J Heuristics
DOI 10.1007/s10732-008-9101-3

An adaptive memory algorithm for the split delivery vehicle routing problem

Rafael E. Aleman · Xinhui Zhang · Raymond R. Hill

Received: 26 November 2007 / Revised: 28 November 2008 / Accepted: 28 November 2008

Abstract The split delivery vehicle routing problem (SDVRP) relaxes routing restrictions forcing unique deliveries to customers and allows multiple vehicles to satisfy customer demand. Split deliveries are used to reduce total fleet cost to meet those customer demands. We provide a detailed survey of the SDVRP literature and define a new constructive algorithm for the SDVRP based on a novel concept called the route angle control measure. We extend this constructive approach to an iterative approach using adaptive memory concepts, and then add a variable neighborhood descent process. These three new approaches are compared to exact and heuristic approaches by solving the available SDVRP benchmark problem sets. Our approaches are found to compare favorably with existing approaches and we find 16 new best solutions for a recent 21 problem benchmark set.

Keywords Adaptive memory · Vehicle routing · Split delivery · Route angle control · Heuristic · Variable neighborhood descent

1 Introduction

The vehicle routing problem (VRP), or truck dispatching, was first formulated by Dantzig and Ramser (1959) and is a core problem in transportation, logistics, and

R.E. Aleman · X. Zhang
BIE Department, Wright State University, 3640 Colonel Glenn Hwy, Dayton, OH 45435, USA

R.E. Aleman
e-mail: rafael.enrique.aleman@gmail.com

X. Zhang
e-mail: xinhui.zhang@wright.edu

R.R. Hill (✉)
Air Force Institute of Technology, Wright-Patterson AFB, Dayton, OH 45433, USA
e-mail: rayrhill@gmail.com

Published online: 11 December 2008

supply chain management. The VRP involves a fleet of vehicles with fixed characteristics (i.e., speed, capacity, etc.) stationed at a central depot and a set of geographically scattered points (i.e., cities, warehouses, schools, customers, etc.) with fixed demands. Vehicles are used to visit and fully supply the demand of these points. The optimization problem is to determine which customers are visited by each vehicle and what route will the vehicle follow to serve those assigned customers, while minimizing the operational costs of the fleet, such as travel distance, gas consumption, and vehicle depreciation. Routes are designed to start and end at the depot, the demand of every customer is fully supplied by exactly one vehicle, and the total demand met by any route cannot exceed the vehicle capacity.

In reality, however, there may be cases where either a customer demand exceeds the vehicle capacity or a savings in terms of the total distance or the number of vehicles can be obtained by serving customers with more than one vehicle. The split delivery vehicle routing problem (SDVRP) relaxes the VRP restraints and allows the use of multiple vehicles to satisfy customer demand points and potentially reduce the total delivery cost by splitting customer deliveries among vehicles (Dror and Trudeau 1989). The computational complexity of the SDVRP remains NP-hard (Dror and Trudeau 1990). Archetti et al. (2005) define the k-SDVRP as a special case of the SDVRP where vehicles have a capacity of k units, $k \in \mathbb{Z}^+$. The Archetti et al. (2005) study shows that the 2-SDVRP is solvable in polynomial time when some specific conditions on the distances are satisfied, while the problem with $k \geq 3$ remains NP-hard. Archetti et al. (2005) also show that the 2-SDVRP may be reduced to a problem of possible smaller size, where each customer has unitary demand.

The SDVRP is defined on an undirected graph $G = (V, E)$ where $V = \{0, 1, \dots, n\}$ is the set of $n + 1$ nodes of the graph, and $E = \{(i, j) : i, j \in V,\ i < j\}$ is the set of edges connecting the nodes. Node 0 represents a depot where a fleet $M = \{1, \dots, m\}$ of identical vehicles with capacity Q are stationed, while the remaining node set $N = \{1, \dots, n\}$ represents the customers. A non-negative cost, usually a function of distance or travel time, c_{ij} is associated with every edge (i, j). Each customer $i \in N$ has a demand of q_i units. The optimization problem is to determine which customers are served by each vehicle and what route will the vehicle follow to serve those assigned customers, while minimizing the operational costs of the fleet, such as travel distance, gas consumption, and vehicle depreciation.

We propose a solution method that uses a constructive heuristic approach and a fixed number of vehicles to construct an initial solution by inserting unassigned customers sequentially into the solution under construction. A sequence is a list of customers in a specific order. When the solution is complete, the sequence of customers is modified based on the characteristics of the constructed solution. Once a new sequence of customers is determined, the constructive heuristic approach is executed again to find another solution. Again, the sequence of customers is modified and the procedure is repeated until no better solutions can be found. The best solution found during this iterative constructive approach is then improved using a variable neighborhood descent (VND) procedure. As far as we are aware, this is the first time a variable neighborhood search is used to solve the SDVRP.

This paper is organized as follows. Section 2 provides a review of the existing literature on SDVRP. In Sect. 3, we describe our algorithms. In Sect. 4, we present numerical experiments comparing our proposed algorithms with other existing methods found in the literature. Conclusions and future directions are provided in Sect. 5.

2 Literature review

2.1 Benefits of SDVRP

At first glance, one may believe that the benefits of allowing split deliveries are small when the customer demands are either considerably small with respect to the vehicle capacity or close to the vehicle capacity. In their experimental study, Dror and Trudeau (1989) showed that if customer demands are low relative to the vehicle capacity and the triangular inequality holds ($c_{ij} \leq c_{ik} + c_{kj}$, for all i, j, and k), the split demand benefits are actually very little. In contrast, when the customer demand is larger, at least 10% of the vehicle capacity, the cost of a SDVRP solution is considerably lower than the cost of a VRP solution. Figure 1 shows a simple example to illustrate the potential benefits of allowing split deliveries in terms of the number of vehicles and the solution cost. In this figure, there are 12 customers with same demand $q_i = 60$ symmetrically located on a circle of radius r centered at the depot, where a fleet of vehicles with capacity $Q = 100$ is located. The optimal VRP solution is illustrated in Fig. 1(a). It employs 12 vehicles with an average utilization of 60% and has a value of $z = 240r$ distance units. If split deliveries are allowed, all customer demands can be supplied as illustrated in Fig. 1(b) with only 8 vehicles with an average utilization of 90% and a solution value of $z = 201.4r$.

Archetti et al. (2006b) provide a worst-case analysis for the SDVRP and show that the savings in delivery costs that can be obtained by allowing split deliveries is at most 50% and that this bound is tight (i.e., there exists an example in which the value

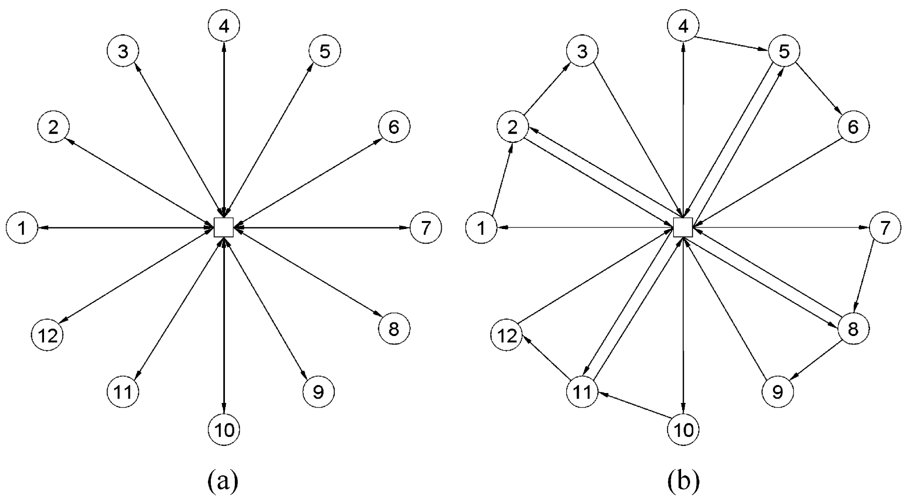


Fig. 1 Illustration of savings by SDVRP: (**a**) VRP solution and (**b**) SDVRP solution

of the optimal VRP solution doubles the value of the optimal SDVRP solution). This analysis, however, does not provide insight into the relation between the customer characteristics and the savings attained by allowing split deliveries.

Archetti et al. (2008) characterize distribution environments and conduct a thorough study (the most detailed found so far) of the value and benefits of allowing split deliveries. The focus of their study is to determine the practical implications of split deliveries for different customer characteristics, particularly in terms of the geographic and demand distribution of customers. The benefits are in: (1) the reduction in the number of routes, and thus vehicles, required to fully supply all customer demands and (2) the reduction in delivery costs.

Archetti et al. (2008) use a mathematical analysis to prove that the maximum reduction in the number of routes that can be achieved by allowing split deliveries is 50%. Moreover, their analysis confirms that the largest reduction can be obtained when the mean customer demand is between 50% and 70% of the vehicle capacity and the demand variances are relatively small. However, while there does not appear to be a dependence of delivery cost reductions on the geographic distribution of customers, there does appear to exist a dependence on the demand variance.

2.2 Existing SDVRP algorithms

Both exact and heuristic algorithms have been used to solve the SDVRP. Although exact algorithms solve instances to guaranteed optimality, they can be unpractical to use in solving large instances due to the computational costs involved. The largest SDVRP instance solved to optimality includes 50 customers (Belenguer et al. 2000).

There are some exact approaches found in the SDVRP literature. Dror et al. (1994) propose an integer linear programming formulation and describe a branch-and-bound algorithm based on new classes of valid inequalities for the SDVRP. Sierksma and Tijssen (1998) apply the SDVRP to building the transportation schedule of helicopters supporting offshore platforms in the North Sea for crew exchange of people employed on those platforms. They propose a set-covering formulation for the SDVRP and solve its relaxation using a simplex algorithm and a column generation technique that includes a Knapsack Problem and several TSPs.

Lee et al. (2006) propose a solution method for the multiple-vehicle routing problem with split pick-ups (mVRPSP) based on a deterministic dynamic program model and a shortest path search algorithm. Based on some properties of optimal solutions of the mVRPSP, they reformulate the original dynamic program to find an equivalent model with a finite action and state space without loss of optimality. The reduced model is associated with a directed network, which is then solved as a shortest path problem. The algorithm is used to solve small instances with 4, 5, and 7 suppliers and the optimal solution is obtained in all cases.

Jin et al. (2007) present an iterative exact method called two-stage approach with valid inequalities (TSVI) to find an optimal solution after a finite number of iterations for SDVRP instances with average customer demands greater than 10% of the vehicle capacity. They divide the problem into a clustering sub-problem and a traveling salesman problem for each vehicle. In a first stage, the clustering sub-problem optimally assigns customer demands to the vehicles without considering distance costs.

In a second stage, a traveling salesman problem is solved via a commercial optimization solver to find the minimal distance traveled by each vehicle. Those distances are added as cuts to the original clustering sub-problem. The process is repeated until no new clusters can be found in the first stage.

Other studies have estimated problem bounds. Belenguer et al. (2000) calculate lower bounds to optimal solutions of SDVRP instances based on a polyhedral study of the problem and a cutting-plane algorithm. The cutting-plane algorithm starts with an initial lower bound, which is calculated by solving the initial problem formulation via a linear programming code. Valid inequalities, or cuts, are developed, added to the formulation, and used to determine the feasibility of the solutions obtained by the algorithm. Any violated inequality is added to the initial formulation and the process is repeated to calculate a better bound. If no inequality violation is found in the new solution, the cutting-plane algorithm stops and provides a final lower bound. Jin et al. (2008) propose a column generation to find lower bounds and an iterative approach to obtain upper bounds for the SDVRP. The approaches are tested on 12 of the 25 instances used by Belenguer et al. (2000) containing large customer demands as the algorithm is not efficient to solve problems with small average customer demands. They suggest solving those instances as capacited vehicle routing problems (CVRPs) rather than SDVRPs. The column generation improves some of the bounds of Belenguer et al. (2000).

Heuristic algorithms are often desirable to solve larger SDVRP instances. Various approaches found in the literature include local, tabu, and scatter search, hybrid approaches, and memetic algorithms. Dror and Trudeau (1989) proposed a local search to solve the routing problem with split deliveries. Theirs is a two-stage algorithm that first constructs a feasible VRP solution and from this generates a feasible SDVRP solution if split deliveries improve the initial VRP solution. Split deliveries are incorporated into the solution by using a *2-split* interchange operator, which creates a neighborhood with all the possible alternatives that remove a demand point from a route and insert it into two other routes whose combined spare capacity is greater than or equal to the demand. A route addition routine may create new routes to try eliminating split deliveries as long as a reduction in the total routing cost is obtained. Frizzell and Giffin (1992) use grid network instead of euclidean distances in the SDVRP. They use a constructive approach that clusters adjacent customers and then allocates vehicles to the clusters until the unassigned demand of the cluster is less than the vehicle capacity. For each cluster, a nearest neighbor blocking is used to first assign the demand of the customers farther from the depot. In case the combined demands in the same cluster exceeds the vehicle capacity, the blocking mechanism produces distance savings as the demands to be split are the ones closer to the depot. Bouzaiene-Ayari et al. (1993) suggest an adaptation of the Clarke and Wright (1964) algorithm to solve the vehicle routing problem with stochastic demands and split deliveries. This study is apparently the first attempt to solve a stochastic vehicle routing problem with split deliveries.

In a second paper, Frizzell and Giffin (1995) incorporate time windows into the problem (SDVRPTW). The algorithm is similar to the one in Frizzell and Giffin (1992), but now customers are sorted according to the distance from the depot and their time windows. The initial solution is changed by moving a customer to alternate routes or by exchanging any two customers between their assigned routes when a saving in the objective function results. Mullaseril et al. (1997) describe a feed distribution problem encountered on a cattle ranch in Arizona. They study the problem of scheduling a fleet of trucks for the feed distribution in a large livestock range. The solution strategy for this problem is an adaptation of the algorithm of Dror and Trudeau (1989). Since the solution must consider time windows, the candidate list is pruned to those routes respecting the time windows constraint. To mitigate a potential reduction in the number of candidates, the *k-split* interchange operator uses $2 \leq k \leq M$, where M is the number of candidate routes generated, usually less than 10. Finally, a route addition improvement approach is used, but a check is done for capacity and time feasibility.

The tabu search by Ho and Haugland (2004) uses an operator called the relocate split operator. The algorithm starts with the construction of an initial solution by checking customers in a pre-defined sequence and appending the nearest un-routed customer to the latest routed customer. During the tabu search, the best candidate among four neighborhoods is selected at each iteration. They use standard operators (e.g., customer relocation, customer exchange, and $2 - \text{opt}^*$) adapted to the SDVRP context to potentially eliminate split deliveries. They also use the relocate split operator, which uses two routes with a shared delivery and relocates the delivery within the two routes subject to obtaining a reduction in the total distance. Archetti et al. (2006a) propose a tabu search where a customer is removed from a set of routes serving it and either inserted into a new route or into an existing route that has spare capacity. The tabu search uses a random tabu tenure selected from an interval defined by the number of customers and the number of routes in the current solution. An improvement phase is used after the tabu search in order to eliminate *k-split* cycles. Chen et al. (2007) develop a heuristic that combines a mixed integer program and a record-to-record travel algorithm that starts with an initial SDVRP solution based on the Clarke and Wright (1964) algorithm. For each route in the initial solution, a mixed integer program considers the endpoints and the closest neighbors to each endpoint to reallocate the demand of the endpoints and maximize the total savings. An endpoint is reallocated in three ways: (1) no change is made; (2) the endpoint is totally removed from its current route(s) and all of its demand is moved to other route(s); and (3) the endpoint is partially removed from its current route(s) and part of its demand is moved to other route(s). The heuristic is tested on the 49 problems of Archetti et al. (2006a), 5 random problems of Belenguer et al. (2000) with large customer demands, and 21 new benchmark problems and is shown to clearly outperform the algorithms of Archetti et al. (2006a).

Other studies covering the SDVRP include the work by Song et al. (2002) who adopt a split delivery scheme to find an allocation of newspaper agents and route vehicles to deliver newspapers while minimizing the delivery costs and reducing the total delay time of the delivery. Various algorithms were used and savings of 15% in the delivery costs and 40% in the delay time were obtained. Nowak (2005) examines a pickup and delivery routing problem with split loads and explore how costs can be reduced by eliminating the constraint that only one vehicle can service a customer. The problem is modeled as a dynamic program and the results show that most benefits with split loads occur when loads are at least half of vehicle capacity. Liu

(2005) proposes a two-stage algorithm with valid inequalities (Jin et al. 2007) and a branch-and-price approach to solve the problem. Wilck and Cavalier (2007) study a modified objective function to consider the impact of the loads in the operational costs and potentially reduce them. Yu et al. (2006) consider an inventory routing problem with split delivery and solve it using Lagrangian relaxation and linear programming. Belfiore and Yoshizaki (2006) implemented a scatter search to solve the problem involving other side constraints including heterogeneous vehicles, time windows, and accessibility constraints applied to a retail market in Brazil. The algorithm was applied to solve real scenarios arising on a daily basis and allowed to reduce the operational costs of the fleet compared to current practices in the company. Ambrosino and Sciomachen (2007) deal with a real application for food distribution in an Italian company. They model the problem as a generalization of the asymmetric VRP with split deliveries to determine an efficient distribution plan of fresh/dry and frozen food along the country. The solution algorithm includes a clustering procedure suitable tailored to the conditions of the real problem and a local search to move customers between routes and to split customer demands to improve the solution. Mota et al. (2007) present a scatter search that uses the minimum possible number of vehicles and performs favorably with respect to the tabu searches of Archetti et al. (2006a) in the tested problems, specially when the customer demands are below half the vehicle capacity. Tavakkoli-Moghaddam et al. (2007) present a simulated annealing method to solve the problem with heterogeneous vehicles and a new term in the objective function to maximize the utilization of the vehicle capacity. The algorithm was tested on randomly generated instances only and no benchmark problems were utilized. Boudia et al. (2007) implemented a memetic algorithm with population management that produces high quality solutions and low running times relative to the Splitabu approach of Archetti et al. (2006a).

3 Proposed SDVRP algorithm

3.1 Constructive heuristic approach (CA)

An initial SDVRP solution is obtained using a construction procedure that sorts the customers based on the distance from the depot c_{0j} and then creates new routes or modifies existing routes to allocate the customers. A list, L, of customers is created and sorted in descending order based on c_{0j}. Customers are then analyzed in sequence to determine the best way to include them in the solution, either by initiating a new route or by inserting them into existing routes. In the former case, new routes are initialized until a maximum number of routes is reached. Our approach utilizes the minimum fleet size required to satisfy the demand constraints. In contrast to the classical VRP where a bin packing problem is solved to find the number of vehicles required to supply all customer demands, any SDVRP instance can be solved using $m = \lceil \sum_{k \in N} q_k / Q \rceil$ vehicles, where $\lceil x \rceil$ is the lowest integer greater than or equal to x. Customers are inserted into existing routes at the cheapest insertion position.

During preliminary experiments, it was found that the insertion method can be improved by using a mechanism we call route angle control (RAC). This mechanism uses the angle formed by customers within routes to help determine the best way to allocate customers in the solution. We define the polar angle of a customer θ_k relative to the depot as:

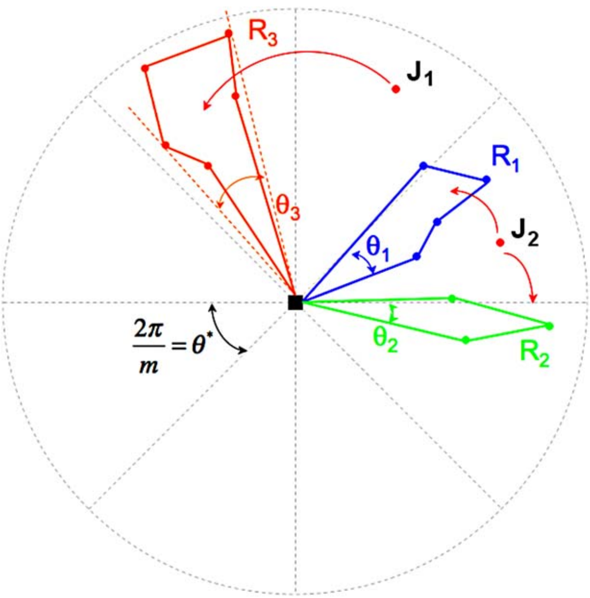
$$\theta_k = \arctan \frac{y_k - y_0}{x_k - x_0}, \tag{1}$$

where (x_k, y_k) represents the location of customer k and customer 0 represents the depot. We define the angle of a route, θ_r, as the maximum angle formed by the customers in the route, i.e., $\theta_r = \max\{\theta_i - \theta_j; \forall (i, j) \in r\}$. Initially, an existing route has an associated angle formed by the customers visited during the route. When a customer is inserted, this angle may either increase or remain constant if the inserted customer is geographically located between the customers in the route. If the increased angle exceeds a threshold angle value, a penalty cost is incurred. The penalty cost can be chosen arbitrarily as long as it makes the move prohibitive.

Inspired by the petal algorithm of Foster and Ryan (1976), the problem is partitioned so that routes serve sectors of the region centered at the depot. We sector the region as the number of sectors equal to the fleet size, m. Thus, sectors are equally distributed with an angle of value $\theta^* = \frac{2\pi}{m}$. This angle is used as the threshold angle value to penalize insertions that spread routes. In addition to the penalty cost, the RAC considers the route angle that results after inserting a customer in the route. This means that when a customer in L is evaluated for insertion in the solution, it may be located between two existing routes that can serve it without exceeding θ^*. In such a case, the RAC mechanism favors the route closest to the customer for servicing the customer.

Figure 2 illustrates the RAC. For this example, assume $m = 8$ so that $\theta^* = 2\pi/8 = \pi/4$. This figure illustrates two cases. In the first case, the insertion of customer j_1

Fig. 2 Route angle control



into R_3 would spread the route giving an angle exceeding θ^*; that insertion is penalized to favor the creation of a new route or the insertion into another route, such as R_1. In the second case, customer j_2 can be inserted either into R_1 or R_2 without incurring a penalty because both routes would remain within θ^*. In such a case, j_2 is inserted into the thiner route among R_1 and R_2. In any case, the insertion cost is proportional to the angle of the route after the insertion of the customer. The evaluation of the insertion candidates is described in the algorithm below.

Our constructive procedure is summarized as follows.

Notation:

- L List containing all customers $i \in N$
- u_i Unserved demand of customer i
- q_i Demand of customer i
- s_r Spare capacity of route r; $r \in M$
- θ_{ir} Angle of route r after inserting customer i

Step 1 (Sort the customers). Create L and sort the customers to produce $L = \{i_1, i_2, \ldots, i_n\}$ with $c_{0i_1} > c_{0i_2} > \cdots > c_{0i_n}$.

Step 2 Set i to the first customer in L and $u_i = q_i$

Step 3 (Insertion candidate). Find the cheapest way to insert customer i within existing feasible routes. The cost of inserting customer i into route r, c_{ir}, includes the distance cost and the value obtained by the route angle mechanism. Let i_b and i_a be the preceding and succeeding customers, respectively, in route r after inserting customer i. The insertion cost is then given by:

$$c_{ir} = c_{i_b i} + c_{i i_a} - c_{i_b i_a} + \alpha \times \theta_{ir} + \beta \times \max\left\{0, \frac{|\theta_{ir} - \theta^*|}{\theta_{ir} - \theta^*}\right\}, \tag{2}$$

where α represents a weight for the angle of route r after inserting customer i and β represents a penalty value incurred when the insertion of customer i produces a route angle θ_{ir} exceeding θ^*. The value of β can be any value large enough to favor the insertion of the customer in other routes. Special care has to be taken in Eq. (2) to avoid a division by zero when $\theta_{ir} = \theta^*$. The route r yielding to the lowest insertion cost is selected as the best insertion candidate.

Step 4 (Insert customer). If the cost of the candidate found in Step 3 is less than the cost of a returning route (this is, $c_{ir} < 2c_{0i}$), insert customer i into the cheapest insertion position of route r. Otherwise, initiate a new route with customer i.

Step 5 (Calculating the quantity). If $s_r \geq u_i$, then $s_r = s_r - u_i$ and $u_i = 0$. Otherwise, $s_r = 0$ and $u_i = u_i - s_r$ (i.e., split delivery occurs).

Step 6 (Optimize route). Using a local search, optimize route r by moving single customers to the cheapest position in the route. If all customers in L are fully supplied, go to Step 8.

Step 7 If $u_i = 0$, go to the next customer i in L and set $u_i = q_i$. Go to Step 3.

Step 8 Stop.

3.2 Iterative constructive approach (ICA)

A big advantage of constructive procedures is their ease of implementation. However, the initial SDVRP solutions for our constructive approach were found to be less than ideal due to the disadvantages of a constructive procedure. Constructive approaches perform moves with the best immediate benefit while ignoring the effects this can have in later stages of the search. When the solution construction starts, the best moves can be performed. As the search progresses, the number of good alternatives are reduced and the final moves usually have a negative impact on the quality of the final solution.

We base our iterative approach on the presumption that customers inserted in the later stages of the procedure are likely to most deteriorate the solution quality. Thus, those customers will have a higher contribution on the solution value than customers inserted earlier in the solution. The impact of those contributions influence the decisions made when constructing new solutions. As new solutions are constructed, customers with a history of high contributions are inserted into the solution earlier by changing the structure of L.

Figure 3 shows an example of the list L used by our iterative constructive approach to solve a benchmark instance involving 50 customers with vehicle capacity $Q = 160$. For simplicity, the figure only shows the customers assigned to routes R_1, R_4, and 5 of the customers assigned to route R_2. These routes are constructed by inserting sequentially the customers in L as follows. Let L_r be the node list containing the customers forming route R_r following the order in which the customer demands are assigned to the route. Thus, for instance, L_1 provides the order customers are placed into the route while R_1 provides the order those customers are visited. The construction of route R_1 commences with the assignment of the 6 demand units of customer 36 (position 1 in L_1), followed by the 17 demand units of customer 35 (position 2 in L_1), until all demands in L_1 are assigned to R_1. Note that customer 11 in the last position of L_1 is partially supplied by R_1 with only 5 demand units as the vehicle has no more capacity at the moment the demand of customer 11 is assigned to the route. In this particular instance, this demand is $q_{11} = 19$. The remaining 14 units are assigned to route R_2 whose construction is out of the scope of this example. Simultaneously, route R_4 is constructed by assigning the demands of customers 43, 31, 26, 7, 24, 8, 23, 48, 32, and 27. In contrast to R_1, all demands can be fully assigned to R_4 and the vehicle still has 20 units left (see spare Q of R_4 in figure). The process continues following the order in list L until all customer demands are assigned and the solution is complete. In Fig. 3 we provide L_1 and L_4 explicitly and show R_1 and R_4 graphically.

The customers in L are sorted based on the distance to the depot, so that customers located closer to the depot are inserted into the solution later in the process. Preliminary experiments found that customers near the depot caused route angles to increase since preferred routes lacked capacity to support the customer insertion. Such angle spreading customers need to be inserted earlier so need to be placed earlier in L. Temporarily removing a customer from a route changes the angle of that route, a value denoted as $\Delta\theta_r$. Let customer i^* have the largest $\Delta\theta_r$, i.e., the customer that most deteriorates the solution. This customer i^* is re-positioned in L to ensure its earlier consideration in the constructive approach.

List L of sorted customers

Position in L	1	2	3	4	5	6	7	8	9	...	16	17
Customer	36	40	35	39	43	33	3	20	21	...	28	31
Position in L	18	19	20	21	22	23	24	...	29	30	31	32
Customer	13	29	10	26	7	50	24	...	8	16	23	49
Position in L	33	34	...	40	...	43	44	...	46	...	49	50
Customer	2	22	...	48	...	1	11	...	**32**	...	27	46

List L_1 of sorted customers forming R_1

Position in L_1	1	2	3	4	5	6	7	8	9	10	11	12
Customer	36	35	3	20	21	28	29	16	2	22	1	11
Delivery	6	17	16	28	8	14	6	15	30	8	7	5
Spare Q of R_1	154	137	121	93	85	71	65	50	20	12	5	0

List L_4 of sorted customers forming R_4

Position in L_4	1	2	3	4	5	6	7	8	9	10	11	12
Customer	43	31	26	7	24	8	23	48	**32**	27		
Delivery	11	11	7	19	10	23	16	17	11	15		
Spare Q of R_4	149	138	131	112	102	79	63	46	35	20		

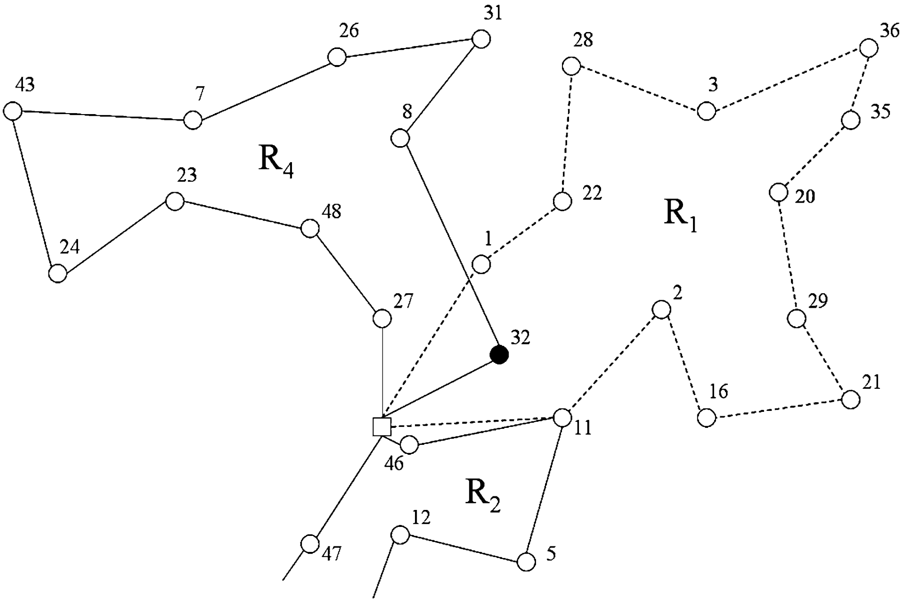


Fig. 3 Example of partial solution to SDVRP, based on a sequential list of customers, L

To relocate i^* in L, we first find a route, r^* closest to i^* such that i^* could be inserted at lowest cost if vehicle capacity were sufficient. The node list of this route, L_{r^*}, is then examined to determine where to place i^* so that it can fit within the vehicle capacity. This position is usually earlier in the L_{r^*} sequence of customers. Designate as i_a the customer that immediately succeeds this insertion point. In other words, the customers in L_{r^*} prior to i_a leave enough spare capacity to accommodate the i^* demand. To obtain L_{new} from L, relocate i^* to occur immediately before i_a in L. This new sequential list of customers, L_{new}, is then used to create a new solution to the SDVRP.

In the example illustrated in Fig. 3, the customer with the highest $\Delta\theta_r$ is shown in bold text in L, $i^* = 32$. Given the design of the routes R_1 and R_4, the insertion of customer 32 into route R_1 seems more reasonable than into route R_4 where it was placed. Based on list L_1, $i_a = 1$ is identified as the last customer in L that spends the capacity of route R_1 necessary to service customer 32. Thus, customer 32 is relocated right before customer 1 in L. This relocation modifies L to produce the list L_{new} shown in Fig. 4.

List L_{new} in Fig. 4 is used to execute the constructive heuristic approach again and produce a new solution, as illustrated. The relocation produced the expected insertion of customer 32 into R_1 plus other perturbations to the solution; since R_1 cannot fully service customer 1 (see spare Q of R_1 after this insertion), this delivery is split among R_1 and R_4. The deliveries of R_1 and R_4 to customer 1 are then 1 and 6 units, respectively, as provided by the new L_1 and L_4 in the figure. The relocation of customer 32 in L produced a reduction in the objective function value of the complete solution from $z = 578.83$ in Fig. 3 to $z = 577.92$ in Fig. 4. Note that this relocation produced the transfer of customer 32 from R_4 to R_1 (similar to the standard customer shift used in multi-route improvement algorithms) plus the relocation of a split delivery in the solution (similar to the relocate split operator of Ho and Haugland 2004). This iterative construction of new solutions continues until no improvements to the best solution found in the search are obtained after a predefined number of consecutive iterations.

3.3 Variable neighborhood descent (VND)

Variable neighborhood search (VNS) is a relatively new meta-heuristic concept based on the principle of systematically changing the neighborhood structure during the search to escape from local optima. This meta-heuristic first appears in the literature in the study of Mladenović and Hansen (1997) where this scheme is shown to outperform other heuristics on the traveling salesman problem. Given N_k ($k = 1, \ldots, k_{\max}$), a finite set of pre-selected neighborhood structures and $N_k(x)$ the set of solutions in the kth neighborhood of x, neighborhoods N_k may be induced from one or more metric functions.

Some variable neighborhood searches have been applied to routing problems. Bräysy (2003) proposes a reactive variable neighborhood search that modifies some parameters and changes the objective function to avoid local optimality. The method is applied successfully to the VRPTW and provided four new best-solutions for the test problems used. Polacek et al. (2004) use variable neighborhood search to solve the multi-depot VRPTW (MDVRPTW). The algorithm outperforms a tabu search, found 10 new best-solutions, and demonstrated superiority on large real-world problems. Kytöjoki et al. (2007) use a variable neighborhood descent to solve large-scale VRPs and accept non-improving solutions by penalizing certain solution features. High quality solutions are found for problems involving up to 20,000 customers.

List L_{new} of sorted customers

Position in L	1	2	3	4	5	6	7	8	9	...	16	17
Customer	36	40	35	39	43	33	3	20	21	...	28	31
Position in L	18	19	20	21	22	23	24	...	29	30	31	32
Customer	13	29	10	26	7	50	24	...	8	16	23	49
Position in L	33	34	...	40	...	43	44	45	...	...	49	50
Customer	2	22	...	48	...	**32**	1	11	...	...	27	46

New list L_1 of sorted customers forming R_1

Position in L_1	1	2	3	4	5	6	7	8	9	10	11	12
Customer	36	35	3	20	21	28	29	16	2	22	**32**	1
Delivery	6	17	16	28	8	14	6	15	30	8	11	1
Spare Q of R_1	154	137	121	93	85	71	65	50	20	12	1	0

New list L_4 of sorted customers forming R_4

Position in L_4	1	2	3	4	5	6	7	8	9	10	11	12
Customer	43	31	26	7	24	8	23	48	1	27	46	
Delivery	11	11	7	19	10	23	16	17	6	15	5	
Spare Q of R_4	149	138	131	112	102	79	63	46	40	25	20	

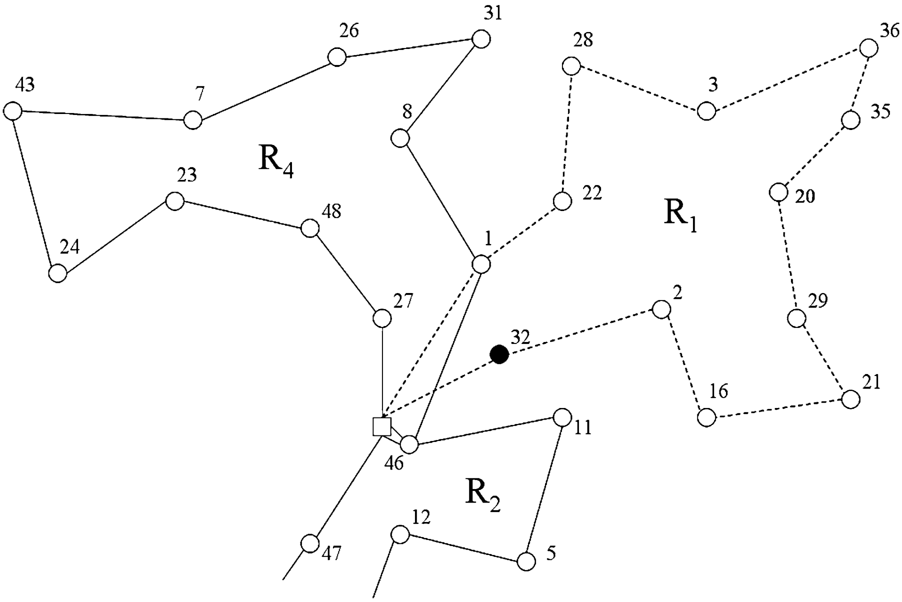


Fig. 4 Example of iterated solution to SDVRP based on solution information from Fig. 3 solution

In Variable Neighborhood Descent (VND), the final solution is a local optimum with respect to all neighborhoods N_k, and thus the chances of finding a global optimum are higher than by using a single neighborhood structure. Our VND is defined as follows:

Step 1 Define the set of neighborhood structures to be used. Set $k_{\max}$ equal to the number of such structures.

Step 2 Find an initial solution x.
Step 3 Set $k = 1$.
Step 4 Find the first improving neighbor x' of x, $x' \in N_k(x)$.
Step 5 If a neighbor x' was found, set $x = x'$ and go to Step 4. Otherwise, set $k = k + 1$.
Step 6 If $k > k_{\max}$ and there were no improvements since $k = 1$, stop.
Step 7 If $k > k_{\max}$ and the solution was improved with any $N_k : k > 1$, go to Step 3. Otherwise, go to Step 4.

3.3.1 Neighborhood structure

Three neighborhoods, N_1, N_2, and N_3, are used in our study and are described below. The first two neighborhoods are based on the well known customer *shift* and customer *swap*, respectively, to move and exchange customers between routes. These operators are adapted from Ho and Haugland (2004) to handle split deliveries. The third neighborhood is based on a new operator, called customer *shift**, that introduces a split delivery when a customer shift is infeasible due to a lack of vehicle capacity in the destination route.

The operators are illustrated in Figs. 5 to 7. In these figures, the depot is graphically represented by a white square, routes R_1 and R_2 represent any two existing routes in the current solution, solid arrows represent the sequence of customers within R_1 and dashed arrows represent the sequence of customers within R_2. Customers are represented by a circle with shading used to differentiate the customers; customers

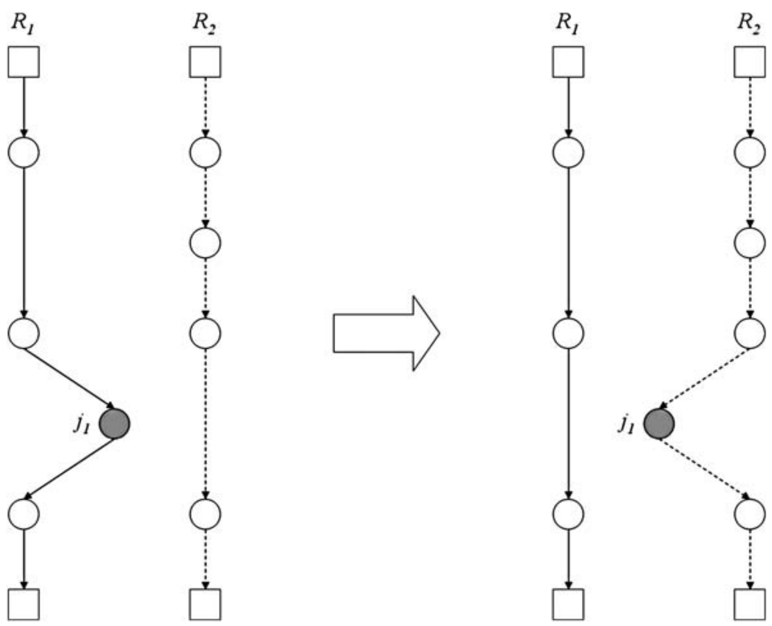


Fig. 5 Illustration of the shift operator

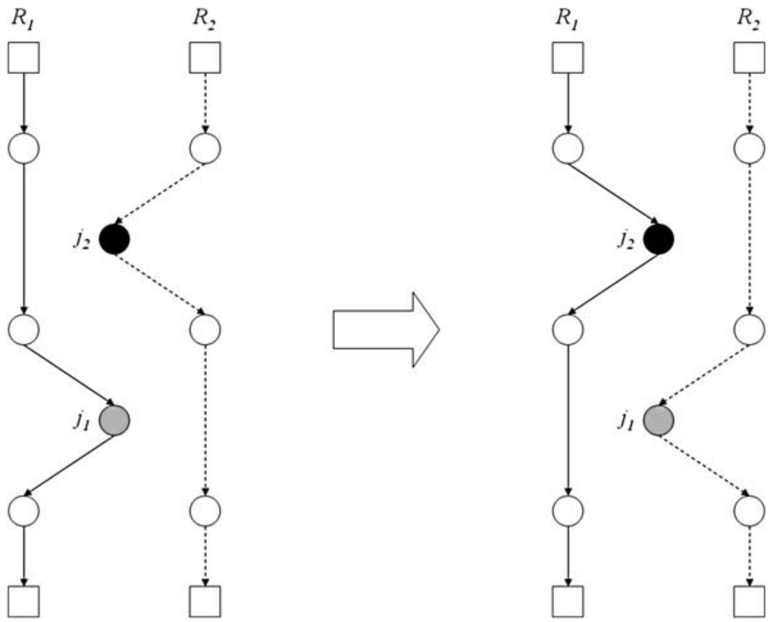


Fig. 6 Illustration of the swap operator

not involved in the transformation and remaining in their original routes are white, customers originally within R_1 and moved to R_2 are gray, and customers originally within R_2 and moved to R_1 are black. Finally, and for the purpose of the operator's description, the spare capacity of route i is denoted by s_i, the demand of customer j is denoted q_j, and the delivery made by route i to customer j is denoted by y_{ij}.

- The customer *shift* is illustrated in Fig. 5. This operator moves customer $j_1 \in R_1$ to the cheapest position in R_2. If j_1 is split among R_1 and R_2, it is removed from R_1 and the quantity y_{2j_1} is increased to q_{j_1}. This move is feasible when $s_2 \geq y_{1j_1}$.
- The customer *swap* is illustrated in Fig. 6. This operator exchanges two customers by removing $j_1 \in R_1$ and $j_2 \in R_2$ and inserting them into the cheapest positions in R_2 and R_1, respectively. If j_2 is split among R_1 and R_2, it is removed from R_2 and the quantity y_{1j_2} is increased to q_{j_2}. This move is feasible when $s_2 + y_{2j_2} \geq q_{j_1}$ and $s_1 + q_{j_1} \geq y_{2j_2}$.
- The customer *shift** is illustrated in Fig. 7. This operator is a variant of the standard customer *shift* and moves customer $j_1 \in R_1$ to the cheapest position in R_2 and inserts a partial delivery of customer $j_2 \in R_2$ into the cheapest position in R_1. This move is feasible when $s_2 < q_{j_1}$ and $q_{j_1} - s_2 < q_{j_2}$. In other words, the move is feasible when R_2 does not have enough capacity to service j_1 and the demand q_{j_2} of customer $j_2 \in R_2$ is large enough to cover the lack of capacity in R_2. As this transformation allows both R_1 and R_2 to service j_2, the quantities y_{1j_2} and y_{2j_2} are possibly adjusted to avoid any route exhausting the entire capacity. This adjustment helps increase the number of feasible candidates that may be found later in the search.

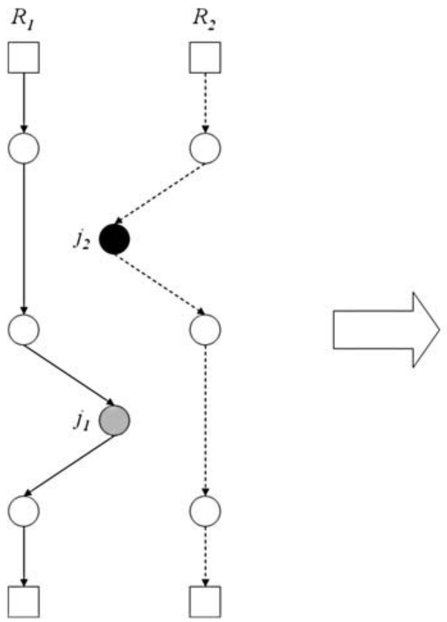


Fig. 7 Illustration of the *shift** operator

4 Experimental results

Our algorithms were tested on existing problem sets available in the literature: Archetti et al. (2006a), Belenguer et al. (2000), Chen et al. (2007), and Jin et al. (2007). These problems total 96. The instances from Archetti et al. (2006a) include the problems with both original and randomly generated customer demands. These instances involve 50, 75, 100, 120, 150, and 199 customers and their demands are randomly generated as a function of the vehicle capacity to test the performance of the algorithms of Archetti, Hertz, and Speranza when the customer demands get large, as in Dror and Trudeau (1989). Instances 1–5 have customers uniformly located around the depot, whereas instances 6 and 7 have clustered customers. In contrast to problem 7, in problem 6 the depot is not centered with respect to the customer locations. The description of these instances can be found in Christofides and Eilon (1969) and Christofides et al. (1979). It is important to mention that the instances with random demands generated by Archetti et al. (2006a) are unavailable. Instances from Boudia et al. (2007) are the same instances tested and generated by Mota et al. (2007) with the generator of Archetti, Hertz, and Speranza. When required, special notes are used at the bottom of our tables to differentiate the instances used by the authors in their corresponding publications. The instances used by Belenguer et al. (2000) include 11 problems taken from TSPLIB and another 14 problems created by randomly generating the customer demands also as a function of the vehicle capacity. These instances involve 21 to 100 customers. Chen et al. (2007) recently generated a set of 21 problems involving 8 to 288 customers having a geometric symmetry, a star shape, with the customers located in concentric circles around the depot. Jin et al. (2007) used

Table 1 Test problems naming scheme

Value first naming field p	Publication
a	Archetti et al. (2006a)
b	Belenguer et al. (2000)
c	Chen et al. (2007)
j	Jin et al. (2007)

one TSPLIB instance involving 21 customers and generated four instances with 18, 21, and 22 customers.

In our analysis, the notation `p-aaa-nnn` is used to name each instance, where `p` is an alphabetical character to identify the publication where the problem is given (see Table 1), `aaa` is a string of variable length corresponding to the name of the instance adopted on the publication, and the third naming field is a three-digit integer denoting the number of customers. For instance, a-01-050 corresponds to problem 1 from Archetti et al. (2006a) with 50 customers. The algorithms were implemented in `C#` and the experiments were carried out on a PC Pentium 4, 2.8 GHz processor, and 512 MB of RAM. During our experiments, parameters α and β in Eq. 2 were set to 200 and 1,000,000, respectively. These values were empirically found to work reasonably well for the tested problems.

In this paper, we propose a constructive approach with a RAC mechanism to best allocate customers to routes. To test the performance of RAC, we construct initial solutions with and without this mechanism and calculate the savings obtained in the total traveled distance when the RAC is actually used. Except on one case, i.e., problem b-eil33-032, where the depot is not geographically centered with respect to the customer locations, the RAC obtains better solutions. On average, RAC savings are 32.84% on the 96 tested problems. However, these savings were found to vary with two problem characteristics: the geographical distribution of customers and the customer demand range. The RAC does well on problems with a centered depot, less well on problems where the depot is not centered. We classified the tested problems as GC when the depot is geographically centered with respect to the customers (e.g., problems a-01-050 and a-07-100), non-GC (NGC) when the depot is not geographically centered (e.g., problems a-06-120 and b-eil33-032), random (R) when the customers are randomly scattered (e.g., problems a-01-050 and b-eil33-032), or clustered (C) when customers form clusters (e.g., problems a-06-120 and a-07-100). Table 2 shows the percentage savings obtained when the RAC is used within the constructive approach for each problem type. Note that RAC is more effective when the depot is centered, i.e., problem types GC-R and GC-C. Problems with clustered or scattered customers do not seem to affect the performance of the proposed angle control measure.

Table 3 shows the percentage savings due to the RAC for different demand ranges, $D1$ to $D6$. These ranges are used by Belenguer et al. (2000) and Archetti et al. (2006a) with $D1 = [0.01 - 0.10]$ and $D6 = [0.70 - 0.90]$. The results in the table reveal that the RAC produces solutions of considerable less quality to problems with low customer demands in the range $D1$. The reason for this is that less vehicles

Table 2 Impact of problem type on the benefits of the route angle control

Problem type	Tested problems	Savings due to RAC		
		Min	Avg	Max
GC-R	77	10.76	35.16	46.94
GC-C	11	27.85	37.33	46.92
NGC-R	1	0.00	12.83	26.19
NGC-C	7	8.49	13.97	17.86

Table 3 Impact of demand range on the benefits of the route angle control

Demand range	Tested problems	Savings due to RAC		
		Min	Avg	Max
$D1$	10	8.49	21.83	37.66
$D2$	10	17.15	41.22	46.85
$D3$	10	17.86	39.56	45.38
$D4$	9	17.49	34.79	39.28
$D5$	9	14.75	35.90	41.72
$D6$	8	13.07	31.20	35.28

are required to satisfy the capacity constraints so the threshold angle $\theta^* = \frac{2\pi}{m}$ gets larger. As a consequence, the term $\beta \times \max\{0, \frac{|\theta_{ir}-\theta^*|}{\theta_{ir}-\theta^*}\}$ in Eq. 2 becomes zero for most insertion candidates so no penalties are applied to insertions into far routes.

Tables 4 to 6 show the solution values obtained with our constructive approach (CA), iterative constructive approach (ICA), and variable neighborhood descent along with the percentage improvement with respect to the CA solution value and the running times. The VND was tested using the solutions obtained with the CA and ICA approaches as the initial solutions. However, using the ICA as the initial solution (ICA + VND in the table) produces better results in 78% of the tested problems. As we see in Tables 4 and 5, although the ICA approach outperforms CA in 35 of the 63 problems, the improvements occur mainly in problems with small customer demands. As demands get larger, ICA does not improve the solution found with CA. The ICA uses a very conservative scheme to improve the search based on the information of constructed solutions. Thus, it is difficult to find solutions of lower value by simply re-positioning a single customer in the list L when the customer demands are large. This fact is reinforced by the results on the Chen et al. (2007) problem set presented in Table 6. In this set, customer demands are 60 and 90, the vehicle capacity is 100, and the ratio of the total demand and total capacity is 1 for all problems, which means that all vehicles are fully loaded in the final solution. ICA was able to slightly improve CA in only problem c-SD15-144. In contrast, the ICA + VND approach has a stronger neighborhood structure and a more aggressive search process so the improvements over CA are noticeably higher.

Our results on the instances from Archetti et al. (2006a) are compared with those obtained by the best among the three tabu searches of Archetti et al. (2006b) (Splitabu-DT), the scatter search (SS) of Mota et al. (2007), the hybrid approach (EMIP + VRTR) of Chen et al. (2007), and the memetic algorithm with population

Table 4 Computational results on problems of Archetti et al. (2006b)

Problem	Demand	CA		ICA			ICA + VND		
		z	CPU	z	IMP	CPU	z	IMP	CPU
a-01-050		578.83	0.08	568.67	1.76	2.69	540.82	6.57	10.89
a-02-075		899.11	0.06	889.05	1.12	3.25	880.28	2.09	9.81
a-03-100		873.46	0.16	863.18	1.18	9.34	854.13	2.21	43.50
a-04-150		1121.33	0.33	1108.97	1.10	20.77	1088.91	2.89	129.23
a-05-199		1412.18	0.55	1412.18	–	26.66	1390.55	1.53	534.83
a-06-120		1257.48	0.41	1257.48	–	20.27	1223.28	2.72	257.30
a-07-100		827.59	0.11	826.03	0.19	6.20	824.82	0.33	21.02
a-01-050	[0.01–0.10]	477.66	0.09	477.66	–	4.42	473.22	0.93	4.52
a-02-075	[0.01–0.10]	638.10	0.19	628.30	1.53	11.14	617.65	3.20	51.28
a-03-100	[0.01–0.10]	864.06	0.42	845.95	2.10	27.34	789.16	8.67	415.47
a-04-150	[0.01–0.10]	907.36	0.56	902.48	0.54	45.80	893.49	1.53	666.20
a-05-199	[0.01–0.10]	1163.38	1.06	1126.78	3.15	181.28	1079.04	7.25	3750.44
a-06-120	[0.01–0.10]	1175.43	0.53	1169.57	0.50	28.66	1101.14	6.32	341.59
a-07-100	[0.01–0.10]	706.74	0.48	687.12	2.78	28.72	673.54	4.70	222.42
a-01-050	[0.10–0.30]	787.03	0.02	777.75	1.18	1.09	777.75	1.18	1.59
a-02-075	[0.10–0.30]	1157.90	0.05	1152.97	0.43	2.36	1099.47	5.05	13.19
a-03-100	[0.10–0.30]	1513.33	0.08	1512.37	0.06	3.97	1452.52	4.02	34.09
a-04-150	[0.10–0.30]	2124.89	0.19	2124.89	–	9.30	1978.01	6.91	164.19
a-05-199	[0.10–0.30]	2584.94	0.33	2584.94	–	16.45	2502.54	3.19	248.83
a-06-120	[0.10–0.30]	2996.54	0.13	2979.88	0.56	6.28	2806.92	6.33	54.25
a-07-100	[0.10–0.30]	1555.18	0.09	1490.76	4.14	4.56	1428.27	8.16	22.56
a-01-050	[0.10–0.50]	1098.88	0.03	1098.88	–	1.06	1045.93	4.82	2.81
a-02-075	[0.10–0.50]	1574.85	0.05	1529.71	2.87	2.77	1503.02	4.56	11.25
a-03-100	[0.10–0.50]	2029.21	0.09	2015.64	0.67	4.41	1957.55	3.53	25.16
a-04-150	[0.10–0.50]	2774.54	0.17	2774.54	–	9.06	2685.33	3.22	111.66
a-05-199	[0.10–0.50]	3615.66	0.33	3615.66	–	16.20	3450.84	4.56	339.36
a-06-120	[0.10–0.50]	4212.58	0.13	4212.58	–	6.31	4085.36	3.02	40.53
a-07-100	[0.10–0.50]	2108.74	0.08	2078.99	1.41	4.69	2046.15	2.97	11.92
a-01-050	[0.10–0.90]	1604.25	0.03	1602.49	0.11	1.38	1547.32	3.55	2.83
a-02-075	[0.10–0.90]	2367.26	0.06	2348.75	0.78	3.36	2212.93	6.52	10.80
a-03-100	[0.10–0.90]	3026.61	0.09	3018.59	0.27	5.08	2925.13	3.35	19.00
a-04-150	[0.10–0.90]	4395.14	0.22	4395.14	–	10.89	4192.50	4.61	141.27
a-05-199	[0.10–0.90]	5559.98	0.38	5545.57	0.26	28.06	5192.06	6.62	662.77
a-06-120	[0.10–0.90]	6541.28	0.16	6541.28	–	7.22	6483.06	0.89	42.00
a-07-100	[0.10–0.90]	3295.74	0.11	3269.30	0.80	4.88	3178.28	3.56	12.89
a-01-050	[0.30–0.70]	1605.45	0.03	1591.06	0.90	1.55	1557.52	2.99	2.20
a-02-075	[0.30–0.70]	2348.60	0.06	2348.60	–	2.97	2241.59	4.56	11.28
a-03-100	[0.30–0.70]	3068.10	0.11	3057.85	0.33	4.78	2945.19	4.01	15.14

Table 4 (*Continued*)

Problem	Demand	CA		ICA			ICA + VND		
		z	CPU	z	IMP	CPU	z	IMP	CPU
a-04-150	[0.30–0.70]	4395.14	0.22	4395.14	–	10.89	4192.50	4.61	143.05
a-05-199	[0.30–0.70]	5680.50	0.45	5680.50	–	18.92	5366.06	5.54	349.97
a-06-120	[0.30–0.70]	6826.12	0.14	6814.97	0.16	7.31	6591.40	3.44	59.20
a-07-100	[0.30–0.70]	3402.47	0.09	3348.16	1.60	5.03	3318.08	2.48	13.69
a-01-050	[0.70–0.90]	2246.82	0.03	2246.82	–	1.63	2215.34	1.40	2.59
a-02-075	[0.70–0.90]	3400.01	0.09	3400.01	–	3.92	3341.26	1.73	10.25
a-03-100	[0.70–0.90]	4526.40	0.13	4526.40	–	6.61	4455.14	1.57	14.31
a-04-150	[0.70–0.90]	6665.56	0.31	6665.56	–	15.34	6513.36	2.28	93.78
a-05-199	[0.70–0.90]	8692.00	0.56	8662.98	0.33	32.63	8368.35	3.72	460.89
a-06-120	[0.70–0.90]	10585.01	0.20	10585.01	–	9.77	10302.16	2.67	59.28
a-07-100	[0.70–0.90]	5196.44	0.13	5196.44	–	6.75	5058.76	2.65	20.70

z denotes objective function value obtained

IMP denotes percentage objective function improvement over CA

CPU denotes running time in seconds on a P4, 2.8 GHz, 512 MB

Table 5 Computational results on the random problems of Belenguer et al. (2000)

Problem	Demand	CA		ICA			ICA + VND		
		z	CPU	z	IMP	CPU	z	IMP	CPU
b-S51D1-050	[0.01–0.10]	477.66	0.09	477.66	–	4.59	473.22	0.93	4.53
b-S51D2-050	[0.10–0.30]	759.56	0.03	745.46	1.86	1.27	732.38	3.58	4.05
b-S51D3-050	[0.10–0.50]	1034.90	0.02	1034.90	–	0.98	1001.22	3.25	2.50
b-S51D4-050	[0.10–0.90]	1740.38	0.03	1740.38	–	1.38	1708.00	1.86	2.89
b-S51D5-050	[0.30–0.70]	1421.74	0.03	1421.74	–	1.16	1404.54	1.21	1.80
b-S51D6-050	[0.70–0.90]	2266.58	0.03	2266.58	–	1.73	2230.06	1.61	2.27
b-S76D1-075	[0.01–0.10]	642.18	0.22	626.72	2.41	21.91	610.23	4.98	63.55
b-S76D2-075	[0.10–0.30]	1199.42	0.06	1196.42	0.25	2.48	1169.80	2.47	7.73
b-S76D3-075	[0.10–0.50]	1584.35	0.05	1584.35	–	2.25	1490.08	5.95	12.23
b-S76D4-075	[0.10–0.90]	2326.64	0.06	2326.64	–	2.73	2220.87	4.55	6.91
b-S101D1-100	[0.01–0.10]	854.05	0.41	831.64	2.62	47.45	765.48	10.37	210.36
b-S101D2-100	[0.10–0.30]	1510.85	0.09	1510.85	–	4.36	1444.96	4.36	26.20
b-S101D3-100	[0.10–0.50]	2167.71	0.08	2144.46	1.07	4.23	1990.28	8.19	27.84
b-S101D5-100	[0.30–0.70]	3062.17	0.09	3046.95	0.50	6.27	2999.31	2.05	18.36

z denotes objective function value obtained

IMP denotes percentage objective function improvement over CA

CPU denotes running time in seconds on a P4, 2.8 GHz, 512 MB

Table 6 Computational results for problems of Chen et al. (2007)

Problem	CA		ICA			ICA + VND		
	z	CPU	z	IMP	CPU	z	IMP	CPU
c-SD01-008	25478.71	0.02	25478.71	–	0.06	22828.43	10.40	0.06
c-SD02-016	73478.71	0.00	73478.71	–	0.19	70828.43	3.61	0.22
c-SD03-016	43058.22	0.02	43058.22	–	0.16	43058.22	–	0.17
c-SD04-024	70448.03	0.00	70448.03	–	0.36	63583.51	9.74	0.55
c-SD05-032	139056.83	0.02	139056.83	–	0.61	139056.83	–	0.69
c-SD06-032	85288.45	0.03	85288.45	–	0.81	83124.14	2.54	0.94
c-SD07-040	364000.00	0.02	364000.00	–	0.94	364000.00	–	1.03
c-SD08-048	509478.71	0.03	509478.71	–	1.36	506828.43	0.52	1.75
c-SD09-048	213794.48	0.03	213794.48	–	1.45	207102.79	3.13	2.91
c-SD10-064	277291.42	0.06	277291.42	–	2.52	274783.08	0.90	3.58
c-SD11-080	1328000.01	0.08	1328000.01	–	3.56	1328000.01	–	3.97
c-SD12-080	727997.00	0.06	727997.00	–	3.59	727997.00	–	4.00
c-SD13-096	1011057.51	0.09	1011057.51	–	5.25	1011057.51	–	5.80
c-SD14-120	1092000.85	0.16	1092000.85	–	7.84	1089349.80	0.24	15.49
c-SD15-144	1522449.07	0.23	1522342.27	0.01	11.75	1516827.58	0.37	18.33
c-SD16-144	375542.10	0.25	375542.10	–	11.77	363526.95	3.20	39.71
c-SD17-160	2655992.75	0.28	2655992.75	–	14.16	2655992.75	0.00	17.42
c-SD18-160	1455999.62	0.28	1455999.62	–	13.88	1444059.28	0.82	40.38
c-SD19-192	2021283.59	0.39	2021283.59	–	20.39	2019119.29	0.11	27.64
c-SD20-240	3983999.63	0.63	3983999.63	–	32.25	3981348.58	0.07	63.18
c-SD21-288	1244552.35	1.05	1244552.35	–	52.90	1179960.15	5.19	738.49

z denotes objective function value obtained

IMP denotes percentage objective function improvement over CA

CPU denotes running time in seconds on a P4, 2.8 GHz, 512 MB

management (MA|PM) of Boudia et al. (2007). Since the tabu search has random elements, Archetti et al. (2006a) ran each problem five times. Thus, their average values are provided in our analysis. Similarly, values reproduced from Chen et al. (2007) correspond to median solution values from 30 different instances for each problem. The computational results of our best approach, ICA + VND, are presented in Tables 7 and 8. Solution values z and percentage improvements IMP of the other approaches over the objective function value of the ICA + VND solution are shown in Table 7. Improvements in bold font denote the cases where the ICA + VND solution has a better value. EMIP + VRTR was not tested on problems a-03-100 so these values are omitted in our analysis. The results are grouped according to the actual instances used in the experiments. The instances used in our experiments have the same customer demands used to test SS and MA|PM. However, we do not have evidence to support the equality of our customer demands with the other instances (Archetti et al. 2006a; Chen et al. 2007).

Table 7 Computational results of ICA + VND on instances of Archetti et al. (2006b)

Problem	Demand	m	ICA + VND[a]	SS[a]		MA|PM[a]		m'	Splitabu-DT[b]		EMIP + VRTR[c]	
			z	z	IMP	z	IMP		z	IMP	z	IMP
a-01-050		5	540.82	531.02	1.81	524.61	3.00	5	533.55	1.34	524.61	3.00
a-02-075		10	880.28	839.75	4.60	823.89	6.41	10	849.54	3.49	840.18	4.56
a-03-100		8	854.13	835.82	2.14	829.44	2.89	8	835.62	2.17	–	–
a-04-150		12	1088.91	1056.92	2.94	1042.37	4.27	12	1069.84	1.75	1041.99	4.31
a-05-199		16	1390.55	1340.44	3.60	1311.59	5.68	16	1342.85	3.43	1307.40	5.98
a-06-120		7	1223.28	1042.97	14.74	1041.20	14.88	7	1056.01	13.67	1043.18	14.72
a-07-100		10	824.82	820.92	0.47	819.56	0.64		825.32	**−0.06**	819.56	0.64
Average					4.33		5.40			3.69		5.53
a-01-050	[0.01–0.10]	3	473.22	460.79	2.63	460.79	2.63		463.76	2.00	457.21	3.38
a-02-075	[0.01–0.10]	4	617.65	602.67	2.43	600.06	2.85		605.24	2.01	598.25	3.14
a-03-100	[0.01–0.10]	5	789.16	729.67	7.54	726.81	7.90		752.20	4.68	–	–
a-04-150	[0.01–0.10]	8	893.49	883.05	1.17	875.61	2.00		890.95	0.28	875.16	2.05
a-05-199	[0.01–0.10]	10	1079.04	1039.51	3.66	1018.71	5.59		1056.27	2.11	1040.20	3.60
a-06-120	[0.01–0.10]	6	1101.14	979.57	11.04	976.57	11.31		1084.70	1.49	985.17	10.53
a-07-100	[0.01–0.10]	5	673.54	633.80	5.90	649.73	3.53		648.74	3.68	651.44	3.28
Average					4.91		5.12			2.32		4.33
a-01-050	[0.10–0.30]	**10**	777.75	769.60	1.05	751.41	3.39	11	761.40	2.10	723.57	6.97
a-02-075	[0.10–0.30]	**15**	1099.47	1074.01	2.32	1074.46	2.27	16	1095.32	0.38	1081.10	1.67
a-03-100	[0.10–0.30]	**20**	1452.52	1416.48	2.48	1392.85	4.11	22	1424.81	1.91	–	–
a-04-150	[0.10–0.30]	**29**	1978.01	1974.70	0.17	1878.71	5.02	32	1918.25	3.02	1844.96	6.73
a-05-199	[0.10–0.30]	**38**	2502.54	2435.08	2.70	2340.14	6.49	41	2384.15	4.73	2258.66	9.75
a-06-120	[0.10–0.30]	**23**	2806.92	2783.10	0.85	2720.38	3.08	26	2918.71	**−3.98**	2568.90	8.48
a-07-100	[0.10–0.30]	20	1428.27	1423.49	0.34	1417.28	0.77		1462.01	**−2.36**	1414.33	0.98
Average					1.41		3.59			0.83		5.76

Table 7 (*Continued*)

Problem	Demand	m	ICA + VND[a]	SS[a]		MA|PM[a]		m'	Splitabu-DT[b]		EMIP + VRTR[c]	
			z	z	IMP	z	IMP		z	IMP	z	IMP
a-01-050	[0.10–0.50]	**15**	1045.93	1025.91	1.91	988.31	5.51	16	1008.67	3.56	943.86	9.76
a-02-075	[0.10–0.50]	**22**	1503.02	1484.62	1.22	1413.80	5.94	24	1443.62	3.95	1393.53	7.28
a-03-100	[0.10–0.50]	**29**	1957.55	1926.15	1.60	1845.30	5.73	33	1894.72	3.21	–	–
a-04-150	[0.10–0.50]	**43**	2685.33	2649.97	1.32	2561.65	4.61	49	2632.71	1.96	2532.93	5.68
a-05-199	[0.10–0.50]	**56**	3450.84	3310.71	4.06	3191.25	7.52	63	3284.47	4.82	3202.57	7.19
a-06-120	[0.10–0.50]	**34**	4085.36	3996.29	2.18	3934.39	3.70	40	4206.12	−**2.96**	3687.06	9.75
a-07-100	[0.10–0.50]	29	2046.15	2022.30	1.17	1994.59	2.52		2029.99	0.79	1973.34	3.56
Average					1.92		5.07			2.19		7.20
a-01-050	[0.10–0.90]	**25**	1547.32	1580.77	−**2.16**	1467.06	5.19	26	1469.92	5.00	1408.34	8.98
a-02-075	[0.10–0.90]	**37**	2212.93	2233.08	−**0.91**	2102.58	4.99	41	2124.43	4.00	2056.54	7.07
a-03-100	[0.10–0.90]	**48**	2925.13	2932.34	−**0.25**	2780.95	4.93	56	2794.08	4.48	–	–
a-04-150	[0.10–0.90]	**73**	4192.50	4185.68	0.16	4045.87	3.50	84	3909.72	6.74	3945.38	5.89
a-05-199	[0.10–0.90]	**93**	5192.06	5085.64	2.05	4941.22	4.83	107	4853.83	6.51	5094.61	1.88
a-06-120	[0.10–0.90]	**56**	6483.06	6361.46	1.88	6318.37	2.54	67	6583.97	−**1.56**	6079.14	6.23
a-07-100	[0.10–0.90]	48	3178.28	3187.44	−**0.29**	3113.72	2.03		3101.53	2.41	3162.22	0.51
Average					0.07		4.00			3.94		5.09
a-01-050	[0.30–0.70]	**25**	1557.52	1568.04	−**0.68**	1477.01	5.17	26	1496.90	3.89	1408.68	9.56
a-02-075	[0.30–0.70]	**37**	2241.59	2228.90	0.57	2132.16	4.88	39	2160.51	3.62	2112.61	5.75
a-03-100	[0.30–0.70]	**49**	2945.19	2986.33	−**1.40**	2858.87	2.93	53	2870.50	2.54	–	–
a-04-150	[0.30–0.70]	**73**	4192.50	4185.68	0.16	4045.87	3.50	80	4039.70	3.64	4011.74	4.31
a-05-199	[0.30–0.70]	**96**	5366.06	5265.01	1.88	5155.36	3.93	103	5102.84	4.91	5088.08	5.18
a-06-120	[0.30–0.70]	**58**	6591.40	6481.09	1.67	6424.71	2.53	65	6639.55	−**0.73**	6123.96	7.09
a-07-100	[0.30–0.70]	49	3318.08	3248.76	2.09	3155.69	4.89		3038.02	8.44	3134.56	5.53
Average					0.61		3.98			3.76		6.24

Table 7 (*Continued*)

Problem	Demand	m	ICA + VND[a]	SS[a]		MA|PM[a]		m'	Splitabu-DT[b]		EMIP + VRTR[c]	
			z	z	IMP	z	IMP		z	IMP	z	IMP
a-01-050	[0.70–0.90]	**40**	2215.34	2312.48	**−4.38**	2154.35	2.75	42	2165.21	2.26	2056.01	7.19
a-02-075	[0.70–0.90]	**60**	3341.26	3387.86	**−1.39**	3200.35	4.22	61	3180.64	4.81	3067.19	8.20
a-03-100	[0.70–0.90]	**80**	4455.14	4580.98	**−2.82**	4312.95	3.19	82	4302.31	3.43	–	–
a-04-150	[0.70–0.90]	**119**	6513.36	6479.46	0.52	6267.48	3.78	123	6196.36	4.87	5950.35	8.64
a-05-199	[0.70–0.90]	**158**	8368.35	8323.72	0.53	8081.58	3.43	162	7944.63	5.06	7207.04	13.88
a-06-120	[0.70–0.90]	**95**	10302.16	10158.32	1.40	10063.47	2.32	99	10304.08	**−0.02**	8941.79	13.20
a-07-100	[0.70–0.90]	80	5058.76	5065.26	**−0.13**	4919.48	2.75		4867.79	3.78	4779.13	5.53
Average					−0.90		3.20			3.46		9.44

z denotes objective function value obtained

IMP denotes percentage objective function reduction over ICA + VND

m denotes number of vehicles in final ICA + VND, SS, and MA|PM final solutions

m' denotes number of vehicles in final Splitabu-DT solutions

[a] Tested instances were generated by Mota et al. (2007)

[b] Tested instances were generated by Archetti et al. (2006b). ICA + VND run using similar problem generator

[c] Tested instances were generated by Chen et al. (2007)

Table 8 Running time in seconds for existing approaches and the ICA + VND approach on instances of Archetti et al. (2006b)

Problem	Demand	ICA + VND[a]	SS[b]	MA|PM[c]	Splitabu-DT[d]	EMIP + VTR[e]
a-01-050		10.89	24.80	8.53	13.20	1.80
a-02-075		9.81	61.66	35.72	35.80	4.00
a-03-100		43.50	108.80	34.59	57.60	–
a-04-150		129.23	261.28	103.69	389.00	10.00
a-05-199		534.83	352.31	353.84	386.40	18.10
a-06-120		257.30	131.34	50.92	38.40	5.60
a-07-100		21.02	108.41	42.89	49.00	3.70
a-01-050	[0.01–0.10]	4.52	26.86	12.38	4.80	1.90
a-02-075	[0.01–0.10]	51.28	68.80	18.75	13.00	25.80
a-03-100	[0.01–0.10]	415.47	125.06	37.12	31.20	–
a-04-150	[0.01–0.10]	666.20	352.09	100.27	172.80	107.80
a-05-199	[0.01–0.10]	3750.44	963.84	356.22	525.80	413.40
a-06-120	[0.01–0.10]	341.59	163.28	72.98	42.40	36.40
a-07-100	[0.01–0.10]	222.42	80.56	34.97	57.80	53.90
a-01-050	[0.10–0.30]	1.59	26.31	10.22	21.80	3.40
a-02-075	[0.10–0.30]	13.19	86.02	34.14	45.40	57.00
a-03-100	[0.10–0.30]	34.09	98.00	78.06	95.80	–
a-04-150	[0.10–0.30]	164.19	10.06	147.89	393.20	308.00
a-05-199	[0.10–0.30]	248.83	19.11	347.14	754.80	618.50
a-06-120	[0.10–0.30]	54.25	11.33	144.19	142.60	136.40
a-07-100	[0.10–0.30]	22.56	151.25	43.27	146.00	126.50
a-01-050	[0.10–0.50]	2.81	3.84	12.49	28.20	14.70
a-02-075	[0.10–0.50]	11.25	6.09	37.38	123.20	214.00
a-03-100	[0.10–0.50]	25.16	7.55	28.39	136.20	–
a-04-150	[0.10–0.50]	111.66	16.17	224.89	739.20	630.50
a-05-199	[0.10–0.50]	339.36	20.64	436.20	2668.00	1775.70
a-06-120	[0.10–0.50]	40.53	63.80	163.14	268.00	220.70
a-07-100	[0.10–0.50]	11.92	41.23	51.31	292.80	287.60
a-01-050	[0.10–0.90]	2.83	3.91	21.42	60.80	55.40
a-02-075	[0.10–0.90]	10.80	6.64	46.11	193.40	401.10
a-03-100	[0.10–0.90]	19.00	9.16	84.38	648.60	–
a-04-150	[0.10–0.90]	141.27	25.03	244.91	2278.00	2220.00
a-05-199	[0.10–0.90]	662.77	71.09	725.69	3297.20	3038.10
a-06-120	[0.10–0.90]	42.00	15.86	196.14	877.80	722.80
a-07-100	[0.10–0.90]	12.89	9.08	52.13	259.60	251.20
a-01-050	[0.30–0.70]	2.20	4.25	24.53	48.60	47.90
a-02-075	[0.30–0.70]	11.28	7.14	51.78	128.60	509.60
a-03-100	[0.30–0.70]	15.14	10.36	100.16	810.20	–

Table 8 (*Continued*)

Problem	Demand	ICA + VND[a]	SS[b]	MA|PM[c]	Splitabu-DT[d]	EMIP + VTR[e]
a-04-150	[0.30–0.70]	143.05	19.38	244.86	3008.00	3028.30
a-05-199	[0.30–0.70]	349.97	120.28	749.94	3565.60	3035.70
a-06-120	[0.30–0.70]	59.20	17.16	271.39	658.60	605.40
a-07-100	[0.30–0.70]	13.69	9.73	91.31	777.80	716.50
a-01-050	[0.70–0.90]	2.59	4.13	22.91	106.40	135.40
a-02-075	[0.70–0.90]	10.25	7.66	27.48	869.20	811.00
a-03-100	[0.70–0.90]	14.31	12.06	55.75	1398.40	–
a-04-150	[0.70–0.90]	93.78	131.91	401.62	10223.20	10038.80
a-05-199	[0.70–0.90]	460.89	165.28	571.70	21849.20	12542.30
a-06-120	[0.70–0.90]	59.28	20.17	298.08	1825.60	725.40
a-07-100	[0.70–0.90]	20.70	9.19	180.11	1004.40	1024.30

[a]P4, 512 MB, 2.8 GHz

[b]P4, 1.0 GB, 2.4 GHz

[c]3 GHz

[d]P4, 256 MB, 2.4 GHz

[e]P4, 512 MB, 1.7 GHz

In terms of solution values z in Table 7, ICA + VND is comparable with SS and MA|PM. On average, our solutions are within 1.77% and 4.34%, respectively. We are not able to improve MA|PM in any case, but we do improve SS in 10 problems with large demands (see the three largest demand ranges in the table). We note a tendency of ICA + VND to perform better as the customer demands get larger. While the other approaches tend to outperform our ICA + VND, across the board the ICA + VND is competitive with those approaches. Column m shows the minimum possible fleet size to satisfy all customer demands, which is also the number of vehicles in the final ICA + VND, SS, and MA|PM solutions. Column m' shows the number of vehicles in the final feasible solutions of the tabu search. Chen et al. (2007) do not report the fleet size for EMIP + VRTR in their computational results. A very important consideration is that our ICA + VND usually utilizes less vehicles than the tabu search (see bold numbers in column m). Using more vehicles may reduce objective function values thereby somewhat obscuring solution comparisons. Table 8 summarizes the reported running times for the existing algorithms and the running times for our ICA + VND approach. The ICA + VND approach only requires a single run, and per Table 8 obtains those solutions quicker than the other approaches in most cases.

Computational results on the TSPLIB instances solved by Belenguer et al. (2000) are presented in Table 9. This table compares the ICA + VND solution values z with the bounds obtained by Belenguer et al. (2000) using a heuristic method and a cutting plane algorithm, the bounds found by Liu (2005) using a branch-and-price approach (B&P), and the bounds obtained by Jin et al. (2008) with a column generation approach (omitted values in the table are not published). Our solution values are also compared with those found by Boudia et al. (2007) with MA|PM. Our ICA + VND is

Table 9 Computational results of ICA + VND on some TSPLIB VRP instances

Problem	ICA + VND		Belenguer et al. (2000)			MA|PM		
	z[a]	CPU[b]	UB[a]	LB	%ALB	z[a]	CPU[c]	IMP
b-eil22-021	375	0.70	375	375	0.00	375	4.11	0.00
b-eil23-022	570	0.59	569	569	0.18	569	5.47	0.18
b-eil30-029	520	2.22	510	508	2.31	503	5.70	3.27
b-eil33-032	869	1.86	835	833	4.14	835	5.19	3.91
b-eil51-050	538	10.89	521	511.57	4.91	521	7.28	3.16
b-eilA76-075	875	9.81	832	782.7	10.55	828	35.94	5.37
b-eilB76-075	1055	16.42	1023	937.47	11.14	1019	13.09	3.41
b-eilC76-075	751	26.25	735	706.01	5.99	738	14.75	1.73
b-eilD76-075	714	29.92	683	659.43	7.64	682	23.12	4.48
b-eilA101-100	842	43.50	817	793.48	5.76	818	25.25	2.85
b-eilB101-100	1129	31.13	1077	1005.85	10.91	1082	21.81	4.16

Problem	ICA + VND		B&P			Jin et al. (2008)			
	z	CPU[b]	UB	LB	%ALB	UB	LB	CPU[d]	%ALB
b-eil22-021	375.28	0.70	**376.00**	373.60	0.45	–	–	–	–
b-eil23-022	569.75	0.59	**608.00**	564.30	0.96	–	–	–	–
b-eil30-029	521.48	2.22	515.30	507.20	2.74	–	–	–	–
b-eil33-032	870.35	1.86	**873.40**	830.20	4.61	–	–	–	–
b-eil51-050	540.82	10.89	**558.50**	507.60	6.14	–	–	–	–
b-eilA76-075	880.28	9.81	**900.70**	800.30	9.09	–	–	–	–
b-eilB76-075	1059.57	16.42	**1163.10**	965.70	8.86	**1063.75**	981.14	45084.00	7.40
b-eilC76-075	758.49	26.25	**809.30**	711.20	6.23	–	–	–	–
b-eilD76-075	719.41	29.92	**768.80**	652.30	9.33	–	–	–	–
b-eilA101-100	854.13	43.50	**910.20**	797.50	6.63	–	–	–	–
b-eilB101-100	1142.02	31.13	**1174.10**	1013.90	11.22	–	–	–	–

z denotes objective function value obtained

CPU denotes running time in seconds

IMP denotes percentage objective function reduction over ICA + VND

%ALB denotes percent ICA + VND above lower bound

MA|PM uses one more vehicle than the minimum fleet size on instance b-eil30-029

[a] Objective function value obtained with euclidean distances truncated to the nearest integer

[b] P4, 512 MB, 2.8 GHz

[c] PC 3.0 GHz

[d] P4, 2 GB, 2.8 GHz

competitive with the other approaches and clearly dominates B&P on these instances. To compare with Belenguer et al. (2000) and MA|PM, our euclidean inter-node distances are also truncated to the nearest integer. In these instances, our solutions are within 5.78% above the lower bounds of Belenguer et al. and 2.96% above MA|PM, on average. Although the percentages above the lower bounds of B&P are higher, note that these percentages change with the bounds and also with the strategy to calculate the inter-node distances. However, we improve the upper bounds of B&P in all but one problem, b-eil30-029 (see bold type in columns UB).

The computational results of ICA + VND on the random problems of Belenguer et al. (2000) are presented in Table 10. In this table we also compare with the results of Chen et al. (2007), who used the EMIP + VRTR to solve only the problems with large average demands (omitted values in the table are not published). In these instances we are able to improve the upper bounds of Belenguer et al. in problems with large demands with 50 and 75 customers and find the same solution value on two other problems in demand ranges $D1$ and $D2$. With respect to MA|PM, we are within 4.04% on average in this problem set. In terms of running time, our ICA + VND is highly competitive and finds those solutions quicker than the other approaches in almost all cases.

Table 11 shows the computational results of ICA + VND on the problem set recently generated by Chen et al. (2007). Each algorithm is presented with the objective function value z and the running time in seconds. The last column, m, in the table contains the number of vehicles in the ICA + VND final solutions. Bold type indicates the best known solution value z to each problem. These problems were generated to have a geometric symmetry, a star shape, with the customers located in concentric circles around the depot (Chen et al. 2007). As one might expect our heuristics, each of which rely upon our route angle computation, do extremely well on these problems. With ICA + VND, we improved on 16 of the 21 problems. If we examine Table 6 we see that CA improved upon 12 of the 21 problems. We cannot compare the number of vehicles used in our solutions as this information is not published for EMIP + VRTR. In Fig. 8 we show the ICA + VND final solution to problem c-SD10-64. This problem is also illustrated in Chen et al. (2007). Our solution has a lower value and uses one less vehicle.

In Table 12, we present the solution values z and running times obtained with our ICA + VND on instances of Jin et al. (2007). In this table, our results are compared with the optimal solutions found by Jin et al. (2007) with their TSVI approach. Our ICA + VND found the optimal solution in problems j-eil22-021 and j-J2-021, whereas a small deviation from optimality was obtained in problem j-J1-018. We find solutions within 4.18% of the optimal value, on average.

5 Conclusions and future directions

This paper provided a fairly comprehensive background on the SDVRP and approaches to solve the problem. We present three local heuristic search algorithms to solve the SDVRP with the minimum fleet size, examine their performance on available benchmark test problems, and offer insight into heuristic performance. These algorithms are then compared to available algorithms based on a thorough empirical study. The first algorithm is a constructive approach that uses a new route angle control mechanism to quickly find high quality solutions on seven benchmark problems. This approach provides solutions within 9% of the best known solutions on a

Table 10 Computational results of ICA + VND on the random problems of Belenguer et al. (2000)

Problem	ICA + VND		Belenguer et al. (2000)			MA|PM		
	z[a]	CPU[b]	UB[a]	LB	%ALB	z[a]	CPU[c]	IMP
b-S51D1-050	469	4.53	458	454	3.20	458	8.77	2.35
b-S51D2-050	726	4.05	726	676.63	6.80	707	7.44	2.62
b-S51D3-050	994	2.50	972	905.22	8.93	945	7.84	4.93
b-S51D4-050	1700	2.89	1677	1520.67	10.55	1578	11.98	7.18
b-S51D5-050	1399	1.80	**1440**	1272.86	9.02	1351	16.72	3.43
b-S51D6-050	2221	2.27	**2327**	2113.03	4.86	2182	9.92	1.76
b-S76D1-075	603	63.55	594	584.87	3.01	592	15.23	1.82
b-S76D2-075	1165	7.73	1147	1020.32	12.42	1089	30.5	6.52
b-S76D3-075	1485	12.23	1474	1346.29	9.34	1427	12.89	3.91
b-S76D4-075	2205	6.91	**2257**	2011.64	8.77	2117	8.76	3.99
b-S101D1-100	757	210.36	716	700.56	7.46	717	49.75	5.28
b-S101D2-100	1431	26.20	1393	1270.97	11.18	1372	31.72	4.12
b-S101D3-100	1975	27.84	1975	1739.66	11.92	1891	33.98	4.25
b-S101D5-100	2985	18.36	2915	2630.43	11.88	2854	18.66	4.39

Problem	ICA + VND		B&P			EMIP + VRTR			Jin et al. (2008)			
	z	CPU[b]	UB	LB	%ALB	z	CPU[d]	IMP	UB	LB	CPU[e]	%ALB
b-S51D1-050	473.22	4.53	**513.90**	449.90	4.93	–	–	–	–	–	–	–
b-S51D2-050	732.38	4.05	**1296.50**	556.70	23.99	–	–	–	723.37	694.98	5978	5.11
b-S51D3-050	1001.22	2.50	986.00	956.00	4.52	–	–	–	968.85	922.72	607	7.84
b-S51D4-050	1708.00	2.89	1654.00	1623.00	4.98	1586.50	201.74	7.11	1657.61	1505.35	260	11.86
b-S51D5-050	1404.54	1.80	**1434.00**	1416.00	−0.82	1355.50	201.62	3.49	**1439.92**	1297.46	46	7.62
b-S51D6-050	2230.06	2.27	**2316.00**	2270.00	−1.79	2197.80	301.90	1.45	**2300.21**	2108.59	243	5.45
b-S76D1-075	610.23	63.55	–	–	–	–	–	–	–	–	–	–
b-S76D2-075	1169.80	7.73	–	–	–	–	–	–	**1185.72**	1066.17	12806	8.86
b-S76D3-075	1490.08	12.23	–	–	–	–	–	–	**1504.94**	1397.43	2030	6.22
b-S76D4-075	2220.87	6.91	2205.00	2178.00	1.93	2136.40	601.92	3.80	2219.07	2019.91	1813	9.05
b-S101D1-100	765.48	210.36	–	–	–	–	–	–	–	–	–	–
b-S101D2-100	1444.96	26.20	–	–	–	–	–	–	**1474.51**	1349.77	47658	6.59
b-S101D3-100	1990.28	27.84	–	–	–	–	–	–	**2012.86**	1837.33	7959	7.69
b-S101D5-100	2999.31	18.36	–	–	–	2846.20	645.99	5.10	2954.96	2725.5	847	9.13

z denotes objective function value obtained

CPU denotes running time in seconds

IMP denotes percentage objective function reduction over ICA + VND

%ALB denotes percent ICA + VND above lower bound

MA|PM uses one more vehicle than the minimum fleet size on instance b-eil30-029

[a] Objective function value obtained with euclidean distances truncated to the nearest integer

[b] P4, 512 MB, 2.8 GHz

[c] PC 3.0 GHz

[d] P4, 512 MB, 1.7 GHz

[e] P4, 2 GB, 2.8 GHz

Table 11 Comparison of ICA + VND on Chen et al. (2007) problem set and approach

Problem	EMIP + VRTR		ICA + VND		m
	z	CPU[b]	z[a]	CPU[c]	
c-SD01-008	**228.28**	0.7	**228.28**	0.06	6
c-SD02-016	714.40	54.4	**708.28**	0.22	12
c-SD03-016	430.61	67.3	**430.58**	0.17	12
c-SD04-024	**631.06**	400	635.84	0.55	18
c-SD05-032	1408.12	402.7	**1390.57**	0.69	24
c-SD06-032	**831.21**	408.3	831.24	0.94	24
c-SD07-040	3714.40	403.2	**3640.00**	1.03	30
c-SD08-048	5200.00	404.1	**5068.28**	1.75	36
c-SD09-048	**2059.84**	404.3	2071.03	2.91	36
c-SD10-064	2749.11	400	**2747.83**	3.58	48
c-SD11-080	13612.12	400.1	**13280.00**	3.97	60
c-SD12-080	7399.06	408.3	**7279.97**	4.00	60
c-SD13-096	10367.06	404.5	**10110.58**	5.80	72
c-SD14-120	11023.00	5021.7	**10893.50**	15.49	90
c-SD15-144	15271.77	5042.3	**15168.28**	18.33	108
c-SD16-144	**3449.05**	5014.7	3635.27	39.71	108
c-SD17-160	26665.76	5023.6	**26559.93**	17.42	120
c-SD18-160	14546.58	5028.6	**14440.59**	40.38	120
c-SD19-192	20559.21	5034.2	**20191.19**	27.64	144
c-SD20-240	40408.22	5053	**39813.49**	63.18	180
c-SD21-288	**11491.67**	5051	11799.60	738.49	216

z denotes objective function value obtained:

[a]values divided by 100 for comparison purposes (see Table 6)

CPU denotes running time in seconds:

[b]P4, 512 MB, 1.7 GHz

[c]P4, 512 MB, 2.8 GHz

m denotes the number of vehicles in the ICA + VND final solution. EMIP + VRTR vehicles not published

set of previously employed benchmark problems. The second algorithm is an iterative approach that executes the constructive approach repeatedly. This algorithm uses the knowledge from past solutions to influence future decisions in the constructive approach and can provide good feasible solutions at a relatively low computational time, again when applied to the set of previously employed benchmark problems. The third algorithm is a variable neighborhood descent that produces our best solutions. On average, our ICA + VND found solutions whose values are within 4.18% of optimality on existing benchmark problems, while the optimal solutions were obtained within a second for two problems involving 21 customers. When tested on the newest benchmark problems available for SDVRP research each of our approaches improved significantly upon existing solutions. Overall, our new algorithms were shown to be

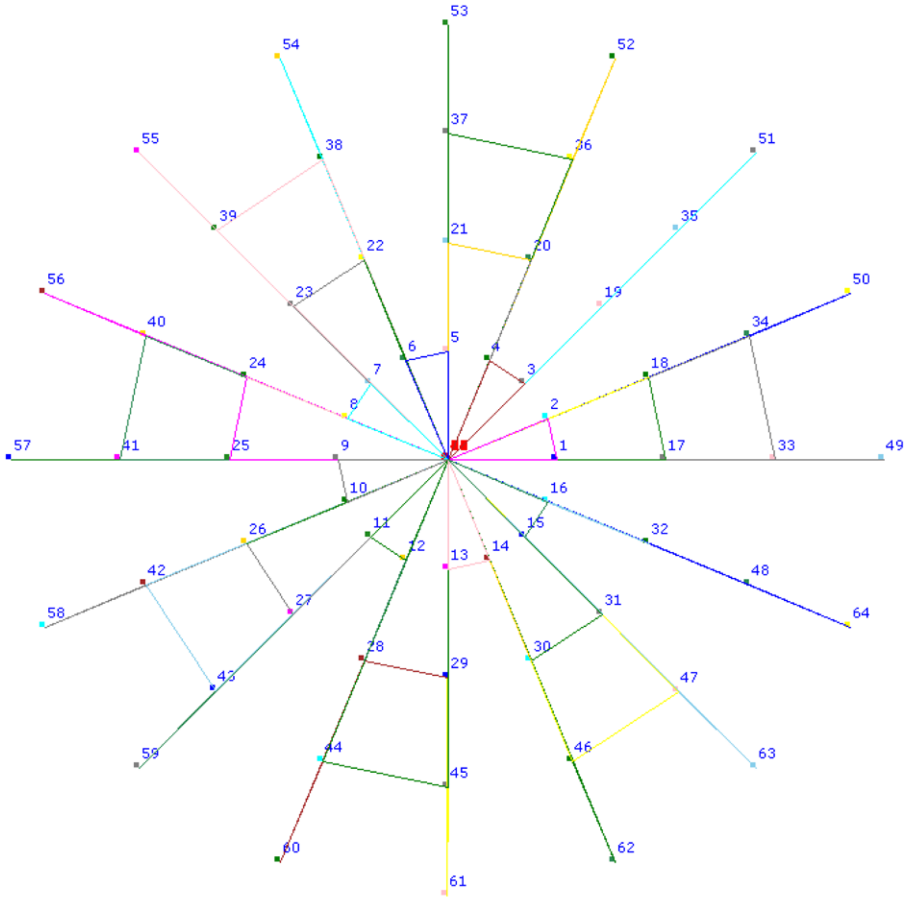


Fig. 8 ICA + VND solution to problem cheSD10-64. Total distance is 2,747.83 with 48 vehicles

Table 12 Computational results of ICA + VND versus optimality on instances of Jin et al. (2007)

Problem	TSVI		ICA + VND		% above optimality
	z^*	CPU	z	CPU	
j-eil22-021	375.28	17 h	375.28	0.7 s	**0.00**
j-J1-018	127.39	13 h	127.76	0.3 s	0.29
j-J2-021	388.44	84 h	388.44	0.5 s	**0.00**
j-J3-022	367.93	17 h	409.19	0.5 s	11.21
j-J4-022	372.2	13 h	407.16	0.5 s	9.39

z^* denotes optimal objective function value

z denotes objective function value obtained

CPU denotes running time

competitive on general forms of SDVRP instances and a particularly good choice for special classes of SDVRP instances.

The proposed route angle control mechanism is easy to implement and looks useful to solve the SDVRP, specially in problems with large customer demands. In the future, we want to explore other mechanisms to estimate the threshold angle used by this mechanism and be able to perform better with the constructive approach, especially in problems where the depot is not centered with respect to the customer locations. Although our constructive approach tends to produce non-crossing routes, optimal solutions can have crossing as well as inner routes so we want to find a way to guide the CA to design such routes. We also want to investigate more aggressive strategies to modify the sequence of customers L used by the constructive approach and force a better exploration of the search space. We have found that these strategies can produce solutions with common attributes and similarities with the best known solutions and can potentially be used for recombination operators and produce high quality solutions.

Acknowledgements The authors are grateful to the referees for their suggestions and comments on our paper. We want to thank C. Archetti for providing us the computational results of the tabu searches and Reghioui Mohamed for facilitating the instances used in their computational experiments.

References

Ambrosino, D., Sciomachen, A.: A food distribution network problem: a case study. IMA J. Manag. Math. **18**(1), 33–53 (2007)

Archetti, C., Mansini, R., Speranza, M.G.: Complexity and reducibility of the skip delivery problem. Transp. Sci. **39**(2), 182–187 (2005)

Archetti, C., Hertz, A., Speranza, M.G.: A tabu search algorithm for the split delivery vehicle routing problem. Transp. Sci. **40**(1), 64–73 (2006a)

Archetti, C., Savelsbergh, M.W.P., Speranza, M.G.: Worst-case analysis for split delivery vehicle routing problems. Transp. Sci. **40**(2), 226–234 (2006b)

Archetti, C., Savelsbergh, M.W.P., Speranza, M.G.: To split or not to split: That is the question. Transp. Res. Part E **44**(1), 114–123 (2008)

Belenguer, J., Martinez, M., Mota, E.: A lower bound for the split delivery vehicle routing problem. Oper. Res. **48**(5), 801–810 (2000)

Belfiore, P.P., Yoshizaki, H.T.Y.: Scatter search for heterogeneous fleet vehicle routing problems with time windows and split deliveries. Prod. Publ. Assoc. Bras. Engenh. Prod. **16**(3), 455–469 (2006)

Boudia, M., Prins, C., Reghioui, M.: An effective memetic algorithm with population management for the split delivery vehicle routing problem. In: Hybrid Metaheuristics. Lecture Notes in Computer Science, vol. 4771, pp. 16–30. Springer, Berlin (2007)

Bouzaiene-Ayari, B., Dror, M., Laporte, G.: Vehicle routing with stochastic demands and split deliveries. Found. Comput. Decis. Sci. **18**(2), 63–69 (1993)

Bräysy, O.: A reactive variable neighborhood search for the vehicle-routing problem with time windows. INFORMS J. Comput. **15**(4), 347–368 (2003)

Chen, S., Golden, B., Wasil, E.: The split delivery vehicle routing problem: Applications, algorithms, test problems, and computational results. Networks **49**(4), 318–329 (2007)

Christofides, N., Eilon, S.: An algorithm for the vehicle-dispatching problem. Oper. Res. Q. **20**(3), 309–318 (1969)

Christofides, N., Mingozzi, A., Toth, P.: The vehicle routing problem. In: Christofides, N., Mingozzi, A., Toth, P., Sandi, C. (eds.) Combinatorial optimization, pp. 315–338. Wiley, Chichester (1979). Chap. 11

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

Dantzig, G., Ramser, J.: The truck dispatching problem. Manag. Sci. **6**(1), 80–91 (1959)

Dror, M., Trudeau, P.: Savings by split delivery routing. Transp. Sci. **23**(2), 141–145 (1989)

Dror, M., Trudeau, P.: Split delivery routing. Nav. Res. Logist. **37**(3), 383–402 (1990)

Dror, M., Laporte, G., Trudeau, P.: Vehicle routing with split deliveries. Discrete Appl. Math. **50**(3), 239–254 (1994)

Foster, B.A., Ryan, D.M.: An integer programming approach to the vehicle scheduling problem. Oper. Res. Q. **27**(2), 367–384 (1976)

Frizzell, P., Giffin, J.: The bounded split delivery vehicle routing problem with grid network distances. Asia-Pac. J. Oper. Res. **9**, 101–116 (1992)

Frizzell, P., Giffin, J.: The split delivery vehicle scheduling problem with time windows and grid network distances. Comput. Oper. Res. **22**(6), 655–667 (1995)

Ho, S., Haugland, D.: A tabu search heuristic for the vehicle routing problem with time windows and split deliveries. Comput. Oper. Res. **31**(12), 1947–1964 (2004)

Jin, M., Liu, K., Bowden, R.O.: A two-stage algorithm with valid inequalities for the split delivery vehicle routing problem. Int. J. Prod. Econ. **105**(1), 228–242 (2007)

Jin, M., Liu, K., Eksioglu, B.: A column generation algorithm for the vehicle routing problem with split delivery. Oper. Res. Lett. **36**(2), 265–270 (2008)

Kytöjoki, J., Nuortio, T., Bräysy, O., Gendreau, M.: An efficient variable neighborhood search heuristic for very large scale vehicle routing problems. Comput. Oper. Res. **34**(9), 2743–2757 (2007)

Lee, C.-G., Epelman, M.A., White, C.C., Bozer, Y.: A shortest path approach to the multiple-vehicle routing problem with split pick-ups. Transp. Res. Part B **40**(4), 265–284 (2006)

Liu, K.: A study on the split delivery vehicle routing problem. Ph.D. thesis, Mississippi State University (2005)

Mladenović, N., Hansen, P.: Variable neighborhood search. Comput. Oper. Res. **24**(11), 1097–1100 (1997)

Mota, E., Campos, V., Corberan, A.: A new metaheuristic for the vehicle routing problem with split demands. In: Evolutionary Computation in Combinatorial Optimization. Lecture Notes in Computer Science, vol. 4446, pp. 121–129. Springer, Berlin (2007)

Mullaseril, P.A., Dror, M., Leung, J.: Split-delivery routing heuristics in livestock feed distribution. J. Oper. Res. Soc. **48**(2), 107–116 (1997)

Nowak, M.: The pickup and delivery problem with split loads. Ph.D. thesis, Georgia Institute of Technology (2005)

Polacek, M., Hartl, R.F., Doerner, K., Reimann, M.: A variable neighborhood search for the multi depot vehicle routing problem with time windows. J. Heur. **105**(6), 613–627 (2004)

Sierksma, G., Tijssen, G.: Routing helicopters for crew exchanges on off-shores locations. Ann. Oper. Res. **76**(1–4), 261–286 (1998)

Song, S.H., Lee, K.S., Kim, G.S.: A practical approach to solving a newspaper logistics problem using a digital map. Comput. Ind. Eng. **43**(1–2), 315–330 (2002)

Tavakkoli-Moghaddam, R., Safaei, N., Kah, M., Rabbani, M.: A new capacitated vehicle routing problem with split service for minimizing fleet cost by simulated annealing. J. Franklin Inst. **344**(5), 406–425 (2007)

Wilck, J., Cavalier, T.: Solving the split delivery vehicle routing problem with a loaded travel cost objective. In: G. Bayraksan, W. Lin, Y. Son, R. Wysk (eds.) Proceedings 2007 Industrial Engineering Research Conference, Nashville, TN. Institute of Industrial Engineers (2007)

Yu, Y., Chen, H., Chu, F.: Large scale inventory routing problem with split delivery: a new model and Lagrangian relaxation approach. Int. J. Serv. Oper. Inform. **1**(3), 304–320 (2006)