Computers & Operations Research 53 (2015) 234–249

Contents lists available at ScienceDirect

Computers & Operations Research

journal homepage: www.elsevier.com/locate/caor

An iterated local search heuristic for the split delivery vehicle routing problem

Marcos Melo Silva [a,*], Anand Subramanian [b], Luiz Satoru Ochi [a]

[a] Universidade Federal Fluminense, Instituto de Computação, Rua Passo da Pátria 156, Bloco E – 3° andar, São Domingos, Niterói-RJ 24210-240, Brazil
[b] Universidade Federal da Paraíba, Departamento de Engenharia de Produção, Centro de Tecnologia, Campus I – Bloco G, Cidade Universitária, João Pessoa-PB 58051-970, Brazil

ARTICLE INFO

Available online 19 August 2014

Keywords:
Metaheuristics
Vehicle routing
Split deliveries
Iterated local search

ABSTRACT

This paper concerns the Split Delivery Vehicle Routing Problem (SDVRP). This problem is a relaxation of the Capacitated Vehicle Routing Problem (CVRP) since the customers' demands are allowed to be split. We deal with the cases where the fleet is unlimited (SDVRP-UF) and limited (SDVRP-LF). In order to solve them, we implemented a multi-start Iterated Local Search (ILS) based heuristic that includes a novel perturbation mechanism. Extensive computational experiments were carried out on benchmark instances available in the literature. The results obtained are highly competitive, more precisely, 55 best known solutions were equaled and new improved solutions were found for 243 out of 324 instances, with an average and maximum improvement of 1.15% and 2.81%, respectively.

1. Introduction

The Split Delivery Vehicle Routing Problem (SDVRP) is a relaxation of the well-known Capacitated Vehicle Routing Problem (CVRP) where the customers' demands are allowed to be split. The SDVRP can be formally defined as follows. Let $G=(V,E)$ be an undirected graph, where $V=\{0,\ldots,n\}$ is the set of vertices and $E=\{\{i,j\}: i,j\in V, i<j\}$ is the set of edges. Vertex 0 is the depot where a fleet of K identical vehicles with capacity Q are placed, while other vertices, $V'=\{1,\ldots,n\}$, represent customers, each one with a positive demand d_i ($i\in V; d_0=0$). A non-negative travel cost c_{ij} is associated with every edge $\{i,j\}\in E$. The objective is to minimize the sum of the travel costs in such a way that (i) the vehicle capacity is not exceeded; (ii) the demand $d_i, i\in V'$, which can be less than, equal, or greater than Q, is either delivered entirely by a single vehicle or split among different vehicles (therefore customers are allowed to be visited more than once); (iii) every route starts and ends at the depot.

A number of real-life applications of the SDVRP are available in the literature. Sierksma and Tijssen [32] described a practical case of specifying a flight schedule for helicopters to off-shore platforms, located in the Dutch continental shelf of the North Sea, for transporting workers to these platforms. Song et al. [34] dealt with the problem of determining an optimal assignment of newspaper agents as well as optimal routes and schedules for newspaper delivery in South Korea. Ambrosino and Sciomachen [4] studied the food distribution problem of an Italian company by considering a split delivery scheme. Belfiore and Yoshizaki [12] presented a case study in a large retail market in Brazil where the authors were faced with the SDVRP with additional features such as time windows, heterogeneous fleet and site dependence. All these practical problems were solved using heuristic approaches.

In spite of being a relaxation of the CVRP, the SDVRP was proven to be $\mathcal{NP}$-hard by Dror and Trudeau [21]. Archetti et al. [7] showed that the SDVRP can be solved in polynomial time when the capacity of the vehicle is 2. Furthermore, there are two main versions of the SDVRP studied in the literature: the one with limited fleet (SDVRP-LF) and the one with unlimited fleet (SDVRP-UF). With respect to the first, the minimum possible number of vehicles ($K_{min}=\lceil\sum_{i=1}^{n} d_i/Q\rceil$) must be used. It can be verified that there always exists a feasible solution when using $K_{\min}$ vehicles [21]. Archetti et al. [6] studied the complexity of these versions when the graph has special structures, namely line, tree, star and circle. Regarding the SDVRP-LF, the authors showed that the problem has complexity $\mathcal{O}(n^2)$ when the graph is a circle, $\mathcal{O}(n)$ when the graph is a line and $\mathcal{NP}$-hard for the remaining cases. Moreover, they also proved that the SDVRP-UF is $\mathcal{NP}$-hard for a tree, $\mathcal{O}(n^2)$ for a circle and $\mathcal{O}(n)$ for the other graphs.

The objective of this work is to present an Iterated Local Search (ILS) based heuristic for both the SDVRP-LF and SDVRP-UF. This metaheuristic proved to be highly efficient when applied to a variety of important routing problems as can be seen in the recent

* Corresponding author. Tel.: +55 21 2629 5665; fax: +55 21 2629 5666.
E-mail addresses: mmsilva@ic.uff.br (M.M. Silva), anand@ct.ufpb.br (A. Subramanian), satoru@ic.uff.br (L.S. Ochi).

http://dx.doi.org/10.1016/j.cor.2014.08.005
0305-0548/

works of Subramanian et al. [36], Penna et al. [30], Silva et al. [33] and Subramanian and Battarra [35]. The proposed algorithm differs from these previous works by incorporating efficient procedures especially devoted to the SDVRP such as (i) specific local search operators to enhance the intensification phase; (ii) a new perturbation scheme to improve the diversification phase; and (iii) auxiliary procedures that are useful in several parts of the algorithm such as when repairing a given solution or when trying to empty a route. These three modifications together play a remarkable role in the performance of the algorithm, resulting in a number of new improved solutions. Furthermore, when compared to other heuristic approaches presented in the literature, the algorithm developed here relies on very few parameters, thus requiring less tuning efforts. Extensive computational experiments were carried out on benchmark instances available in the literature. The results obtained are highly competitive, more precisely, 55 best known solutions were equaled and new improved solutions were found for 243 out of 324 instances, with an average and maximum improvement of 1.15% and 2.81%, respectively.

The remainder of the paper is organized as follows. Section 2 presents some related works. Section 3 describes the proposed ILS algorithm. Section 4 contains the computational results. Finally, Section 5 concludes the work.

2. Related work

The SDVRP was introduced by Dror and Trudeau [20,21] where the authors demonstrated empirically that split deliveries can generate savings both in terms of number of vehicles and travel costs. This claim was verified through computational experiments involving a set of 540 instances with up to 150 customers.

Some exact approaches for the SDVRP were suggested in the literature. Dror et al. [19] presented a mixed integer programming (MIP) formulation for the SDVRP-UF along with new classes of valid inequalities that were incorporated in a cutting-plane algorithm. Belenguer et al. [11] performed a polyhedral study on the SDVRP-LF and, as a result, several classes of valid inequalities were developed and a cutting-plane based algorithm was implemented. Jin et al. [26] proposed a two-phase exact algorithm for the SDVRP-LF, while Jin et al. [27] developed a column generation algorithm for the same variant. Moreno et al. [28] put forward a cut-and-price based approach over an extended formulation together with new valid inequalities for the SDVRP-LF. More recently, Archetti et al. [5] devised a branch-cut-and-price algorithm for the SDVRP-LF and SDVRP-UF.

In spite of the efforts in developing exact solution approaches for the SDVRP, such methods are still only capable of solving instances with up to 30 customers in a systematic fashion. This number is limited when compared to the CVRP where there are exact algorithms capable of systematically solving instances with up to 135 customers. Heuristic algorithms, however, appear to be more suitable for dealing with medium-large SDVRP instances.

Archetti et al. [9] proposed a three-phase Tabu Search (TS) heuristic for the SDVRP-UF. Chen et al. [15] developed a hybrid algorithm for the SDVRP-UF by combining a savings based heuristic with a mixed integer programming formulation and a record-to-record procedure. Campos et al. [14] proposed a Scatter Search based algorithm for the SDVRP-LF with two different procedures for generating initial populations. Boudia et al. [13] developed a Memetic Algorithm with population management for the SDVRP-UF and three new neighborhood structures especially devoted to the SDVRP. Archetti et al. [10] put forward a hybrid approach that combines a TS heuristic with an integer programming formulation. Aleman et al. [2,3] presented constructive and local search procedures for the SDVRP-LF, whereas Aleman and Hill [1] implemented a TS heuristic with vocabulary building for the SDVRP-UF. Derigs et al. [17] presented a study where they compared the behavior of several local search based metaheuristics for the SDVRP-UF. More recently, Wilck IV and Cavalier [37] developed a two-phase constructive procedure for the SDVRP-LF. They also proposed a Genetic Algorithm [38] for the same variant.

A detailed and comprehensive survey of SDVRPs can be found in the recent work of Archetti and Speranza [8].

3. Description of the iterated local search heuristic

The proposed algorithm, called SplitILS, consists of a multi-start heuristic mainly based on ILS. Initial solutions are generated using an extension of the cheapest insertion procedure, while local search is performed using Randomized Variable Neighborhood Descent (RVND) [25,36]. Every time a solution gets trapped in a local optimum, i.e., none of the local search operators is capable of improving it, a perturbation mechanism is applied over this solution.

Algorithm 1 presents the pseudocode of SplitILS. The algorithm executes I_{Max} iterations (lines 3–18) and at each of them an initial solution is generated using an insertion-based constructive procedure (line 4). The initial solution is improved by applying a RVND procedure in the local search phase (lines 7–15). The solutions initially generated always contain the minimum number of vehicles (K_{min}). However, during the local search, the number of routes can increase due to a neighborhood operator called *RouteAddition* (see Section 3.3.2) designed to bring together parts of customers' demands that were previously split. The number of routes associated to the solution returned at the end of the local search may need to be adjusted or not depending on the version of the problem, i.e., SDVRP-LF or SDVRP-UF. If the fleet is limited (SDVRP-LF) and the current solution contains more routes than the minimum number of vehicles, the least-loaded routes are emptied using a procedure called *EmptyRoutes* (see Section 3.1) (lines 9–10). Otherwise (SDVRP-UF), the solution is not modified. If an improved solution is found, *iterILS* is reinitialized (lines 11–13). The best current solution is then perturbed (line 14) using the perturbation mechanism described in Section 3.4.

Algorithm 1. SplitILS.

```
1  Procedure SplitILS(I_Max, I_ILS)
2  f* ← ∞;
3  for i ← 1, ..., I_Max do
4  |  s ← Construction();
5  |  s' ← s;
6  |  iterILS ← 0;
7  |  while iterILS < I_ILS do
8  |  |  s ← RVND(s);
9  |  |  if SDVRP − LF then
10 |  |  |s ← EmptyRoutes(s);
11 |  |  if f(s) < f(s') then
12 |  |  |  s' ← s;
13 |  |  |  iterILS ← 0;
14 |  |  s ← Perturb(s');
15 |  |  iterILS ← iterILS + 1;
16 |  |
17 |  if f(s') < f* then
18 |  |  s* ← s';
   |  |  f* ← f(s');
19 return s*;
```

3.1. Auxiliary procedures

In this section we describe two important auxiliary procedures that are used in other parts of the algorithm. The first one is called

SplitReinsertion, which is based on a generalization suggested by Boudia et al. [13] of the *k-Split* neighborhood structure proposed by Dror and Trudeau [20]. Algorithm 2 presents the pseudocode of the *SplitReinsertion* procedure. The input data of this procedure are (i) a solution s; (ii) a customer i; (iii) a demand $d_i' \leq d_i$; and (iv) a given forbidden route $\bar{r}$. The latter will be useful when emptying a route or when the demand of i is completely met by a single vehicle. Initially, each route $r \neq \bar{r}$, whose residual capacity a_r is greater than zero, is added to a list L (lines 5–10). If the sum of the residual capacities is not enough to meet the demand of i the procedure is aborted. Otherwise, for each route $r \in L$, the least additional cost u_r of inserting customer i in route r is computed (lines 12–14). The problem of deciding the combination that minimizes the sum of the insertion costs in such a way that the demand of i is fully met can be seen as a knapsack problem. Although there are highly efficient dynamic programming based algorithms to solve it, for the sake of computing time we decided to apply a simple and well-known greedy heuristic, in which insertions are performed in increasing order of u_r/a_r (line 15). Boudia et al. [13] pointed out that use of dynamic programming increases the running time by 30% without significantly improving the solution found by this greedy heuristic.

Algorithm 2. SplitReinsertion.

```
1  Procedure SplitReinsertion(s, i, d_i', r̄)
2  L ← A ← U ← ∅;
3  Let K_s be the number of vehicles in s;
4  sumResidual ← 0;
5  for r ← 1, ..., K_s do
6  |  Let a_r be the residual capacity in route r;
7  |  if r ≠ r̄ and a_r > 0 then
8  |  |  L ← L ∪ {r};
9  |  |  A ← A ∪ {a_r};
10 |  |  sumResidual ← sumResidual + a_r;
11 if sumResidual ≥ d_i' then
12 |  for r ← 1, ..., |L|
13 |  |  Let u_r be the least additional cost of inserting customer i in r;
14 |  |  U ← U ∪ {u_r};
15 |  s ← knapGreedyHeuristic(s, i, d_i', L, A, U);
16 return s;
```

The second auxiliary procedure, called *EmptyRoutes*, aims at reducing the number of routes of a solution. Algorithm 3 shows the pseudocode of *EmptyRoutes*. While the current number of vehicles of a solution s, that is K, is larger than K_{min}, the procedure tries to empty the least-loaded route $\bar{r}$ in s by applying the *SplitReinseration* method on every customer of $\bar{r}$.

Algorithm 3. EmptyRoutes.

```
1  Procedure EmptyRoutes(s)
2  while K > K_min do
3  |  r̄ ← least − loaded route in s;
4  |  while r̄ ≠ ∅ do
5  |  |  i ← first customer of r̄;
6  |  |  s ← SplitReinsertion(s, i, d_i', r̄);
7  |  |  r̄ ← r̄ − {i};
8  return s;
```

3.2. Constructive procedure

Initial solutions are generated using a simple construction procedure that is based on two strategies, namely the sequential insertion strategy and the parallel insertion strategy; and on two insertion criteria, namely the nearest feasible insertion criterion and the modified cheapest feasible insertion criterion. We refer to Penna et al. [30] for a detailed explanation on these strategies and criteria.

The constructive procedure works as follows. Firstly, $K_{\min} = \lceil \sum_{i=1}^{n} d_i/Q \rceil$ empty routes are considered. Next, every route is filled with a seed customer that is selected at random. It is important to mention that each route must contain a different customer. After that the insertion strategy and the insertion criterion are selected at random. The unrouted customers are then iteratively inserted to the partial solution. If it comes to a point where it is no longer possible to perform a feasible insertion due to a possible lack of available residual capacity of the current fleet, an extra vehicle is added and the procedure continues until all customers are inserted. Up to this stage, splits are not admitted. If the solution initially generated contains more than $K_{\min}$ vehicles, then the *EmptyRoutes* procedure is called with a view of obtaining a solution with exactly $K_{\min}$ vehicles.

3.3. Local search

As already mentioned, the local search is performed by a procedure based on RVND that is composed of CVRP and SDVRP inter-route neighborhood structures (see Sections 3.3.1 and 3.3.2). One of the advantages of this randomized approach is that one does not need to specify a neighborhood ordering, thus avoiding the use of an extra parameter. In addition, it has been empirically shown that RVND is capable of producing better results compared to VND when applied under an ILS scheme [30].

It is important to mention that, in case of improvement due to an inter-route move, the algorithm checks whether the modified routes contain customers that are visited twice by the same vehicle. If so, a repair operator is called in order to delete the copy whose removal yields the largest cost reduction. In this case, the partial demand carried by the deleted copy is aggregated to the one that remained in the route. Also, an intra-route local search is performed, using classical Traveling Salesman Problem (TSP) neighborhood operators, on the routes that have been modified. These operators are *2-opt* [16], *Reinsertion*, *Or-opt* [29] and *Exchange*.

3.3.1. CVRP inter-route neighborhoods

Six well-known CVRP inter-route neighborhood structures were implemented and their description is given as follows:

- *Shift* (1,0): One customer i is moved from route r_1 to r_2.
- *Swap* (1,1): Interchange between customer i from route r_1 and customer j from route r_2.
- *Shift* (2,0): Two adjacent customers i and j (or an arc (i,j)) are moved from route r_1 to route r_2. The transfer of the opposite arc (j,i) is also considered.
- *Swap* (2,1): Interchange between adjacent customers i and j from route r_1 and customer k from route r_2. As in the previous case, the opposite arc (j, i) is also considered.
- *Swap* (2,2): Interchange between adjacent customers i and j from route r_1 and two other adjacent customers k and l, from route r_2. The opposite arcs (j, i) and (l, k) are also considered, thus yielding 4 possible combinations.
- *Cross*: The arc between adjacent customers i and j from route r_1 and the arc between the adjacent customers k and l from route r_2 are removed. Next, two arcs (i, l) and (k, j) are inserted.

3.3.2. SDVRP inter-route neighborhoods

The following four SDVRP inter-route neighborhood structures were considered:

- *Swap* (1,1)*: Introduced by Boudia et al. [13], this neighborhood structure extends the *Swap(1,1)* by modifying the amounts to be delivered to the customers involved in the move. Given the customers $i \in r_1$ and $j \in r_2$, two cases are considered: (i) if $d_i > d_j$, then j is inserted before or after i, with $d'_i = d_i - d_j$, in route r_1 (the cheapest option) and j is replaced with a copy of i with $d'_i = d_j$ in r_2; (ii) if $d_j > d_i$, then i is inserted before or after j, with $d'_j = d_j - d_i$, in route r_2 (the cheapest option) and i is replaced with a copy of j with $d'_j = d_i$ in route r_1. The case where $d_i = d_j$ corresponds to a particular case of *Swap* (1,1) and it is not considered in this neighborhood structure. Fig. 1 shows an example where customer 8 is interchanged with customer 5. In this situation, $d_8 > d_5$, which corresponds to case (i).
- *Swap* (2,1)*: Also proposed by Boudia et al. [13], it consists of an extension of Swap(2,1) by employing a scheme similar to the previous neighborhood. Given the adjacent customers $i \in r_1$ and $j \in r_1$ and $k \in r_2$, we also consider the following two cases: (i) if $(d_i + d_j) > d_k$ and $(d_i < d_k)$, then i and a copy of j with $d'_j = (d_k - d_i)$ replace k in r_2 and k is inserted before or after j with $d'_j = d_j - (d_k - d_i)$ in r_1 (the cheapest option); (ii) if $d_k > (d_i + d_j)$, then i and j are inserted before or after k, with $d'_k = d_k - (d_i + d_j)$, in route r_2 (the cheapest option) and i and j are replaced with a copy of k with $d'_k = (d_i + d_j)$ in r_1. The case where $(d_i + d_j) = d_k$ corresponds to a particular case of *Swap* (2,1) and it is not considered. Fig. 2 shows an example where customer 8 is interchanged with customers 5 and 1. In this situation, $d_8 > (d_5 + d_1)$, which corresponds to case (ii).
- *RouteAddition*: Assume that customer i appears in at least two distinct routes r_1 and r_2. Firstly, the split is removed from i and a new empty route is generated. Secondly, take the four main route segments of r_1 and r_2, i.e., from the depot up to customer i and from the customer after i up to the depot (in both routes). Finally, choose the best form of combining these four route segments together with customer i in such a way that three new routes are created. If i is presented in more than two routes, all possible combinations involving two routes are considered. This neighborhood structure was suggested by Dror and Trudeau [20], but the authors also consider combinations involving up to three routes.
- *k-Split*: A customer i is removed from the current solution s and the *SplitReinsertion* procedure is called in order to generate a new solution s'. If there is only a single occurrence of i in s, the route $\bar{r}$ associated with i is considered to be forbidden for reinsertion.

The above neighborhood structures are exhaustively examined and the best feasible improvement strategy is considered. We also adopt auxiliary data structures in the spirit of Penna et al. [30] to enhance the performance of the local search. In spite of the large number of neighborhoods embedded in the local search, preliminary experiments revealed that each of them plays an important role depending on the characteristics of the instance. For example, when the value of the average demand is small, the classical VRP structures appear to be the most effective since the occurrence of splits is seldom observed in these cases.

3.4. Perturbation mechanism

Local optimal solutions are perturbed using a novel and simple mechanism, called *Multiple-k-Split*, whose description can be found in Algorithm 4. Firstly, a list composed of 5, 6 or 7 customers selected at random is built (line 2). Next, these customers are removed from the solution. Finally, the *SplitReinsertion* (see Section 3.1) is applied for every customer in the list.

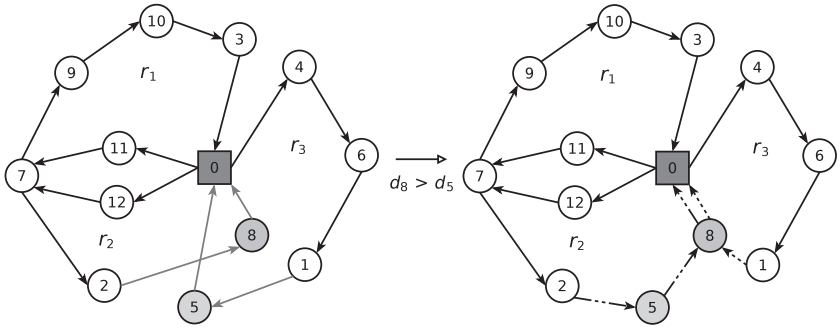

Fig. 1. *Swap*(1,1)*.

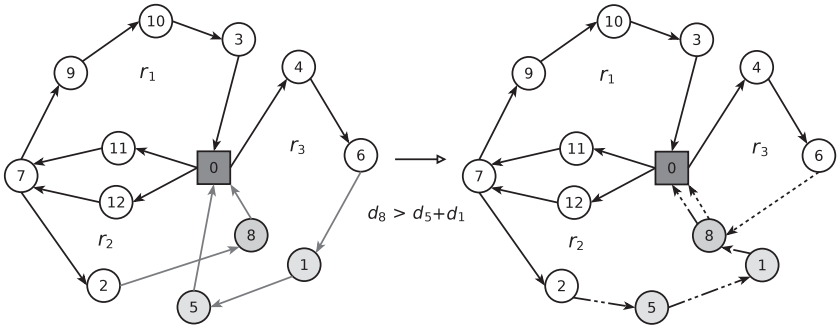

Fig. 2. *Swap*(2,1)*.

Algorithm 4. Multiple-*k*-Split.

```
1 procedure Multiple-k-Split(s)
2 Initialize the Customer List CL choosing k ∈ {5, 6, 7}
  customers at random;
3 for i ← 1, ..., k do
4 | Remove customer CL[i] from s;
5 for i ← 1, ..., k do
6 | c ← CL[i];
7 | s ← SplitReinsertion(s, c, d'_c, ∅);
8 return s;
```

It is worth mentioning that other CVRP based perturbation mechanisms, such as performing multiple *Swap*(1,1) moves or applying the *Ejection Chain* operator [23], were implemented, but the results found were not as good as those obtained when using *Multiple-k-Split*. This can be explained by the fact that those perturbations do not exploit the split characteristic of the problem and therefore they become less effective in terms of diversification. Moreover, we also tried applying several *Swap*(1,1)* moves as an alternative perturbation approach. Computational experiments, however, suggested that *Multiple-k-Split* is still more effective in generating diversified solutions.

4. Computational results

The algorithm was coded in C++ (g++ 4.4.3) and executed on an *Intel*® Core™ i7 2.93 GHz, with 8.0 GB of RAM memory running under GNU/Linux Ubuntu 10.04 (*kernel* 2.6.32-25). Only a single thread was used and the algorithm was executed 20 times for each instance.

In the tables presented hereafter, **Problem** indicates the name of the instance, **LB** is the lower bound (LB), **BKS** is the best known solution (BKS), **Best Sol.** is the best solution found by the work identified in the column header, **Avg. Sol** is the average solution obtained by SplitILS, **Avg. Gap** is the gap between the average solution found by SplitILS and the BKS (Avg. Gap = (100(Avg. Sol − BKS)/BKS)), **Avg. Gap LB** is the gap between the average solution obtained by SplitILS and the LB (Avg. Gap = (100(Avg. Sol − LB)/LB)). **Avg. Time** is the average time in seconds of the 20 executions. **Avg. Time BKS** is the average time spent by SplitILS to find or improve the BKS, **Best All Exp.** is the best solution found by SplitILS considering all experiments, **cTime** indicates the scaled time of each computer [18], with respect to our 2.93 GHz. The best solutions are highlighted in boldface and the solutions improved by SplitILS are underlined.

The proposed algorithm was tested on four set of instances. The first one was generated by Belenguer et al. [11] and it is composed of 25 test-problems, where 11 were adapted from the TSPLIB [31] and other 14 were generated at random as follows. The coordinates of the customers are those of the instances *eil51, eil76* and *eil101* from the TSPLIB. The capacity of the vehicle is $Q = 160$ and the customers demands were generated according to Dror and Trudeau [20]. In the 14 instances randomly generated, D1 indicates that the demand d_i of customer i was selected at random from the interval $[\lceil 0.01Q \rceil, \lfloor 0.1Q \rfloor]$, D2 $[\lceil 0.1Q \rceil, \lfloor 0.3Q \rfloor]$, D3 $[\lceil 0.1Q \rceil, \lfloor 0.5Q \rfloor]$, D4 $[\lceil 0.1Q \rceil, \lfloor 0.9Q \rfloor]$, D5 $[\lceil 0.3Q \rceil, \lfloor 0.7Q \rfloor]$ and D6 $[\lceil 0.7Q \rceil, \lfloor 0.9Q \rfloor]$.

A set of instances was generated by Archetti et al. [9] by modifying the demands (using the intervals described above) of the VRP test-problems 1–5, 11, and 12 described in Gendreau et al. [22]. Unfortunately, this set is no longer available. Therefore, the second set of instances that we actually considered was generated by Campos et al. [14] following the same procedure employed by Archetti et al. [9]. The set is composed of 49 test-problems with up to 199 customers and it is divided into seven groups. The first one considers the original demands and the other six take into account the intervals D1–D6, respectively. This is the same set employed by Boudia et al. [13], Aleman et al. [2,3], and Aleman and Hill [1].

The third set of instances was proposed by Archetti et al. [10] also following the approach suggested by Archetti et al. [9]. The characteristics of the instances of this set are similar to those generated by Campos et al. [14], but the demand values are different.

The fourth set of instances was suggested by Chen et al. [15]. This set is composed of 21 test-problems varying from 8 to 288 customers. The vehicle capacity in all cases is $Q = 100$ and the demands are set to either 60 or 90. In addition, the customers are concentrically distributed around the depot. The authors state that this allows for estimating near-optimal solutions visually.

4.1. Parameter tuning

The proposed algorithm relies mainly on three parameters, namely the number of restarts (I_{max}), the number of ILS iterations (I_{ILS}) and the size of the customer list of the perturbation mechanism (k).

Parameter k was the first one to be tuned. This parameter controls the level of diversification of a solution during the perturbation phase. It can be noted that the value of k directly affects the solution quality as well as the computing time: for small values of k, the algorithm usually faces difficulties in escaping from local optimal solutions and the local search phase becomes less time consuming; for large values of k, more moves are likely to be performed during the local search phase, thus increasing the possibility of finding new improved solutions at a cost of increased computing time.

In order to calibrate the value of k, we arbitrarily assigned values for I_{max} and I_{ILS}. Since we were interested in examining the

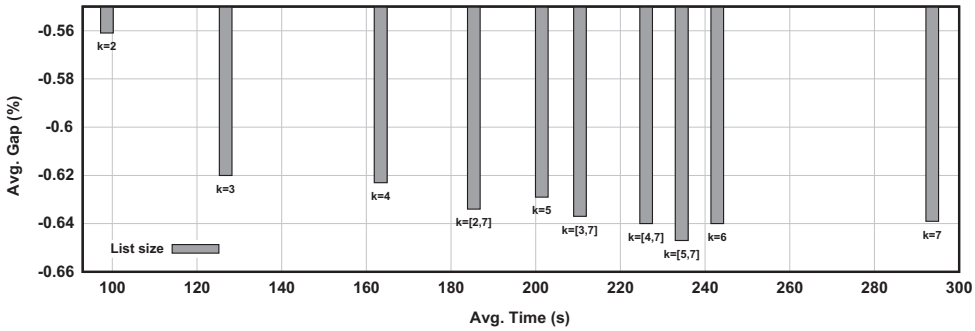

Fig. 3. Results for different values of k for the *Multiple-k-Split* perturbation.

Table 1
Experiments considering different values for I_{ILS}.

Benchmark dataset	$I_{ILS} = (K_{\min} + n)$		$I_{ILS} = 10 \times (K_{\min} + n)$		$I_{ILS} = \min\{K_{\min} \times n, 5000\}$	
	Avg. gap (%)	Avg. time (s)	Avg. gap (%)	Avg. time (s)	Avg. gap (%)	Avg. time (s)
Belenguer et al. [11][a,b]	−1.60	5.20	−1.84	36.31	−1.87	63.77
Belenguer et al. [11][a]	−1.24	8.06	−1.43	52.04	−1.46	75.97
Archetti et al. [10][a]	−2.52	124.45	−2.77	697.17	−2.81	1159.19
Campos et al. [14][a]	−1.98	95.19	−2.21	551.71	−2.23	925.59
Chen et al. [15][a]	0.05	58.10	−0.01	376.65	−0.02	504.44
Belenguer et al. [11][c,b]	−0.04	4.79	−0.30	33.71	−0.32	56.87
Belenguer et al. [11][c]	−0.93	6.78	−1.12	44.50	−1.13	74.51
Archetti et al. [10][c]	−0.92	106.69	−1.20	629.63	−1.22	1062.86
Campos et al. [14][c]	−0.17	84.53	−0.38	518.86	−0.41	872.02
Chen et al. [15][c]	0.06	65.49	−0.02	436.19	−0.03	591.62

[a] SDVRP-LF.
[b] Rounded costs.
[c] SDVRP-UF.

performance of the interaction between local search and perturbation phases, it was not necessary to perform a large number of restarts of the method, but at the same time we believed that using a considerable number of ILS iterations seemed to be more appropriate in this context. Therefore, we set $I_{\max} = 10$ and $I_{ILS} = [\mu \times (K_{min} + n)]$, where μ is the number of inter-route neighborhoods plus the number of intra-route neighborhoods. We experimented various settings for k, more precisely, $\{2, 3, 4, 5, 6, 7, [2, 7], [3, 7], [4, 7], [5, 7]\}$. It should be noted that when k was set as a discrete interval (e.g. [2,7]), its value is selected at random from the possible values of the interval. For each configuration, we ran the algorithm 20 times in a subset composed of 30 instances with different characteristics, chosen from the four sets previously described. The results of the tests are shown in Fig. 3, where it can be verified that the algorithm performs better when using the interval $[5, 7]$, as already specified in Section 3.4.

The value of the parameter I_{ILS} was defined as a function of n and $K_{\min}$. Three different functions were tried and the algorithm was executed 20 times for all instances of the four sets considered in this work, as shown in Table 1. It can be clearly observed that the larger the I_{ILS}, smaller the average gap and larger the computing time. These experiments were ran considering $I_{\max} = 10$. Moreover, we also partially applied the same tests for $I_{\max} = 20$ and $I_{\max} = 30$ to investigate if more restarts were necessary for substantially improving the average gaps, which ended up not happening. It was observed that $I_{\max} = 10$ and $I_{ILS} = \min\{K_{\min} \times n, 5000\}$ seem to provide a good compromise between computing time and solution quality, always leading to negative average gaps for all sets of instances.

4.2. Impact of the new perturbation mechanism and the use of VRP and SDVRP neighborhoods

We conducted a series of experiments in order to show the importance of the new perturbation mechanism and the relevance of using both VRP and SDVRP based neighborhood structures in our proposed algorithm. We compared SplitILS with other three alternate versions, namely SplitILS2, SplitILS3 and SplitILS4, whose characteristics are specified in Table 2.

It is important to mention that the VRP based perturbation *MultipleSwap*(1,1), used in SplitILS2, is the same as the one utilized in Penna et al. [30] and it consists of applying multiple *Swap*(1,1) moves consecutively. Note that the application of this perturbation may possibly lead to infeasible solutions. Such solutions are repaired by applying the following procedure to each unfeasible route. At first, customers are removed from the route at random until the vehicle capacity is no longer violated. Next, these customers are split among the routes whose residual capacity are greater than zero using a procedure based on *SplitReinsertion*.

Table 2
Different versions of SplitILS.

Method	Perturbation		Neighborhoods	
	Multiple-k-Split	*Multiple-Swap*(1,1)	VRP based	SDVRP based
SplitILS	✓		✓	✓
SplitILS2		✓	✓	✓
SplitILS3	✓		✓	
SplitILS4	✓			✓

We executed each of the four versions 20 times in all instances of the four sets. For a fair comparison, we imposed the average time spent by SplitILS in each instance as stopping criterion for the other three versions. Table 3 presents a comparison among the methods where we can verify that SplitILS outperformed the other methods in all set of instances, thus illustrating the effectiveness of combining the *Multiple-k-Split* perturbation with VRP and SDVRP based neighborhoods. Nonetheless, the other three methods still yielded high quality solutions, especially when the instances favored the respective method. However, their overall performance were not as robust as the one of SplitILS, which in turn had a very consistent behavior regardless of the characteristics of the instances. Moreover, the three best versions (SplitILS, SplitILS3 and SplitILS4) used the *Multiple-k-Split* perturbation.

Figs. 4 and 5 depict a comparison among the four methods when varying the average demand of the customers. It is worth mentioning that this analysis was only performed in the instances of Archetti et al. [10] and Campos et al. [14], since they were the only authors that divided their instances into distinct groups based on customers' demands. It can be verified that SplitILS outperforms the other versions even in almost all VRP-like instances, i.e., those in which the average demand of the customers is smaller than 50% of the vehicle capacity (D1, D2, D3). SplitILS performed slightly worse only in the group D1 of Campos et al. [14] for the SDVRP-UF version, as can be seen in Fig. 5(b), where the average demand of the customers is smaller than 10%. These results confirm those presented by Dror and Trudeau [20], where the authors concluded that there is no clear advantage in performing split when the average demand of the customers is smaller than 10%.

Table 3
Average gaps obtained by different versions of SplitILS.

Benchmark dataset	SplitILS Avg. gap (%)	SplitILS2 Avg. gap (%)	SplitILS3 Avg. gap (%)	SplitILS4 Avg. gap (%)	Avg. time (s)
Belenguer et al. [11][a,b]	**−1.87**	−1.41	−1.68	−1.45	63.77
Belenguer et al. [11][a]	**−1.46**	−1.11	−1.25	−1.17	75.97
Archetti et al. [10][a]	**−2.81**	−1.74	−2.52	−2.47	1159.19
Campos et al. [14][a]	**−2.23**	−1.39	−2.04	−1.78	925.59
Chen et al. [15][a]	**−0.02**	0.26	0.00	0.14	504.44
Belenguer et al. [11][b,c]	**−0.32**	0.14	−0.10	0.14	56.87
Belenguer et al. [11][c]	**−1.13**	−0.81	−0.92	−0.85	74.51
Archetti et al. [10][c]	**−1.22**	−0.25	−0.94	−0.80	1062.86
Campos et al. [14][c]	**−0.41**	0.42	−0.21	0.07	872.02
Chen et al. [15][c]	**−0.03**	0.27	−0.02	0.16	591.62

[a] SDVRP-LF.
[b] Rounded costs.
[c] SDVRP-UF.

4.3. Results for the SDVRP-LF

Table 4 presents the results obtained on the set of instances of Belenguer et al. [11]. It can be verified that SplitILS equaled three BKSs and it was found capable of improving the other 22. The average gap between the average solutions obtained by SplitILS and the BKSs was −1.46%. Similar results were found for the same set of instances but with rounded costs, as can be observed in Table 5. In this case, seven solutions were equaled, 18 were improved and the average gap was −1.87%. These results shows that SplitILS clearly outperforms the algorithm of Aleman et al. [2] in terms of solution quality. Furthermore, despite spending a larger total computing time when compared to the algorithm of Aleman et al. [2], SplitILS consumed, on average, only 1.69 s to find or improve the BKSs.

Table 6 shows the results found on the set of instances of Campos et al. [14]. The SplitILS algorithm performed very well on this benchmark dataset since the BKSs of 44 out of the 49 instances were improved and the five remaining ones were equaled. Comparing with the literature, the average gap of the proposed algorithm with respect to the BKSs is −2.23%. Moreover, it can be observed that SplitILS was capable, on average, of quickly finding or improving the BKSs when compared to the algorithms of Campos et al. [14] and Aleman et al. [2]. For this set of instances it is not possible to estimate how far the solutions found by SplitILS are from the optimal, since no LBs are currently available.

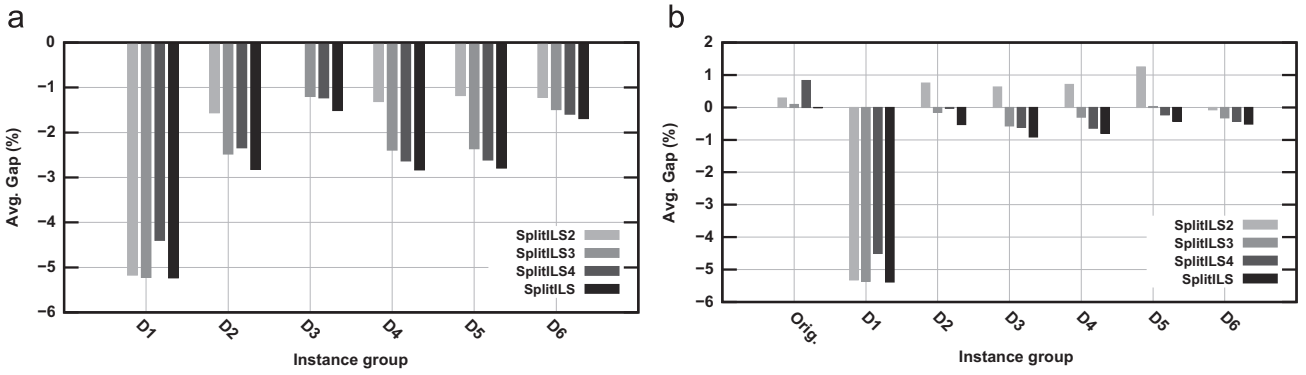

Fig. 4. Comparing the performance of different versions of SplitILS when varying the average demand of the customers on the instances of Archetti et al. [10]: (a) SDVRP-LF version. (b) SDVRP-UF version.

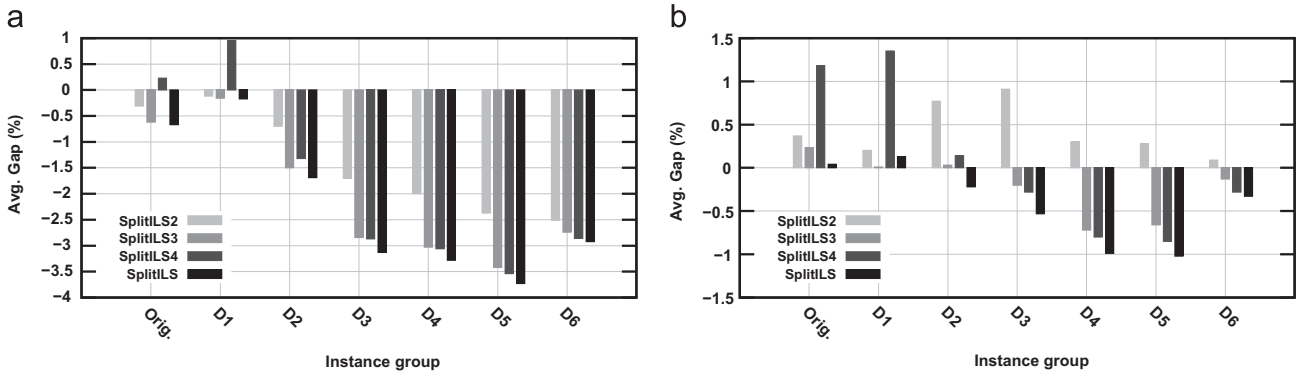

Fig. 5. Comparing the performance of different versions of SplitILS when varying the average demand of the customers on the instances of Campos et al. [14]. (a) SDVRP-LF version. (b) SDVRP-UF version.

Table 4
Results for the SDVRP-LF on the instances of Belenguer et al. [11].

Problem	LB	BKS	Aleman et al.		SplitILS						
			Best sol.	Time[a] (s)	Best sol.	Avg. sol.	Avg. gap LB (%)	Avg. gap (%)	Avg. time (s)	Avg. time BKS (s)	Best sol. exp.
eil22	–	375.28[b]	**375.28**	4.19	**375.28**	**375.28**	–	0.00	0.14	0.00	375.28
eil23	525.65	569.75[b]	**569.75**	3.42	**568.56**	**568.56**	8.16	−0.21	0.12	0.00	568.56
eil30	–	512.72[b]	**512.72**	14.47	**512.72**	**512.72**	–	0.00	0.32	0.00	512.72
eil33	–	853.10[b]	**853.10**	14.03	**837.06**	**837.06**	–	−1.88	0.45	0.00	837.06
eil51	518.26	524.61[b]	**524.61**	54.91	**524.61**	**524.61**	1.23	0.00	1.63	0.24	524.61
eilA76	809.67	829.58[c]	851.24	83.28	**823.89**	**825.22**	1.92	−0.53	27.25	3.19	823.89
eilB76	985.42	1023.23[c]	1059.57	79.00	**1009.04**	**1011.19**	2.62	−1.18	44.98	0.15	1009.04
eilC76	723.55	753.29[b]	**753.29**	148.20	**738.67**	**739.83**	2.25	−1.79	15.68	0.01	738.67
eilD76	672.54	699.35[b]	**699.35**	140.83	**687.60**	**688.37**	2.35	−1.57	9.92	0.03	687.60
eilA101	803.62	851.39[c]	852.74	319.33	**826.14**	**826.26**	2.82	−2.95	36.59	0.01	826.14
eilB101	1055.40	1118.93[c]	1139.27	185.84	**1076.26**	**1078.58**	2.20	−3.61	101.26	0.06	1076.26
S51D1	457.10	464.78[c]	471.92	40.53	**459.50**	**459.50**	0.53	−1.14	1.07	0.02	459.50
S51D2	700.40	719.82[c]	731.01	28.34	**708.42**	**709.54**	1.30	−1.43	9.98	0.15	708.42
S51D3	938.50	955.56[c]	1001.22	14.70	**948.01**	**949.96**	1.22	−0.59	14.15	0.34	947.97
S51D4	1549.70	1614.36[c]	1680.66	16.53	**1561.01**	**1563.25**	0.87	−3.17	59.96	0.01	1560.92
S51D5	1318.90	1343.07[c]	1389.40	13.94	**1333.67**	**1333.85**	1.13	−0.69	32.41	0.21	1333.67
S51D6	2154.70	2179.65[c]	2218.23	16.83	**2169.10**	**2174.71**	0.93	−0.23	83.79	30.90	2169.10
S76D1	592.60	606.47[b]	**606.47**	476.27	**598.94**	**598.98**	1.08	−1.24	4.54	0.25	598.94
S76D2	1071.30	1129.06[c]	1143.36	46.94	**1087.99**	**1089.69**	1.72	−3.49	74.51	0.08	1087.99
S76D3	1407.54	1442.08[c]	1490.08	53.34	**1427.81**	**1429.01**	1.53	−0.91	88.72	1.76	1427.81
S76D4	2059.80	2100.15[c]	2173.61	51.84	**2079.76**	**2080.76**	1.02	−0.92	173.55	2.69	2079.76
S101D1	716.80	749.19[b]	**749.19**	2125.58	**726.59**	**728.44**	1.62	−2.77	14.16	0.01	726.59
S101D2	1358.90	1417.87[c]	1443.44	217.91	**1383.35**	**1386.45**	2.03	−2.22	129.94	0.06	1378.19
S101D3	1853.10	1906.28[c]	1988.78	146.61	**1876.97**	**1881.26**	1.52	−1.31	277.62	1.84	1876.48
S101D5	2767.60	2873.08[d]	2984.48	104.05	**2792.01**	**2795.73**	1.02	−2.69	696.64	0.38	2789.54
Average				176.04			1.87	−1.46	75.97	1.69	

[a] Pentium 4, 2.8 GHz, Single run, Avg. cTime (s): 73.68.
[b] Aleman et al. [2].
[c] Archetti et al. [5].
[d] Wilck IV and Cavalier [38], Xeon 2.49 GHz, Single run.

Table 5
Results for the SDVRP-LF on the instances of Belenguer et al. [11] using rounded costs.

Problem	LB	BKS	Aleman et al.		SplitILS						
			Best Sol.	Time[a] (s)	Best Sol.	Avg. Sol.	Avg. Gap LB (%)	Avg. Gap (%)	Avg. Time (s)	Avg. Time BKS (s)	Best Sol. Exp.
eil22	375.00	375[b,c]	**375**	4.19	**375**	**375.00**	0.00	0.00	0.13	0.01	375
eil23	569.00	569[b,c]	570	3.42	**569**	**569.00**	0.00	0.00	0.09	0.00	569
eil30	510.00	510[b,c]	**510**	14.47	**510**	**510.00**	0.00	0.00	0.30	0.00	510
eil33	834.70	835[b,c]	851	14.03	**835**	**835.00**	0.04	0.00	0.39	0.03	835
eil51	517.80	521[b,c]	**521**	54.91	**521**	521.55	0.72	0.11	1.63	0.14	521
eilA76	807.60	832[c]	847	83.28	**818**	**820.45**	1.59	−1.39	25.68	0.35	818
eilB76	981.40	1023[c]	1055	79.00	**1002**	**1005.80**	2.49	−1.68	38.05	0.33	1002
eilC76	717.80	735[c]	746	148.20	**733**	**733.55**	2.19	−0.20	15.17	1.78	732
eilD76	666.10	683[c]	695	140.83	**681**	**683.00**	2.54	0.00	11.02	2.87	681
eilA101	799.80	817[c]	843	319.33	**815**	**815.85**	2.01	−0.14	32.70	8.57	814
eilB101	1040.60	1077[c]	1122	185.84	**1061**	**1065.40**	2.38	−1.08	75.43	8.34	1061
S51D1	454.40	458[c]	466	40.53	**458**	**458.00**	0.79	0.00	1.21	0.09	458
S51D2	694.20	725[d]	**725**	28.34	**703**	**704.65**	1.51	−2.81	8.32	0.06	703
S51D3	930.70	972[c]	994	14.70	**943**	**944.20**	1.45	−2.86	13.58	0.03	943
S51D4	1539.00	1672[d]	**1672**	16.53	**1552**	**1555.55**	1.08	−6.96	47.34	0.01	1551
S51D5	1313.40	1385[d]	**1385**	13.94	**1328**	**1329.15**	1.20	−4.03	33.46	0.02	1328
S51D6	2141.70	2211[d]	**2211**	16.83	**2163**	**2165.70**	1.12	−2.05	65.68	0.01	2162
S76D1	586.60	594[c]	600	476.27	**592**	**592.45**	1.00	−0.26	4.75	1.33	592
S76D2	1061.10	1138[d]	**1138**	46.94	**1081**	**1083.35**	2.10	−4.80	59.20	0.03	1080
S76D3	1395.90	1474[c]	1485	53.34	**1419**	**1422.05**	1.87	−3.52	8.07	0.03	1419
S76D4	2046.10	2160[d]	**2160**	51.84	**2071**	**2074.30**	1.38	−3.97	148.48	0.01	2071
S101D1	704.90	716[c]	740	2125.58	**716**	718.40	1.92	0.34	14.17	13.00	716
S101D2	1337.10	1393[c]	1426	217.91	**1364**	**1370.95**	2.53	−1.58	116.33	0.45	1364
S101D3	1832.20	1974[d]	1974	146.61	**1859**	**1868.75**	1.99	−5.33	233.36	0.01	1859
S101D5	2737.10	2915[c]	2970	104.05	**2772**	**2779.65**	1.55	−4.64	579.68	0.06	2772
Average				176.04			1.42	−1.87	63.77	1.50	

[a] Pentium 4, 2.8 GHz, Single run, Avg. cTime (s): 73.68.
[b] Optimal solution.
[c] Belenguer et al. [11].
[d] Aleman et al. [2].

The results obtained on the set of instances of Archetti et al. [10] are depicted in Table 7. It can be verified that SplitILS was capable of improving all BKS with an average gap of −2.81%. As in previous cases, the BKSs were quickly found or improved.

Finally, on the set of instances of Chen et al. [15], SplitILS equaled five results and improved another seven, with an average gap of −0.02%, as shown in Table 8. The average gap with respect to the LBs was only 0.53%, which indicates that the solutions found by our algorithm are, on average, of high quality. Nevertheless, this is the only SDVRP-LF benchmark dataset that SplitILS fails to either equal or improve the result of all instances. In the successful cases, SplitILS consumes a very small amount of time to either find or improve the BKS.

4.4. Results for the SDVRP-UF

In Table 9 one can observe the results found on the set of instances of Belenguer et al. [11]. In most cases, the solutions found are equivalent to those obtained for the SDVRP-LF. However,

Table 6
Results for the SDVRP-LF on the instances of Campos et al. [14].

Problem	BKS	Campos et al.		Aleman et al.		SplitILS					
		Best Sol.	Time[a] (s)	Best Sol.	Time[b] (s)	Best Sol.	Avg. Sol.	Avg. Gap (%)	Avg. Time (s)	Avg. Time BKS (s)	Best All Exp.
p01-50	524.61[c]	**524.61**	49.70	**524.61**	54.91	**524.61**	**524.61**	0.00	1.87	0.22	524.61
p02-75	829.01[c]	**829.01**	166.50	851.24	83.28	**823.89**	**824.77**	−0.51	30.84	6.72	823.89
p03-100	829.45[c]	**829.45**	276.10	852.74	319.33	**826.14**	**826.39**	−0.37	40.81	3.20	826.14
p04-150	1045.22[c]	**1045.22**	527.10	1074.11	1361.16	**1024.59**	**1026.60**	−1.78	251.66	1.06	1023.66
p05-199	1324.73[c]	**1324.73**	588.30	1368.67	3284.64	**1289.40**	**1296.37**	−2.14	1594.46	10.70	1286.08
p06-120	1042.12[c]	**1042.12**	270.30	1201.83	3414.41	**1037.88**	1043.41	0.12	90.06	20.49	1037.88
p07-100	819.56[c]	**819.56**	192.40	824.78	126.08	**819.56**	**819.56**	0.00	27.99	0.13	819.56
p01-50D1	460.79[c]	**460.79**	51.80	471.92	33.70	**460.79**	**460.79**	0.00	1.16	0.02	460.79
p02-75D1	596.99[c]	**596.99**	144.00	597.46	303.77	**596.25**	**596.25**	−0.12	5.00	0.61	596.25
p03-100D1	726.81[c]	**726.81**	272.10	745.35	2194.23	**726.81**	730.01	0.44	17.72	13.70	726.81
p04-150D1	871.26[c]	**871.26**	743.30	891.98	3461.44	**866.31**	**866.31**	−0.57	119.63	2.77	866.31
p05-199D1	1023.14[c]	**1023.14**	1874.80	1073.55	15505.22	**1017.30**	**1018.40**	−0.46	438.21	27.99	1017.28
p06-120D1	976.57[c]	**976.57**	370.90	1087.80	3952.67	**975.96**	**976.42**	−0.02	46.16	8.15	975.96
p07-100D1	636.00[c]	**636.00**	166.50	673.54	1207.42	**632.63**	**633.11**	−0.45	14.38	2.75	632.63
p01-50D2	741.06[c]	**741.06**	66.40	766.19	19.77	**741.06**	741.26	0.03	9.87	4.12	741.06
p02-75D2	1071.87[c]	**1071.87**	143.80	1099.47	73.05	**1064.49**	**1066.87**	−0.47	53.42	3.14	1064.49
p03-100D2	1397.50[c]	**1397.50**	305.10	1425.90	190.53	**1376.09**	**1380.28**	−1.23	182.16	1.85	1374.19
p04-150D2	1937.20[c]	**1937.20**	326.60	1978.01	878.55	**1861.63**	**1866.48**	−3.65	1055.54	0.29	1861.53
p05-199D2	2433.17[c]	**2433.17**	32.10	2464.65	1457.16	**2307.82**	**2313.37**	−4.92	2440.32	0.16	2306.80
p06-120D2	2742.60[c]	**2742.60**	380.80	2806.92	558.56	**2703.75**	**2708.51**	−1.24	762.81	6.43	2703.75
p07-100D2	1418.81[c]	**1418.81**	206.30	1428.27	123.00	**1413.85**	**1413.91**	−0.35	130.70	1.21	1413.85
p01-50D3	997.83[c]	**997.83**	87.10	1039.89	18.16	**982.77**	**983.70**	−1.42	18.44	0.40	982.77
p02-75D3	1463.60[c]	**1463.60**	126.80	1478.67	67.80	**1393.11**	**1393.11**	−4.82	101.77	0.04	1393.11
p03-100D3	1908.02[c]	**1908.02**	225.20	1956.13	154.47	**1823.17**	**1827.47**	−4.22	326.55	0.06	1822.36
p04-150D3	2649.97[c]	**2649.97**	21.30	2671.62	625.83	**2528.51**	**2531.79**	−4.46	1514.55	0.08	2525.51
p05-199D3	3291.96[c]	**3291.96**	31.20	3411.38	2173.84	**3153.01**	**3163.89**	−3.89	3895.07	0.32	3153.01
p06-120D3	3979.67[c]	**3979.67**	329.00	4026.53	358.56	**3907.27**	**3910.03**	−1.75	1543.98	26.71	3907.27
p07-100D3	1995.34[c]	**1995.34**	266.50	2007.11	107.47	**1967.41**	**1967.93**	−1.37	260.65	0.41	1967.41
p01-50D4	1522.43[d]	1554.38	92.60	**1522.43**	16.36	**1456.00**	**1456.87**	−4.31	46.74	0.01	1456.00
p02-75D4	2182.34[c]	**2182.34**	119.90	2200.51	71.11	**2081.38**	**2084.62**	−4.48	219.74	0.02	2081.37
p03-100D4	2865.86[d]	2894.21	177.90	**2865.86**	126.52	**2751.13**	**2754.52**	−3.89	629.59	0.03	2749.05
p04-150D4	4062.88[c]	**4062.88**	50.40	4165.18	671.36	**3988.06**	**3997.49**	−1.61	1986.49	2.41	3988.06
p05-199D4	5074.57[c]	**5074.57**	50.70	5184.57	3650.59	**4844.58**	**4855.82**	−4.31	3806.84	0.25	4842.97
p06-120D4	6357.33[c]	**6357.33**	20.60	6364.87	458.91	**6201.66**	**6215.87**	−2.23	1975.24	0.57	6196.48
p07-100D4	3156.31[d]	3166.31	272.70	**3156.31**	96.98	**3087.75**	**3088.96**	−2.13	435.09	0.04	3087.75
p01-50D5	1532.19[c]	**1532.19**	92.40	1540.39	15.33	**1467.47**	**1467.47**	−4.22	48.93	0.01	1467.47
p02-75D5	2228.90[c]	**2228.90**	11.10	2238.98	80.30	**2112.19**	**2113.38**	−5.18	267.72	0.02	2110.20
p03-100D5	2941.64[d]	2986.33	17.00	**2941.64**	103.94	**2813.82**	**2817.05**	−4.24	737.35	0.04	2813.35
p04-150D5	4165.18[d]	4185.68	23.00	**4165.18**	675.39	**3986.49**	**3996.85**	−4.04	2076.38	0.16	3986.49
p05-199D5	5265.01[c]	**5265.01**	327.30	5363.65	3026.22	**5061.25**	**5070.77**	−3.69	4570.46	0.33	5061.25
p06-120D5	6481.09[c]	**6481.09**	20.50	6545.50	469.17	**6372.58**	**6375.64**	−1.63	2289.79	1.93	6370.97
p07-100D5	3225.63[d]	3248.76	16.00	**3225.63**	110.05	**3125.47**	**3126.22**	−3.08	619.62	0.03	3125.47
p01-50D6	2215.34[d]	2312.48	5.80	**2215.34**	18.70	**2154.21**	**2154.51**	−2.75	83.85	0.01	2154.21
p02-75D6	3304.24[d]	3387.86	10.50	**3304.24**	58.05	**3179.20**	**3181.30**	−3.72	441.77	0.01	3179.08
p03-100D6	4429.21[d]	4576.13	38.30	**4429.21**	94.98	**4294.12**	**4298.50**	−2.95	731.49	0.03	4294.12
p04-150D6	6479.46[c]	**6479.46**	30.50	6482.11	584.84	**6231.01**	**6233.76**	−3.79	1660.06	0.09	6230.49
p05-199D6	8323.72[c]	**8323.72**	215.00	8329.55	2124.66	**8045.18**	**8047.68**	−3.32	4718.09	0.25	8041.01
p06-120D6	10,158.32[c]	**10,158.32**	20.40	10,302.16	636.72	**10,001.95**	**10,005.18**	−1.51	2209.90	0.08	10001.95
p07-100D6	5028.78[d]	5065.26	13.80	**5028.78**	178.19	**4902.81**	**4907.00**	−2.42	823.19	0.04	4902.37
Average			201.40		1130.15			−2.23	925.59	3.06	

[a] Pentium 4, 2.4 GHz, Single run, Avg. cTime (s): 65.91.
[b] Pentium 4, 2.8 GHz, Single run, Avg. cTime (s): 473.04.
[c] Campos et al. [14].
[d] Aleman et al. [2].

in some instances, such as eil30 and S101D5, there is a gain in terms of solution cost when an extra vehicle is considered. The same interpretation can be derived for the results found on the set of instances with rounded costs (see Table 10).

Table 11 shows the superiority of SplitILS, especially in terms of solution quality, on the set of instances of Campos et al. [14] when compared to the algorithms of Boudia et al. [13] and Aleman and Hill [1]. Our algorithm was capable of improving the BKS of 40 instances and equaling the other 9, with an average gap between the average solutions and the BKSs of −0.41%. Regarding the computational effort, SplitILS spent, on average, 63.55 s to obtain or improve the BKSs.

Table 12 illustrates the results found by SplitILS on the set of instances of Archetti et al. [10]. It can be observed that our algorithm only fails to find the BKS in the instance p11-120D5. Nevertheless, an improved solution was found when considering all our experiments. As for the remaining instances, SplitILS clearly outperforms, on average, the results of Derigs et al. [17]. The average gap between the average solutions of SplitILS and the BKSs was −1.22%. Equal or improved solutions were found in few seconds for most of the cases.

On the set of instances of Chen et al. [15], SplitILS equaled the result of five instances, while six improved solutions were found, as can be seen in Table 13. Despite failing to equal/improve the result of 10 instances, the average gap between the average solution and LB was smaller than 1%, which is a significant result in terms of solution quality. When comparing our results with those reported by Aleman and Hill [1], one can notice that SplitILS found a worse solution only on instance SD17. Although not explicitly depicted in Table 13, SplitILS actually spent, on average, 0.8 s to either find or improve the best solutions of Aleman and Hill [1], disregarding instances SD12 and SD17.

4.5. Summary of the results

The summary of the results obtained by SplitILS is presented in Table 14. In this table, **#Instances** stands for the number of instances of the given benchmark dataset, **#Improv.** denotes the number of solutions improved and **#Ties** represents the number of ties. It can be observed that SplitILS was capable of improving the result of 243 instances and equal the result of another 55.

Table 7
Results for the SDVRP-LF on the instances of Archetti et al. [10].

Problem	LB[a]	BKS[a]	SplitILS						
			Best Sol.	Avg. Sol.	Avg. Gap LB (%)	Avg. Gap (%)	Avg. Time (s)	Avg Time BKS (s)	Best All Exp.
p01-50	–	–	524.61	524.61	–	–	1.83	–	524.61
p02-75	–	–	823.89	824.39	–	–	30.31	–	823.89
p03-100	–	–	826.14	826.45	–	–	42.16	–	826.14
p04-150	–	–	1023.87	1026.48	–	–	243.93	–	1023.66
p05-199	–	–	1285.79	1292.79	–	–	1672.72	–	1285.25
p11-120	–	–	1037.88	1038.68	–	–	85.14	–	1037.88
p01-50D1	457.17	459.98	**459.50**	**459.50**	0.51	−0.10	1.21	0.09	459.50
p02-75D1	604.89	637.36	**617.85**	**620.19**	2.53	−2.69	5.72	0.02	617.85
p03-100D1	742.97	794.21	**760.00**	**760.70**	2.39	−4.22	22.00	0.03	760.00
p04-150D1	896.03	1065.94	**921.47**	**923.74**	3.09	−13.34	164.58	0.03	921.44
p05-199D1	1042.37	1188.45	**1074.18**	**1080.65**	3.67	−9.07	629.08	0.04	1074.18
p11-120D1	1023.37	1063.73	**1043.19**	**1043.21**	1.94	−1.93	93.35	8.15	1042.89
p01-50D2	747.25	771.65	**756.71**	**760.52**	1.78	−1.44	14.55	0.14	756.71
p02-75D2	1093.56	1124.70	**1110.43**	**1112.70**	1.75	−1.07	54.11	0.63	1109.62
p03-100D2	1435.23	1492.05	**1458.46**	**1462.37**	1.89	−1.99	200.43	0.48	1458.46
p04-150D2	1986.79	2109.45	**2017.00**	**2021.78**	1.76	−4.16	1156.99	0.13	2016.96
p05-199D2	2423.64	2632.22	**2481.44**	**2487.28**	2.63	−5.51	2846.16	0.14	2478.02
p11-120D2	2879.63	2988.61	**2899.91**	**2905.28**	0.89	−2.79	898.52	0.22	2898.46
p01-50D3	996.93	1011.64	**1005.75**	**1005.93**	0.90	−0.56	21.48	4.19	1005.75
p02-75D3	1483.17	1509.79	**1502.05**	**1503.42**	1.37	−0.42	110.02	2.86	1502.05
p03-100D3	1971.43	2018.09	**1997.76**	**2001.83**	1.54	−0.81	366.31	24.78	1996.76
p04-150D3	2811.64	2956.18	**2849.66**	**2856.41**	1.59	−3.37	1699.36	0.30	2849.66
p05-199D3	3420.17	3548.13	**3472.79**	**3481.37**	1.79	−1.88	3015.92	4.94	3471.71
p11-120D3	4162.99	4308.17	**4219.01**	**4220.59**	1.38	−2.03	2260.39	4.37	4217.59
p01-50D4	1470.91	1490.61	**1488.27**	**1489.05**	1.23	−0.10	52.71	3.54	1488.06
p02-75D4	2270.44	2372.22	**2301.61**	**2304.89**	1.52	−2.84	283.77	0.12	2301.15
p03-100D4	3043.27	3167.29	**3090.65**	**3094.91**	1.70	−2.29	746.44	0.14	3090.40
p04-150D4	4474.18	4708.11	**4543.18**	**4550.63**	1.71	−3.34	2467.51	0.21	4542.52
p05-199D4	5425.69	5894.69	**5526.28**	**5530.56**	1.93	−6.18	5799.52	0.34	5524.97
p11-120D4	6808.07	7020.87	**6856.11**	**6863.96**	0.82	−2.23	3363.54	0.79	6854.51
p01-50D5	1467.09	1498.30	**1481.71**	**1484.62**	1.19	−0.91	42.90	0.57	1481.71
p02-75D5	2192.25	2235.61	**2219.52**	**2222.58**	1.38	−0.58	261.25	18.55	2219.11
p03-100D5	2945.76	3044.92	**2991.22**	**2991.89**	1.57	−1.74	756.18	0.38	2990.27
p04-150D5	4269.77	4637.69	**4336.80**	**4342.45**	1.70	−6.37	2366.91	0.08	4332.45
p05-199D5	5306.11	5669.69	**5404.44**	**5415.31**	2.06	−4.49	5706.50	0.32	5399.91
p11-120D5	6584.11	6860.65	**6674.97**	**6678.58**	1.43	−2.65	2306.51	0.07	6662.11
p01-50D6	2133.94	2162.58	**2156.14**	**2160.60**	1.25	−0.09	84.35	19.39	2155.80
p02-75D6	3177.69	3259.82	**3223.06**	**3226.79**	1.55	−1.01	377.02	0.01	3223.06
p03-100D6	4316.42	4494.53	**4387.32**	**4389.19**	1.69	−2.34	719.27	0.03	4386.72
p04-150D6	6287.09	6529.63	**6396.68**	**6402.63**	1.84	−1.94	2180.59	0.09	6396.44
p05-199D6	8062.24	8400.74	**8188.47**	**8195.06**	1.65	−2.45	3528.41	0.22	8188.47
p11-120D6	10,111.11	10,456.19	**10,215.90**	**10,218.78**	1.06	−2.27	2006.24	0.07	10202.34
Average					1.69	−2.81	1159.19		

[a] Archetti et al. [5].

Table 8
Results for the SDVRP-LF on the instances of Chen et al. [15].

Problem	LB	BKS	Wilck et al.		SplitILS						
			Best Sol.	Time[a] (s)	Best Sol.	Avg. Sol.	Avg. Gap LB (%)	Avg. Gap (%)	Avg. Time (s)	Avg. Time BKS (s)	Best All Exp.
SD1	228.28	228.28[b,c]	**228.28**	0.27	**228.28**	**228.28**	0.00	0.00	0.05	0.00	228.28
SD2	708.28	708.28[b,c]	**708.28**	1.95	**708.28**	**708.28**	0.00	0.00	0.58	0.00	708.28
SD3	430.40	430.40[b,c]	430.58	1.94	430.58	430.58	0.04	0.04	0.59	–	430.58
SD4	630.62	630.62[b,c]	631.05	6.24	631.05	631.05	0.07	0.07	2.16	–	631.05
SD5	1381.76	1389.94[c]	1390.57	14.20	1390.57	1390.57	0.64	0.05	5.90	–	1390.57
SD6	830.86	830.86[b,c]	833.58	14.97	831.24	831.24	0.05	0.05	5.62	–	831.24
SD7	3640.00	3640.00[c]	**3640.00**	28.61	**3640.00**	**3640.00**	0.00	0.00	13.74	0.00	3640.00
SD8	5068.28	5068.28[b,c]	**5068.28**	48.26	**5068.28**	**5068.28**	0.00	0.00	24.07	0.01	5068.28
SD9	2031.22	2042.88[c]	2054.84	48.91	2044.20	2044.43	0.65	0.08	35.86	–	2044.20
SD10	2679.04	2683.73[c]	2746.54	114.16	2684.88	2684.88	0.22	0.04	81.76	–	2684.88
SD11	13,275.00	13,280.00[c]	**13,280.00**	231.64	**13,280.00**	**13,280.00**	0.04	0.00	136.43	0.01	13,280.00
SD12	7175.80	7239.57[c]	7279.97	227.11	**7213.61**	**7216.34**	0.56	−0.32	179.19	0.02	7213.61
SD13	10,053.60	10,105.86[c]	10,110.57	421.95	10,110.58	10,110.58	0.57	0.05	168.07	–	10110.58
SD14	10,588.20	10,725.38[c]	10,786.52	718.65	**10,715.53**	**10,722.73**	1.27	−0.02	432.26	196.13	10,715.53
SD15	14,908.50	15,129.68[c]	15,160.04	1278.35	**15,093.85**	**15,102.85**	1.30	−0.18	658.54	0.06	15,093.85
SD16	3379.33	3379.33[b,c]	3433.83	1225.88	3395.11	3395.16	0.47	0.47	580.27	–	3381.25
SD17	26,317.20	26,533.39[c]	26,559.92	1722.20	**26,493.56**	**26,499.23**	0.69	−0.13	484.43	0.10	26,493.56
SD18	14,029.20	14,283.51[c]	14,302.22	1735.83	**14,197.80**	**14,202.85**	1.24	−0.56	676.77	0.10	14,197.80
SD19	19,707.20	20,152.53[d]	**20,152.53**	3093.17	**19,989.95**	**20,000.54**	1.49	−0.75	1261.95	0.17	19,989.95
SD20	39,252.80	39,706.51[d]	**39,706.51**	6208.16	**39,641.91**	**39,648.42**	1.01	−0.15	1518.12	0.33	39,637.53
SD21	11,271.00	11,271.00[b,e]	11,461.20	10,565.70	11,344.96	11,357.62	0.77	0.77	4326.99	–	11,344.96
Average				1319.44			0.53	−0.02	504.44		

[a] Xeon 2.49 GHz, Single run.
[b] Optimal solution.
[c] Archetti et al. [5].
[d] Wilck IV and Cavalier [38].
[e] Moreno et al. [28].

Table 9
Results for the SDVRP-UF on the instances of Belenguer et al. [11].

Problem	LB	BKS	Aleman and Hill			SplitILS							
			K	Best Sol.	Time[a] (s)	Avg. *K*	Best Sol.	Avg. Sol.	Avg. Gap LB (%)	Avg. Gap (%)	Avg. Time (s)	Avg. Time BKS (s)	Best all exp.
eil22	–	375.28[b]	4	**375.28**	2.58	4.00	**375.28**	**375.28**	–	0.00	0.15	0.00	375.28
eil23	451.80	569.75[b]	3	**569.75**	1.59	3.00	**568.56**	**568.56**	25.84	−0.21	0.13	0.00	568.56
eil30	218.92	505.01[c]	4	**505.01**	7.45	4.00	**505.01**	**505.01**	130.68	0.00	0.24	0.00	505.01
eil33	–	843.64[b]	4	**843.64**	8.38	4.00	**837.06**	**837.06**	–	−0.78	0.51	0.00	837.06
eil51	518.23	527.67[b]	5	**527.67**	49.84	5.00	**524.61**	**524.61**	1.23	−0.58	1.79	0.16	524.61
eilA76	809.58	832.71[c]	10	853.20	145.78	10.00	**823.89**	**824.92**	1.89	−0.94	30.76	0.52	823.89
eilB76	984.13	1034.21[b]	14	**1034.21**	91.36	14.00	**1009.04**	**1012.07**	2.84	−2.14	51.83	0.16	1009.04
eilC76	721.39	761.55[b]	8	**761.55**	151.13	8.00	**738.67**	**739.89**	2.56	−2.84	16.96	0.02	738.67
eilD76	672.34	695.96[b]	7	**695.96**	122.52	7.00	**687.60**	**689.36**	2.53	−0.95	11.16	0.04	687.60
eilA101	804.27	844.21[b]	8	**844.21**	295.22	8.00	**826.14**	**826.58**	2.77	−2.09	38.90	0.04	826.14
eilB101	1055.59	1112.15[b]	14	**1112.15**	173.13	14.00	**1076.26**	**1079.15**	2.23	−2.97	110.61	0.16	1076.26
S51D1	457.08	459.50[c]	3	468.79	13.56	3.00	**459.50**	**459.50**	0.53	0.00	1.24	0.01	459.50
S51D2	697.00	717.18[c]	9	718.69	31.66	9.00	**709.29**	**709.49**	1.79	−1.07	11.20	0.03	708.42
S51D3	933.97	960.38[c]	15	969.78	18.75	15.00	**948.06**	**950.12**	1.73	−1.07	15.74	0.03	947.97
S51D4	1545.19	1569.92[c]	27	1628.20	19.77	27.60	**1562.01**	**1563.29**	1.17	−0.42	56.28	2.23	1561.34
S51D5	1316.93	1339.38[c]	23	1362.19	15.39	23.00	**1333.67**	**1333.67**	1.27	−0.43	36.69	1.36	1333.67
S51D6	2149.55	2182.13[c]	41	2236.16	14.38	41.60	**2169.10**	**2177.78**	1.31	−0.20	62.55	0.19	2169.10
S76D1	590.92	613.70[b]	4	**613.70**	252.28	4.00	**598.94**	**598.94**	1.36	−2.41	4.86	0.02	598.94
S76D2	1066.88	1104.56[c]	15	1128.15	60.44	15.00	**1087.40**	**1089.45**	2.12	−1.37	69.36	0.94	1087.40
S76D3	1406.85	1435.10[c]	23	1472.92	51.13	23.00	**1427.86**	**1429.26**	1.59	−0.41	96.50	23.13	1427.81
S76D4	2053.66	2104.87[d]	37	2180.13	53.56	37.00	**2079.76**	**2081.16**	1.34	−1.13	188.38	0.32	2079.76
S101D1	714.50	749.93[b]	5	**749.93**	860.31	5.00	**726.59**	**728.45**	1.95	−2.86	15.93	0.27	726.59
S101D2	1356.78	1397.38[d]	20	1409.03	219.52	20.00	**1378.43**	**1386.03**	2.16	−0.81	151.66	2.67	1378.43
S101D3	1845.07	1911.40[c]	31	1947.62	132.19	31.00	**1874.81**	**1880.62**	1.93	−1.61	317.29	0.32	1874.81
S101D5	2758.21	2824.16[c]	48	2910.71	131.16	49.00	**2791.22**	**2795.36**	1.35	−1.02	572.13	2.18	2790.70
Average					116.92				8.44	−1.13	74.51	1.39	

[a] Pentium 4, 2.8 GHz, Single run, Avg. cTime (s): 48.94.
[b] Aleman and Hill [1].
[c] Archetti et al. [5].
[d] Gulczynski et al. [24], Pentium 4, 3.0 GHz, Single run.

Table 10
Results for the SDVRP-UF on the instances of Belenguer et al. [11] using rounded costs.

Problem	BKS	Boudia et al.		Aleman and Hill			SplitILS						
		Best Sol.	Time[a] (s)	*K*	Best Sol.	Time[b] (s)	Avg. *K*	Best Sol.	Avg. Sol.	Avg. Gap (%)	Avg. Time (s)	Avg. Time BKS (s)	Best All Exp.
eil22	375[c]	**375**	4.11	4	**375**	2.58	4.00	**375**	**375.00**	0.00	0.15	0.00	375
eil23	569[c]	**569**	5.47	3	570	1.59	3.00	**569**	**569.00**	0.00	0.11	0.00	569
eil30	503[c]	**503**	5.70	4	**503**	7.45	4.00	**503**	**503.00**	0.00	0.23	0.00	503
eil33	835[c]	**835**	5.19	4	844	8.38	4.00	**835**	**835.00**	0.00	0.45	0.01	835
eil51	521[c]	**521**	7.28	5	526	49.84	5.00	**521**	**521.00**	0.00	1.75	0.10	521
eilA76	818[c]	828	35.94	10	847	145.78	10.00	**818**	821.75	0.46	24.63	11.69	818
eilB76	1007[c]	1019	13.09	14	1027	91.36	14.00	**1002**	1007.05	0.00	37.68	25.37	1002
eilC76	733[c]	738	14.75	8	754	151.13	8.00	**733**	733.75	0.10	14.75	6.24	732
eilD76	682[c]	**682**	23.12	7	691	122.52	7.00	**682**	683.05	0.15	10.39	6.43	681
eilA101	815[c]	818	25.25	8	834	295.22	8.00	**814**	816.20	0.15	32.61	20.95	814
eilB101	1082[c]	**1082**	21.81	14	1104	173.13	14.00	**1061**	**1064.00**	−1.66	78.42	5.54	1061
S51D1	458[c]	**458**	8.77	3	465	13.56	3.00	**458**	**458.00**	0.00	1.17	0.05	458
S51D2	706[c]	707	7.44	9	715	31.66	9.00	**703**	**704.75**	−0.18	8.12	0.79	703
S51D3	945[c]	**945**	7.84	15	966	18.75	15.00	**943**	**944.05**	−0.10	13.06	0.81	943
S51D4	1578[c]	**1578**	11.98	27	1621	19.77	27.35	**1553**	**1556.50**	−1.36	39.25	0.04	1552
S51D5	1336[c]	1351	16.72	23	1357	15.39	23.00	**1328**	**1329.25**	−0.51	32.07	1.75	1328
S51D6	2177[c]	2182	9.92	41	2228	14.38	41.15	**2163**	**2166.15**	−0.50	52.95	0.01	2163
S76D1	592[c]	**592**	15.23	4	606	252.28	4.00	**592**	592.30	0.05	4.75	2.06	592
S76D2	1087[c]	1089	30.50	15	1124	60.44	15.00	**1082**	**1083.15**	−0.35	53.60	15.28	1081
S76D3	1420[c]	1427	12.89	23	1466	51.13	23.00	**1420**	1423.05	0.21	67.81	56.14	1419
S76D4	2094[c]	2117	8.76	37	2170	53.56	37.00	**2073**	**2074.95**	−0.91	144.89	7.53	2070
S101D1	716[c]	717	49.75	5	741	860.31	5.00	**716**	718.35	0.33	14.76	12.13	716
S101D2	1372[c]	**1372**	31.72	20	1398	219.52	20.00	**1366**	**1371.40**	−0.04	112.47	60.90	1364
S101D3	1891[c]	**1891**	33.98	31	1936	132.19	31.00	**1864**	**1868.05**	−1.21	236.05	2.91	1860
S101D5	2854[c]	**2854**	18.66	48	2897	131.16	49.00	**2770**	**2779.10**	−2.62	439.49	0.11	2769
Average			17.03			116.92				−0.32	56.87	9.47	

[a] PC 3.0 GHz, Single run, Avg. cTime (s): 9.28.
[b] Pentium 4, 2.8 GHz, Single run, Avg. cTime (s): 48.94.
[c] Boudia et al. [13].

Table 11
Results for the SDVRP-UF on the instances of Campos et al. [14].

Problem	BKS	Boudia et al.		Aleman and Hill			SplitILS						
		Best Sol.	Time[a] (s)	*K*	Best Sol.	Time[b] (s)	Avg. *K*	Best Sol.	Avg. Sol.	Avg. Gap (%)	Avg. Time (s)	Avg. Time BKS (s)	Best All Exp.
p01-50	524.61[c]	**524.61**	8.53	5	527.67	49.84	5.00	**524.61**	**524.61**	0.00	1.82	0.28	524.61
p02-75	823.89[c]	**823.89**	35.72	11	853.20	145.78	10.00	**823.89**	824.80	0.11	30.52	10.20	823.89
p03-100	827.39[c]	829.44	34.59	8	844.21	295.22	8.00	**826.14**	**826.39**	−0.12	40.01	3.09	826.14
p04-150	1025.49[c]	1042.37	103.69	12	1079.55	2217.67	12.00	**1023.66**	1026.89	0.14	250.37	198.64	1023.66
p05-199	1293.00[c]	1311.59	353.84	17	1339.49	4514.28	16.05	**1286.92**	1293.71	0.05	1298.10	1040.29	1285.55
p06-120	1037.88[c]	1041.20	50.92	7	1051.24	1944.19	7.00	**1037.88**	1039.13	0.12	81.50	51.32	1037.88
p07-100	819.56[c]	**819.56**	42.89	10	819.60	75.33	10.00	**819.56**	**819.56**	0.00	27.67	0.13	819.56
p01-50D1	460.79[c]	**460.79**	12.38	3	466.74	19.69	3.00	**460.79**	**460.79**	0.00	1.17	0.04	460.79
p02-75D1	596.25[c]	600.06	18.75	4	614.09	136.14	4.00	**596.25**	**596.25**	0.00	4.98	0.15	596.25
p03-100D1	726.81[c]	**726.81**	37.12	5	741.60	1944.09	5.00	**726.81**	730.80	0.55	16.67	14.08	726.81
p04-150D1	866.31[c]	875.61	100.27	8	891.10	2640.95	8.00	**866.31**	**866.31**	0.00	120.92	18.68	866.31
p05-199D1	1018.38[c]	1018.71	356.22	10	1069.24	11.215.52	10.00	**1017.28**	1018.59	0.02	431.97	261.10	1017.28
p06-120D1	976.57[c]	**976.57**	72.98	6	990.59	2736.34	6.00	**975.96**	**976.57**	0.00	44.82	13.57	975.96
p07-100D1	634.57[c]	649.73	34.97	6	658.99	461.75	5.15	**632.63**	636.76	0.35	12.28	7.29	632.63
p01-50D2	741.06[c]	751.41	10.22	10	753.98	23.17	10.00	**741.06**	741.26	0.03	9.72	5.86	741.06
p02-75D2	1067.80[c]	1074.46	34.14	15	1085.70	97.17	15.00	**1064.49**	**1066.36**	−0.13	53.25	23.82	1064.49
p03-100D2	1377.28[c]	1392.85	78.06	20	1416.35	160.95	20.00	**1376.22**	1380.23	0.21	187.76	173.95	1374.15
p04-150D2	1875.09[c]	1878.71	147.89	30	1929.91	755.08	29.00	**1861.63**	**1866.95**	−0.43	1041.64	147.76	1861.51
p05-199D2	2329.37[c]	2340.14	347.14	39	2408.16	1544.36	38.00	**2305.70**	**2313.04**	−0.70	2296.08	445.68	2305.70
p06-120D2	2720.35[c]	2720.38	144.19	24	2744.74	463.97	23.00	**2707.52**	**2710.15**	−0.37	704.44	25.93	2703.88
p07-100D2	1415.78[c]	1417.28	43.27	–	1441.48	98.31	20.00	**1413.85**	**1413.99**	−0.13	128.52	1.08	1413.85
p01-50D3	988.31[c]	**988.31**	12.49	15	1023.24	17.72	15.00	**982.79**	**983.59**	−0.48	18.04	0.21	982.77
p02-75D3	1398.53[c]	1413.80	37.38	23	1458.59	67.66	22.00	**1393.11**	**1393.11**	−0.39	97.13	12.47	1393.11
p03-100D3	1827.65[c]	1845.30	28.39	29	1886.70	145.05	29.00	**1823.58**	1827.81	0.01	313.83	123.39	1823.41
p04-150D3	2539.75[c]	2561.65	224.89	45	2647.17	470.34	43.00	**2527.96**	**2531.50**	−0.32	1445.25	256.58	2526.04
p05-199D3	3180.30[c]	3191.25	436.20	56	3296.69	1216.69	56.15	**3156.02**	**3163.26**	−0.54	3316.93	131.70	3155.85
p06-120D3	3934.39[c]	**3934.39**	163.14	35	4010.80	340.53	34.00	**3907.27**	**3909.28**	−0.64	1487.97	32.80	3907.27
p07-100D3	1994.59[c]	**1994.59**	51.31	30	2010.00	84.50	29.00	**1967.41**	**1968.09**	−1.33	265.71	0.37	1967.41
p01-50D4	1467.06[c]	**1467.06**	21.42	26	1530.81	19.11	25.00	**1456.00**	**1457.37**	−0.66	42.86	0.64	1456.00
p02-75D4	2087.22[c]	2102.58	46.11	39	2164.74	61.81	37.05	**2081.38**	**2084.91**	−0.11	191.06	47.31	2076.41
p03-100D4	2780.95[c]	**2780.95**	84.38	48	2874.86	125.28	48.00	**2749.53**	**2753.99**	−0.97	647.44	2.39	2749.53
p04-150D4	4045.87[c]	**4045.87**	244.91	74	4151.90	451.95	73.15	**3988.64**	**3996.55**	−1.22	1901.05	4.31	3987.58
p05-199D4	4941.22[c]	**4941.22**	725.69	93	5066.24	108.63	93.00	**4843.83**	**4855.49**	−1.73	3739.98	3.38	4843.83
p06-120D4	6308.76[d]	6318.37	196.14	56	**6308.76**	418.98	56.00	**6195.37**	**6219.01**	−1.42	1805.91	1.97	6195.37
p07-100D4	3113.72[c]	**3113.72**	52.13	48	3157.48	97.58	48.00	**3088.47**	**3089.41**	−0.78	416.34	0.22	3088.23
p01-50D5	1477.01[c]	**1477.01**	24.53	26	1505.38	19.09	25.00	**1467.47**	**1467.47**	−0.65	49.42	2.16	1467.47
p02-75D5	2132.16[c]	**2132.16**	51.78	38	2182.33	55.17	37.95	**2111.83**	**2114.03**	−0.85	212.38	6.17	2110.28
p03-100D5	2858.87[c]	**2858.87**	100.16	49	2929.29	134.84	49.00	**2813.52**	**2817.05**	−1.46	737.91	1.63	2813.37
p04-150D5	4045.87[c]	**4045.87**	244.86	74	4151.90	449.34	73.05	**3985.76**	**3995.25**	−1.25	1836.14	3.54	3985.76
p05-199D5	5155.36[c]	**5155.36**	749.94	96	5281.55	119.04	96.00	**5063.89**	**5072.74**	−1.60	4222.85	3.52	5056.93
p06-120D5	6399.42[c]	6424.71	271.39	58	6511.08	436.80	58.00	**6373.24**	**6376.25**	−0.36	2303.70	16.18	6372.92
p07-100D5	3155.69[c]	**3155.69**	91.31	50	3200.62	96.39	49.00	**3125.47**	**3125.98**	−0.94	568.97	0.61	3125.35
p01-50D6	2154.35[c]	**2154.35**	22.91	41	2219.32	24.41	41.00	**2150.97**	**2152.95**	−0.06	51.94	5.02	2150.97
p02-75D6	3200.35[c]	**3200.35**	27.48	60	3278.33	86.27	60.25	**3178.47**	**3181.28**	−0.60	412.86	0.59	3178.47
p03-100D6	4312.95[c]	**4312.95**	55.75	80	4435.56	185.55	80.00	**4294.12**	**4299.40**	−0.31	737.85	0.03	4294.10
p04-150D6	6267.48[c]	**6267.48**	401.62	119	6416.12	678.94	119.00	**6232.37**	**6234.56**	−0.53	1543.69	0.09	6229.85
p05-199D6	8081.58[c]	**8081.58**	571.70	158	8333.61	153.12	158.00	**8037.88**	**8048.57**	−0.41	4616.79	3.46	8037.88
p06-120D6	10,017.47[c]	10,063.47	298.08	95	10,186.06	30.32	95.00	**10,003.99**	**10,005.29**	−0.12	2161.59	1.91	10,001.12
p07-100D6	4919.48[c]	**4919.48**	180.11	80	4996.88	152.92	80.00	**4903.00**	**4906.56**	−0.26	799.40	8.30	4902.73
Average			152.73			771.18				−0.41	872.02	63.55	

[a] PC 3.0 GHz, Single run, Avg. cTime (s): 83.21.
[b] Pentium 4, 2.8 GHz, Single run, Avg. cTime (s): 322.79.
[c] Boudia et al. [13].
[d] Aleman and Hill [1].

Table 12
Results for the SDVRP-UF on the instances of Archetti et al. [10].

Problem	LB	BKS	Derigs et al.		SplitILS							
			K	Best[a] Sol.	Avg. *K*	Best Sol.	Avg. Sol.	Avg. Gap LB (%)	Avg. Gap (%)	Avg. Time (s)	Avg. Time BKS (s)	Best All Exp.
p01-50	–	524.61[b]	5	**524.61**	5.00	**524.61**	**524.61**	–	0.00	1.88	0.48	524.61
p02-75	–	823.89[b]	10	829.89	10.00	**823.89**	825.06	–	0.14	29.69	14.04	823.89
p03-100	–	826.14[b]	8	**826.14**	8.00	**826.14**	826.45	–	0.04	40.52	12.91	826.14
p04-150	–	1028.42[b]	12	**1028.42**	12.00	**1023.87**	**1027.28**	–	−0.11	233.73	59.28	1023.66
p05-199	–	1296.66[b]	16	1302.89	16.00	**1289.89**	**1293.24**	–	−0.26	1355.25	564.00	1283.27
p11-120	–	1042.12[b]	7	**1042.12**	7.00	**1037.88**	1043.38	–	0.12	84.84	9.02	1037.88
p01-50D1	456.63	459.50[c]	–	–	3.00	**459.50**	**459.50**	0.63	0.00	1.16	0.03	459.50
p02-75D1	604.70	652.93[c]	–	–	5.00	**617.85**	**619.59**	2.46	−5.11	5.70	0.00	617.85
p03-100D1	743.06	788.23[c]	–	–	6.00	**760.00**	**760.46**	2.34	−3.52	21.65	0.09	760.00
p04-150D1	895.46	984.69[c]	–	–	9.00	**921.91**	**923.69**	3.15	−6.19	161.90	0.04	921.47
p05-199D1	1042.37	1268.79[c]	–	–	12.00	**1074.18**	**1080.64**	3.67	−14.83	626.12	0.04	1074.18
p11-120D1	1023.39	1071.58[c]	–	–	8.00	**1043.19**	**1043.22**	1.94	−2.65	109.76	0.14	1042.89
p01-50D2	750.45	758.20[c]	11	776.42	11.00	**757.15**	761.12	1.42	0.39	13.24	10.87	756.71
p02-75D2	1095.65	1116.75[b]	16	1123.97	16.00	**1109.62**	**1112.11**	1.50	−0.42	53.26	8.69	1109.62
p03-100D2	1437.78	1472.53[b]	22	1478.59	22.00	**1458.46**	**1462.68**	1.73	−0.67	179.69	18.52	1458.46
p04-150D2	1986.34	2037.00[b]	32	2055.18	32.00	**2016.97**	**2021.36**	1.76	−0.77	1109.88	98.57	2016.93
p05-199D2	2423.99	2528.82[b]	41	2540.06	41.05	**2478.40**	**2486.54**	2.58	−1.67	2661.25	4.76	2477.44
p11-120D2	2867.79	2907.39[b]	26	2913.09	26.00	**2898.50**	**2907.07**	1.37	−0.01	895.11	413.06	2898.46
p01-50D3	996.72	1007.51[b]	16	1012.56	16.00	**1005.75**	**1005.75**	0.91	−0.17	20.70	2.39	1005.75
p02-75D3	1482.50	1504.74[b]	24	1508.73	24.00	**1502.05**	**1503.57**	1.42	−0.08	108.37	26.32	1502.05
p03-100D3	1971.34	2018.94[b]	33	2035.91	33.00	**1996.76**	**2002.23**	1.57	−0.83	362.02	6.92	1996.76
p04-150D3	2811.98	2901.62[b]	49	2912.08	49.00	**2849.66**	**2857.28**	1.61	−1.53	1518.82	3.72	2849.66
p05-199D3	3420.23	3548.31[b]	63	3581.66	63.00	**3471.41**	**3480.76**	1.77	−1.90	3014.82	6.03	3471.33
p11-120D3	4156.68	4259.94[c]	40	4270.38	40.00	**4219.01**	**4220.79**	1.54	−0.92	1957.37	83.32	4218.50
p01-50D4	1471.54	1488.58[b]	27	1489.64	27.00	**1488.58**	1488.89	1.18	0.02	43.04	28.41	1488.40
p02-75D4	2272.05	2318.28[c]	41	2340.09	41.00	**2298.58**	**2301.85**	1.31	−0.71	207.22	2.60	2298.33
p03-100D4	3042.93	3116.61[b]	56	3145.33	56.00	**3085.69**	**3093.64**	1.67	−0.74	736.52	8.56	3085.69
p04-150D4	4474.92	4581.32[b]	84	4638.74	83.05	**4545.46**	**4550.85**	1.70	−0.67	2410.48	60.96	4543.23
p05-199D4	5422.95	5669.26[b]	105	**5669.26**	106.00	**5521.57**	**5529.06**	1.96	−2.47	4349.61	0.33	5520.69
p11-120D4	6780.19	6881.04[b]	67	6890.39	67.00	**6854.09**	**6865.23**	1.25	−0.23	3442.31	159.93	6854.03
p01-50D5	1466.34	1487.81[b]	26	1488.28	26.00	**1481.71**	**1483.36**	1.16	−0.30	44.11	6.69	1481.71
p02-75D5	2195.44	2228.69[b]	39	2243.93	39.00	**2219.97**	**2224.06**	1.30	−0.21	265.00	85.83	2219.52
p03-100D5	2945.42	3002.64[b]	53	3014.08	53.00	**2989.30**	**2992.75**	1.61	−0.33	742.88	55.01	2988.38
p04-150D5	4267.33	4374.56[b]	79	4435.95	79.00	**4334.71**	**4341.15**	1.73	−0.76	2357.52	19.48	4334.71
p05-199D5	5304.09	5487.55[b]	102	5541.09	102.95	**5409.76**	**5417.75**	2.14	−1.27	4524.33	5.43	5403.35
p11-120D5	6593.28	6658.52[b]	64	6671.04	64.00	6673.95	6678.11	1.29	0.29	2354.02	–	6652.64
p01-50D6	2134.96	2160.66[b]	41	2174.54	41.00	**2156.14**	2161.31	1.23	0.03	78.49	39.87	2155.80
p02-75D6	3177.20	3234.64[b]	61	3266.78	61.00	**3223.40**	**3226.20**	1.54	−0.26	378.95	3.86	3223.40
p03-100D6	4319.16	4411.32[b]	82	4447.47	82.00	**4387.32**	**4389.43**	1.63	−0.50	675.01	0.08	4387.32
p04-150D6	6284.76	6467.17[b]	122	**6467.17**	122.85	**6395.41**	**6402.15**	1.87	−0.94	1926.20	0.09	6393.22
p05-199D6	8062.14	8297.71[b]	161	**8297.71**	161.00	**8192.03**	**8195.67**	1.66	−1.23	3258.29	0.22	8188.15
p11-120D6	10,113.55	10,233.37[b]	98	**10,233.37**	98.00	**10,204.81**	**10,216.80**	1.02	−0.16	2279.57	13.91	10,204.79
Average								1.68	−1.22	1062.86		

[a] PC 3.0 GHz, 3600 s for all problems (cTime (s): 1944), best of five runs using the ABHC heuristic.
[b] Derigs et al. [17].
[c] Archetti et al. [5].

Table 13
Results for the SDVRP-UF on the instances of Chen et al. [15].

Problem	LB	BKS	Aleman and Hill			SplitILS							
			K	Best Sol.	Time[a] (s)	Avg. *K*	Best Sol.	Avg. Sol.	Avg. Gap LB (%)	Avg. Gap (%)	Avg. Time (s)	Avg. Time BKS (s)	Best All Exp.
SD1	228.28	228.28[b,c]	6	**228.28**	0.00	6.00	**228.28**	**228.28**	0.00	0.00	0.05	0.00	228.28
SD2	708.28	708.28[b,c]	12	**708.28**	0.02	12.00	**708.28**	**708.28**	0.00	0.00	0.63	0.02	708.28
SD3	430.40	430.40[b,c]	12	430.58	0.03	12.00	430.58	430.58	0.04	0.04	0.62	–	430.58
SD4	630.62	630.62[b,c]	18	631.05	0.08	18.00	631.05	631.05	0.07	0.07	2.26	–	631.05
SD5	1373.25	1389.94[c]	24	1390.57	0.13	24.00	1390.57	1390.57	1.26	0.05	6.07	–	1390.57
SD6	830.86	830.86[b,c]	24	831.24	0.14	24.00	831.24	831.24	0.05	0.05	5.81	–	831.24
SD7	3638.45	3640.00[c]	30	**3640.00**	0.09	30.00	**3640.00**	**3640.00**	0.04	0.00	14.12	0.00	3640.00
SD8	5068.28	5068.28[b,c]	36	**5068.28**	0.14	36.00	**5068.28**	**5068.28**	0.00	0.00	24.93	0.01	5068.28
SD9	2028.08	2042.88[c]	36	2071.03	0.36	36.00	2044.20	2044.20	0.79	0.06	38.78	–	2044.20
SD10	2678.83	2683.73[c]	48	2747.83	0.89	48.00	2684.88	2684.88	0.23	0.04	101.10	–	2684.88
SD11	13,111.11	13,280.00[c]	60	**13,280.00**	0.41	60.00	**13,280.00**	**13,280.00**	1.29	0.00	152.42	0.02	13,280.00
SD12	7089.27	7213.62[d]	60	**7213.62**	0.84	60.00	**7213.61**	7216.60	1.80	0.04	210.71	160.60	7213.61
SD13	9969.93	10,105.86[c]	72	10,110.58	1.20	72.00	10,110.58	10,110.58	1.41	0.05	189.45	–	10,110.58
SD14	10,502.76	10,754.70[c]	90	10,802.87	2.31	90.00	**10,717.53**	**10,723.79**	2.10	−0.29	479.85	0.04	10,715.52
SD15	14,787.05	15,152.88[e]	108	15,153.45	3.20	108.00	**15,094.48**	**15,105.90**	2.16	−0.31	731.98	0.06	15,093.85
SD16	3379.33	3379.33[b,c]	108	3446.43	7.59	108.00	3381.26	3394.48	0.45	0.45	930.72	–	3381.25
SD17	26,166.80	26,493.56[d]	120	**26,493.56**	7.27	120.15	26,496.06	26,499.32	1.27	0.02	577.29	–	26,496.06
SD18	13,892.74	14,323.04[d]	120	**14,323.04**	27.95	120.25	**14,202.53**	**14,205.07**	2.25	−0.82	834.60	0.10	14,199.22
SD19	19,584.84	20,157.10[d]	144	**20,157.10**	11.95	144.55	**19,995.69**	**20,007.52**	2.16	−0.74	1524.67	0.16	19,991.18
SD20	38,901.37	39,722.86[d]	180	**39,722.86**	11.02	180.00	**39,635.51**	**39,647.61**	1.92	−0.19	1563.38	0.32	39,635.51
SD21	11,254.83	11,271.06[e]	216	11,458.76	111.56	216.35	11,345.68	11,365.37	0.98	0.84	5034.56	–	11,345.68
Average					8.91				0.96	−0.03	591.62	14.67	

[a] Pentium 4, 2.8 GHz, Single run, Avg. cTime (s): 3.73.
[b] Optimal solution.
[c] Archetti et al. [5].
[d] Aleman and Hill [1].
[e] Derigs et al. [17], PC 3.0 GHz, Single run.

Table 14
Summary of SplitILS results.

Benchmark dataset	#Instances	#Improv.	#Ties	Avg. Gap (%)	Avg. Time (s)	Avg. Time BKS (s)
Belenguer et al. [11][a,b]	25	18	7	−1.87	63.77	1.50
Belenguer et al. [11][a]	25	22	3	−1.46	75.97	1.69
Campos et al. [14][a]	49	44	5	−2.23	925.59	3.06
Archetti et al. [10][a]	42	36	–	−2.81	1159.19	2.68
Chen et al. [15][a]	21	7	5	−0.02	504.44	16.41[c]
Belenguer et al. [11][d,b]	25	13	12	−0.32	56.87	9.47
Belenguer et al. [11][d]	25	22	3	−1.13	74.51	1.39
Campos et al. [14][d]	49	40	9	−0.41	872.02	63.55
Archetti et al. [10][d]	42	36	5	−1.22	1062.86	44.74[c]
Chen et al. [15][d]	21	5	6	−0.03	591.62	14.67[c]
Total	324	243	55			

[a] SDVRP-LF;
[b] Rounded costs.
[c] Disregarding the instances where SplitILS fails to find or improve the BKS.
[d] SDVRP-UF.

5. Concluding remarks

In this paper we propose a heuristic approach for the Split Delivery Vehicle Routing Problem considering both limited and unlimited fleet. The proposed algorithm, called SplitILS, is based on Iterated Local Search (ILS) and Randomized Variable Neighborhood Descent (RVND). Extensive computational experiments were carried out on 324 instances from the literature and SplitILS was, in most cases, capable of finding or improving the best known solution in a matter of seconds. Moreover, it is important to mention that SplitILS relies mainly on only three parameters, more precisely the number of restarts, the number of ILS iterations and the number of moves applied by the perturbation mechanism. As for future work, one may extend SplitILS to solve other SDVRP variants that may include additional features such as time windows, heterogeneous fleet and pickup and delivery services.

Acknowledgments

We would like to thank the anonymous reviewers and Dr. Thibaut Vidal for the valuable comments that helped us to significantly improve the quality of the paper. We would also like to thank Dr. Aleman for providing the benchmark datasets.

This research was partially supported by the following Brazilian research agencies: CNPq (Grant 471158/2012-7), CAPES (Grant no. 922/2008) and FAPERJ (Grant no. E-26/103.057/2011).

References

[1] Aleman RE, Hill RR. A tabu search with vocabulary building approach for the vehicle routing problem with split demands. Int J Metaheurist 2010;1(1):55–80.
[2] Aleman RE, Zhang X, Hill RR. A ring-based diversification scheme for routing problems. Int J Math Oper Res 2009;1(1):163–90.
[3] Aleman RE, Zhang X, Hill RR. An adaptive memory algorithm for the split delivery vehicle routing problem. J Heurist 2010;16(3):441–73.
[4] Ambrosino D, Sciomachen A. A food distribution network problem: a case study. IMA J Manag Math 2007;18(1):33–53.
[5] Archetti C, Bianchessi N, Speranza MG. A column generation approach for the split delivery vehicle routing problem. Networks 2011;58(4):241–54.
[6] Archetti C, Feillet D, Gendreau M, Speranza MG. Complexity of the VRP and SDVRP. Transp Res Part C: Emerg Technol 2011;19(5):741–50.
[7] Archetti C, Mansini R, Speranza MG. Complexity and reducibility of the skip delivery problem. Transp Sci 2005;39(2):182–7.
[8] Archetti C, Speranza MG. Vehicle routing problems with split deliveries. Int Trans Oper Res 2012;19(1-2):3–22.
[9] Archetti C, Speranza MG, Hertz A. A tabu search algorithm for the split delivery vehicle routing problem. Transp Sci 2006;40(1):64–73.
[10] Archetti C, Speranza MG, Savelsbergh MWP. An optimization-based heuristic for the split delivery vehicle routing problem. Transp Sci 2008;42(1):22–31.
[11] Belenguer JM, Martinez MC, Mota E. A lower bound for the split delivery vehicle routing problem. Oper Res 2000;48(5):801–10.
[12] Belfiore P, Yoshizaki HTY. Scatter search for a real-life heterogeneous fleet vehicle routing problem with time windows and split deliveries in Brazil. Eur J Oper Res 2009;199(3):750–8.
[13] Boudia M, Prins C, Reghioui M. An effective memetic algorithm with population management for the split delivery vehicle routing problem. In: Bartz-Beielstein T, Blesa Aguilera M, Blum C, Naujoks B, Roli A, Rudolph G, et al., editors. Hybrid metaheuristics. Lecture notes in computer science, vol. 4771. Berlin, Heidelberg: Springer; 2007. p. 16–30.
[14] Campos V, Corberán A, Mota E. A scatter search algorithm for the split delivery vehicle routing problem. In: Fink A, Rothlauf F, editors. Advances in computational intelligence in transport, logistics, and supply chain management studies in computational intelligence, vol. 144. Berlin, Heidelberg: Springer; 2008. p. 137–52.
[15] Chen S, Golden B, Wasil E. The split delivery vehicle routing problem: applications, algorithms, test problems, and computational results. Networks 2007;49(4):318–29.
[16] Croes GA. A method for solving traveling-salesman problems. Oper Res 1958;6(6):791–812.
[17] Derigs U, Li B, Vogel U. Local search-based metaheuristics for the split delivery vehicle routing problem. J Oper Res Soc 2010;61(9):1356–64.
[18] Dongarra JJ. Performance of various computers using standard linear equations software. Technical report CS-89-85. Computer Science Department, University of Tennessee; 2011.
[19] Dror M, Laporte G, Trudeau P. Vehicle routing with split deliveries. Discret Appl Math 1994;50(3):239–54.
[20] Dror M, Trudeau P. Savings by split delivery routing. Transp Sci 1989;23(2):141–5.
[21] Dror M, Trudeau P. Split delivery routing. Naval Res Logist 1990;37(3):383–402.
[22] Gendreau M, Hertz A, Laporte G. A tabu search heuristic for the vehicle routing problem. Manag Sci 1994;40(10):1276–90.
[23] Glover F, Rego C. Ejection chain and filter-and-fan methods in combinatorial optimization. 4OR 2006;4(4):263–96.
[24] Gulczynski D, Golden B, Wasil E. The split delivery vehicle routing problem with minimum delivery amounts. Transp Res Part E: Logist Transp Rev 2010;46(5):612–26.
[25] Hansen P, Mladenović N, Moreno-Pérez JA. Variable neighbourhood search: methods and applications. Ann Oper Res 2010;175(1):367–407.
[26] Jin M, Liu K, Bowden RO. A two-stage algorithm with valid inequalities for the split delivery vehicle routing problem. Int J Prod Econ 2007;105(1):228–42.
[27] Jin M, Liu K, Eksioglu B. A column generation approach for the split delivery vehicle routing problem. Oper Res Lett 2008;36(2):265–70.
[28] Moreno L, deAragão MP, Uchoa E. Improved lower bounds for the split delivery vehicle routing problem. Oper Res Lett 2010;38(4):302–6.
[29] Or I. Traveling salesman-type combinatorial problems and their relation to the logistics of regional blood banking [Thesis]. Northwestern University; 1976.
[30] Penna PHV, Subramanian A, Ochi LS. An iterated local search heuristic for the heterogeneous fleet vehicle routing problem. J Heurist 2013;19(2):201–32.
[31] Reinelt G. TSPLIB—a traveling salesman problem library. INFORMS J Comput 1991;3(4):376–84.
[32] Sierksma G, Tijssen GA. Routing helicopters for crew exchanges on off-shore locations. Ann Oper Res 1998;76(0):261–86.
[33] Silva MM, Subramanian A, Vidal T, Ochi LS. A simple and effective metaheuristic for the minimum latency problem. Eur J Oper Res 2012;221(3):513–20.
[34] Song SH, Lee KS, Kim GS. A practical approach to solving a newspaper logistics problem using a digital map. Comput Ind Eng 2002;43(1-2):315–30.
[35] Subramanian A, Battarra M. An iterated local search algorithm for the travelling salesman problem with pickups and deliveries. J Oper Res Soc 2013;64(3):402–9.
[36] Subramanian A, Drummond L, Bentes C, Ochi L, Farias R. A parallel heuristic for the vehicle routing problem with simultaneous pickup and delivery. Comput Oper Res 2010;37(11):1899–911.
[37] Wilck IV J, Cavalier T. A construction heuristic for the split delivery vehicle routing problem. Am J Oper Res 2012;2(2):153–62.
[38] Wilck IV J, Cavalier T. A genetic algorithm for the split delivery vehicle routing problem. Am J Oper Res 2012;2(2):207–16.