An Effective Memetic Algorithm with Population Management for the Split Delivery Vehicle Routing Problem

Mourad Boudia, Christian Prins, and Mohamed Reghioui*

ICD - OSI, University of Technology of Troyes,
12, Rue Marie Curie, BP 2060, 10010 Troyes France
{mourad.boudia,christian.prins,mohamed.reghioui_hamzaoui}@utt.fr
http://www.utt.fr/labos/LOSI

Abstract. This paper studies the Split Delivery Vehicle Routing problem (SDVRP), a variant of the VRP in which multiple visits to customers are allowed. This NP-hard problem is solved by a recent metaheuristic called Memetic Algorithm with Population Management or MA|PM (Sörensen, 2003). It consists in a genetic algorithm, combined with a local search procedure for intensification and a distance measure to control population diversity. Special moves dedicated to split deliveries are introduced in the local search. This solution approach is evaluated and compared with the tabu search algorithm of Archetti et al. (2006) and with lower bounds designed by Belenguer et al. (2000). Our method outperforms the tabu search both in solution quality and running time. On a set of 49 instances, it improves the best-known solution 32 times. The savings obtained confirm the interest and the power of the MA|PM.

Keywords: Split delivery vehicle routing problem, Memetic algorithm, Distance measure.

1 Introduction and Literature Review

The Vehicle Routing Problem (VRP) is one of the most studied problem in operations research. It consists of servicing customers with known demands, using capacitated vehicles based at a depot-node, to minimize the total cost of the routes. In 2002, Toth and Vigo edited a book entirely devoted to this problem [23]. Today, the best exact method for the VRP is a branch-and-cut-and-price developed by Fukasawa et al. [13]. Efficient metaheuristics for large instances, including memetic algorithms like the one of Prins [18], are surveyed in a book chapter by Cordeau et al. [7].

Contrary to the VRP, each customer may be visited several times in the Split Delivery VRP (SDVRP). The studies on this problem are scarce and relatively

* Corresponding author.

recent, even if Dror and Trudeau [9,10] underlined in the 90's that splitting can improve both the routing cost and the number of vehicles. These authors proved some properties of the optimal SDVRP solution and proposed a heuristic. Their tests show that the savings become significant when the average demand per customer exceeds 10% of vehicle capacity. A branch and bound algorithm was developed later by Dror et al. [8], with several integer linear formulations and reduced gaps between lower and upper bounds.

Belenguer et al. [4] presented a better lower bound for the SDVRP, based on a polyhedral approach. They studied a new family of valid inequalities and solved to optimality some instances of literature. We use this new bound to evaluate our memetic algorithm.

In two recent papers, Archetti et al. analyse the conditions in which splitting is interesting [3] and prove that the cost ratio VRP/SDVRP is 2 in the worst case [2]. In [1], they present a tabu search procedure called SplitTabu, which improves the results of Dror and Trudeau [10]. Some extensions of the SDVRP have been investigated. For example, Mullaseril et al. [17] adapt a SDVRP resolution method for a split delivery arc routing problem met in livestock feed distribution. Feillet et al. [11] and Frizzell and Griffin [12] study an SDVRP with simple and multiple time windows.

In this paper, a memetic algorithm with population management or MA|PM [21,22] is developed for the SDVRP. Section 2 states the problem and introduces some notation. The main components and the global structure of the MA|PM are described in section 3, while section 4 is devoted to computational evaluations. A conclusion and some future directions close the paper.

2 Problem Statement

The SDVRP is defined on a complete weighted and undirected network $G = (N, E, C)$. N is a set of $n+1$ nodes indexed from 0 onwards. Node 0 corresponds to a depot with identical vehicles of capacity W. Each other node i, $i = 1, 2, \ldots, n$, has a known demand q_i. The weight $c_{ij} = c_{ji}$ on each edge (i, j) of E is the travelling cost between nodes i and j. We assume that no demand q_i exceeds vehicle capacity W. Otherwise, for each customer i such that $q_i > W$, an amount of demand W can be deducted from q_i to build one dedicated trip with a full load, until the residual demand fits vehicle capacity, as shown in [1].

Partial deliveries are allowed, so some customers (called *split customers*) can be visited more than once. The objective is to determine a set of vehicle trips of minimum total cost. Each trip starts and ends at the depot and supplies a subset of customers. The number of trips or vehicles used is a decision variable. It is assumed that the triangle inequality holds: in that case, solutions in which each trip visits its customers only once dominate the others. In other words, if one customer is visited several times, these visits are done by distinct trips.

3 MA|PM Components

3.1 General Principles of MA|PM

In combinatorial optimization, classical genetic algorithms (GA) are not agressive enough, compared to other metaheuristics like tabu search. The memetic algorithms (MA) proposed by Moscato [16] are more powerful versions, in which intensification is performed by applying an improvement procedure (local search) to new solutions. For some classical optimization problems, memetic algorithms are currently the best solution methods. For instance, Prins designed for the VRP one MA which outperforms most published metaheuristics [18].

The MA with population management or MA|PM was introduced by Sörensen in his Ph.D. thesis [21] and recently published in a journal [22]. Like in any incremental MA, starting from an initial population, each iteration selects two parents, applies a crossover operator, improves the offspring using a local search procedure and replaces some existing solutions by the offspring. However, the mutation operator of traditional GAs and MAs is replaced in MA|PM by a diversity control based on a distance measure in solution space.

More precisely, let $d(B, C)$ be the distance between two solutions B and C and, by extension, $D_P(C) = \min\{d(B, C) : B \in P\}$ the distance between a new solution C and a population P. C is accepted to replace one solution in P if and only if $D_P(C) \geq \Delta$, where Δ is a given diversity threshold which can be adjusted during the search. In other words, new solutions enter the population only if they differ enough from existing solutions, in terms of structure.

Sörensen recommends a systematic local search after crossovers. When $D_P(C) < \Delta$, he suggests to apply a mutation operator until C is accepted, but the resulting solution can be strongly degraded. Prins et al. prefer to discard the child in that case, but then the local search rate must be limited to 20-50% to avoid loosing too much time in unproductive iterations. They obtain better results with this option on the Capacitated Arc Routing Problem (CARP) [19].

3.2 Chromosome and Evaluation

Like in the VRP MA of Prins [18], each solution is encoded as a list T of customers, without trip delimiters. The list can be viewed as a TSP tour and a procedure called *Split* is required to split it into trips to get a feasible SDVRP solution and its cost. However, in the SDVRP, each customer may appear more than once in T and a parallel list D is required to define the amounts delivered for each visit. Figure 1 shows two examples for an instance with $n = 6$ customers. No demand is split in the first chromosome, while customers 1, 3 and 6 are visited twice in the other. Note that the sum of the amounts delivered to each customer must be equal to his demand, e.g., $1 + 4 = 5$ for customer 3.
The procedure *Split* is easy to understand for the VRP, so we recall it first. An auxiliary graph $H = (X, A, Z)$ is constructed to describe all possible ways of splitting the TSP tour T into trips compatible with vehicle capacity. X contains $n+1$ nodes indexed from 0 onwards. A contains one arc $(i-1, j)$ for each feasible trip (sublist of customers) $(T_i, T_{i+1}, \cdots, T_j)$. The weight $z_{i-1,j}$ of arc $(i-1, j)$

Chromosome U without split demands

List of customer visits T: 5 3 6 2 1 4
Amounts delivered D : 9 5 7 4 8 6

Chromosome V with 3 split demands

List of customer visits T: 5 3 3 6 6 2 1 1 4
Amounts delivered D : 9 1 4 6 1 4 5 3 6

Fig. 1. Two examples of chromosomes with 6 customers

is the cost of the associated trip. The optimal splitting, subject to the sequence imposed by T, is obtained by computing a shortest path from node 0 to node n in H. Since H contains $O(n^2)$ arcs in the worst case, the shortest path can be computed with the same complexity, using the version of Bellman's algorithm for directed acyclic graphs.

For the SDVRP, due to split demands, T may contain more than n customer visits and a trip $(T_i, T_{i+1}, \ldots, T_j)$ may visit several times some customers, here called *split customers*. Since the triangle inequality holds, the trip cost does not increase if all visits except one are suppressed for each split customer. The problem of setting the best possible cost on the arc which models the trip in H is equivalent to removing superfluous visits to minimize the cost of the resulting trip.

This problem is sometimes easy. For instance, if T contains a trip $(1, 5, 1, 3, 2, 1, 4)$, the only split customer is 1, and there are only three ways of removing redundant visits: keep either the first, second or third occurrence. However, for SDVRP solutions with a high splitting level, the length of T is not bounded by a polynomial in n and the splitting procedure cannot be polynomial. We decided to use a simple rule: for each split customer in a trip, we keep the first visit and delete the other ones. Compared to the VRP the splitting is no longer optimal but can still be implemented in $O(n^2)$. In fact, after some preliminary tests, we realized that using this sub-optimal policy is not a real drawback: in most cases, the local search of the MA is able to move each customer visited by a vehicle to a better position in the trip.

3.3 Initial Population

The initial population P contains a fixed number nc of distinct chromosomes. Two are built using VRP heuristics: the Clarke and Wright saving algorithm [6] and the sweep heuristic of Gillett and Miller [15]. The saving algorithm starts from a trivial solution, with one dedicated trip per customer. Then, it performs successive mergers (concatenations of two trips) to reduce the total routing cost, until additional mergers would violate vehicle capacity or increase total cost. In the sweep heuristic, clusters of customers compatible with vehicle capacity are generated by the rotation of a half-line centered on the depot. A vehicle route is then computed in each cluster, by solving a traveling salesman problem.

By construction, no demand is split in these two solutions. Each solution is converted into a chromosome by concatenating its trips. When the average demand per customer is important compared to vehicle capacity, the two resulting chromosomes are often identical and, of course, only one is put in P. The $nc-2$ or $nc-1$ other initial chromosomes are randomly generated as below, to include a few split demands:

Step 1. Generate a random permutation S of the customers.

Step 2. Starting from its first customer, S is partitioned into successive trips with a greedy heuristic. Each trip is completed when the vehicle gets full or if adding one more customer would violate vehicle capacity. In the second case, the customer is split to fill the vehicle completely and his remaining demand is used to start a new trip. Hence, only the first and last customers of each trip may be split in the resulting solution. Finally, the trips are concatenated to form the chromosome.

In figure 1, assume that $W = 10$ and that the random permutation generated in step 1 is the list T of chromosome U. The heuristic in step 2 extracts one full trip (5,3), one full trip (3,6) in which customer 6 is split, one full trip (6,2,1) in which 6 and 1 are split, and a residual trip (1,4). Once concatenated, these trips yield chromosome V.

3.4 Selection and Crossover

In each MA iteration, the two parents A and B are chosen using the binary tournament method: two solutions are randomly drawn in P and the best one is taken for A. The same process is repeated to get B.

The recombination operator is a cyclic, one-point crossover designed for chromosomes with different lengths. Only one child-solution C is generated. The cut-point k is randomly drawn in the integer interval $[1, \min\{|A|, |B|\} - 1]$. Child C is initialized with the first k customers of A and their delivered amounts. The child solution is completed by browsing circularly the second parent B, from

Chromosome A

$$\begin{array}{l|l} \text{Customers : } 5\ 3\ 4\ 6 & 5\ 2\ 1\ 6\ 4\ 5 \\ \text{Demands : } \ \ 3\ 5\ 1\ 3 & 5\ 4\ 8\ 4\ 5\ 1 \end{array}$$

Chromosome B

$$\begin{array}{l|l} \text{Customers : } 4\ 1\ 6\ 2 & 5\ 3\ 4\ 1\ 2\ 5\ 6 \\ \text{Demands : } \ \ 3\ 3\ 5\ 1 & 6\ 5\ 3\ 5\ 3\ 3\ 2 \end{array}$$

Chromosome C

$$\begin{array}{l} \text{Customers : } 5\ 3\ 4\ 6\ 5\ 4\ 1\ 2\ 6\ 4\ 1\ 6\ 2 \\ \text{Demands : } \ \ 3\ 5\ 1\ 3\ 6\ 3\ 5\ 3\ 2\ 2\ 3\ 2\ 1 \end{array}$$

Fig. 2. Example of crossover

position $k+1$ onwards. There are two cases when inspecting one customer i. If his demand is already satisfied in C, we skip him. Otherwise, the customer is appended to C and the quantity associated to this visit is eventually truncated to guarantee that the total amount he receives in C does not exceed his demand q_i. Figure 2 depicts one example of crossover for 6 customers and $k = 4$.

3.5 Local Search

After each crossover, child C is evaluated and converted into an SDVRP solution, using the *Split* procedure of 3.2. In this solution, recall that no customer is visited several times by the same trip. The solution is then improved by the local search and reconverted into a chromosome by concatenating its trips. Finally, the distance test described in 3.6 is applied to know if the new chromosome may be accepted. As explained in 3.1, too many children are rejected if the local search is systematic. In practice, it is called with a fixed probability $\beta < 1$.

The first-improvement local search evaluates two groups of moves. The first group does not split demands. The solution to be improved can be viewed as a string with trips separated by copies of the depot node. In the first group, all pairs (i, j) of nodes in this string are considered, *even if they belong to distinct trips*, and the following moves are evaluated if vehicle capacity is respected. The notation $s(i)$ denotes the successor of customer i in his trip.

- move i or $(i, s(i))$ after j, if i and $s(i)$ are customers.
- exchange i and j if they are customers.
- 2-opt moves in one trip: the subsequence from $s(i)$ to j is inverted.
- 2-opt moves on two trips: edges $(i, s(i))$ and $(j, s(j))$ are removed and replaced either by (i, j) and $(s(i), s(j))$ or by $(i, s(j))$ and $(j, s(i))$.

These moves are classical for the VRP, but there is a modification for the SDVRP: if a customer i is visited by a trip and if a second visit to i is moved to this trip, the amount transferred is aggregated with the first visit. This ensures that all stops in a trip concern different customers.

The second group with three moves may split a customer or change the amounts delivered to each visit. The first move is an improvement of the *k-split procedure* of Dror and Trudeau [10] and used later by Archetti et al. [1]. This procedure consists of removing one customer i from all his trips and to reinsert it (possibly with splitting) into a given number of trips k, in order to decrease the total routing cost. Note that the best insertion position in a trip is unique and can be pre-computed. Following discussions with these authors, it appears that they only evaluate insertions for small values of k, to avoid excessive running times.

We found an optimal method for which k does not need to be fixed. Let S be the set of trips in which i can be inserted (the vehicle is not full), a_j the residual capacity of trip j, u_j the fixed cost if i is inserted into j, y_j the amount of demand inserted into j and x_j a binary variable equal to 1 if trip j is used for insertion. The optimal reinsertion corresponds to the following integer linear program.

$$\min \sum_{j \in S} u_j x_j \tag{1}$$

$$\sum_{j \in S} y_j = q_i \tag{2}$$

$$0 \le y_j \le a_j x_j \qquad \forall j \in S \tag{3}$$

$$x_j \in \{0,1\} \qquad \forall j \in S \tag{4}$$

In fact, this ILP can be replaced by the 0-1 knapsack below. Indeed, constraints (2) and (3) imply (6), the optimal solution x^* of the knapsack problem tells us which trips must be used, and the demand q_i may be assigned as we wish to these trips, e.g., using the very simple algorithm 1.

$$\min \sum_{j \in S} u_j x_j \tag{5}$$

$$\sum_{j \in S} a_j x_j \ge q_i \tag{6}$$

$$x_j \in \{0,1\} \qquad \forall j \in S \tag{7}$$

The knapsack problem can be quickly solved in practice using dynamic programming (DP) and the number of variables (trips) is often much smaller than the number n of customers. We actually implemented the DP method and the MA|PM running time increased by 30%, which is tolerable. However, we decided to use the greedy heuristic in which insertions are done in increasing order of u_j/a_j. This heuristic is known to solve optimally the dual of the continuous relaxation of (5)-(7). It is much faster and optimal in 66% of cases for our instances, and we observed no significant increase in the MA|PM final solution cost, compared to the exact algorithm.

Algorithm 1. Insertion of i using the knapsack problem solution

```
1: repeat
2:     $j := \arg\min\{u_p | p \in S \wedge x^*_p = 1 \wedge a_p \neq 0\}$
3:     $z := \min\{q_j, a_j\}$
4:     Insert customer $i$ in trip $j$, with an amount $z$
5:     $q_i := q_i - z; a_j := a_j + z$
6: until $q_i = 0$
```

The two last moves are new. The second one depicted in the upper part of figure 3 exchanges two customers i and j pertaining to distinct trips and modify their received amounts. Let $R(i)$, $p(i)$, $s(i)$, and $y(i)$ respectively denote the trip containing one given visit to i, the predecessor and successor of i in this trip and the amount received. If $y(i) > y(j)$, j is removed with his amount from $R(j)$ and inserted before or after i in $R(i)$ (the best position is selected). In parallel, a copy of i with an amount $y(i) - y(j)$ is created in $R(j)$, to replace the visit to j which has been moved. If $y(i) < y(j)$, the move is analogous, except that the roles of i and j are exchanged. The case $y(i) = y(j)$ is ignored, since this is

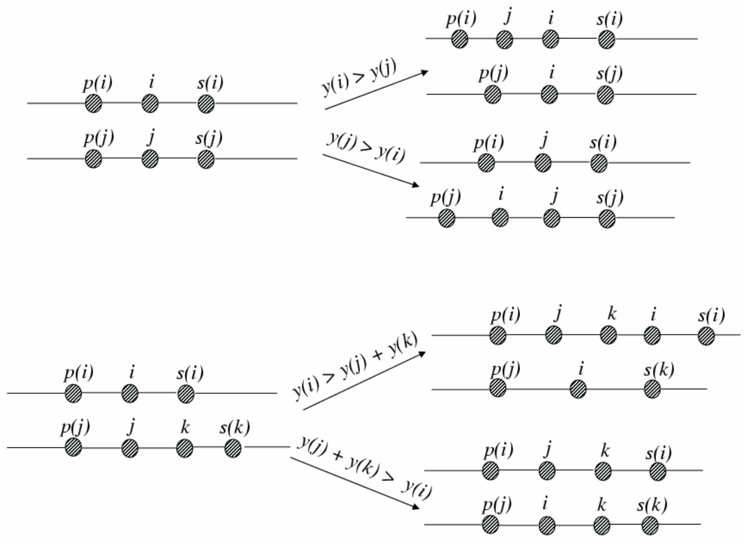

Fig. 3. Examples of splitting moves

a standard exchange already treated in the first group. Note that the new move does not affect the load of the trips: in particular, it works on two full vehicles.

The third move is similar but i is swapped with a pair (j, k) of adjacent customers, see the lower part of figure 3. If $y(i) > y(j) + y(k)$, (j, k) is removed from $R(j)$ with its two amounts and inserted before or after i in $R(i)$ (whichever is the best), while a copy of i with a quantity $y(i) - y(j) - y(k)$ is inserted at the old location. If $y(i) < y(j) + y(k)$, by convention, i and j are exchanged without splitting but a copy of k is moved to $R(i)$, with an amount $y(j)+y(k)-y(i)$. Finally, whenever $y(i) = y(j) + y(k)$, a simple exchange of i with (j, k) is performed.

3.6 Distance Measure

Campos et al. [5] proposed what they call the “distance for R-permutations”. For two permutation chromosomes A and B, it is equal to the number of pairs of adjacent customers in A which are broken (no longer adjacent) in B. An adaptation of this distance is proposed here to handle SDVRP chromosomes A and B with different lengths. In this case, the distance is the number of pairs present in chromosomes A and not in B plus the number of pairs present in B and not in A. Two $n \times n$ matrices M^A and M^B are used. For any chromosome X, M^X_{ij} is equal to the number of pairs (i, j) of adjacent customers in X. Since we have symmetric edge costs, a trip is equivalent to its reverse trip, i.e. performed backward by the vehicle. To make our distance reversal-independent, M^X is symmetric, i.e. we set $M^X_{ij} = M^X_{ji} = 1$ if X contains the substring (i, j).

The distance is then defined as $d(A, B) = \sum_{i=1}^{n} \sum_{j=1}^{n} |M^A_{ij} - M^B_{ij}|/2$. Figure 4 gives one example. The sum of the elements in matrix $|A - B|$, divided by

Chromosome A	Chromosome B
1 2 4 3 2 4 5	1 4 3 2 5 4

$$\text{Matrix } M^A \quad \begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & 0 & 1 & 0 & 0 & 0 \\ 2 & 1 & 0 & 1 & 2 & 0 \\ 3 & 0 & 1 & 0 & 1 & 0 \\ 4 & 0 & 2 & 1 & 0 & 1 \\ 5 & 0 & 0 & 0 & 1 & 0 \end{array} \qquad \text{Matrix } M^B \quad \begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & 0 & 0 & 0 & 1 & 0 \\ 2 & 0 & 0 & 1 & 0 & 1 \\ 3 & 0 & 1 & 0 & 1 & 0 \\ 4 & 1 & 0 & 1 & 0 & 1 \\ 5 & 0 & 1 & 0 & 1 & 0 \end{array} \qquad |M^A - M^B| \quad \begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & 0 & 1 & 0 & 1 & 0 \\ 2 & 1 & 0 & 0 & 2 & 1 \\ 3 & 0 & 0 & 0 & 0 & 0 \\ 4 & 1 & 2 & 0 & 0 & 0 \\ 5 & 0 & 1 & 0 & 0 & 0 \end{array}$$

Fig. 4. Example of distance measure

2, gives a distance equal to 5. The reason is that the pair $(1, 2)$ and the two pairs $(2, 4)$ of A do not exist in B and the pairs $(1, 4)$ and $(2, 5)$ of B do not exist in A. This metric is a generalization of the distance of Campos et al. It is reversal-independent, i.e. $d(A, B) = 0$ if $A = B$ or if B is the mirror of A. Indeed, if no customer is split, our distance is equal to twice the one of Campos.

3.7 Algorithm Overview

The general structure of the MA|PM can now be summarized in algorithm 2. The initial population P includes the two good VRP solutions obtained with the Clarke and Wright heuristic and the sweep heuristic. The distance measure is used to avoid adding a solution S already in P, by checking that $d_P(S) \neq 0$. The resulting population is sorted in ascending order of costs.

The main loop performs a fixed number of cycles, *maxcycles* and returns the best solution P_1 at the end. Each cycle initializes the diversity threshold Δ, executes a given number of basic iterations *maxcross* and modifies the population for the next cycle, using a partial renewal procedure. Each basic iteration chooses two parents A and B using the binary tournament technique. The resulting child C undergoes the local search with a given probability β. The solution R to be replaced by C is pre-selected at random in the worst half of P. C is accepted if its distance to P minus R is no smaller than Δ. Note that R is not included in the test, because it will no longer be in P in case of replacement. To avoid missing a new best solution, C is also accepted if it improves the current best solution, i.e., if $cost(C) < cost(P_1)$. If C is accepted, it replaces R and a simple shift is sufficient to keep P sorted. Otherwise C is simply discarded. The distance threshold Δ is updated at the end of each basic iteration.

A decreasing policy is used to vary Δ. Starting from an initial value Δ_{max}, Δ is linearly decrease to reach 1 (the minimum value to avoid duplicate solutions) at the end of each cycle. The renewal procedure consists of keeping the best solution and of replacing the others by new random solutions. Like in the initial population, all solutions must be distinct.

Algorithm 2. MA|PM overview

```
1: $P \leftarrow \emptyset$
2: initialize $P_1, P_2$ using the Clarke and Wright and Gillet and Miller heuristics
3: for $k = 3$ to $nc$ do
4:    repeat
5:       $S \leftarrow random\_solution()$
6:    until $d_P(S) \geq 1$
7:    $P_k \leftarrow S$
8: end for
9: sort $P$ in ascending cost order
10: for $cycle = 1$ to $maxcycles$ do
11:    initialize diversity threshold $\Delta$
12:    for $cross = 1$ to $maxcross$ do
13:       select two parent-solutions $A$ and $B$ in $P$ using binary tournament
14:       $C \leftarrow crossover(A, B)$
15:       if $random < \beta$ then
16:          $C \leftarrow local\_search(C)$
17:       end if
18:       select $R$ to be replaced in the worst half of $P$ ($P(\lfloor nc/2 \rfloor)$ to $P(nc)$)
19:       if $(D_{P \setminus \{R\}}(C) \geq \Delta)$ or $(cost(C) < cost(P_1))$ then
20:          remove solution $R$: $P \leftarrow P \setminus \{R\}$
21:          add solution $C$: $P \leftarrow P \cup \{C\}$
22:          shift $C$ to keep population $P$ sorted
23:       end if
24:       update $\Delta$
25:    end for
26:    $P \leftarrow partial\_renewal(P)$
27: end for
28: return the best solution $(P_1)$
```

4 Computational Results

All algorithms designed in this paper were implemented in Delphi and executed on a 3.0 GHz PC with Windows XP. Some preliminary testing was required to tune the parameters, the objective being a good tradeoff between solution quality and running time. The best results on average were achieved using a population size $nc = 10$, four cycles of 2000 crossovers each ($maxcycles = 4$ and $maxcross = 2000$, and a local search rate $\beta = 0.1$. The diversity threshold Δ is initialized with a Δ_{max} equal to one quarter of the average inter-solution distance calculated on the initial population. Δ is then decreased by $(\Delta_{max} - 1)/maxcross$ after each crossover. The tuning seems robust, since it gives very good results on the three sets of instances tested.

The MA|PM was first compared with the tabu search *SplitTabu* of Archetti et al. [1], using the same instances. These instances are derived from the VRP files 1 to 5, 11 and 12 used by Gendreau et al. [14]. They contain 50 to 199 customers. The demands q_i were regenerated randomly for each instance, using six different intervals with respect to vehicle capacity W: $[0.01W, 0.1W]$, $[0.1W,$

Table 1. Computational results of the SDVRP instances of Archetti

		SplitTabu		MA|PM			
File	Demand	Cost	Time	Cost	Time	Saving	Best
p1-50		5 335 535	13	**5 246 111**	8.53	1.68	5 246 111
p2-75		8 495 410	36	**8 238 883**	35.72	3.02	8 238 883
p3-100		8 356 191	58	**8 294 397**	34.59	0.74	8 273 926
p4-150		10 698 369	389	**10 423 740**	103.69	2.57	10 254 927
p5-199		13 428 515	386	**13 115 937**	353.84	2.33	12 929 951
p6-120		10 560 148	38	**10 412 000**	50.92	1.40	10 378 753
p7-100		8 253 184	49	**8 195 575**	42.89	0.70	8 195 575
p1-50	0.01-0.1	4 637 571	5	**4 607 896**	12.38	0.64	4 607 896
p2-75	0.01-0.1	6 052 376	13	**6 000 642**	18.75	0.85	5 962 499
p3-100	0.01-0.1	7 522 012	31	**7 268 076**	37.12	3.38	7 268 076
p4-150	0.01-0.1	8 909 533	73	**8 756 127**	100.27	1.72	8 663 116
p5-199	0.01-0.1	10 562 679	526	**10 187 055**	356.22	3.56	10 183 801
p6-120	0.01-0.1	10 846 959	42	**9 765 688**	72.98	9.97	9 765 688
p7-100	0.01-0.1	**6 487 359**	58	6 497 338	34.97	-0.15	6 345 694
p1-50	0.1-0.3	7 614 021	22	**7 514 139**	10.22	1.31	7 410 562
p2-75	0.1-0.3	10 953 225	45	**10 744 585**	34.14	1.90	10 678 012
p3-100	0.1-0.3	14 248 114	96	**13 928 469**	78.06	2.24	13 772 801
p4-150	0.1-0.3	19 182 459	393	**18 787 090**	147.89	2.06	18 750 888
p5-199	0.1-0.3	23 841 545	755	**23 401 362**	347.14	1.85	23 293 715
p6-120	0.1-0.3	29 187 092	143	**27 203 752**	144.19	6.80	27 203 534
p7-100	0.1-0.3	14 620 077	146	**14 172 757**	43.27	3.06	14 157 818
p1-50	0.1-0.5	10 086 663	28	**9 883 062**	12.49	2.02	9 883 062
p2-75	0.1-0.5	14 436 243	123	**14 137 965**	37.38	2.07	13 985 257
p3-100	0.1-0.5	18 947 210	136	**18 452 995**	28.39	2.61	18 276 498
p4-150	0.1-0.5	26 327 126	739	**25 616 472**	224.89	2.70	25 397 546
p5-199	0.1-0.5	32 844 723	2 668	**31 912 475**	436.20	2.84	31 802 981
p6-120	0.1-0.5	42 061 210	268	**39 343 880**	163.14	6.46	39 343 880
p7-100	0.1-0.5	20 299 948	293	**19 945 947**	51.31	1.74	19 815 453*
p1-50	0.1-0.9	14 699 221	61	**14 670 635**	21.42	0.19	14 438 367*
p2-75	0.1-0.9	21 244 269	193	**21 025 762**	46.11	1.03	20 872 242
p3-100	0.1-0.9	27 940 774	649	**27 809 492**	84.38	0.47	27 467 515*
p4-150	0.1-0.9	**39 097 249**	2 278	40 458 715	244.91	-3.48	38 497 320*
p5-199	0.1-0.9	**48 538 254**	3 297	49 412 170	725.69	-1.80	47 374 671*
p6-120	0.1-0.9	65 839 735	878	**63 183 734**	196.14	4.03	62 596 720*
p7-100	0.1-0.9	**31 015 273**	260	31 137 187	52.13	-0.39	30 105 041*
p1-50	0.3-0.7	14 969 009	49	**14 770 135**	24.53	1.33	14 770 135
p2-75	0.3-0.7	21 605 050	129	**21 321 601**	51.78	1.31	21 321 601
p3-100	0.3-0.7	28 704 954	810	**28 588 684**	100.16	0.41	27 642 538*
p4-150	0.3-0.7	**40 396 994**	3 008	40 458 715	244.86	-0.15	39 671 062*
p5-199	0.3-0.7	**51 028 379**	3 566	51 553 604	749.94	-1.03	50 014 512*
p6-120	0.3-0.7	66 395 522	659	**64 247 101**	271.39	3.24	63 994 197
p7-100	0.3-0.7	**30 380 225**	778	31 556 915	91.31	-3.87	28 821 235*
p1-50	0.7-0.9	21 652 085	106	**21 543 510**	22.91	0.50	21 483 778*
p2-75	0.7-0.9	**31 806 415**	869	32 003 533	27.48	-0.62	31 381 780*
p3-100	0.7-0.9	**43 023 114**	1 398	43 129 461	55.75	-0.25	42 788 332*
p4-150	0.7-0.9	**61 963 577**	10 223	62 674 827	401.62	-1.15	60 998 678*
p5-199	0.7-0.9	**79 446 339**	21 849	80 815 768	571.70	-1.72	76 761 141*
p6-120	0.7-0.9	103 040 778	1 826	**100 634 739**	298.08	2.34	100 174 729
p7-100	0.7-0.9	**48 677 857**	1 004	49 194 826	180.11	-1.06	47 735 921*

$0.3W$], [$0.1W$, $0.5W$], [$0.1W$,$0.9W$], [$0.3W$, $0.7W$] and [$0.7W$, $0.9W$]. Including the original instances for which the demands are not changed, the 49 instances listed in Table 1 are obtained. The first group of 7 instances contains the original demands.

The four first columns correspond respectively to the file name with the number n of customers, the interval of demands, the average solution cost found by *SplitTabu* over 5 runs (using different random initial solutions) and the average running time per run in seconds. *SplitTabu* was implemented in C++ on a 2.4 GHz PC. The MA|PM solution value and running time (for one run and the standard setting of parameters) are given in columns 5 and 6. The last but one column indicates the saving in % obtained by our method, computed using the following formula:

$$Saving = \frac{SplitTabu\ cost - MA|PM\ cost}{SplitTabu\ cost} \times 100$$

The last column *Best* gives the new best-known solution values, found either by the tabu search (indicated by an asterisk) or by the MA|PM, when different settings of parameters can be used.

Using one single run, our algorithm improves 37 out of the 49 instances, while being faster than *SplitTabu* for 42. All instances with demands not greater than $W/2$ are improved, except one. The saving for these instances varies between 0.64% and 9.97%. Only instance p7-100 with demands in $[0.01W,\ 0.1W]$ is not improved, but the loss is marginal (-0.15%). Concerning the other instances, in which some demands may exceed $W/2$, 10 out of 21 are improved. Moreover, MA|PM is much faster than *SplitTabu* on these instances and, when the tabu search is the best, the deviation exceeds 3% only twice: for p4-150 and demands in $[0.1W, 0.9W]$ and p7-100 with demands in $[0.3W, 0.7W]$. Concerning best-known solutions, 32 are obtained using MA|PM versus 17 for *SplitTabu*.

The second evaluation was conducted with 25 instances used by Belenguer at al. [4], downloadable at `http://www.uv.es/∼belengue/SDVRP/sdvrplib.html`. 11 are VRP instances from the TSPLIB [20], the other 14 are randomly generated. The number in the file names denotes the total number of vertices (customers plus depot). For the random instances, the coordinates come from the TSPLIB instances *eil51*, *eil76* and *eil101*. The demands were randomly generated with

Table 2. Results for TSPLIB instances

			MA|PM				
File	UB	LB	Cost	Time	MA/LB	UB/LB	MA Best
eil22	375	375.00	375	4.11	0.00	0.00	375
eil23	569	569.00	569	5.47	0.00	0.00	569
eil30	510	508.00	**503**	5.70	-0.98	0.39	503
eil33	835	833.00	835	5.19	0.24	0.24	835
eil51	521	511.57	521	7.28	1.81	1.81	521
eilA76	832	782.70	**828**	35.94	5.79	6.30	818
eilB76	1 023	937.47	**1 019**	13.09	8.70	9.12	1 007
eilC76	**735**	706.01	738	14.75	4.53	4.11	733
eilD76	683	659.43	**682**	23.12	3.42	3.57	682
eilA101	**817**	793.48	818	25.25	3.09	2.96	815
eilB101	**1 077**	1 005.85	1 082	21.81	7.57	7.03	1 007
Average					3.10	3.23	

different intervals, as in Archetti et al. [1]. In the names of the randomly generated instances in Table 3, the code Dx corresponds to the interval used: D1 for $[\lceil 0.01W \rceil, \lfloor 0.1W \rfloor]$, D2 for $[\lceil 0.1W \rceil, \lfloor 0.3W \rfloor]$, D3 for $[\lceil 0.1W \rceil, \lfloor 0.5W \rfloor]$, D4 for $[\lceil 0.1W \rceil, \lfloor 0.9W \rfloor]$, D5 for $[\lceil 0.3W \rceil, \lfloor 0.7W \rfloor]$ and D6 for $[\lceil 0.7W \rceil, \lfloor 0.9W \rfloor]$.

Table 2 gives the results obtained for the 11 VRP instances from the TSPLIB. The first column mentions the file name of each instance, the second and third give an upper bound (UB) and a lower bound (LB) obtained by Belenguer et al. using a heuristic method and a cutting plane algorithm. Columns 4 and 5 indicate the solution cost and running time in seconds achieved by our algorithm with the standard setting of parameters. Column 6 provides the deviation of our method to the lower bound in %, while the gap UB/LB achieved by Belenguer et al. is given in column 7 for comparison. The last column gives the best MA|PM solution found when parameters may be modified.

The average deviation to the lower bound is only slightly improved (around 3%), but the MA retrieves the two optima found by Belenguer et al., improves 4 instances and obtains the same results for two others. Belenguer et al. have a better solution for 3 instances only. Here, the MA|PM is quite fast, with 36 seconds in the worst case. For the instance *eil30*, the result obtained is better than the LB. This is due to the method used to compute this lower bound which considers the minimum number of vehicles. For the instance *eil30*, the MA|PM uses one more vehicle than the LB. This result does not put the comparison to LB in jeopardy because the MA|PM finds the same number of vehicles for all remaining instances.

The comparison with the instances randomly generated by Belenguer et al. is given in Table 3. The same table format is used. For these instances, the average gap UB/LB is significantly impoved by the MA: 4.97% versus 8.63%. 13 out of 14 instances are improved and the MA is still reasonably fast, with 50 seconds in the worst case.

Table 3. Results for the random SDVRP instances of Belenguer

			MA|PM				
File	UB	LB	Cost	Time	MA/LB	UB/LB	MA Best
S51D1	458	454.00	**458**	8.77	0.88	0.88	458
S51D2	726	676.63	**707**	7.44	4.48	7.30	706
S51D3	972	905.22	**945**	7.84	4.39	7.38	945
S51D4	1 677	1 520.67	**1 578**	11.98	3.77	10.28	1 578
S51D5	1 440	1 272.86	**1 351**	16.72	6.14	13.13	1 336
S51D6	2 327	2 113.03	**2 182**	9.92	3.26	10.12	2 177
S76D1	594	584.87	**592**	15.23	1.21	1.56	592
S76D2	1 147	1 020.32	**1 089**	30.50	6.73	12.41	1 087
S76D3	1 474	1 346.29	**1 427**	12.89	5.99	9.49	1 420
S76D4	2 257	2 011.64	**2 117**	8.76	5.24	12.15	2 094
S101D1	**716**	700.56	717	49.75	2.35	2.20	716
S101D2	1 393	1 270.97	**1 372**	31.72	7.95	9.60	1 372
S101D3	1 975	1 739.66	**1 891**	33.98	8.70	13.53	1 891
S101D5	2 915	2 630.43	**2854**	18.66	8.50	10.82	2 854
Average				4.97	8.63		

5 Conclusion and Future Directions

In this paper, the very recent MA|PM framework is applied to the very hard Split Delivery VRP. This requires the design of non-trivial components: an ad-hoc encoding, a crossover operator, an effective local search able to split customers and a distance adapted to chromosomes with varying lengths.

A comparison with Archetti et al. indicates that the proposed algorithm competes with a state of the art tabu search method, while being much faster. The lower bounds available for Belenguer's instances show that solution gaps can be reduced, especially for the randomly generated instances.

Future possible research directions include the design of more effective moves when the average demand represents a large fraction of vehicle capacity, in order to improve the instances which resist. We also intend to tackle additional constraints like time windows. The study of stochastic demands seems very interesting, because demand variations could have a strong impact on a planned solution with split customers. Finally, the deviations to lower bounds are moderate but more important than the typical gaps reachable for the VRP. This raises the need for tighter bounds.

Acknowledgement. This research has been funded by the Champagne-Ardenne Regional Council and by the European Social Fund.

References

1. Archetti, C., Hertz, A., Speranza, M.G.: A tabu search algorithm for the split delivery vehicle routing problem. Transportation Science 40(1), 64–73 (2006)
2. Archetti, C., Savelsbergh, M.W.P., Speranza, M.G.: Worst-case analysis for split delivery vehicle routing problems. Transportation Science 40(2), 226–234 (2006)
3. Archetti, C., Savelsbergh, M.W.P., Speranza, M.G.: To split or not to split: that is the question. Transportation Research Economics (to appear)
4. Belenguer, J.M., Martinez, M.C., Mota, E.: A lower bound for the split delivery vehicle routing problem. Operations Research 48(5), 801–810 (2000)
5. Campos, V., Martí, R., Laguna, M.: Context-independent scatter search and tabu search for permutation problems. INFORMS Journal on Computing 17, 111–122 (2005)
6. Clarke, G., Wright, J.W.: Scheduling of vehicles from a central depot to a number of delivery points. Operations Research 12, 568–581 (1964)
7. Cordeau, J.F., Gendreau, M., Hertz, A., Laporte, G., Sormany, J.S.: New heuristics for the vehicle routing problem. In: Langevin, A., Riopel, D. (eds.) Logistic systems: design and optimization, pp. 279–298. Wiley, Chichester (2005)
8. Dror, M., Laporte, G., Trudeau, P.: Vehicle routing with split deliveries. Discrete Applied Mathematics 50, 239–254 (1994)
9. Dror, M., Trudeau, P.: Savings by split delivery routing. Transportation Science 23, 141–145 (1989)
10. Dror, M., Trudeau, P.: Split delivery routing. Naval Research Logistics 37, 383–402 (1990)
11. Feillet, D., Dejax, P., Gendreau, M., Gueguen, C.: Vehicle routing with time windows and split deliveries. In: Proceedings of Odysseus 2003, Palermo (May 2003)

12. Frizzell, P.W., Giffin, J.W.: The split delivery vehicle scheduling problem with time windows and grid network distances. Computers and Operations Research 22(6), 655–667 (1995)
13. Fukasawa, R., Lysgaard, J., de Aragão, M.P., Reis, M., Uchoa, E., Werneck, R.F.: Robust branch-and-cut-and-price for the capacitated vehicle routing problem. Mathematical Programming 106, 491–511 (2006)
14. Gendreau, M., Hertz, A., Laporte, G.: A tabu search heuristic for the vehicle routing problem. Management Science 40, 1276–1290 (1994)
15. Gillett, B.E., Miller, L.R.: A heuristic algorithm for the vehicle dispatch problem. Operations Research 22, 340–349 (1974)
16. Moscato, P.: Memetic algorithms: a short introduction. In: Corne, D., Dorigo, M., Glover, F. (eds.) New ideas in optimization, pp. 219–234. McGraw-Hill, New York (1999)
17. Mullaseril, P.A., Dror, M., Leung, J.: Split-delivery routing heuristics in livestock feed distribution. Journal of Operational Research Society 48, 107–116 (1997)
18. Prins, C.: A simple and effective evolutionary algorithm for the vehicle routing problem. Computers and Operations Research 31, 1985–2002 (2004)
19. Prins, C., Sevaux, M., Sörensen, K.: A genetic algorithm with population management (GA|PM) for the CARP. In: 5th Triennal Symposium on Transportation Analysis (Tristan V), Le Gosier, Guadeloupe (2004)
20. Reinelt, G.: Tsplib-travelling salesman problem library. ORSAJ. Comput. 3, 376–384 (1991)
21. Sörensen, K.: A framework for robust and flexible optimisation using metaheuristics with applications in supply chain design. PhD thesis, University of Antwerp, Belgium (2003)
22. Sörensen, K., Sevaux, M.: MA|PM: memetic algorithms with population management. Computers and Operations Research 33, 1214–1225 (2006)
23. Toth, P., Vigo, D.: The vehicle routing problem. SIAM, Philadelphia (2002)