We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

6,900
Open access books available

186,000
International authors and editors

200M
Downloads

Our authors are among the

154
Countries delivered to

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities

WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com

1

Scatter Search for Vehicle Routing Problem with Time Windows and Split Deliveries

Patrícia Belfiore[1], Hugo Tsugunobu[2] and Yoshida Yoshizaki[2]
[1]*Department of Production Engineering – FEI University Center, São Bernardo do Campo-SP,*
[2] *Department of Production Engineering – University of São Paulo (USP) Brazil*

1. Introduction

The classical vehicle routing problem (VRP) aims to find a set of routes at a minimal cost (finding the shortest path, minimizing the number of vehicles, etc) beginning and ending the route at the depot, so that the known demand of all nodes are fulfilled. Each node is visited only once, by only one vehicle, and each vehicle has a limited capacity. Some formulations also present constraints on the maximum traveling time.

The VRPSD is a variation of the classical VRP, where each customer can be served by more than one vehicle. Thus, for the VRPSD, besides the delivery routes, the amount to be delivered to each customer in each vehicle must also be determined. The option of splitting a demand makes it possible to service a customer whose demand exceeds the vehicle capacity. Splitting may also allow decreasing costs. The vehicle routing problem with time windows and split deliveries (VRPTWSD) is an extension of the VRPSD, adding to it the time window restraints.

Lenstra and Rinnooy Kan (1981) have analyzed the complexity of the vehicle routing problem and have concluded that practically all the vehicle routing problems are NP-hard (among them the classical vehicle routing problem), since they are not solved in polynomial time.

According to Solomon and Desrosiers (1988), the vehicle routing problem with time windows (VRPTW) is also NP-hard because it is an extension of the VRP.

Although the vehicle routing problem with split deliveries (VRPSD) is a relaxation of the VRP, it is still NP-hard (Dror and Trudeau, 1990, Archetti et al., 2005).

Therefore, the VRPTWSD is NP-hard, since it is a combination of the vehicle routing problem with time windows (VRPTW) and the vehicle routing problem with split delivery (VRPSD), and that makes a strong point for applying heuristics and metaheuristic in order to solve the problem.

This work develops a scatter search (SS) algorithm to solve a vehicle routing problem with time windows and split deliveries (VRPTWSD). To generate the initial solutions of SS we propose an adaptation of the sequential insertion heuristic of Solomon (1987).

Ho and Haugland (2004) modified the customers' demands of the Solomon's test problems in order to perform split deliveries. Numerical results of SS are reported as well as comparisons with the Ho and Haugland algorithm.

Open Access Database www.i-techonline.com

Source: Vehicle Routing Problem, Book edited by: Tonci Caric and Hrvoje Gold, ISBN 978-953-7619-09-1, pp. 142, September 2008, I-Tech, Vienna, Austria

The sequence of the chapter is described next. Section 2 describes the literature review for VRPSD and its extensions. Section 3 presents the problem definition, including the mathematical formulation. Section 4 describes the scatter search overview. Section 5 describes the heuristic and the scatter search approach proposed in order to solve the model. Section 6 presents the computational results. Finally, some conclusions are drawn in the last section.

2. Literature review for the VRPSD (TW)

The vehicle routing problem with split deliveries (VRPSD) was introduced in the literature by Dror and Trudeau (1989, 1990), who presented the mathematical formulation of the problem and analyzed the economy that can be made when it is allowed that a customer is fulfilled by more than one vehicle, economy both related to number of vehicles and total distance traveled.

Dror et al. (1994) have presented an integer programming formulation of the VRPSD and have developed several families of valid inequalities, and a hierarchy between these is established. A constraint relaxation branch and bound algorithm for the problem was also described.

Frizzell and Giffin (1992, 1995) have developed construction and improvement heuristics for the VRPSD with grid network distances. In their second publication they also considered time windows constraints.

Mullaseril et al. (1997) have described a feed distribution problem encountered on a cattle ranch in Arizona. The problem is cast as a collection of capacitated rural postman problem with time windows and split deliveries. They presented an adaptation of the heuristics proposed by Dror and Trudeau (1990).

Belenguer et al. (2000) proposed a lower bound for the VRPSD based on a polyhedral study of the problem. This study includes new valid inequalities. The authors developed a cutting-plane algorithm to solve small instances. For bigger instances, integer values are obtained via branch-and-bound.

Archetti et al. (2006b) have done the worst-case performance analysis for the vehicle routing problem with split deliveries. The authors have shown that the cost savings that can be realized by allowing split deliveries is at most 50%. They also study the variant of the VRPSD in which the demand of a customer may be larger than the vehicle capacity, but where each customer has to be visited a minimum number of times. Archetti et al. (2006a) have described a tabu search algorithm for the VRPSD. At each iteration, a neighbour solution is obtained by removing a customer from a set of routes where it is currently visited, and by inserting it either into a new route, or into an existing route which has enough residual capacity. The algorithm also considers the possibility of inserting a customer into a route without removing it from another route.

Ho and Haugland (2004) have developed a tabu search algorithm for the VRPTWSD. The first stage constructs a VRP solution using node interchanges, and the second stage improves the VRP solution by introducing and eliminating splits. There is a pool of solutions that are defined by different move operators. The best solution in the current pool is always chosen.

Belfiore (2006) proposed a scatter search algorithm to solve a real-life heterogeneous fleet vehicle routing problem with time windows and split deliveries that occurs in a major

Brazilian retail group. The results show that the total distribution cost can be reduced significantly when such methods are used.
In this chapter, we developed a new scatter search algorithm to solve the VRPTWSD. The results will be compared with Ho and Haugland modified problems.

3. Problem formulation and definitions

In this section we define the problem under study, and the notation used throughout the chapter.
Customers
The problem is given by a set of customers $N = \{1, 2, .., n\}$, residing at n different locations. Every pair of locations (i, j), where $i, j \in N$ and $i \neq j$, is associated with a travel time t_{ij} and a distance traveled d_{ij} that are symmetrical $(t_{ij} = t_{ji}$ and $d_{ij} = d_{ji})$. Denote by q_i, $i = 1, 2, ..., n$, the demand at point i. The central depot is denoted by 0.
Fleet of vehicles
The customers are served from one depot with a homogeneous and limited fleet. The vehicles leave and return to the depot. There is a set V of vehicles, $V = \{1, ..., m\}$, with identical capacities. The capacity of each vehicle $k \in V$ is represented by a_k.
We let $R_i = \{r_i(1), ..., r_i(n_i)\}$ denote the route for vehicle i, where $r_i(j)$ is the index of the jth customer visited and n_i is the number of customers in the route. We assume that every route finishes at the depot, i.e. $r_i(n_i + 1) = 0$.
Time Windows
Each customer $i \in N$ has a time windows, i.e. an interval $[e_i, l_i]$, where $e_i \leq l_i$ which corresponds, respectively, to the earliest and latest time to start to service customer i. Let s_i be the service time at customer i.
Split deliveries
The demand of a customer may be fulfilled by more than one vehicle. This occurs in all cases where some demand exceeds the vehicle capacity, but can also turn out to be cost effective in other cases.
The decision variables of the model are:

$x_{ij}^k =$ 1, if j is supplied after i by vehicle k;
0, otherwise.

$b_i^k =$ moment at which service begins at customer i by vehicle k, $i = 1, .., n \quad k = 1, ..., m$

$y_i^k =$ fraction of customer's demand i delivered by vehicle k.

The objective of the model is to minimize the total distance traveled respecting the time window constraints. The mathematical programming formulation is presented below based on Dror and Trudeau (1990) and Ho and Haugland (2004).

The objective function can be written as follows:

$$\min \sum_{i=0}^{n} \sum_{j=0}^{n} \sum_{k=1}^{m} d_{ij} x_{ij}^{k}$$

The model constraints are:

$$\sum_{j=1}^{n} x_{0j}^{k} = 1 \qquad k = 1, \ldots, m$$

Constraint (1) guarantees that each vehicle will leave the depot and arrive at a determined customer.

$$\sum_{i=0}^{n} x_{ip}^{k} - \sum_{j=0}^{n} x_{pj}^{k} = 0 \qquad p = 0, \ldots, n; \qquad k = 1, \ldots, m$$

Constraint (2) is about entrance and exit flows, guarantees that each vehicle will leave a determined customer and arrive back to the depot.

$$\sum_{k=1}^{m} y_{i}^{k} = 1, \qquad i = 1, \ldots, n$$

Constraint (3) guarantees that the total demand of each customer will be fulfilled.

$$\sum_{i=1}^{n} q_{i} y_{i}^{k} \leq a_{k} \qquad k = 1, \ldots, m$$

Constraint (4) guarantees that the vehicle capacity will not be exceed.

$$y_{i}^{k} \leq \sum_{j=0}^{n} x_{ji}^{k} \qquad i = 1, \ldots, n; \quad k = 1, \ldots, m$$

Constraint (5) guarantees that the demand of each customer will only be fulfilled if a determined vehicle goes by that place. We can notice that, adding to constraint (5) the sum of all vehicles and combining to equation (3) we have the constraint $\sum_{k=1}^{m} \sum_{i=0}^{n} x_{ij}^{k} \geq 1 \quad j = 0, \ldots, n$, which guarantees that each vertex will be visited at least once by at least one vehicle.

$$b_{i}^{k} + s_{i} + t_{ij} - M_{ij}(1 - x_{ij}^{k}) \leq b_{j}^{k} \qquad i = 1, \ldots, n; \; j = 1, \ldots, n; \; k = 1, \ldots, m$$

Equation (6) sets a minimum time for beginning the service of customer *j* in a determined route and also guarantees that there will be no sub tours. The constant M_{ij} is a large enough number, for instance, $M_{ij} = l_{i} + t_{ij} - e_{j}$.

$$e_i \le b_i^k \le l_i \quad i = 1,...,n$$

Constraint (7) guarantees that all customers will be served within their time windows.

$$y_i^k \ge 0 \quad i = 1,...,n; \quad k = 1,...,m$$
$$b_i^k \ge 0 \quad i = 1,...,n; \quad k = 1,...,m$$

Equation (8) guarantees that the decision variables y_i^k and b_i^k are positive.

$$x_{ij}^k \in \{0,\ 1\} \quad i = 0,...,n;\ j = 0,...,n;\ k = 1,...,m$$

Finally equation (9) guarantees the decision variables x_{ij}^k to be binary.

4. Scatter search overview

Scatter search (SS) is an instance of evolutionary methods, because it violates the premise that evolutionary approaches must be based on randomization – though they likewise are compatible with randomized implementations. Glover (1977, 1998) proposed the first description of the method and a scatter search template. SS operates on a set of reference solutions to generate new solutions by weighted linear combinations of structured subsets of solutions. It uses strategies for search diversification and intensification that have proved effective in a variety of optimization problems.
The following parameters are used in the method discussion:

$PSize$ = size of the set of diverse solutions generated by the Diversification Generation Method
b = size of the reference set (*RefSet*)
b_1 = size of the high-quality subset of *RefSet*
b_2 = size of the diverse subset of *RefSet*
MaxIter = maximum number of iterations

A sketch of the scatter search method is presented in figure 1 based on Alegre et al. (2004) and Yamashita et al. (2006). The first step (diversification generation method) generate a set *P* of diverse trial solutions, with *PSize* elements, using one or more arbitrary trial solutions as an input. The improvement method (step 2) is applied to transform a trial solution into one or more enhanced trial solutions. If no improvement occurs for a given trial solution, the enhanced solution is considered to be the same as the one submitted for improvement. Step 3 chooses b solutions from P, according to their quality or diversity, in order to build the reference set (*RefSet*), which is a collection of both high quality solutions and diverse solutions. In the step 4, solutions are combined. For each combined solution we apply the improvement method (step 5). At this point, the reference set can be updated (step 6), depending on the quality or diversity of the new combined solution. In this work, dynamic and static *RefSet* updating are implemented, as described in section 5.3. If the search converges, i.e., no new solutions are found for inclusion in *RefSet*, then Step 7 rebuilds *RefSet*. The algorithm stops when a termination criterion – the maximum number of iterations, *MaxIter*, is reached.

Step 1: Generate solutions – generate a set *P* of *Psize* diverse trial solutions through the diversification generation method.
Step 2: Improve solutions – apply the improvement method to improve solutions generated in Step 1.
Step 3: Build the reference set: Put b_1 best solutions and b_2 diverse solutions *P* in the reference set (*RefSet*). Number of iterations = 0.
NewSolutions = TRUE
While (number of iterations < *MaxIter*) **do**
  While *NewSolutions* in *RefSet* **do**
    Step 4: Combine solutions
    Step 5: Improve solutions – Apply the improvement method for each combined solution.
    Step 6: Update reference set
  End while
  Step 7: Rebuild reference set: Remove the worst b_2 solutions from the *RefSet*. Generate *PSize* diverse trial solutions and apply the improvement method (step 1 and 2). Choose b_2 diverse solutions and add them to *RefSet*. Number of iterations = Number of iterations + 1.
End while

Figure 1. Scatter search algorithm

5. Solution method

The initial reference set is composed of solutions generated using the constructive heuristic described at section 5.1. The scatter search procedure to solve the VRPTWSD is presented at section 5.2.

5.1 Adaptation of sequential insertion's heuristic of Solomon (1987)

This is an extension of Solomon's sequential insertion heuristics I1 (1987), although, in order to generate split deliveries, the vehicle capacity constraint must not be respected, so that the demand of a customer is added while there is capacity. The initialization criterion of the route is the farthest customer yet not allocated. The insertion criterion aims to minimize the addition of distance and time caused by a customer's insertion.
The heuristics begins with the farthest customer yet not allocated (customer *i*). If the demand of customer *i* is larger than the vehicle capacity, a vehicle with full truckload is sent and the remaining demand is added to a new vehicle. Next step is the insertion of a new customer *j* to the route. If the total demand of the customer *j,* plus the demand of customer *i,* exceeds the capacity of the actual vehicle, the demand of customer *j* is added while there is still capacity, and the remaining demand is added to a new vehicle. This process goes on until all the customers belong to a route.

5.2. Scatter search procedure

This section describes the scatter search procedure proposed to solve the vehicle routing problem with time windows and split deliveries.

Diversification generation method

The initial population must be a wide set of disperse individuals. However, it must also include good individuals. So, individuals of the population are created by using a random procedure in the constructive heuristics to achieve a certain level of diversity. In the next step, an improvement method must be applied to these individuals in order to get better solutions.

So, a randomized version of the constructive heuristic, called H-random, may be achieved by modifying the initialization criterion and the insertion criterion. In H-random, customer j is randomly selected from a candidate list. The candidate list is created by first defining *r*_first and *r*_last as the first and last customer in the list. If the initialization criterion considers the customer with the earliest deadline, *r*_first and *r*_last are the customers with the earliest and latest deadline, respectively. If the initialization criterion considers the customer farthest from the depot, *r*_first and and *r*_last are the customers farthest and nearest from the depot, respectively. In the insertion criterion, *r*_first and *r*_last are the customers with the cheapest and more expensive insertion cost, respectively. A possible customer *i* is added to the list if

$$r_i \leq r\_\text{first} + \alpha(r\_\text{first} - r\_\text{last})$$

where $\alpha(0 \leq \alpha \leq 1)$ is the parameter that controls the amount of randomization permitted. If α is set to a value of zero, H-random becomes the deterministic procedure described in subsections 5.1. On the contrary, the candidate list reaches its maximum possible cardinality when α is set to a value of one.

Improvement method

To each one of the *PSize* solutions obtained through the diversification generation method we apply the improvement method that is composed by 5 phases: swap in the same route, demand reallocation, route elimination and combination, insertion and route addition. The first exchange procedure is once again applied at the end of the process.

Swap in the same route

This exchange procedure has the goal of reducing the length of a route. The procedure is applied to each route R_k, for $k = 1,...,m$. From $i = 1,...,n_k - 1$ and $j = i+1,...,n_k$, the procedure tests the reduction of the length of a route R_k produced by exchanging the positions of $r_k(i)$ and $r_k(j)$. If the length is reduced and the constraints are respected, the positions are exchanged. The procedure stops when no more feasible exchanges are possible that result in a shorter route length.

Demand reallocation

This procedure is applied for each customer *i*, $i = 1,...,n$ which is split into more than one route. We begin with the farthest customer *i* from the depot. For a determined customer *i*, we initially choose the route R_j which deliveries the most quantity for customer *i*, because customers with superior demands have a larger probability of violating the capacity constraint if they are not inserted at the beginning (Salhi and Rand, 1993). Therefore, we calculate the demand reallocation cost of customer *i* of route R_j for each one of the other routes R_k where customer *i* is inserted and we choose the cheapest one. The exchange is done only if all the problem constraints are respected and the total cost reduced. The procedure is repeated for the other routes where customer *i* is inserted.

Route Elimination and Combination

This exchange procedure aims to eliminate routes with n or less customers and routes with idle capacity. For each possible route R_j eliminated (we begin with the longest route, that is, the one with the maximum length), we aim to combine it to another route R_k, chosen though a candidate list. This candidate list is based on Corberán et al. (2002) and is described below.

For each one of the routes evaluated (R_j), there is a candidate list, based on the best pairs of routes (R_j , R_k). The best pair is the one with the minimum distance between two routes. When the routes have only one customer, the distance between the routes is simply the distance between two customers. When a route has more then one customer, we consider the extreme points to calculate the minimum distance. The minimum distance between routes R_j and R_k is given by:

$$\min\left\{d_{r_j(1)r_k(n_k)}, d_{r_k(1)r_j(n_j)}\right\}$$

where:

$d_{r_j(1)r_k(n_k)}$ is the distance between the first customer on route R_j to the last one on route R_k

$d_{r_k(1)r_j(n_j)}$ is the distance between the first customer on route R_k to the last one on route R_j.

For each route R_j, we randomly choose a candidate route R_k from the candidate list. We attempt to merge the routes, considering R_j first and then R_k. If the merging is feasible, we stop and go to the next route R_j. A feasible merging of routes R_j and R_k is such that the resulting route does not violate the capacity and time windows constraints. If the merging is not feasible, we choose another candidate route R_k from the candidate list, while it will have a candidate route R_k.

Insertion

This exchange method is based on removing a customer from one route and inserting it into another route. The insertion movement is implemented for each route R_j, where all the feasible customers will be tested in all positions of another route R_k, chosen through a candidate list, similar to the one described in the routes elimination and combination procedure. We begin with the longest route R_j, which is the one with the maximum length, and for each route R_j chosen we begin with the farthest customer i from the depot. We select the best insertion position of customer i in route R_k. The customer is only inserted if the cost is reduced and all the constraints are respected.

Routes addition

Like in demand reallocating, we consider all customers i whose demand is split into more than one route. We begin with the farthest customer i from the depot. For each customer i, we choose the route R_j that deliveries the smallest quantity for customer i. In the next step, we relocate the demand of customer i from route R_j to a new route R_k and calculate the reduction or addition to the total cost after this movement. This process is repeated for the other routes where customer i is inserted. The demand is added to the actual route R_k while there is space and the remaining demand is added to a new vehicle. In the end, the best combination is chosen, if there is any saving.

Reference set update method

Solutions are included in the reference set by quality or diversity. The subset of quality solutions ($RefSet_1$) contains the b_1 best solutions and the subset of diverse solutions ($RefSet_2$) contains the b_2 diverse solutions. The initial *RefSet* consists of b_1 best solutions that belong to *P* and b_2 elements from P that maximize the minimum distance to *RefSet*. The distance between two solutions is calculated by adding the number of non-common arcs of each solution before the combination. If an arc belongs to more than one route, we add one unit for each non-common arc. We consider arc x_{i0} or x_{0i} only for routes with full truckloads $(0-i-0)$.

There are two main aspects that must be considered when updating the *RefSet*. The first one refers to the timing of the update. There are two possibilities: static update (S) and dynamic update (D). The second aspect deals with choosing the criteria for adding to and deleting elements from *RefSet*. We consider two types of updates: by quality (Q) and by quality and diversity (QD).

In the static update (S), the reference set doesn't change until all solutions combinations of *RefSet* were done. In the dynamic update (D), the reference set is updated when a new better solution is found.

In the quality update (Q), if a new solution is better than the worst element of $RefSet_1$, then the worst element of $RefSet_1$ is substituted by the new solution. In the quality and diversity update (QD), if a new solution is better than the worst element of $RefSet_1$, then the worst element of $RefSet_1$ is substituted by the new solution. If a new solution is worse than the worst element of $RefSet_1$, but it increases the minimum distance between the solutions in $RefSet_2$, the solution with the minimum distance in $RefSet_2$ is substituted by the new solution.

Solution combination method

The solution combination method is applied to all pairs of solutions in the current referent set.

This method is divided into two steps. Step 1 aims to combine only the common elements of the combined routes and step 2 supplies the remaining demand of the customers. The combination method was based on the ideas of Corberán et al. (2002) and Rego and Leão (2000).

Step 1

Let A be a solution with m routes and B a solution with n routes, where A_i is the *i*-th route for solution A, $i = \{1,...,m\}$, and B_k is the *k*-th route for solution B, $k = \{1,...,n\}$. The solutions A and B are combined. The combination procedure of step 1 is built from a matrix $A \times B$, where its component (A_i, B_k) have the number of common elements between the route *i* of solution *A* and route *k* of solution *B*. Firstly we define which routes are combined, that is, which component (A_i, B_k) is chosen. It begins with the component with a greater number of common elements. If there is a tie, the combination that minimizes the function *f*, $f = \sum_{j \in i,k} \left| y_{j,i}^{A} - y_{j,k}^{B} \right|$ is chosen, where *j* is the common element of the route i of solution *A* and of the route k in the solution *B*; $y_{j,i}^{A}$ is the quantity delivered to customer *j* from route i in solution *A* and $y_{j,i}^{B}$ is the quantity delivered to the customer *j* from route k in solution *B*.

The combined route is formed by the routes' common elements. The quantity delivered to each element is the smallest quantity between two solutions combined. The route of the combined solution is similar to the one of the best solution and, if there is a tie, we choose randomly the route of one of the solutions. Each combined route is excluded from the list (we delete the line and column referring to components A_i, B_k). The procedure follows while there are routes (which have not been combined yet) with common elements.

Step 2

This step aims to fulfill the remaining demand of customers who have already been served by some route in step 1, and fulfill the complete demand of customers who still do not belong to any route. This is done through an insertion procedure. Customer i is the farthest one from the depot, and is going to be inserted. First it is verified if it already belongs to any route combined, and after that two steps can be taken:

Step 2.1

If the customer i already belongs to at least one combined route (step 1 of the combination method), the one with larger idle capacity is chosen and it is delivered the minimum between the idle capacity of the vehicle and the customer demand. While the total customer demand i is not fulfilled and there is a route with idle capacity, in which the customer i is inserted, this procedure is repeated.

By the end of this step, if the total demand of customer i has not been fulfilled, we go to step 2.2. Otherwise, the next farther customer is chosen and inserted.

Step 2.2

If customer *i* does not belong to any combined route or the total demand of the customer i has not been fulfilled through step 2.1, step 2.2 is taken. Based on the initial solutions *A* and *B* (before combination), all the routes in which customer *i* is inserted are verified and also all the arcs in which $x_{ij} = 1$ (customer *i* is attended before customer *j*) or $x_{ji} = 1$ (customer *i* is attended after customer *j*). Therefore, for each *j* belonging to one of the combined routes in the step 2.1, except the depot $(j = 0)$, we calculate the cost for inserting customer *i* (addition of fixed cost, routing cost and time) before customer *j* (when $x_{ij} = 1$) or the cost for inserting customer *i* after customer *j* (when $x_{ji} = 1$). It is considered $x_{i0} = 1$ or $x_{0i} = 1$ only for routes with full truckload $(0 - i - 0)$.

For each possible position of customer insertion *i*, we deliver the minimum between the idle capacity of the vehicle and the demand of customer *i*, and we choose the one with the minimum cost, since the time window constraint has been respected. The procedure is repeated until the total demand of the customer *i* is fulfilled or while there is a position to insert customer *i*.

Once all the routes from solutions *A* and *B* where customer *i* is inserted have been verified, if the total demand has not been fulfilled, we add a new route. The procedure is repeated until the total demand of customer *i* is fulfilled.

The combination method guarantees the feasibility of the final solutions. For each combined solution we apply the improvement method.

The algorithm stops when $MaxIter = 5$ or when reaching the maximum time of 1 hour, until the last iteration is finished.

6. Experimental results

6.1 Problem sets

The Solomon's problem set consists of 100 customers with Euclidean distance. The percentage of customers with time windows varies between 25, 50, 75 and 100% according to the problem. The author considers six sets of problems: R1, R2, C1, C2, RC1 and RC2. In sets R1 and R2 the customers' position is created randomly through a uniform distribution. In sets C1 and C2 the customers are divided in groups. In sets RC1 and RC2, customers are in sub-groups, that is, part of the customers is placed randomly and part is placed in groups. Besides, R1, C1 and RC1 problems have got a short term planning horizon and, combined to lighter capacity vehicles, they allow only some customers (3-8) in each route. Sets R2, C2 and RC2 have got a long term planning horizon and, since they have got higher capacity vehicles, they are able to supply more than 10 customers per route.

In each set of problems, the customers' geographical distributions, the demand and the service time do not change. Therefore, on set R1, the problems from R101 to R104 are identical, except for the customer with time window percentage, which is 100% in problem R101, 75% in problem R102, 50% in problem R103 and 25% in problem R104. Problems from R105 to R108 are identical to problems from R101 to R104, the only difference is the time window interval. The same occurs for problems R106 to R112.

Ho and Haugland (2004) have changed the demands from Solomon's problems, aiming to allow more than one delivery per customer. We consider m the vehicle capacity and w_i the customer i's demand $i=1,\ldots, n$. Each demand is recalculated within the period $[lm, um]$, where $l < u$ are defined within the period [0, 1]. Therefore, for every $i \in C$, we define a new demand $w_i^{'} = lm + m((u-l)/(\overline{w}-\underline{w}))(w_i - \underline{w})$, where $\underline{w} = \min\{w_i : i \in C\}$ and $\overline{w} = \max\{w_i : i \in C\}$. The new demand $w_i^{'}$ is evened to the closest integer number.

For each problem set the authors use the following values of $[l, u]$:

l	u
0,01	0,50
0,02	1,00
0,50	1,00
0,70	1,00

Using this method, sets totaling 224 problems were created.

Table 1. Demand Values

6.2 Experiments on Solomon's problems with augmented demands

Initially, many different values for b_1, b_2, $PSize$, updating criteria and frequency were tested. The best results obtained were: $PSize = 30$, $b_1 = 5$, $b_2 = 5$, static (S) and quality and diversity (QD) update.

Table 2 compares the results of the algorithm presented in this chapter with the results from Ho and Haugland (2004), for each class of problems, considering different [l, u] values.

l, u		R1	C1	RC1
0.01, 0.50	VRPTWSD$^{H\&H}$	18.25/1471.49	12.22/1182.12	20.13/1965.05
	VRPTWSD$^{B\&Y}$	18.42/1475.54	**12.22/1160.74**	21.00/**1941.25**
0.02, 1.00	VRPTWSD$^{H\&H}$	35.00/2291.46	22.22/2168.57	40.00/3339.20
	VRPTWSD$^{B\&Y}$	35.83/2302.58	24.00/**2009.37**	41.75/3425.96
0.50, 1.00	VRPTWSD$^{H\&H}$	67.00/4040.67	61.00/3979.78	70.00/5453.10
	VRPTWSD$^{B\&Y}$	69.50/**4035.84**	**60.75/3975.49**	73.75/**5231.85**
0.70, 1.00	VRPTWSD$^{H\&H}$	79.00/4581.54	77.00/4962.28	81.00/6095.20
	VRPTWSD$^{B\&Y}$	82.75/**4464.85**	**76.88/4950.81**	82.50/**6013.92**
l, u		R2	C2	RC2
0.01, 0.50	**VRPTWSD$^{H\&H}$**	**18.00/1430.62**	11.13/1174.29	20,00/1946,07
	VRPTWSD$^{B\&Y}$	18.00/1425.40	11.75/1180.34	21.00/1941.42
0.02, 1.00	**VRPTWSD$^{H\&H}$**	35.00/2318.04	22.00/1995.59	39,00/3419,85
	VRPTWSD$^{B\&Y}$	35.82/**2314.65**	23.13/**1993.47**	41.50/**3410.65**
0.50, 1.00	VRPTWSD$^{H\&H}$	68.00/4059.26	61.00/4268.02	71,00/5546,20
	VRPTWSD$^{B\&Y}$	69.27**/4055.29**	**60.88/4259.14**	**71.00/5498.32**
0.70, 1.00	VRPTWSD$^{H\&H}$	80.00/4574.17	77.00/5246.12	82,00/6155.49
	VRPTWSD$^{B\&Y}$	82.82/4625.87	**76.88/5214.79**	83.25/6217.43

Legend
VRPTWSD$^{H\&H}$: Ho and Haugland's VRPTWSD results (2004)
VRPTWSD$^{B\&Y}$: Belfiore and Yoshizaki's VRPTWSD results (2008)

Table 2. Comparison to Ho and Haugland (2004) results for Solomon's modified demands.

According to table 2, we can conclude that, in 6 classes of problems, the scatter search metaheuristic's average has overcome the best results from Ho and Haugland (2004). Besides, in other 11 problems, it was possible to reduce the average distance traveled, but the average number of used vehicles was higher. It has also been verified that the best results were found for $[l,u]=[0.50,\ 1.00]$ e $[0.70,\ 1.00]$, because higher demand values increases the possibility of split deliveries.

Table 3 shows the average processing time (seconds), for the classes of problems R1, C1, RC1, R2, C2 and RC2, considering different values of [*l*, *u*].

l, *u*	**R1**	**R2**	**C1**	**C2**	**RC1**	**RC2**
0.01, 0.50	807	982	517	1121	678	1030
0.02, 1.00	840	1022	785	888	905	926
0.50, 1.00	1025	1014	574	876	1014	885
0.70, 1.00	812	899	755	922	901	957

Table 3. Average processing time (seconds) for Ho and Haugland (2004) modified demands.

7. Conclusions and future research

We have proposed a scatter search algorithm for the Vehicle Routing Problem with Time Windows and Split Deliveries. The initial solution is an extension of Solomon's sequential insertion heuristic I1. The scatter search framework provided a means to combine solutions, diversify, and intensify the metaheuristic search process.

The algorithm was applied on Solomon's problems with augmented demands. For the problems sets, many parameters of scatter search metaheuristic was tested: *PSize*, b_1, b_2, updating criteria and updating frequency.

Computational testing revealed that our algorithm matched some of the best solutions on the Solomon's problem set with augmented demands. It has also been verified that the best results were found for $[l,u]=[0.50,\ 1.00]$ e $[0.70,\ 1.00]$, because with higher demand values higher is the possibility of split delivery.

As future research, other constructive heuristic can be applied as an initial solution. Other improvement heuristics can also be tested. Besides, the Scatter Search metaheuristic proposed can be adapted for other problems, as VRP with heterogeneous fleet, multiple depots, etc.

8. Acknowledgments

This research was partially funded by CAPES. The authors also thank Vanzolini Foundation for giving support to presentations.

9. References

Archetti, C, M W P Savelsbergh and M G Speranza (2003a), Worst-Case Analysis of Split Delivery Routing Problems, Department of Quantitative Methods, University of Brescia and School of Industrial and Systems Engineering, Georgia Institute of Technology, Accept in *Transportation Science*.

Archetti, C, A Hertz and M G Speranza (2003b), A Tabu Search Algorithm for the Split Delivery Vehicle Routing Problem, Les Cahiers du GERAD, Accept in *Transportation Science*.

Archetti, C, M Mansini and M G Speranza (2005), Complexity and Reducibility of the Skip Delivery Problem, *Transportation Science* 39, 182-187.

Belenguer, J M, M C Martinez and E Mota (2000), A Lower Bound for the Split Delivery Vehicle Routing Problem. *Operations Research* 48, 801-810.

Belfiore, P P (2006), Scatter Search para Problemas de Roteirização de Veículos com Janelas de Tempo e Entregas Fracionadas. Tese (Doutorado), Universidade de São Paulo.

Corberán, A, E Fernández, M Laguna, R Martí (2002), Heuristic solutions to the problem of routing school buses with multiple objectives, *Journal of the Operational Research Society* 53, 427-435.

Dror, M and P Trudeau (1989), Savings by Split Delivery Routing, *Transportation Science* 23, 141-145.

Dror, M and P Trudeau (1990), Split Delivery Routing, *Naval Research Logistics* 37, 383-402.

Dror, M, G Laporte and P Trudeau (1994), Vehicle routing with split deliveries, *Discrete Applied Mathematics* 50, 229-254.

Feo, T A and M G C Resende (1989), A probabilistic heuristic for a computationally difficult set covering problem, *Operations Research Letters* 8, 67-71.

Frizzell, P W and J W Giffin (1992), The bounded split delivery vehicle routing problem with grid network distances, *Asia Pacific Journal of Operational Research* 9, 101-106.

Frizzell, P W and J W Giffin (1995), The Split Delivery Vehicle Scheduling Problem with Time Windows and Grid Network Distances, *Computers & Operations Research* 22, 655-667.

Glover, F (1977), Heuristics for Integer Programming Using Surrogate Constraints, *Decision Sciences* 8, 156-166.

Glover, F (1998), A Template for Scatter Search and Path Relinking, in *Artificial Evolution*, Lecture Notes in Computer Science 1363, Hao, J.-K., E. Lutton, E. Ronald, M. Schoenauer and D. Snyers (Eds.), Springer-Verlag, p13-54.

Ho, S C and D Haugland (2004), A tabu search heuristic for the vehicle routing problem with time windows and split deliveries, *Computers & Operations Research* 31, 1947-1964.

Lenstra, J K and A H G Rinnooy Kan (1981), Complexity of Vehicle and Scheduling Problems, *Networks* 11, 221-227.

Mullaseril, P A, M Dror and J Leung (1997), Split-delivery Routing Heuristics in Livestock Feed Distribution, *Journal of the Operational Research Society* 48, 107-116.

Rego, C and P A Leão (2000), Scatter Search Tutorial for Graph-Based Permutation Problems, Hearin Center for Enterprise, University of Mississipi, HCES-10-00, USA, in *Metaheuristic Optimization via Memory and Evolution: Tabu Search and Scatter Search*, Rego, C and B Alidaee, B (Eds.), Kluwer Academic Publishers, p1-24, 2005.

Salhi, S and G K Rand (1993), Incorporating vehicle routing into the vehicle fleet composition problem, *European Journal of Operational Research* 66, 313-330.

Solomon, M M (1987), Algorithms for the Vehicle Routing and Scheduling Problems with Time Windows Constraints, *Operations Research* 35, 254-265.

Solomon, M M and J Desrosiers (1988), Time Window Constrained Routing and Scheduling Problem, *Transportation Science* 22, 1-13.

Vehicle Routing Problem

Edited by Tonci Caric and Hrvoje Gold

ISBN 978-953-7619-09-1
Hard cover, 142 pages
Publisher InTech
Published online 01, September, 2008
Published in print edition September, 2008

The Vehicle Routing Problem (VRP) dates back to the end of the fifties of the last century when Dantzig and Ramser set the mathematical programming formulation and algorithmic approach to solve the problem of delivering gasoline to service stations. Since then the interest in VRP evolved from a small group of mathematicians to a broad range of researchers and practitioners from different disciplines who are involved in this field today. Nine chapters of this book present recent improvements, innovative ideas and concepts regarding the vehicle routing problem. It will be of interest to students, researchers and practitioners with knowledge of the main methods for the solution of the combinatorial optimization problems.

How to reference
In order to correctly reference this scholarly work, feel free to copy and paste the following:

Patrícia Belfiore, Hugo Tsugunobu and Yoshida Yoshizaki (2008). Scatter Search for Vehicle Routing Problem with Time Windows and Split Deliveries, Vehicle Routing Problem, Tonci Caric and Hrvoje Gold (Ed.), ISBN: 978-953-7619-09-1, InTech, Available from: http://www.intechopen.com/books/vehicle_routing_problem/scatter_search_for_vehicle_routing_problem_with_time_windows_and_split_deliveries

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

© 2008 The Author(s). Licensee IntechOpen. This chapter is distributed under the terms of the Creative Commons Attribution-NonCommercial-ShareAlike-3.0 License, which permits use, distribution and reproduction for non-commercial purposes, provided the original is properly cited and derivative works building on this content are distributed under the same license.

IntechOpen

IntechOpen