A Scatter Search Algorithm for the Split Delivery Vehicle Routing Problem

Campos,V., Corberán, A., Mota, E.

Dep. Estadística i Investigació Operativa. Universitat de València. Spain
Corresponding author: mota@uv.es

Abstract. In this paper we present a metaheuristic procedure constructed for the special case of the Vehicle Routing Problem in which the demands of the clients can be split, i.e., any client can be serviced by more than one vehicle. The proposed algorithm, based on the scatter search methodology, produces a feasible solution using the minimum number of vehicles. The results obtained compare with the best results known up to date on a set of instances previously published in the literature.

1 Introduction

In this paper we consider a variant of the Vehicle Routing Problem (VRP) in which the demand of any client can be serviced by more than one vehicle, the Split Delivery Vehicle Routing Problem (SDVRP). This relaxation of the classical VRP was first proposed by Dror and Trudeau ([9] and [10]), who showed that important savings on the total solution cost could be obtained as well as a reduction in the total number of vehicles used in the solution by allowing clients to be serviced by more than one vehicle. They also showed that this problem is also NP-hard. The SDVRP has received great attention in the last years.

Mullaseril, Dror and Leung [18] studied the problem of distributing feed to cattle at a large livestock ranch in Arizona. They modelled this problem as an arc routing one (in fact, as a Capacitated Rural Postman Problem with split deliveries and time windows). The about 100.000 head of cattle are fed each day, within a specified time window, by six trucks that deliver feed to the large pens connected by a road network. Sometimes, the last pen on a route does not receive its full load and another truck, servicing a different route, has to visit it again in order to complete the load. The computational experiments showed that allowing split deliveries produced a significant reduction in the total distance travelled by the vehicles in most of the considered situations.

In 1998, Sierksma and Tijssen [19] presented another application. The problem was to schedule the helicopter flights from an airport near Amsterdam to 51 off-shore platforms in the North Sea in order to exchange the employees, which work every other week. A person leaving a platform is exchanged for another person arriving for working at the same platform. The helicopters have a fixed capacity and, because of fuel constraints, a maximum flying distance for each route. The problem was modelled as a Split Delivery Vehicle Routing Problem, in which the total number of exchanges at a given platform could be done by more than one helicopter.

More details about the above applications, as well as some comments on their resolution and results, can be found in the recent paper by Chen, Golden and Wasil [6]. Moreover, that paper also mentions another interesting application by Song, Lee and Kim [20] related to the distribution of newspapers from printing plants to agents in Seul (South Korea).

Dror, Laporte and Trudeau [8] proposed a branch and bound algorithm based on an integer and linear SDVRP formulation, to which several classes of new valid inequalities were added. The procedure was tested on three small instances up to 20 clients and varying client demands. The SDVRP was studied from a polyhedral point of view in [5]. Based on the partial description of the SDVRP polyhedron, the same authors implemented a branch and cut algorithm capable of solving some medium size instances up to 51 clients. The strengthened linear relaxation produces a good lower bound to the optimal solution value.

In their original work, Dror and Trudeau ([9], [10]) propose a two stage procedure that first obtains a feasible VRP solution and then improves it using, among others, specific routines such as the route addition and k-split interchanges. The Route addition routine consists of creating a new route to service a client whose demand is split in several routes if the total distance is reduced. A k-split interchange considers a client i with demand d_i and removes i from all the routes that service it. Then, the routine considers all subsets of routes having a "residual" capacity greater than d_i and computes the total insertion cost of client i into all the routes of the subset. Finally, the subset leading to the least insertion cost is chosen and the interchange takes place. These basic but important procedures have also been used in successive heuristic and metaheuristic procedures later on. In order to test their algorithm, Dror and Trudeau selected three basic instances with 75, 115 and 150 clients, with randomly generated demands from six different scenarios, expressed as a fraction of the vehicle capacity. The same pattern has been used since then, so most of the published and available test instances are generated from known VRP instances, varying the demands of the clients. In two cases, as far as we know, other instances are proposed: in [3] some of the Solomon instances are used and in [6] the authors propose new and geometric instances. However the published computational results do not always allow an easy comparison.

In [11], Frizzell and Giffin studied the SDVRP with time windows but on a special network (the clients are located on a grid) and proposed a constructive heuristic followed by some improvement procedures (1-0 exchanges and 1-1 interchanges) that will be described later.

A Tabu Search procedure was developed by Archetti, Hertz and Speranza [1]. It produces an initial feasible solution using the GENIUS algorithm for the TSP ([12]). In the Tabu Search phase moves are made according to two procedures: one orders the routes servicing client *i* according to the saving obtained by removing *i,* while the other looks for the "best neighbor" solution of the current one. In a final and improvement phase, GENIUS is applied to each individual route. A variant of this algorithm (SPLITABU) consists of applying 2-opt and node interchange procedures each time the best solution encountered so far is improved. This variant produces better results and is denoted as SPLTABU-DT.

Archetti, Savelsbergh and Speranza [4] use the above Tabu Search procedure to identify parts of the solution space that are likely to contain high quality solutions. Once a set of promising routes is selected, an integer program (a route-based formulation for the SDVRP) is run trying to obtain improved feasible solutions.

In a recent paper, Chen, Golden and Wasil [6] propose a procedure that first constructs a feasible VRP solution using the Clarke and Wright [7] savings algorithm. The solution of an integer program provides then the optimal reallocation of the endpoints of a route in order to maximize the total savings. The program is run for a maximum time of T seconds, with a given neighbour list for each endpoint that is a function of the number of endpoints, and the best feasible solution found is saved. Then, using the feasible solution as the initial one, a new program is run, with a larger size for the neighbour list and a smaller limit for the running time. To the final and best solution obtained, a variable length record-to-record travel algorithm [16] that considers 1-0 exchanges, 1-1 interchanges and 2-opt moves, is applied.

Before closing this introduction, we should mention some structural properties of the problem. In [9] Dror and Trudeau showed that “if the cost matrix satisfies the triangle inequality, then there exists an optimal solution to the SDVRP where no two routes have more than one client with a split demand in common”. Archetti, Savelsbergh and Speranza ([2]) define by n_i the number of vehicles servicing client i and by $n_i - 1$ the number of splits at client i. Then they show that when the cost matrix satisfies the triangle inequality there exists an optimal solution to the SDVRP where the total number of splits (the sum of the number of splits of every client) is less than the number of routes.

Moreover, Archetti, Savelsbergh and Speranza addressed in [3] the question: To split or not to split? They showed first that, assuming that all the distances among clients and the depot satisfy the triangle inequality, the ratio between the minimum number of routes required to satisfy the client demands in a VRP solution over the minimum number in a SDVRP solution is always less than or equal to 2. They also proved that this bound is tight. In what refers to the ratio between optimal solution values, the same authors showed in [2] that the same bound applies. So, allowing splitting the demands may produce important savings both in the total number of vehicles used and in the total solution cost, as already pointed out by Dror and Trudeau. Moreover, Archetti, Savelsbergh and Speranza conducted an empirical study of the last mentioned ratio, as a function of client’s location and client’s demand, concluding: Cost’s reductions seem to be due to the ability to reduce the number of routes, without a dependence on client’s locations, and mainly depend on the relation between mean demand and vehicle capacity and on the variance of the demands. They obtained the largest benefits when the mean demand is greater than half the vehicle capacity but less than three quarters of the vehicle capacity. Our own computational study also points to this direction.

The paper is organized as follows: in Section 2 the problem is defined and some notation is presented. Section 3 describes the main features of the proposed metaheuristic and in Section 4 we present the computational results. Conclusions and future work are summarized in Section 5.

2 Problem definition and notation

The Vehicle Routing Problem with Split Demands is defined on an undirected and complete graph $G = (V, E)$, where $V = \{0,1,2,...n\}$ is the set of vertices (vertex 0 denotes the depot and $1,...,n$ represent the set of clients). Each edge $e = (i,j)$ has an associated cost or distance c_e between clients i and j. Moreover, each vertex has a known demand d_i ($d_0=0$) and there is a fleet of identical vehicles of capacity Q located at the depot. A feasible solution consists of a set of routes, each one beginning and ending at the depot, such that:

- The demand of every client is satisfied, and
- The sum of the demands serviced by any vehicle does not exceed its capacity Q

The SDVRP version defined above is a very difficult problem that presents an outstanding characteristic that makes it different from the classical VRP: *there is always a feasible solution using the minimum number of vehicles k*. It is easy to see that this minimum number corresponds to the smallest integer greater than or equal to $\Sigma_i d_i /Q$. This is not always true if the demand of a client can not be split, since in this case the minimum number of vehicles corresponds to the optimal solution of a Bin Packing Problem.

To the usual and explicit objective of minimizing the total solution cost, we add the implicit one of minimizing the number of vehicles used in the solution. We consider that this is a very important objective, since in most of the practical applications involving several vehicles there is a fixed cost associated to each used vehicle. Moreover, the total fixed cost of a fleet is usually greater than the total and variable cost of a feasible solution. This variable cost usually depends on the total distance travelled by the fleet. Note that a term in the objective function penalizing the excess of vehicles could be added, or bicriteria techniques could also be taken into account, since it is possible in some instances to decrease the total cost by increasing the number of vehicles. Instead we propose a Scatter Search procedure, following the framework presented in [14] and [15], which generates a population of feasible solutions with the minimum number of vehicles.

3 A Scatter Search Procedure

In this section we describe the main features of a Scatter Search procedure designed for the SDVRP. This is, as far as we know, the first time that such a technique is applied to this routing problem. Figure 1 presents a schematic summary of the overall procedure. The main characteristics and particularities introduced by the authors are briefly described in the next subsections.

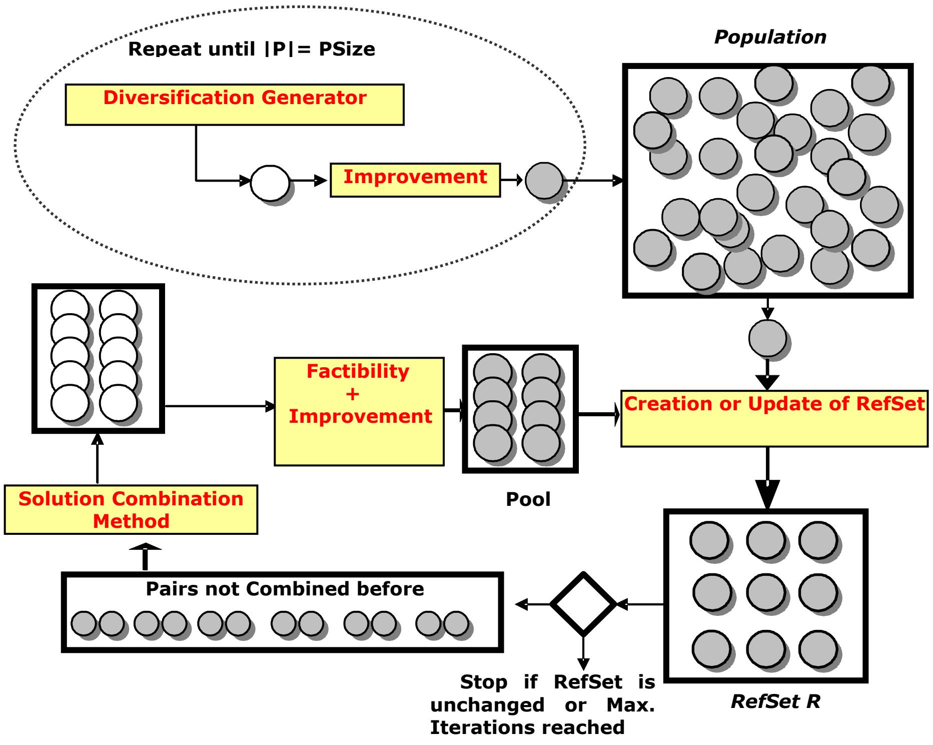
Repeat until |P|= PSize
Diversification Generator
Improvement
Population
Factibility + Improvement
Creation or Update of RefSet
Pool
Solution Combination Method
Pairs not Combined before
Stop if RefSet is unchanged or Max. Iterations reached
RefSet R

Figure 1: Scheme of a Scatter Search procedure

3.1 Creating a Population

We have adapted two standard VRP heuristic procedures to the split demands case in order to obtain SDVRP feasible solutions. The first one, called here Big Tour, uses the Lin and Kernighan [17] heuristic to build a giant tour through the *n* clients and the depot. From this tour it is always possible to obtain *k* routes and, thus, a feasible SDVRP solution: Let us first renumber the clients so that the TSP tour is 0-1-2-… -(n-1)-n-0. The first vehicle leaves the depot using edge (0,1) and services, successively, clients 1,2,… up to client i_1 for which the total demand serviced by this first route is at least equal to the vehicle's capacity. Client i_1 is either completely serviced or its demand is split between routes 1 and 2. In the first case, edges $(i_1,0)$ and $(0, i_1+1)$ are added, corresponding to the last edge in route 1 and the first one in route 2. In the second case edge $(i_1,0)$ is added twice and route 2 continues using edge (i_1,i_1+1). The feasible solution finally obtained satisfies the two structural properties mentioned in the introduction, i.e., the total number of splits is less than the number of routes and any two routes have, at most, one client in common. We can take into account the difference between the total capacity $k\,Q$ of the vehicles' fleet and the total demand of the clients and adjust the load in each vehicle to an *average load* so that the solution finally obtained uses k balanced (in terms of load) routes. In this way, we avoid obtaining a solution having $k-1$ routes with load Q and a last route with usually a very small load.

The same Big Tour is used to generate additional solutions, all of them following the same sequence of clients but starting each one at a different client, i.e., the second feasible solution (for instance) could use as the first edge in the first route edge (0, 2) and then edge (2, 3) and so on. The last edge in route k would then be edge (1, 0). However, in order to obtain solutions that differ substantially, the starting clients are selected in a nonconsecutive order.

The second procedure is a modified and accelerated version of the classical Clarke and Wright parallel savings algorithm [7]. According to this procedure, from an initial solution consisting of n return trips to each client, the best available saving, computed as $s_{ij} = c_{0i} + c_{0j} - \lambda\, c_{ij}$, is used to merge the single routes $(0,i,0)$ and $(0,j,0)$ into a new route $(0,i,j,0)$ and the procedure is repeated until no merge is feasible, in terms of vehicle capacity, or there are no more available savings. For each client, its neighborhood is computed as the subset of its closest clients, and only these savings are calculated. We allow to split the demand of a client l only when the best available saving corresponds to merging a given route r with a return trip from client l and the total demand exceeds the vehicle capacity Q; in this case, part of the demand of client l is serviced in route r and we maintain a return trip from client l with the unsatisfied demand. In order to limit the complexity of the procedure, any client is serviced by, at most, two vehicles. This procedure does not guarantee a feasible solution using the minimum number of vehicles but in all our computational experiences, feasible solutions using k vehicles are obtained. In order to generate more than one solution, we prohibit half of the savings used in a solution when computing the next one. Savings are prohibited with probabilities directly proportional to the frequency of use of each saving in the previously generated solutions.

Finally, we have also implemented a sequential version of Clarke and Wright savings algorithm [7] in which routes are generated one after the other. Thus, the procedure has always just one *active route* which grows by adding new clients either to the first serviced client or to the last one (the two "endpoints" of the route). In this case, only the demand of the last client added to the route can be split, but the number of vehicles that may service this client is not limited. Savings are again computed, for a pair of clients i and j as $s_{ij} = c_{0i} + c_{0j} - \lambda\, c_{ij}$ and, in both versions, different values of parameter λ are considered in order to diversify the solutions. We have chosen λ=1, 0.6 and 0.4.

We have tested the quality of the feasible solutions obtained by these three procedures on the set of test instances used in this paper, after applying to each feasible solution the improvement procedures described next. Table 1 shows the average values on 50 solutions obtained for each instance by the three constructive algorithms and Figure 2 summarizes the results obtained. These results indicate that the values of feasible solutions obtained by the sequential version are always worst than the values obtained by the other two procedures, which are similar.

Instance class	Big Tour	Sequential CW	Parallel CW
OD	10160398,7	12167770,7	9863394,7
0.01-0.1	8196954,3	9405272,8	8160844,3
0.1-0.3	17922662,7	22271986,9	17572631,2
0.1-0.5	24246567,9	29550626,9	24172589,7
0.1-0.9	37063229,9	41902227,9	37405057,9
0.3-0.7	37575097,1	42143493,7	38055871,1
0.7-0.9	58107028,0	66310339,9	60157794,6

Table 1: Average values for three constructive algorithms for the SDVRP

We have decided then to discard the sequential version of Clarke and Wright savings algorithm and use procedure Big Tour to generate half of the population of feasible solutions, of size *P,* and generate the remaining feasible solutions using the parallel version of the Clarke and Wright heuristic.

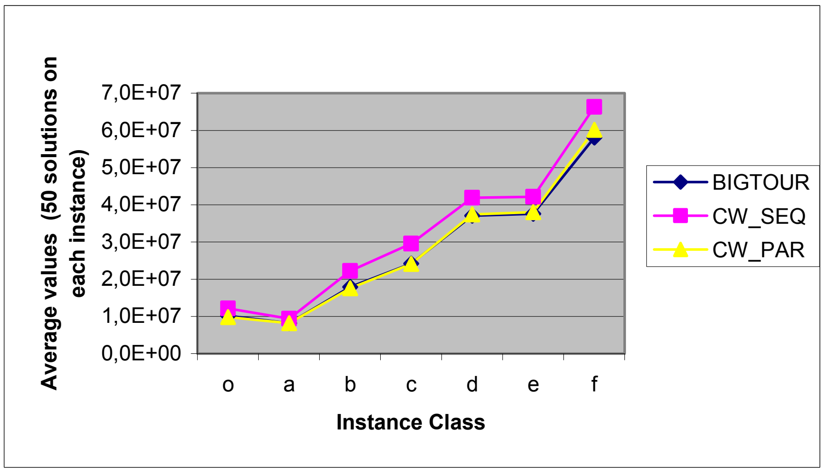

Figure 2: Quality comparison for the three constructive heuristics

3.2 Improving a Feasible Solution

A local search phase is applied to each solution in the original population in order to reduce its cost, if possible. We have implemented procedures for client moves, such as the *1-0 exchanges*, tried first and consisting of shifting one client from one route to another route, and *1-1 interchanges*, consisting of interchanging one client from a route with another client in another route. These moves are applied to every non split client. We have also implemented *two-split changes*, that take a client out from every route visiting it and look for a pair of routes that, jointly, could service its demand (two-split changes are a particular case of the k-split changes, first introduced by Dror and Trudeau in [9]. Finally, *2-2 interchanges* are checked. With them, we try to interchange one edge of a route with another from another route. When such improvements are no longer possible, the routes in the solution are re-optimized using a 2-opt procedure or the more complex Lin and Kernighan algorithm. The same procedures are applied to a feasible solution entering the reference set, as described in the next subsection.

3.3 The Reference Set

The *P* feasible solutions in the population are ordered according to the cost and *b* of them are selected to be in the reference set. One half corresponds to the best feasible solutions and the remaining solutions add the necessary diversity to this set, since they correspond to those solutions in the population that are the most different when compared to the best ones. As a measure of the difference between two solutions we compute the total number of edges in one solution but not in the other. Each pair of solutions in the reference set is combined to produce another solution that enters the set only when its cost is greater than the cost of the worst solution, which is eliminated. The overall procedure stops when, after every possible combination (one iteration), no new feasible solution enters in the reference set or a maximum number of iterations, previously fixed, is reached.

In our computational experiments, we have tested different population's sizes combined with the size of the Reference Set. As expected, solutions' quality improves as *P* and *b* increase, although at cost of greater computing times. Accordingly, we have chosen *P*=150 and *b*=25 as the final values for the computational experiences and comparisons.

3.4 The Combination Method

We have devised a procedure that captures the essential characteristics of a feasible SDVRP solution and tries to maintain those that could be satisfied by the good solutions. In order to do that, for each solution in the reference set we define a set of *critical* clients, consisting of:

1. all its split clients,
2. all the clients in routes with just 1 or 2 clients,
3. the client whose removal from a route produces the greatest saving cost, for each route with at least 3 clients, and finally
4. every client such that at least one among its three closest neighbours belongs to a different route.

When combining feasible solutions A and B in the reference set we consider, in turn, a *critical client* in A, in classes 1 to 3 above, and we move this client, thus modifying solution A, following the recommendation for this same client in solution B. If it is a split client in B, we consider that there is no recommendation and so we take the next *critical client* in A. If it is not, we consider its two adjacent clients, say α and β, in B and we locate these clients in solution A. If clients α and β are two consecutive clients in the same route in solution A, we understand that solution B recommends to insert the critical client between α and β; otherwise, the critical client is moved to the "best recommended" position, i.e., inserted after client α or inserted before client β. In order to simplify the combination procedure and since the moves already performed may produce a route overload, to move a client to an already unfeasible route is prohibited. Note that combining solutions B and A is also possible and produces a different combination.

When all the *critical clients* of A have been considered, the combination method has produced a new and maybe unfeasible solution because of the load in each route. A routine is then applied that considers some moves aimed at obtaining a feasible solution.

Each time a feasible solution is obtained as a combination of two solutions in the reference set, the improve procedures described in subsection 3.2 are applied. Once all the possible comparisons have been considered, if no new solution enters the reference set, instead of finishing the overall procedure at this moment, we augment the set of *critical clients* of each solution in the reference set by including those in case 4 above and we run all the possible combinations once again. The new neighborhood created for each solution replaces the usual rebuilding phase. If a feasible solution enters now the Reference Set, the procedure continues, otherwise we stop it. The procedure considers also the combinations already performed so that repetitions are avoided and only pairs of feasible solutions not yet considered are combined.

4 Computational Experiments

We present the results obtained by the Scatter Search procedure on a set of instances following the proposal made by Dror and Trudeau. We have repeated the generation parameters used by Archetti, Hertz and Speranza [1] so that the results can be compared.

4.1 The Instances

In order to test our algorithm, we have considered the same set of instances used by Archetti, Hertz and Speranza [1]. They generated the instances starting from the VRP problems 1 to 5, 11 and 12 taken from [13], which have between 50 and 199 clients and satisfy the triangle inequality, and computed the demands of the customers in the following way. First, two parameters α and γ ($\alpha \leq \gamma$) are chosen in the interval [0, 1]. Then, the demand d_i of customer i is set equal to $d_i = \lfloor \alpha\, Q + \delta\,(\gamma - \alpha)\, Q \rfloor$, where δ is a random number in [0, 1]. As in [9], Archetti, Hertz and Speranza have considered the following combinations of parameters (α, γ): (0.01, 0.1), (0.1, 0.3), (0.1, 0.5), (0.1, 0.9), (0.3, 0.7) and (0.7, 0.9). The procedure is equivalent to generate the demands of an instance in the interval *(αQ, γQ)*. Considering also the case where the original demands are not changed (represented as O.D. in the tables), a total of 49 instances are obtained.

4.2 Computational Results

Computational results on the whole set of instances are shown in Table 4 in the Appendix, which compares the results obtained by the Scatter Search procedure with those obtained with two Tabu Search (TS) algorithms presented in [1]. Other characteristics of the feasible solutions are also included in the table. The first two columns show the instance name, which also indicates the number of clients, and the values of α and γ and thus, the corresponding interval where demands have been generated. Instances in the first seven rows have original demands (denoted by O.D.). Column 3 presents the best value (z) obtained by our Scatter Search procedure (SS). A value in bold indicates that this value is at least as good as all the ten values obtained by applying the SPLITABU and the SPLITABU-DT procedures ([1]). All the values have been obtained using as distance between clients i and j the cost c_{ij}:

$$c_{ij} = round\left(10000\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}\right) \qquad \textbf{(1)}$$

Column 4 indicates the number of vehicles in the feasible solution obtained (k), which corresponds always to the minimum number. Total time in seconds is presented in column 5 (T). The procedure was implemented in C and run on a PC Pentium IV, 1Gb Ram, CPU 2.40 GHz. The minimum solution value (zmin) among the five executions that each instance is run with SPLITABU and SPLITABU-DT is shown in columns 6 and 9, respectively. Similarly, columns 7 and 10 present the average solution value for the 5 runs (zmean). Columns 8 and 11 give the average times, in seconds of a PC Pentium IV, 256 Mb Ram, CPU 2.40 GHz. Finally, column 12 gives the number of vehicles in the feasible solution as presented in a previous version of [4].

Table 2 summarizes the results shown in Table 4; it gives the average number of vehicles used (k), the average solution cost (z) and the average computing time in seconds (t) on each subset of 7 instances grouped by a given interval of client's demand. Last row shows the addition of the average number of vehicles and average running times. Again, best average values are denoted in bold. Figure 3 illustrates the behavior of the three metaheuristics.

	SS-SDVRP			SP-TABU			SP-TABU-DT		
	k	**z**	**t**	**k**	**z**	**t**	**k**	**z**	**t**
O.D.	9.7	**916.4**	295.8	9.7	957.1	420.0	9.7	930.4	138.5
.01-.1	6.0	**755.9**	517.7	NA	818.1	197.1	NA	786.0	121.1
.1-.3	22.5	**1677.5**	208.8	24.7	1759.0	418.7	24.7	1709.2	228.5
.1-.5	33.3	**2326.6**	155.2	37.5	2407.4	827.6	37.5	2357.2	607.9
.1-.9	55.0	3613.1	112.2	63.5	3674.7	1035.9	63.5	**3548.2**	1087.9
.3-.7	56.3	3704.0	72.5	61.0	3830.3	826.4	61.0	**3621.1**	1285.3
.7-.9	92.0	5757.6	47.9	94.8	5837.2	4108.3	94.8	**5565.9**	5325.2
Total	268.8		1410.1	291.2			291.2		8794.4

Table 2: A summary

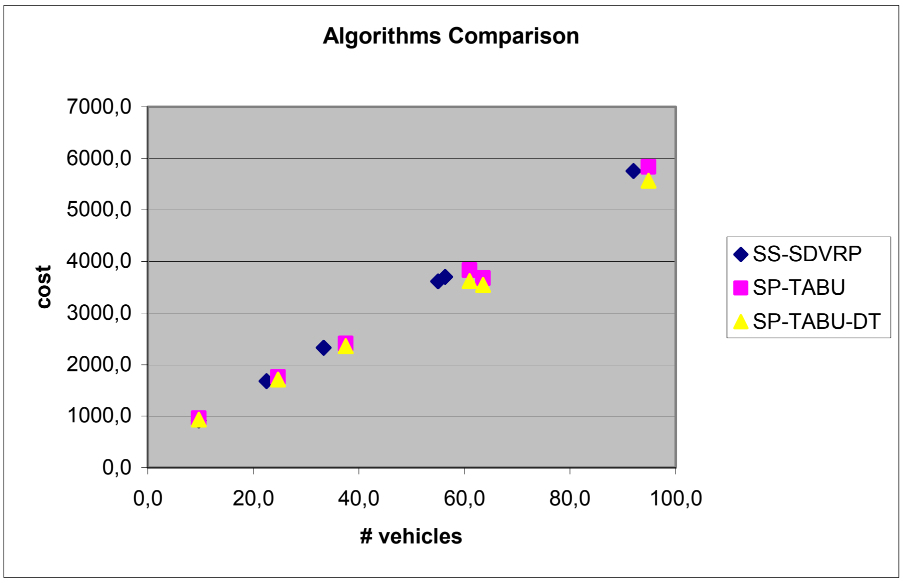

Figure 3: Number of vehicles used by the SS and TS metaheuristics

Considering the original demands, the quality of the solutions obtained with the SS algorithm is better, although at a greater computational effort. Every solution uses the minimum number of vehicles. Note that the values obtained by the Scatter Search algorithm were produced maintaining all the parameters unchanged for all the instances and with only one execution per instance. The number of vehicles used is not available, (NA), for both TS methods on the second group of instances and the solutions obtained by the SS are also better than the best solutions obtained with the TS procedures. When the demand is generated in the interval *(0.1Q, 0.3Q)* the solution's quality is also better in the case of the SS algorithm. The number of vehicles in the solutions obtained with the Tabu Search procedures is no longer the minimum one and the difference reaches 3 vehicles in 2 out of 6 instances. On the instances with demands in *(0.1Q, 0.5Q)* the SS produce worst solutions than the best ones obtained in the 5 runs of the two TS procedures, although on average the behavior of the SS algorithm is still slightly better (see Table 2) and it uses 4.2 vehicles less on average. On the remaining instances, the SS solutions are worse than the ones obtained with the TS procedures. Clearly our algorithm does not perform well on this kind of instances with big demands. This could be explained by the fact that the SS algorithm is designed to find solutions with the minimum number of vehicles while the TS algorithms minimize the total distance traveled. Note that the solutions obtained with the TS methods always use a bigger number of vehicles that, in some cases increases up to 14 vehicles, as in instance p5-199 with demands in *(0.1Q, 0.9Q).*

Finally, comparisons on all the instances with the results presented in [4] and [6] were not possible since the instances and other significant details were not available to the authors. Though in [6] the general characteristics of the instances are maintained as in [1], the instances themselves are different. However a comparison is possible among the SS, Tabu_DT and the Chen et al. algorithm on 6 of the instances with original demands. Results are summarized in Table 3. Columns z and t show the solution value and the time used to obtain it in the case of the SS and the Chen et al.'s procedure. In the case of the Tabu-DT algorithm, column z shows the value of the best solution obtained in 5 executions of the algorithm, while t shows the average computing time. Times shown for the Chen et al.'s procedure are obtained in a slower machine (PC Pentium IV, 512 Mb Ram, CPU 1.70 Ghz). As it can be seen, the best results are obtained by the SS and the Chen et al.'s procedure, although this last method is faster than ours.

Number Clients	**SS-SDVRP**		**SP-TABU-DT**		**EMIP+VRTR**	
	z	**t**	**z**	**t**	**z**	**t**
50	**524.61**	49.7	530.65	13.0	**524.61**	1.8 (3.4)
75	**829.01**	166.6	845.82	36.0	840.18	4.0 (57.0)
100	**819.56**	192.4	833.35	58.0	**819.56**	3.7 (126.5)
120	**1042.11**	270.3	1053.54	38.0	1043.18	5.6 (136.4)
150	1045.22	527.1	1064.38	389.0	**1041.99**	10.0 (308.0)
199	1324.73	588.3	1339.98	386.0	**1307.40**	18.1 (618.5)

Table 3: Comparison on the instances with original demands

Note however that a time limit is a parameter in that procedure and that greater CPU times were given to the algorithm in order to compare in [6] its results with the Tabu-DT procedure. Therefore, we have include in brackets the average CPU times, used in [6] for solving instances of similar sizes when the client's demands are generated in the range (0.1Q, 0.3Q). These average CPU times are the second best times given in [6] for the six scenarios considered.

5 Conclusions and Further Research

The first results obtained with the Scatter Search procedure indicate that it is able to obtain very good feasible solutions *with the minimum number of vehicles* within reasonable computing times. When the demands are well over half the capacity of the vehicle, the values of the solutions are not so good, because our procedure was not initially designed for these situations. The set of published and available instances is limited and quite small. In the future, we want to work on the elaboration of bigger test instances that will be publicly available and include some other refinements to the procedure. Among them, another generator of feasible solutions and more procedures to be applied to the unfeasible solutions produced in the combination phase.

Acknowledgments Authors want to acknowledge the support of the Spanish Ministerio de Educación y Ciencia, through grants MTM 2006-14961-C05-02 and TIN2006-02696. We are also grateful to C. Archetti and M.G. Speranza that kindly facilitated the instances used in the computational experiences.

References

1. Archetti, C., Hertz, A., Speranza, M.G.: A tabu search algorithm for the split delivery vehicle routing problem. Transportation Science 40 (2006) 64 -73.
2. Archetti, C., Savelsbergh, M.W.P., Speranza, M.G.: Worst-case analysis for split delivery vehicle routing problems. Transportation Science 40 (2006) 226-234.
3. Archetti, C., Savelsbergh, M.W.P., Speranza, M.G.: To Split or Not to Split: That is the Question. Transportation Research. To appear.
4. Archetti, C., Savelsbergh, M.W.P., Speranza, M.G.: An Optimization-Based Heuristic for the Split Delivery Vehicle Routing Problem. Submitted to Transportation Science.
5. Belenguer, J.M., Martínez, M.C., Mota, E.: A lower bound for the split delivery vehicle routing problem. Operations Research 48 (2000) 801-810.
6. Chen, S., Golden, B. and Wasil, E.: The Split Delivery Vehicle Routing Problem: Applications, algorithms, test problems and computational results. Networks 49 (2007) 318-329.
7. Clarke, G., Wright, J.V.: Scheduling of vehicles from a central depot to a number of delivery points. Operations Research 12 (1964) 568-581.
8. Dror, M., Laporte, G., Trudeau, P.: Vehicle Routing with Split Deliveries. Discrete Applied Mathematics 50 (1994) 239-254.
9. Dror, M., Trudeau, P.: Savings by split delivery routing. Transportation Science 23 (1989) 141-145.
10. Dror, M., Trudeau, P.: Split Delivery Routing. Naval Research Logistics 37 (1990) 383-402.
11. Frizzell, P.W., Giffin, J.W.: The split delivery vehicle routing problem with time windows and grid network distances. Computers and Operations Research 22 (1995) 655-667.

12. Gendreau, M., Hertz, A., Laporte, G.: New insertion and postoptimization procedures for the travelling salesman problem. Operations Research 40 (1992) 1086-1094.
13. Gendreau, M., Hertz, A., Laporte, G.: A tabu search heuristic for the vehicle routing problem. Management Science 40 (1994) 1276–1290.
14. Glover, F.: A template for scatter search and path relinking. In Hao, J.-K., Lutton, E., Schoenauer, M., Snyers, D. (eds.), Artificial Evolution. Lecture Notes in Computer Science 1363 (1998) 13-54.
15. Laguna, M., Martí, R.: Scatter Search – Methodology and implementations in C. Kluwer Academic Publishers, Boston (2003)
16. Li, F., Golden, B. and Wasil, E.: Very large-scale vehicle routing: New test problems, algorithms and results. Computers and Operations Research 32 (2005) 1197-1212.
17. Lin, S., Kernighan, B.W.: An effective heuristic algorithm for the travelling salesman problem. Operations Research 21 (1973) 498-516
18. Mullaseril, P.A., Dror, M., Leung, J.: Split-delivery routing in livestock feed distribution. Journal of the Operational Research Society 48 (1997) 107-116.
19. Sierksma, G., Tijssen, G.A.: Routing helicopters for crew exchanges on off-shore locations. Annals of Operations Research 76 (1998) 261-286
20. Song, S., Lee, K. and Kim, G.: A practical approach to solving a newspaper logistic problem using a digital map. Comput. Ind. Eng. 43 (2002) 315-330.

APPENDIX

Table 4. Computational results on the whole set of instances

Inst.	Dem.	SS			SP-TABU			SP-TABU-DT			
		z	**k**	**T**	**zmin**	**zmean**	**T**	**zmin**	**zmean**	**T**	**k**
p1-50	O.D.	**5246113**	5	49.7	5276751	5300570	17	5306520	5335535	13	5
p2-75	O.D.	**8290121**	10	166.5	8404529	8516729	64	8458221	8495410	36	10
p3-100	O.D.	**8294477**	8	276.1	8341357	8461844	60	8333566	8356191	58	8
p4-150	O.D.	**10452237**	12	527.1	10570279	10621988	440	10643847	10698369	389	12
p5-199	O.D.	**13247325**	16	588.3	13573455	13678177	1900	13399777	13428515	386	16
p6-120	O.D.	**10421158**	7	270.3	10763753	10847331	40	10535425	10560148	38	7
p7-100	O.D.	**8195581**	10	192.4	**8195581**	8226045	86	**8195581**	8253184	49	NA
p1-50	.01-0.1	**4607896**	3	51.8	**4607896**	4638532	9	**4607896**	4637571	5	NA
p2-75	.01-0.1	**5969872**	4	144.0	6041186	6076579	42	6016174	6052376	13	NA
p3-100	.01-0.1	**7268078**	5	272.1	7572459	7727881	59	7374514	7522012	31	NA
p4-150	.01-0.1	**8712624**	8	743.3	8864415	8949843	258	8819995	8909533	173	NA
p5-199	.01-0.1	**10231356**	10	1874.8	10452620	10735985	754	10478739	10562679	526	NA
p6-120	.01-0.1	**9765696**	6	370.9	10641948	10957537	61	10760897	10846959	42	NA
p7-100	.01-0.1	6359953	5	166.5	6407126	6628037	71	6358865	6487359	58	NA
p1-50	0.1-0.3	**7410565**	10	66.4	7507084	7644041	27	7515968	7614021	22	11
p2-75	0.1-0.3	**10718694**	15	143.8	10835052	10990297	78	10879305	10953225	45	16
p3-100	0.1-0.3	**13975030**	20	305.1	14189700	14288683	122	14193762	14248114	96	22
p4-150	0.1-0.3	19372005	29	326.6	19284742	19406720	545	19079235	19182459	393	32
p5-199	0.1-0.3	24331737	38	32.1	24120235	24199773	1224	23780537	23841545	755	41
p6-120	0.1-0.3	**27425985**	23	380.8	28520739	29009898	516	29145714	29187092	143	26
p7-100	0.1-0.3	**14188103**	20	206.3	14531928	14709592	85	14379543	14620077	146	v
p1-50	0.1-0.5	9978334	15	87.1	9940561	10076838	56	9972128	10086663	28	16
p2-75	0.1-0.5	14635982	22	126.8	14407823	14501086	71	14321606	14436243	123	24
p3-100	0.1-0.5	19080227	29	225.2	18788510	18878331	206	18857414	18947210	136	33
p4-150	0.1-0.5	26499703	43	21.3	26168532	26340901	564	26089113	26327126	739	49
p5-199	0.1-0.5	32919600	56	31.2	32765665	32981871	3811	32473125	32844723	2668	63
p6-120	0.1-0.5	**39796717**	34	329.0	41231758	41667801	259	41311259	42061210	268	40
p7-100	0.1-0.5	19953407	29	266.5	19924980	20300366	188	19815453	20299948	293	NA

Inst.	Dem.	SS			SP-TABU			SP-TABU-DT			
		z	**k**	**T**	**zmin**	**zmean**	**T**	**zmin**	**zmean**	**T**	**k**
p1-50	0.1-0.9	15543768	25	92.6	14815710	14939233	34	14438367	14699221	61	26
p2-75	0.1-0.9	21823368	37	119.9	21131636	21212778	311	21072124	21244269	193	41
p3-100	0.1-0.9	28942137	48	177.9	28116117	28266122	412	27467515	27940774	649	56
p4-150	0.1-0.9	40628821	71	50.4	39785729	40062807	1822	38497320	39097249	2278	84
p5-199	0.1-0.9	50745658	93	50.7	50058905	50396524	2598	47374671	48538254	3297	107
p6-120	0.1-0.9	63573283	56	20.6	65072461	65603347	1037	62596720	65839735	878	67
p7-100	0.1-0.9	31663064	48	272.7	31192745	31580865	523	30105041	31015273	260	NA
p1-50	0.3-0.7	15321944	25	92.4	14941462	15093118	52	14870198	14969009	49	26
p2-75	0.3-0.7	22288975	37	11.1	21666219	21759879	184	21497382	21605050	129	39
p3-100	0.3-0.7	29863298	49	17.0	28958054	29147859	454	27642538	28704954	810	53
p4-150	0.3-0.7	41856832	73	23.0	41228032	41467544	1512	39671062	40396994	3008	80
p5-199	0.3-0.7	52650121	96	327.3	53329707	53689192	2279	50014512	51028379	3566	103
p6-120	0.3-0.7	64810943	58	20.5	67207650	68663390	477	64330110	66395522	659	65
p7-100	0.3-0.7	32487607	49	16.0	31631372	32220334	411	28821235	30380225	778	NA
p1-50	0.7-0.9	23124751	40	5.8	21733326	21763923	160	21483778	21652085	106	42
p2-75	0.7-0.9	33878605	60	10.5	32184116	32294627	437	31381780	31806415	869	61
p3-100	0.7-0.9	45761339	80	38.3	43619465	43687723	1891	42788332	43023114	1398	82
p4-150	0.7-0.9	64794550	119	30.5	63345083	63542058	8783	60998678	61963577	10223	123
p5-199	0.7-0.9	83237230	158	215.0	82071543	83439543	11347	76761141	79446339	21849	162
p6-120	0.7-0.9	101583160	95	20.4	103067404	105505745	2033	100726022	103040778	1826	99
p7-100	0.7-0.9	50652558	80	13.8	49334893	49607513	1865	47735921	48677857	1004	NA