	6	150	c	645.86	0.00	0.00	0.00	74
61	86	23	6	6	150	c	777.00	0.00	0.00	0.00	91
62	84	5	21	11	150	c	642.40	0.00	0.00	0.00	68
63	83	21	6	11	150	c	717.47	0.00	0.01	0.11	69
64	33	2	10	6	100	rd	557.56	0.00	0.00	0.00	10
65	34	8	3	6	100	rd	553.54	0.00	0.00	0.00	10
66	49	29	116	6	150	rd	869.02	0.12	0.24	0.35	622
67	47	117	30	6	150	rd	897.21	0.16	0.52	0.90	433
68	44	29	116	11	150	rd	858.61	0.03	0.39	0.65	726

Table 8 Computations on XSTTRP instances

Id	$\lvert V_c^1\rvert$	$\lvert V_c^2 \setminus \overline{V_c^2}\rvert$	$\lvert\overline{V_c^2}\rvert$	$\lvert V_d\rvert$	Q_1	*Type*	*BKS*	*Best*	*Avg*	*Worst*	t_{10}
69	48	112	29	11	150	rd	878.07	0.04	0.51	0.91	388
70	43	27	109	21	150	rd	825.73	0.16	0.63	0.85	725
71	45	107	27	21	150	rd	860.21	0.41	0.71	1.12	416
72	11	6	28	6	100	rd	456.01	0.00	0.00	0.00	18
73	12	26	7	6	100	rd	485.42	0.00	0.00	0.00	10
74	49	9	37	6	150	c	573.69	0.00	0.00	0.00	66
75	47	38	10	6	150	c	660.97	0.00	0.00	0.00	51
76	44	9	37	11	150	c	567.98	0.00	0.00	0.00	68
77	47	34	9	11	150	c	594.29	0.00	0.00	0.00	40
Mean								0.04	0.12	0.24	193.62

6.1. Assignment fitness functions

The quality of the initial solution is tightly linked to the definition of what we called the assignment fitness function, see Section 4.1. Due to the highly complex interactions between heuristic algorithm components, good starting points do not guarantee a final solution of superior quality. In this paragraph we experimentally analyze the effect of using different assignment fitness functions on the quality of the initial and final solutions, as well as on the search diversification achieved. To this end, we compared two general approaches to defining the function, using a range of different parameters. The first approach was the restricted GRASP-based function d-NEAR that we defined in Section 4.1. In contrast, the second was a rank-based function b-RANK, defined as $f^i(j) = 1 + (|\overline{V}_d^+| - r_j)^b$, where r_j is the position for j in a list including all $k \in V$ and sorted according to the assignment costs $\hat{c}$; b is the scaling factor. The b-RANK function can be classified as an empirical bias function. We refer the interested reader to the work by Grasas et al. (2017) for a general survey on biased randomized procedures with theoretical or empirical bias functions. More precisely, we ran algorithm AVXS for ten executions on all instances of the XSTTRP data set, and Table 9 shows, for each assignment function (*Function)*, the average gap of the starting solutions obtained after the construction phase averaged over the Δ restarts (*Start*), the best, average and worst final gap (*Best*, *Avg*, *Worst*) averaged over all instances, the average total computing time for the ten runs in seconds (t_{10}), the average percentage improvement provided by the polishing phase (*sp*), the percentage of time to perform the polishing phase with reference to the total time of the algorithm ($\%t_{sp}$), and the number of routes used in the set-partitioning model F ($|\mathcal{P}|$). From the table it is clear that neither the final solution gap nor the computing time depend on the initial gap, but the differences in computing time are strongly correlated with the post-optimization polishing phase. In fact, the standard deviation of the computing time among the different approaches without considering the set-partitioning resolution is less than five seconds. Moreover, some randomization on the initial

Table 9 Computations on XSTTRP instances with different assignment fitness functions.

Function	*Start*	*Best*	*Avg*	*Worst*	t_{10}	*sp*	$\%t_{sp}$	$\lvert\mathcal{P}\rvert$
1-RANK	341.25	0.05	0.13	0.26	202.57	0.05	32.85	6862.04
2-RANK	281.65	0.04	0.12	0.24	196.01	0.05	30.60	6564.64
3-RANK	243.35	0.04	0.13	0.24	190.97	0.05	29.12	6328.70
4-RANK	217.20	0.04	0.13	0.24	186.70	0.05	28.56	6142.69
5-RANK	196.56	0.04	0.13	0.25	185.74	0.05	27.31	5986.62
1-NEAR	36.22	0.09	0.23	0.35	134.19	0.04	9.31	2591.15
2-NEAR	52.13	0.05	0.15	0.27	145.42	0.05	14.69	3764.73
5-NEAR	109.16	0.05	0.13	0.25	160.36	0.04	19.77	4973.16
10-NEAR	184.40	0.05	0.13	0.23	175.64	0.05	26.11	5859.38
25-NEAR	311.53	0.04	0.12	0.24	193.62	0.04	31.76	6738.41

assignment was beneficial compared to the deterministic assignment of the 1-NEAR function. In fact, all considered approaches (except the 1-NEAR) behaved in a similar way. Among the best performing functions we note that the 25-NEAR was the one that obtained good results while being conceptually simpler than the empirical bias functions of the b-RANK family. As an additional note, we report that using the 3-RANK we were able to find an additional best known solution for instance 28 of the STTRPSD with an objective value of 1445.05 (0.06% better than the current one). Other problem variants produced very similar results, with the exception of LRP in which the deterministic 1-NEAR turned out to be preferable, because of the strong influence of the fixed costs in the solution process (as was discussed in Section 4.5).

6.2. Local search

As extensively described in Section 4, our local search engine is based on an RVND scheme that searches five different neighborhoods in random order according to a randomized first-improvement strategy. In this section, we analyze the impact of the various neighborhoods we used. First of all, we observed that all the neighborhoods contributed significantly to the effectiveness of the local search step. More precisely, let $\mathbf{N}$ be the set of all the neighborhoods we considered, for each neighborhood $\mathcal{N} \in \mathbf{N}$ we stored the total improvement value $D(\mathcal{N})$ it produced and the number $I(\mathcal{N})$ of times a complete application of $\mathcal{N}$ resulted in an improvement. Then, for each neighborhood $\mathcal{N}$, we computed a *Relative Neighborhood Effectiveness Index* or RNEI($\mathcal{N}$), which defines the effectiveness of $\mathcal{N}$ with respect to the other neighborhoods. In particular, RNEI($\mathcal{N}$) is computed as RNEI($\mathcal{N}$) $= D(\mathcal{N})I_\Sigma / D_\Sigma I(\mathcal{N})$ where $D_\Sigma = \sum_{\mathcal{N}' \in \mathbf{N}} T(\mathcal{N}')$ and $I_\Sigma = \sum_{\mathcal{N}' \in \mathbf{N}} I(\mathcal{N}')$. In other words, RNEI($\mathcal{N}$) measures the successful improvement that any application of $\mathcal{N}$ would produce on an XSTTRP solution compared to the application of any other neighborhood in the set.

Figure 8 shows an aggregate measure, averaged over ten runs, for all the XSTTRP instances, for each neighborhood $\mathcal{N}$ computed as $100 \cdot \text{RNEI}(\mathcal{N}) / \sum_{\mathcal{N}' \in \mathbf{N}} \text{RNEI}(\mathcal{N}')$. The values (X, Y) reported

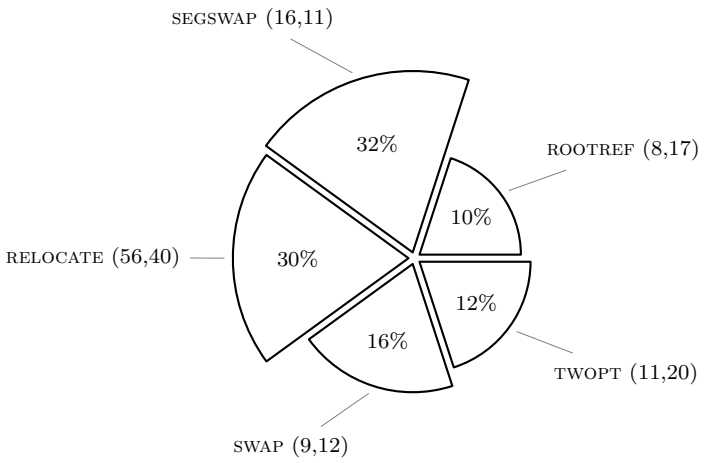


Figure 8 The percentage Relative Neighborhood Effectiveness Index for the XSTTRP instances.

beside each neighborhood $\mathcal{N}$ name are $X = 100 \cdot D(\mathcal{N})/D_\Sigma$ and $Y = 100 \cdot I(\mathcal{N})/I_\Sigma$. For example, the values reported for SEGSWAP show that its application was less frequently successful with respect to other neighborhoods (11%) but the average improvement that it had produced (16%) during those applications made it the most effective operator, with an RNEI percentage score equal to 32%. However, Figure 8 clearly shows that all neighborhoods made a positive contribution to the overall local search effectiveness. We also note that because we used an RVND approach, the RNEI measure is not affected by the order in which the neighborhoods are examined.

Next, we compared our implemented scheme to a classical VND with fixed operators and moves ordering. In particular, we examined a naive reasonable order given by RELOCATE, SWAP, TWOPT, SEGSWAP, and ROOTREF and a best-RNEI order defined as SEGSWAP, RELOCATE, SWAP, TWOPT, and ROOTREF. We executed ten runs of AVXS on all XSTTRP instances with both approaches. The results with the RVND were slightly better than those with the VND. In particular, the final gap obtained by the RVND over all instances was about 0.03%, resp. 0.01%, smaller when compared to the VND with naive, resp. best-RNEI, ordering. The computing time was similar for all the three approaches. Moreover, the solutions produced by the RVND appeared to be more diversified, because the set-partitioning polishing phase obtained better final results with less effort when fed with the RVND solutions rather than with those of the VND. Thus, we can conclude that the adoption of randomization in the selection of the next neighborhood (and of the next move within the neighborhood) may play an important role in improving the final solution quality without lengthening the computational time.

In addition, we evaluated what happens if a certain operator is removed entirely. The results are shown in Figure 9. On the one hand, none of the operators seems mandatory to obtain high quality final solutions. On the other hand, all of them play a role in determining the best possible result.

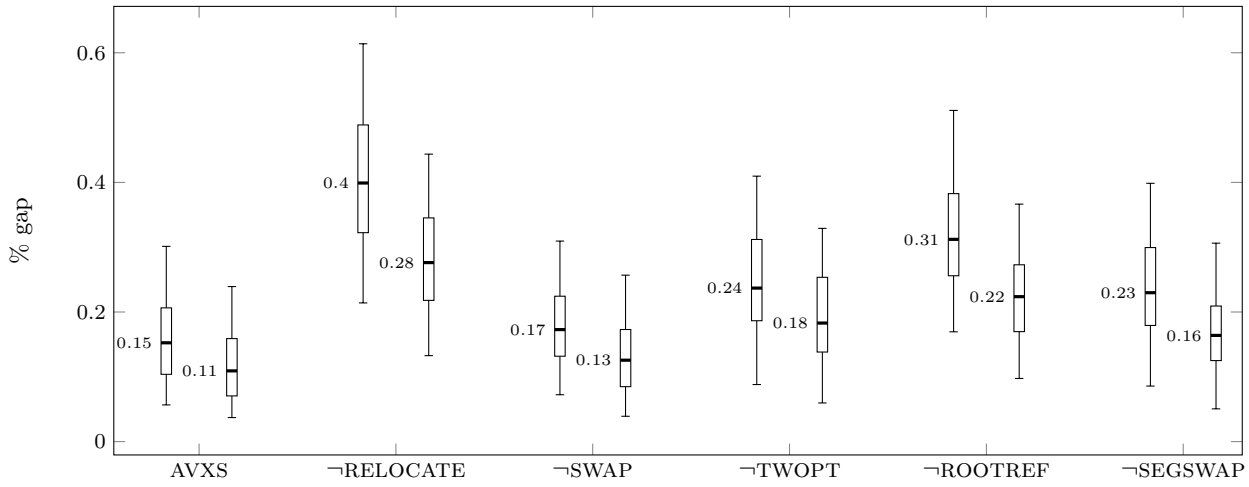


Figure 9 **Computations on XSTTRP instances with the complete approach (AVXS) and disabling one local search operator at a time (¬OPERATOR). For each configuration, the left box-plot represents the results before the polishing phase and the right one, the results after the polishing phase. The median value is shown on the left of each box-plot.**

Finally, we analyzed the impact of the two shake operators we described in Section 4.3. We experimentally observed that both operators proved to be effective in inducing improvements in the solutions. In particular, over all XSTTRP instances, the percentage improvement associated with a local search step following a SUNLOAD shaking was equal to 39% of the total improvement, while that associated with SREM was 61%. Similar results were obtained with the other problem classes; therefore we can conclude that both shaking operators are required for maximum effectiveness.

6.3. Granular neighborhoods adoption

We studied the effect of granular neighborhoods by comparing the average solution quality and the computing time needed to run AVXS, using either the granular or the complete neighborhood exploration. The results confirmed that use of granular neighborhoods is an effective speedup technique, greatly improving computing time while only slightly decreasing solution quality. In particular, we observed that the average running time to execute ten runs over all XSTTRP instances with complete neighborhoods exploration was 904.87 seconds and the average best gap was 0.02%. In contrast, using granular neighborhoods, as reported in Table 8, the average computing time was only 193.62 seconds and the average best gap was 0.04%. We can conclude that the adoption of granular neighborhoods make it possible to reach high-quality solutions with a very limited running time (almost five times shorter than that of an improvement phase considering complete neighborhoods).

6.4. Set-partitioning post-optimization

The polishing phase glues together the different restarts, making the overall algorithm more robust and stable. The bar diagram in Figure 10 shows the average polishing phase improvement, the

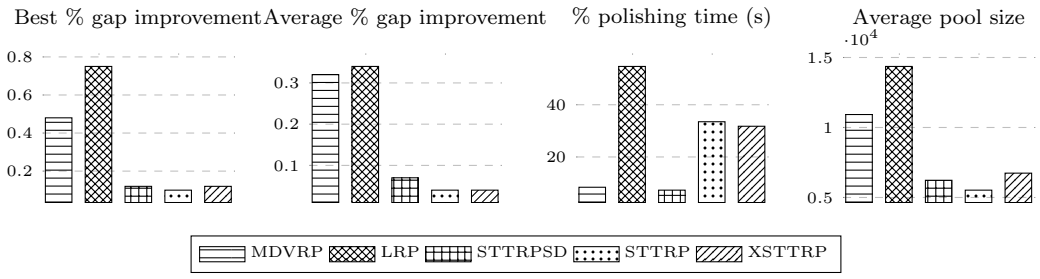


Figure 10 Polishing phase contribution

average percent processing time with reference to that of the complete algorithm, and the average size of the pool $\mathcal{P}$ for the different problem variants we considered. The figure shows that for all variants, the polishing phase was effective. Moreover, with the exception of LRP (which had a considerably larger solution pool), the phase's computing time was always less than 35% of the total. As one can expect, the improvement is much more relevant for problems that required additional care to be modeled as XSTTRP. In fact, both the MDVRP and the LRP have additional constraints which are not explicitly considered by the XSTTRP model and the local search procedures. On the other hand, the core part of AVXS is alone able to generate high quality solutions for the STTRPSD and the STTRP which are straightforward XSTTRP special cases. However, by observing Figure 9, one can still see the beneficial effects of the polishing phase on the XSTTRP instances, in terms of improvement and stabilization of the results.

7. Conclusions

In this paper we presented AVXS, an effective hybrid metaheuristic for a general variant of single-vehicle truck and trailer routing problems called the Extended Single Truck and Trailer Routing Problem (XSTTRP). The AVXS algorithm is based on a four-phase solution approach in which the main improvement phase consists of an iterated local search (ILS) incorporating two shaking procedures and a randomized variable neighborhood descent with granular speedup. A set of high-quality solutions generated during the ILS is stored in a pool, which is used for a final polishing phase based on the solution of a restricted mathematical formulation of the problem. Therefore, AVXS can also be classified as a matheuristic (see Maniezzo, Stützle, and Voß (2010)).

The AVXS algorithm was used to solve, with very short computing times, a large set of instances for several variants of the problem and proved able to reach state-of-the-art results for those variants already studied in the literature. In particular, AVXS was able to detect one new best solution for the extensively studied Location Routing Problem and seven new best solutions for the Single Truck and Trailer Routing Problem with Satellite Depots.

The study of the XSTTRP variant opens different research possibilities that could enrich the problem's definition such as, for example, considering time windows and extending it to multiple

vehicles. Furthermore, the development of exact methods, which are seldom studied in the truck and trailer literature, might be a difficult but extremely interesting research direction.

Acknowledgments

This research was partially funded by Ministero dell'Istruzione dell'Università e della Ricerca, Italy under grant PRIN n. 2015JJLC3E_002 and by United States Air Force Office of Scientific Research under award number FA9550-17-1-0234.

References

Accorsi L, 2017 *Validi Lower Bound al Single Truck and Trailer Routing Problem*. Master's thesis, Università di Bologna.

Baldacci R, Mingozzi A, 2008 *A unified exact method for solving different classes of vehicle routing problems. Mathematical Programming* 120(2):347, URL http://dx.doi.org/10.1007/s10107-008-0218-9.

Baldacci R, Mingozzi A, Wolfler Calvo R, 2011 *An exact method for the capacitated location-routing problem. Operations Research* 59(5):1284–1296, URL http://dx.doi.org/10.1287/opre.1110.0989.

Bartolini E, Schneider M, 2018 *A two-commodity flow formulation for the capacitated truck-and-trailer routing problem. Discrete Applied Mathematics* URL http://dx.doi.org/10.1016/j.dam.2018.07.033.

Belenguer J, Benavent E, Martínez A, Prins C, Prodhon C, Villegas JG, 2016 *A branch-and-cut algorithm for the single truck and trailer routing problem with satellite depots. Transportation Science* 50(2):735–749, URL http://dx.doi.org/10.1287/trsc.2014.0571.

Bodin L, Levy L, 2000 *Scheduling of local delivery carrier routes for the united states postal service. In M. Dror, editor, Arc Routing: Theory, Solutions and Applications* 419–442.

Caramia M, Guerriero F, 2010 *A milk collection problem with incompatibility constraints. Interfaces* 40(2):130–143, URL http://dx.doi.org/10.1287/inte.1090.0475.

Chao I, 2002 *A tabu search method for the truck and trailer routing problem. Computers & OR* 29(1):33–51, URL http://dx.doi.org/10.1016/S0305-0548(00)00056-3.

Clarke G, Wright JW, 1964 *Scheduling of vehicles from a central depot to a number of delivery points. Operations Research* 12(4):568–581, URL http://dx.doi.org/10.1287/opre.12.4.568.

Contardo C, Cordeau JF, Gendron B, 2014 *A grasp + ilp-based metaheuristic for the capacitated location-routing problem. Journal of Heuristics* 20(1):1–38, URL http://dx.doi.org/10.1007/s10732-013-9230-1.

Cordeau JF, Gendreau M, Laporte G, 1997 *A tabu search heuristic for periodic and multi-depot vehicle routing problems. Networks* 30(2):105–119, URL http://dx.doi.org/10.1002/(SICI)1097-0037(199709)30:2<105::AID-NET5>3.0.CO;2-G.

CPLEX, 2017 *Ibm ilog cplex optimizer 12.8 callable library* .

Cuda R, Guastaroba G, Speranza MG, 2015 *A survey on two-echelon routing problems. Computers & OR* 55:185–199, URL http://dx.doi.org/10.1016/j.cor.2014.06.008.

Drexl M, 2011 *Branch-and-price and heuristic column generation for the generalized truck-and-trailer routing problem. Journal of Quantitative Methods for Economics and Business Administration* 12:5–38.

Drexl M, 2014 *Branch-and-cut algorithms for the vehicle routing problem with trailers and transshipments. Networks* 63(1):119–133, URL http://dx.doi.org/10.1002/net.21526.

Duhamel C, Lacomme P, Prins C, Prodhon C, 2010 *A grasp×els approach for the capacitated location-routing problem. Comput. Oper. Res.* 37(11):1912–1923, URL http://dx.doi.org/10.1016/j.cor.2009.07.004.

Escobar JW, Linfati R, Baldoquin MG, Toth P, 2014 *A granular variable tabu neighborhood search for the capacitated location-routing problem. Transportation Research Part B: Methodological* 67:344 – 356, URL http://dx.doi.org/10.1016/j.trb.2014.05.014.

Escobar JW, Linfati R, Toth P, 2013 *A two-phase hybrid heuristic algorithm for the capacitated location-routing problem. Computers & Operations Research* 40(1):70 – 79, URL http://dx.doi.org/10.1016/j.cor.2012.05.008.

Florian A, 2018 *Efficient Heuristics for Routing and Integrated Logistics*. Ph.D. thesis, University of Antwerp, Faculty of Applied Economics.

Gerdessen J, 1996 *Vehicle routing problem with trailers. European Journal of Operations Research* 93:135–147.

Grasas A, Juan AA, Faulin J, de Armas J, Ramalhinho H, 2017 *Biased randomization of heuristics using skewed probability distributions: A survey and some applications. Computers & Industrial Engineering* 110:216 – 228, URL http://dx.doi.org/10.1016/j.cie.2017.06.019.

Hemmelmayr VC, Cordeau JF, Crainic TG, 2012 *An adaptive large neighborhood search heuristic for two-echelon vehicle routing problems arising in city logistics. Computers & Operations Research* 39(12):3215 – 3228, URL http://dx.doi.org/10.1016/j.cor.2012.04.007.

Lin SW, Yu VF, Chou SY, 2010 *A note on the truck and trailer routing problem. Expert Systems with Applications* 37(1):899 – 903, URL http://dx.doi.org/10.1016/j.eswa.2009.06.077.

Lopes RB, Ferreira C, Santos BS, 2016 *A simple and effective evolutionary algorithm for the capacitated locationrouting problem. Computers & Operations Research* 70:155 – 162, URL http://dx.doi.org/10.1016/j.cor.2016.01.006.

Lourenço HR, Martin OC, Stützle T, 2003 *Iterated Local Search*, 320–353 (Boston, MA: Springer US), ISBN 978-0-306-48056-0, URL http://dx.doi.org/10.1007/0-306-48056-5_11.

Maniezzo V, Stützle T, Voß S, eds., 2010 *Matheuristics - Hybridizing Metaheuristics and Mathematical Programming*, volume 10 of *Annals of Information Systems* (Springer), ISBN 978-1-4419-1305-0, URL http://dx.doi.org/10.1007/978-1-4419-1306-7.

Matsumoto M, Nishimura T, 1998 *Mersenne twister: A 623-dimensionally equidistributed uniform pseudo-random number generator. ACM Trans. Model. Comput. Simul.* 8(1):3–30, URL http://dx.doi.org/10.1145/272991.272995.

PassMark Software, 2018 *Professional cpu benchmarks.* URL https://www.passmark.com/index.html.

Schneider M, Drexl M, 2017 *A survey of the standard location-routing problem. Annals of Operations Research* 259(1):389–414, URL http://dx.doi.org/10.1007/s10479-017-2509-0.

Schneider M, Lffler M, 2019 *Large composite neighborhoods for the capacitated location-routing problem. Transportation Science* 53(1):301–318, URL http://dx.doi.org/10.1287/trsc.2017.0770.

Schneider M, Schwahn F, Vigo D, 2017 *Designing granular solution methods for routing problems with time windows. European Journal of Operational Research* 263(2):493–509, URL http://dx.doi.org/10.1016/j.ejor.2017.04.059.

Schrimpf G, Schneider J, Stamm-Wilbrandt H, Dueck G, 2000 *Record breaking optimization results using the ruin and recreate principle. J. Comput. Phys.* 159(2):139–171, URL http://dx.doi.org/10.1006/jcph.1999.6413.

Semet F, 1995 *A two-phase algorithm for the partial accessibility constrained vehicle routing. Annals of Operations Research* 45(11):1233–1246.

Semet F, Taillard ÉD, 1993 *Solving real-life vehicle routing problems efficiently using tabu search. Annals OR* 41(4):469–488, URL http://dx.doi.org/10.1007/BF02023006.

Subramanian A, Uchoa E, Ochi LS, 2013 *A hybrid algorithm for a class of vehicle routing problems. Computers & OR* 40(10):2519 – 2531, URL http://dx.doi.org/10.1016/j.cor.2013.01.013.

Taillard , Badeau P, Gendreau M, Guertin F, Potvin JY, 1997 *A tabu search heuristic for the vehicle routing problem with soft time windows. Transportation Science* 31(2):170–186, URL http://dx.doi.org/10.1287/trsc.31.2.170.

Tan KC, Chew YH, Lee LH, 2006 *A hybrid multi-objective evolutionary algorithm for solving truck and trailer vehicle routing problems. European Journal of Operational Research* 172(3):855–885, URL http://dx.doi.org/10.1016/j.ejor.2004.11.019.

Ting CJ, Chen CH, 2013 *A multiple ant colony optimization algorithm for the capacitated location routing problem. International Journal of Production Economics* 141(1):34 – 44, URL http://dx.doi.org/10.1016/j.ijpe.2012.06.011, meta-heuristics for manufacturing scheduling and logistics problems.

Toth P, Vigo D, 2002 *Branch-and-bound algorithms for the capacitated VRP.* Toth P, Vigo D, eds., *The Vehicle Routing Problem*, 29–51, Monographs on Discrete Mathematics and Applications (Philadelphia, PA: S.I.A.M.).

Toth P, Vigo D, 2003 *The granular tabu search and its application to the vehicle-routing problem. INFORMS Journal on Computing* 15(4):333–346, URL http://dx.doi.org/10.1287/ijoc.15.4.333.24890.

Tuzun D, Burke LI, 1999 *A two-phase tabu search approach to the location routing problem. European Journal of Operational Research* 116(1):87 – 99, URL http://dx.doi.org/10.1016/S0377-2217(98)00107-6.

Uchoa E, Pecin D, Pessoa A, Poggi M, Vidal T, Subramanian A, 2017 *New benchmark instances for the capacitated vehicle routing problem. European Journal of Operational Research* 257(3):845 – 858, URL http://dx.doi.org/10.1016/j.ejor.2016.08.012.

Vidal T, Crainic TG, Gendreau M, Lahrichi N, Rei W, 2012 *A hybrid genetic algorithm for multidepot and periodic vehicle routing problems. Operations Research* 60(3):611–624, URL http://dx.doi.org/10.1287/opre.1120.1048.

Villegas JG, Prins C, Prodhon C, Medaglia AL, Velasco N, 2010 *GRASP/VND and multi-start evolutionary local search for the single truck and trailer routing problem with satellite depots. Eng. Appl. of AI* 23(5):780–794, URL http://dx.doi.org/10.1016/j.engappai.2010.01.013.

Villegas JG, Prins C, Prodhon C, Medaglia AL, Velasco N, 2011 *A GRASP with evolutionary path relinking for the truck and trailer routing problem. Computers & OR* 38(9):1319–1334, URL http://dx.doi.org/10.1016/j.cor.2010.11.011.

Villegas JG, Prins C, Prodhon C, Medaglia AL, Velasco N, 2013 *A matheuristic for the truck and trailer routing problem. European Journal of Operational Research* 230(2):231–244, URL http://dx.doi.org/10.1016/j.ejor.2013.04.026.

Yu VF, Lin SW, Lee W, Ting CJ, 2010 *A simulated annealing heuristic for the capacitated location routing problem. Computers & Industrial Engineering* 58(2):288 – 299, URL http://dx.doi.org/10.1016/j.cie.2009.10.007.