

# A Tabu Search Heuristic for the Vehicle Routing Problem with Soft Time Windows

ÉRIC TAILLARD

*Istituto Dalle Molle di Studi sull'Intelligenza Artificiale (IDSIA), Corso Elvezia 36, CH-6900 Lugano, Switzerland*

PHILIPPE BADEAU

*Centre Universitaire des Sciences et Techniques, Université Blaise Pascal, Clermont Ferrand II, Campus Universitaire des Cézeaux, 63174 Aubière Cedex, France*

MICHEL GENDREAU\*, FRANÇOIS GUERTIN and JEAN-YVES POTVIN\*

*Centre de recherche sur les transports, Université de Montréal, C.P. 6128, Succ. Centre-Ville, Montréal, Québec H3C 3J7, Canada and \*Département d'informatique et de recherche opérationnelle, Université de Montréal, C.P. 6128, Succ. Centre-Ville, Montréal, Québec H3C 3J7, Canada*

*This paper describes a tabu search heuristic for the vehicle routing problem with soft time windows. In this problem, lateness at customer locations is allowed although a penalty is incurred and added to the objective value. By adding large penalty values, the vehicle routing problem with hard time windows can be addressed as well. In the tabu search, a neighborhood of the current solution is created through an exchange procedure that swaps sequences of consecutive customers (or segments) between two routes. The tabu search also exploits an adaptive memory that contains the routes of the best previously visited solutions. New starting points for the tabu search are produced through a combination of routes taken from different solutions found in this memory. Many best-known solutions are reported on classical test problems.*

An important component of many distribution systems is the routing of vehicles to service customers. For various applications, like bank deliveries, postal deliveries, or school bus routing, a time interval or time window is also associated with each customer to constrain the time of service. In practice, these time windows are often relaxed to allow for early or late arrivals at customer locations. In the literature, such problems are referred to as vehicle routing problems with soft time windows, or VRPSTWs.

From a graph theoretical perspective, the VRPSTW can be stated as follows. Let  $G = (V, E)$  be a complete undirected graph with vertex set  $V = \{v_0, v_1, v_2, \dots, v_n\}$  and edge set  $E = \{(v_i, v_j) : v_i, v_j \in V, i < j\}$ . In this graph, vertex  $v_0$  is the depot and the other vertices are customers to be serviced. Each vertex is associated with:

- a fixed quantity  $q_i$  of goods to be delivered (with  $q_0 = 0$  at vertex  $v_0$ );
- a time window  $[e_i, l_i]$ , where  $e_i$  and  $l_i$  are the lower and upper bound of the time window, respectively (with  $e_0$  the earliest start time and  $l_0$  the latest end time of each vehicle route, respectively);
- a service time  $s_i$  for unloading the goods (with  $s_0 = 0$  at vertex  $v_0$ ).

Finally, a symmetric distance matrix  $D = (d_{ij})$  that satisfies the triangle inequality is defined on  $E$  (with travel times  $t_{ij}$  proportional to the distances).

Given a fixed size fleet of  $m$  identical vehicles, each with capacity  $Q$ , the goal is to find a set of minimum cost vehicle routes, originating from and terminating at the depot  $v_0$ , such that:

- each vehicle services one route;

- each vertex  $v_i$ ,  $i = 1, \dots, n$  is visited exactly once;
- the quantity of goods to be delivered on a route never exceeds the vehicle capacity  $Q$ ;
- the start time of each vehicle route is greater than or equal to  $e_0$ ;
- the end time of each vehicle route is less than or equal to  $l_0$ ;
- the time of beginning of service  $b_i$  at each vertex  $v_i$ ,  $i = 1, \dots, n$  is greater than or equal to the time window's lower bound  $e_i$ ; if the arrival time  $t_i$  is less than  $e_i$ , a waiting time  $w_i = (e_i - t_i)$  is incurred.

The objective function  $f$  to be minimized over the set of feasible solutions  $S$  is:

$$f(s) = \sum_{k=1}^m d_k + \sum_{i=1}^n \alpha_i(t_i - l_i)^+, \quad s \in S, \quad (1)$$

where  $y^+ = \max\{0, y\}$ ,  $d_k$  is the total distance traveled on route  $k$ ,  $k = 1, \dots, m$ , and  $\alpha_i$  is a lateness penalty coefficient associated with vertex  $v_i$ ,  $i = 1, \dots, n$ .

This definition implies that each route must satisfy the following hard constraints:

- a. the total load on a route cannot exceed the capacity of the vehicle servicing the route;
- b. each vehicle must start and terminate its route within the time window associated with the depot.

Furthermore, a soft time window constraint is found at each customer location. The time window is "soft" because the vehicle can arrive before the lower bound or after the upper bound. If the vehicle arrives too early, it must wait to start its service. If the vehicle is too late, a penalty for lateness is incurred. That is, the upper bound of the time window is relaxed into the objective function in a Lagrangean relaxation fashion. According to objective function (1), the penalty coefficients can be adjusted to each customer. For example, large coefficients can be associated with customers with rather strict time requirements and small coefficients with customers with some flexibility.

Another closely related problem, known as the vehicle routing problem with hard time windows, or VRPHTW, does not allow late services. That is, a vehicle must arrive before the time window's upper bound at each customer location. Furthermore, the fleet size is typically a decision variable. Quite often, a hierarchical objective function is associated with such problems: first, minimize the number of vehicles and then, for the same number of vehicles, minimize the total distance traveled. Most algorithms

reported in the literature address the VRPHTW (with the exception of KOSKOSIDIS, POWELL, and SOLOMON (1992)). However, there are many good reasons for solving the VRPSTW:

- a. The VRPSTW model is more general and includes the VRPHTW as well. The latter problem can be solved by appropriately raising the lateness penalty coefficients.
- b. The VRPSTW more closely models situations found in practice and can be used to find a good trade-off between fleet size and service quality to customers.
- c. Since there are fewer hard constraints, feasible solutions are easier to find. In cases where the set of feasible solutions for the "hard" version of the problem is empty (assuming a fixed size fleet), a solution with a few time window violations can still be produced.

However, the generality of the VRPSTW does not come for free. For example, hard time windows can be exploited within the VRPHTW model to quickly filter out infeasible solutions. Different techniques are reported in the literature to perform time window reduction (DESROCHERS, DESROSIERS, and SOLOMON, 1992) and to check solution feasibility in constant time after the insertion of a new customer or after a local modification to the solution (SAVELSBERGH, 1986, 1990, 1992; DUHAMEL, GARCIA, and POTVIN, 1995). The total distance of the new solution can also be easily evaluated in constant time. Obviously, the filtering mechanism cannot be applied to soft time windows, and the penalty component in objective function (1) does not lend itself to a constant time evaluation. Rather, approximations must be used to achieve this result (see Section 2).

The paper is organized as follows. Section 1 first presents a brief literature review on vehicle routing problems with time windows. Then, Section 2 introduces our neighborhood structure. Section 3 describes the problem-solving approach and provides a detailed explanation of its different components. Finally, Section 4 reports computational results on standard test problems.

## 1. LITERATURE REVIEW

BECAUSE OF THEIR wide applicability in practical settings, vehicle routing problems with time windows have been intensively studied over the last ten years. These problems being NP-hard, a large spectrum of heuristics are reported in the literature (mostly for the VRPHTW). The interested reader will find excellent surveys in DESROCHERS et al.

(1988); SOLOMON and DESROSIERS (1988); DESROSIERS et al. (1995).

Generally speaking, these problem-solving approaches can be classified as follows:

- a. *Exact algorithms based on branch-and-bound techniques* (KOLEN, RINNOOY KAN, and TRIENEKESN, 1987; Desrochers et al., 1992). Recently, Desrochers et al. found the optimum on a few problems with 100 customers. In this case, a column generation scheme was used to solve the linear programming relaxation of a set partitioning problem. The columns were generated by solving a shortest path problem with time windows.
- b. *Route construction heuristics*. Different sequential insertion heuristics are reported in SOLOMON (1987), and parallel route construction procedures are found in POTVIN and ROUSSEAU (1993); RUSSELL (1995). A worst case analysis of some of these route construction heuristics is provided in SOLOMON (1986).
- c. *Route improvement heuristics*. Edge exchange heuristics for problems with time windows may be found in OR (1976); BAKER and SCHAFFER (1988); SOLOMON, BAKER, and SCHAFFER (1988); THOMPSON and PSARAFTIS (1993); Potvin and Rousseau (1995). Efficient implementations for speeding up the screening of infeasible solutions and the evaluation of the objective function are reported in Savelsbergh (1986, 1990, 1992); Solomon et al. (1988); Duhamel et al. (1995).
- d. *Composite heuristics*. These heuristics mix both route construction and route improvement procedures (see DERIGS and GRABENBAUER, 1993; KONTORAVDIS and BARD, 1995; Russell, 1995).
- e. *Optimization-based heuristics*. In Koskosidis et al. (1992), the authors exploit a mixed integer programming model. Through heuristic means, the original problem is decomposed into an assignment/clustering subproblem and a series of routing and scheduling subproblems.
- f. *Metaheuristics*. This new generation of heuristics is at the core of many recent developments. Near optimal solutions to the VRPHTW can now be produced with tabu search (BARNES and CARLTON, 1995; CARLTON, 1995; ROCHAT and TAILLARD, 1995; POTVIN et al., 1996), simulated annealing (CHIANG and RUSSELL, 1993), and genetic algorithms (THANGIAH, NYGARD, and JUELL, 1991; BLANTON and WAINWRIGHT, 1993; THANGIAH, 1993; THANGIAH, OSMAN, and SUN, 1994; POTVIN and BENGIO, 1996). An application of tabu search on a real-world vehicle routing problem with many side constraints, including

time windows, may also be found in SEMET and TAILLARD (1993).

Finally, asymptotically optimal heuristics are reported in BRAMEL and SIMCHI-LEVI (1993); BRAMEL, LI, and SIMCHI-LEVI (1993).

Tabu search heuristics are of particular interest to us. In Potvin et al. (1996), a standard tabu search heuristic based on 2-opt\* exchanges (POTVIN and ROUSSEAU, 1995) and Or-opt exchanges (Or, 1976) is described. The search alternates between the two neighborhoods and includes intensification procedures aimed at eliminating routes with only a few customers. The tabu search in (Rochat and Taillard, 1995) exploits a neighborhood based on the exchange of customers between routes. An adaptive memory records the best routes produced during the search. This memory is then used to construct new starting solutions for the tabu search (see Section 3). Finally, Carlton (1995) describes a reactive tabu search that dynamically adjusts its parameter values based on the current search status. This approach is applied to several types of problems with time windows, including the VRPHTW. Its robustness comes from a simple neighborhood structure which can be easily adapted to different problems. Namely, each customer is removed and reinserted at some other location in the current solution.

The main contribution of our work is the development of a new neighborhood structure, coupled with approximation methods for evaluating each solution in constant time (in the presence of soft time windows). This is the topic of the next section.

## 2. THE NEIGHBORHOOD STRUCTURE

IN THIS SECTION, edge exchange heuristics are introduced in general terms. Then, our exchange method is described. Finally, simplifications and approximations aimed at speeding up the examination of the neighborhood are discussed.

### 2.1 Edge Exchange Heuristics

Edge exchange heuristics, like 2-opt or 3-opt (LIN, 1965), are widely used to improve vehicle routing solutions. Typically, these methods are embedded within a local search framework of the following type:

1. Start with an initial solution and define this solution to be the current solution.
2. Generate all solutions in the neighborhood of the current solution by applying all modifications associated with the method under consideration.
3. Select the best solution in this neighborhood and

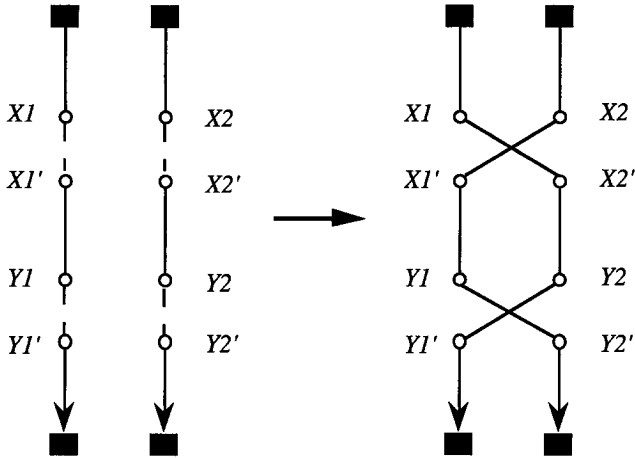


Fig. 1. The CROSS exchange.

define this solution to be the new current solution.

4. Go back to step 2.

Although the number of iterations needed to reach a local optimum can grow exponentially with the problem size (on carefully designed “pathological” problems), the neighborhood generated in step 2 is typically polynomial. This kind of iterative framework is exploited, in particular, within the tabu search heuristic (GLOVER, 1989; GLOVER, 1990).

## 2.2 A New Exchange Heuristic

In this section, a new exchange heuristic is introduced which generalizes two edge exchange heuristics previously proposed for problems with time windows. This method, called the CROSS exchange, is illustrated in Figure 1. In this figure, the black square stands for the depot and the white circles are customers along the routes (with the depot duplicated at the start and at the end of each route). First, the two edges  $(X1, X1')$  and  $(Y1, Y1')$  are removed from the first route while the edges  $(X2, X2')$  and  $(Y2, Y2')$  are removed from the second route. Then, the segments  $X1'-Y1$  and  $X2'-Y2$ , which may contain an arbitrary number of customers, are swapped by introducing the new edges  $(X1, X2')$ ,  $(Y2, Y1')$ ,  $(X2, X1')$ , and  $(Y1, Y2')$ .

Note that the time window constraints define an implicit orientation on each route. In this example,  $Y1$  (respectively,  $Y2$ ) is visited after  $X1'$  (respectively,  $X2'$ ). So, the segment  $X1'-Y1$  (respectively,  $X2'-Y2$ ) has the same orientation after the move.

As illustrated in Figures 2(a) and (b), the 2-opt\* (Potvin and Rousseau, 1995) and Or-opt (Or, 1976) are special cases of this operator. The 2-opt\* only exchanges two edges taken from two different routes, and is obtained when  $Y1$  and  $Y2$  are directly

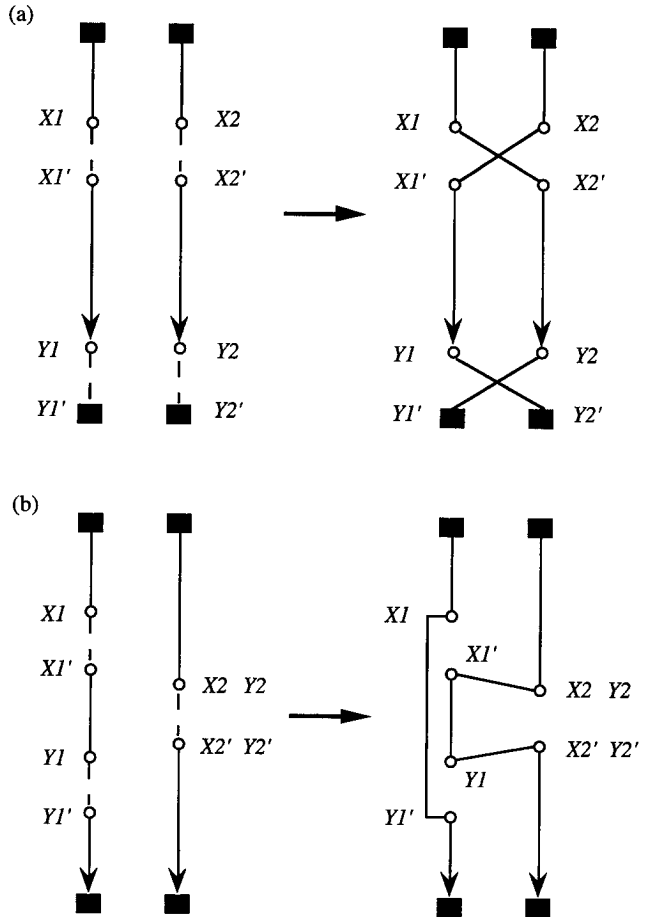


Fig. 2. Special cases of the CROSS exchange: (a) 2-opt\* (b) Or-opt.

connected to the depot. An Or-opt exchange moves a sequence of three consecutive customers or less from one route to another. It is obtained, for example, by setting  $X2 = Y2$  and  $X2' = Y2'$  so that an empty segment is removed from the second route. Because the sequence removed from the first route must contain three customers at most,  $Y1$  is either  $X1'$ , or the first successor of  $X1'$ , or the second successor of  $X1'$ . In a few cases, a CROSS exchange can create empty routes. In Figure 2(b), for example, if  $X1'$  is the first customer on the route while  $Y1$  is the last customer on the same route, the entire route is inserted between  $X2$  and  $X2'$ .

A CROSS exchange preserves the orientation of the routes, which is a nice feature for problems with time windows. Furthermore, by selecting an exchange that leads to the largest improvement to the current solution, the swapping of segments that are close from a spatial and temporal point of view is favored. One drawback is the complexity of this method. By restricting the exchanges to route segments whose length is at most  $L$ , where  $L$  is a

parameter, there are  $O(nL)$  ways to select the first two edges in any given route (on a problem of size  $n$ ) and  $O(nL)$  ways to select two edges in another route. Hence the overall complexity is

$$O(nL) \times O(nL) = O(n^2L^2).$$

It is worth noting that CROSS exchanges, while generalizing previous exchange methods, are also a special case of  $\lambda$ -interchanges (OSMAN, 1993). The latter method selects two subsets of customers (whose cardinality is less than or equal to  $\lambda$ ) from two different routes and exchanges them. However, the size of this neighborhood quickly becomes very large, even for small values of  $\lambda$ . A useful and manageable neighborhood can be obtained by restricting the exchanges to consecutive customers in both routes (thus leading to CROSS exchanges). Our method is also closely related to the chain-exchange procedure described in (FAHRION and WREDE, 1990). Here, two chains or segments of different lengths are removed from the current solution. Then, each chain is reinserted at the location that introduces the smallest detour. Looking for the best insertion place of each chain, however, increases the complexity by a factor of  $O(n)$ .

In the following section, simplification and approximation procedures for reducing the number of CROSS exchanges and speeding up their evaluation will be presented.

### 2.3 Generating the Neighborhood

In this section, we explain how each move can be evaluated in constant time using approximations. Then, we discuss techniques aimed at discarding moves that are unlikely to yield any improvement.

#### 2.3.1 Evaluating a Move

To simplify the following description, we will assume that  $\alpha_i = \alpha$ ,  $i = 1, \dots, n$ . In this case, the objective function (1) becomes:

$$f(s) = \sum_{k=1}^m d_k + \alpha \sum_{i=1}^n (t_i - l_i)^+, \quad s \in S. \quad (2)$$

The second summation in (2) is the total lateness of the solution (similarly, the total lateness on a route would be the summation over the subset of customers found on this route). When we evaluate a CROSS exchange, we are interested in the difference  $\Delta f$  between the value of the neighboring solution and the value of the current solution. An improvement is found when  $\Delta f$  is negative. Here,

$$\Delta f = \Delta d + \alpha \Delta l,$$

where  $\Delta d$  and  $\Delta l$  are the modification to the total distance and total lateness of the solution, respectively. It is easy to evaluate  $\Delta d$  in constant time. Namely, the summation over the lengths of the edges that are removed from the solution is subtracted from the summation over the lengths of the edges that are introduced into the solution. For the lateness penalty component of the objective, however, the impact of the CROSS exchange must be propagated to the end of the route, thus preventing a constant time evaluation. The modification to the total lateness of a solution is obtained by summing up the modifications to the total lateness of both routes involved in the CROSS exchange. Here, the approximate evaluation procedure will be illustrated on a single route, namely the new route servicing customers  $X1$ ,  $X2'$ ,  $Y2$ , and  $Y1'$  in Figure 1. Considering only the contribution of this route,  $\Delta l$  is approximated through the following equation:

$$\tilde{\Delta l} = \Delta l_{X2'-Y2} + \tilde{\Delta l}_{Y1'-\text{depot}}, \quad (3)$$

where

$$\tilde{\Delta l}_{Y1'-\text{depot}} = g_{Y1'}(\Delta b_{Y1'}).$$

The first component of the summation is the modification to the total lateness on the route segment between  $X2'$  and  $Y2$ . This component can be evaluated exactly in constant time. The second component of the summation is the modification to the total lateness on the last route segment between  $Y1'$  and the depot. This component cannot be evaluated exactly in constant time and is approximated through function  $g_{Y1'}$ . We will now go further into the details of equation (3).

**Exact evaluation of  $\Delta l_{X2'-Y2}$ .** In the following, we explain how to update the time of beginning of service at each customer on route segment  $X2'-Y2$  in constant time. Through this result  $\Delta l_{X2'-Y2}$  can also be easily evaluated in constant time. The new link that now connects  $X1$  to  $X2'$  (after the CROSS exchange) implies that the modification  $\Delta b_{X2'}$  to the time of beginning of service at customer  $X2'$  is

$$\begin{aligned} \Delta b_{X2'} &= b_{X2'}^{\text{new}} - b_{X2'} \\ b_{X2'}^{\text{new}} &= \max\{e_{X2'}, b_{X1} + s_{X1} + t_{X1X2'}\} \end{aligned} \quad (4)$$

Now, the impact on customer  $Y2$  can be evaluated by propagating  $\Delta b_{X2'}$  along the route segment  $X2'-Y2$ . This propagation is necessary due to possible waiting times along the segment that can absorb totally or partially  $\Delta b_{X2'}$ . Fortunately, the exploration of the neighborhood is done in such a way that

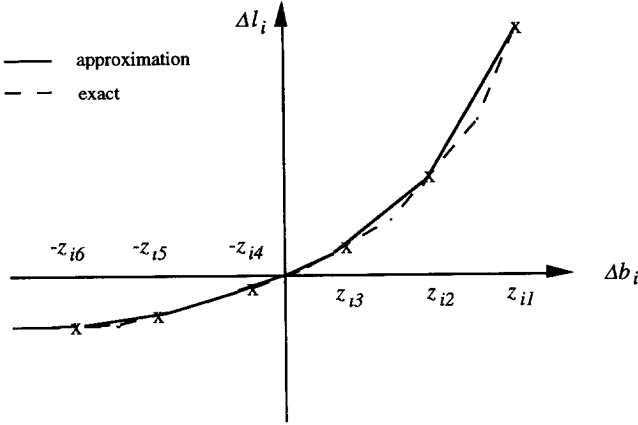


Fig. 3. Approximation function  $g_i$  at customer location  $i$ .

this propagation can be avoided. Four nested loops generate the entire neighborhood, namely:

**For**  $X1$  from depot to last customer  
 Set  $X1'$  to the immediate successor of  $X1$ ,  
 ...  
**For**  $X2$  from depot to last customer  
 Set  $X2'$  to the immediate successor of  $X2$ ;  
 ...  
**For**  $Y1$  from  $X1$  to depot  
 Set  $Y1'$  to the immediate successor of  $Y1$ ;  
 ...  
**For**  $Y2$  from  $X2$  to depot  
 Set  $Y2'$  to the immediate successor of  $Y2$ ;  
 ...  
 (5)

For fixed  $X1$ ,  $X2$ , and  $Y1$ , the first segment to be moved from the second route to the first route only contains customer  $Y2 = X2'$ , that is, the immediate successor of customer  $X2$ . Then, the segment is progressively extended by setting  $Y2$  to the second successor of  $X2$ , the third successor of  $X2$ , etc. Hence,  $\Delta b_{Y2}$  can be easily evaluated in constant time by exploiting the value computed at the previous neighboring solution (given that  $Y2$  in the previous neighboring solution is the predecessor of  $Y2$  in the current neighbor).

**Approximation of  $\Delta l_{Y1'-\text{depot}}$ .** With  $\Delta b_{Y2}$ ,  $\Delta b_{Y1'}$  is easily evaluated in constant time via the new link  $(Y2, Y1')$ . The impact of  $\Delta b_{Y1'}$  on  $\Delta l_{Y1'-\text{depot}}$  is then approximated through function  $g_{Y1'}$  (see equation (3)). Such an approximation function is found at each customer location  $i$ ,  $i = 1, \dots, n$ , and is constructed as follows. First, the modification  $\Delta l_i$  to the total lateness on the route servicing customer  $i$  in the current solution is evaluated exactly for a few  $\Delta b_i$  values, noted  $z_{ij}$ ,  $j = 1, \dots, Z$ . Then, a piecewise linear function is produced by interpolation, as illustrated in Figure 3 for  $Z = 6$ . This function is

quite different on the positive and negative sides. Increasing  $\Delta b_i$  on the positive side, thus shifting the time of beginning of service later in time, increases the total lateness on the route. On the other hand, decreasing  $\Delta b_i$  on the negative side, thus shifting the time of beginning of service earlier in time, causes no additional effect over a certain threshold (for example, when the lateness on the route is completely eliminated). The dotted curve in Figure 3 illustrates what could be the true function.

An update of these approximation functions is performed only when the best CROSS exchange is applied to the current solution to produce a new solution (i.e., these functions remain the same during the neighborhood evaluation). Furthermore, the update is only performed at customer locations found on the two routes involved in the CROSS exchange.

With respect to the implementation,  $Z = 6$  values were used to construct the approximation function  $g_i$  at each customer location. A larger number of values would provide a better approximation of the true function, but would also require more computations. Preliminary experiments have shown that  $Z = 6$  offers a good compromise (a correlation coefficient varying from 0.5 to 0.8 has been observed between the approximation and the true modification). The value  $z_{i1}$  was set to the maximum positive shift to the time of beginning of service observed at any customer location during the search. The values  $z_{i2}$  and  $z_{i3}$  were obtained by dividing  $z_{i1}$  by 50 and 2500, respectively. In our test problems,  $z_{i2}$  is typically around 25 time units, and  $z_{i3}$  around 0.5 time units. On the negative side,  $z_{i6}$  was set to the sum of lateness at customer  $i$  and all its successors on the route. Note that  $\Delta l_i$  cannot be reduced by more than  $z_{i6}$  time units through a shift (earlier in time) at customer  $i$ . The values  $z_{i5}$  and  $z_{i4}$  were set by dividing  $z_{i6}$  by 50 and 2500, respectively.

Although a good exchange according to the approximation is likely to be good according to the true objective, the  $P$  best solutions in the neighborhood (according to the approximation) are kept for further consideration. These solutions are then evaluated *exactly* and the best one (according to the exact evaluation) becomes the new current solution for the next iteration.

### 2.3.2 Feasibility of a Move

The capacity constraints and hard time window at the depot are easily checked in constant time. In the latter case, the latest feasible time of beginning of service is maintained at each customer location in the current solution (Kontoravdis and Bard, 1995). The solution obtained after a CROSS exchange vio-

lates the time window at the depot if the new time of beginning of service at customer  $Y1'$  (or  $Y2'$ ) in Figure 1 occurs after this limit.

### 2.3.3 Neighborhood Reduction

The size of the neighborhood can be reduced by discarding moves that are unlikely to yield any improvement. In (5), four nested loops generate the entire neighborhood. At the level of  $Y2$ , the loop is interrupted when a monotonous degradation of the (approximate) objective value is observed over three consecutive iterations. The same approach is applied at the level of  $X2$ . In this case, the objective value associated with  $X2$  is the best solution found after iterating over  $Y1$  and  $Y2$ . Because the  $X2$  loop is at the second level within the nested structure, its interruption is much more beneficial in terms of computation time than the interruption at the level of  $Y2$ . The interruption at  $X2$  mostly occurs when the segments to be swapped cover different time periods. In this case, huge lateness is typically created on one route after the exchange. Hence, it makes sense to discard such "incompatible" moves.

### 2.3.4 Approximation Matrix

The overall computation time can be reduced by storing previous calculations in a data structure known as the approximation matrix. Each entry  $(i, j)$ ,  $i, j = 1, \dots, m$ ,  $i < j$ , in this upper triangular matrix is associated with a pair of routes  $i$  and  $j$ . It contains information about the best CROSS exchange (according to the approximation) for this pair of routes, namely, the value of the new solution as well as the edges introduced or removed from both routes to produce this solution. When a particular exchange involving routes  $i'$  and  $j'$  is accepted, only the information stored in row  $i'$  and column  $j'$  of the approximation matrix is updated (because only these two routes are modified by the exchange). That is, the information associated with any pair of routes that does not include neither route  $i'$  nor route  $j'$  remains the same and is not recalculated.

## 2.4 Intra-Route Exchanges

The CROSS exchanges of Section 2.2 swap customers between two routes. However, intra-route optimization is equally important to find good solutions. To optimize individual routes, the CROSS neighborhood is enlarged by including exchanges that apply to a single route. These exchanges are similar to those defined on a pair of routes. Namely, two edges are removed from a given route, and the segment between the two edges is moved at another location *within the same route*. This approach generalizes the Or-opt exchanges (Or, 1976), by allow-

ing the relocation of segments with more than three customers.

## 3. THE PROBLEM-SOLVING METHODOLOGY

THE MAIN IDEAS in this section, like the adaptive memory or the decomposition/reconstruction (D&R) procedure, have already been proposed in TAILLARD (1993); Rochat and Taillard (1995). The following presentation will thus omit several details which can be found in the above papers. The tabu search heuristic presented in this section follows the guidelines provided in Glover (1989, 1990). It is embedded within the following problem-solving methodology:

*Construct  $I$  different solutions using a stochastic insertion heuristic. Then, apply the tabu search heuristic to each solution and store the resulting routes in the adaptive memory.*

*While the stopping criterion is not met do:*

*Construct an initial solution from the routes found in the adaptive memory, and define this solution to be the current solution.*

*For  $W$  iterations do:*

*Decompose the current solution into  $C$  disjoint subsets of routes.*

*Apply a tabu search on each subset of routes.*

*Reconstruct a complete solution by merging the new routes found by the tabu search, and define this solution to be the new current solution.*

*Store the routes of the current solution in the adaptive memory.*

*Apply a postoptimization procedure to each individual route of the best solution.*

This methodology includes a number of different components like an initialization procedure, the D&R, an adaptive memory, and the tabu search as such. These components will now be described.

### 3.1 Initialization

To fill the adaptive memory with different types of routes, a randomized insertion heuristic is used to construct  $I$  different initial solutions. First, the routes are initialized by randomly selecting  $m$  seed customers. Hence, each initial route only services a single customer. The remaining unrouted customers are then inserted one by one (in a random order) at the location that minimizes Solomon's insertion cost over the current set of routes (see Section 3.4 and the description of heuristic I1 in Solomon (1987)). This simple construction heuristic does not produce high quality solutions. Accordingly, the tabu search heuristic presented in Section 3.4 is applied to each

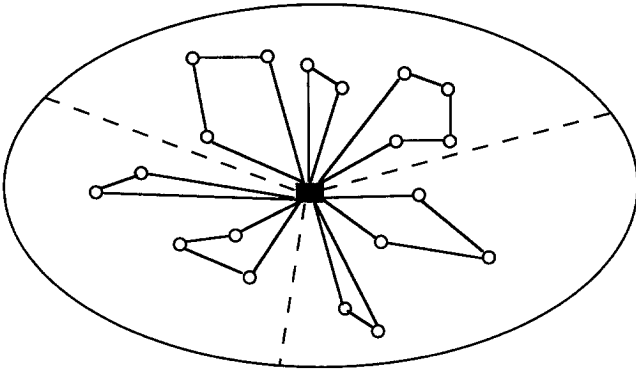


Fig. 4. Decomposition of a solution into subsets of routes.

solution before the resulting routes are stored in the adaptive memory. This phase introduces a high level of diversity in the adaptive memory by storing routes extracted from different types of solutions. This diversity is then exploited by the tabu search.

### 3.2 Decomposition/Reconstruction

To reduce computation time and intensify the search in specific regions of the search space, each initial solution is partitioned into  $C$  disjoint subsets of routes, each subset or subproblem being processed by a different tabu search (Taillard, 1993). The cardinality of the partition is determined through parameter  $U$ , which is an upper bound on the number of routes found in each subproblem. That is,  $C$  is the smallest integer greater than or equal to 1, such that  $m/C \leq U$ . The best routes found for every subproblem are then merged together to form the new solution for the next D&R. After  $W$  D&Rs, the final routes are stored in the adaptive memory if it is not filled yet (see Section 3.3). Otherwise, the routes of the worst solution found in the adaptive memory are discarded and replaced by the new routes (if the new solution is better than the worst solution in memory).

The decomposition uses the polar angle associated with the center of gravity of each route. Through a sweep procedure, the domain is then partitioned into sectors that approximately contain the same number of routes (see Figure 4). The decomposition changes from one D&R to the next by choosing a different starting angle for creating sectors, thus allowing the CROSS exchange heuristic to exploit new pairs of routes.

### 3.3 The Adaptive Memory

The adaptive memory is a pool of routes taken from the best solutions visited during the search. Its purpose is to provide new starting solutions for the tabu search through selection and combination of

routes extracted from this memory (in a manner reminiscent of crossover operators, which are found in genetic algorithms (HOLLAND, 1975); however, the number of parent solutions in our case is generally larger than two).

First, the memory is partially filled with routes produced during the initialization procedure (see Section 3.1). All routes of a given solution are contiguously stored in memory and the solutions are sorted according to their objective value. Hence, the routes associated with the best solutions are found in the first positions of the memory. This pool is then used to construct initial solutions for the tabu search. The selection process for creating a new solution is stochastic and biased in favor of the best routes. Namely, a selection probability is associated with each route, and this probability is higher for routes associated with better solutions (Rochat and Taillard, 1995).

Once the first route is selected, the routes in memory with one or more customers in common with the selected route are discarded from the selection procedure. Then, a second route is selected among the remaining routes. This procedure is repeated until the set of selected routes covers all customers or until there is no admissible route in memory. In the latter case, Solomon's I1 insertion heuristic is invoked to insert the remaining customers (see Solomon (1987) and Section 3.4). If this insertion procedure cannot accommodate all customers, due to the capacity constraints or the time window at the depot, the unserved customers are left aside. However, the insertion procedure is reinvoked at the end of each D&R to try to insert these customers into the current solution.

### 3.4 The Tabu Search

The tabu search presented in this section exploits the neighborhood structure of Section 2. It is applied to a subset of routes or subproblem through the D&R procedure of Section 3.2. The tabu search is quite standard, and can be summarized as follows:

*Set the current solution to some initial subset of routes.*

*While the stopping criterion is not met do:*

*Generate the neighborhood of the current solution by applying CROSS exchanges.*

*Select the best non-tabu solution in this neighborhood and define this solution to be the new current solution.*

*If the current solution is better than the best overall solution then*



*reorder the customers within each route using Solomon's I1 insertion heuristic; define this new solution to be the new current solution and the best overall solution.*

*Update the tabu list.*

*Return the best overall solution.*

The main components of this algorithm are now briefly introduced.

- a. *Initialization.* As mentioned previously, the initial solution is produced by combining routes contained in the adaptive memory (see Section 3.3). Then, this set of routes is partitioned into subsets of routes, each subset being provided as input to a tabu search heuristic.
- b. *Stopping criterion.* The tabu search stops after a certain number of iterations. This number is calculated with the formula:

$$A[1 + (DR - 1)/B]$$

where  $A$  and  $B$  are parameters and  $DR$  is the current D&R,  $DR = 1, \dots, W$ . Hence, the number of iterations increases with the number of D&Rs. Given that the solution improves from one D&R to the next, more iterations are required to significantly improve the solution in the last D&Rs.

- c. *Tabu list.* The tabu list has length  $T$  and its positions are indexed from 0 to  $T - 1$ . Each solution is associated with a position in the list. This position is the objective value of the solution modulo  $T$  and the value stored at this place is the iteration number at which the solution will lose its tabu status. When a neighboring solution is produced through a CROSS exchange, its objective value modulo  $T$  provides its position in the tabu list. If the value found at this position is larger than the current iteration number then the move is tabu, otherwise it is accepted. Clearly, this approach can filter out legitimate solutions. For example, two solutions will collide at the same position within the list if their objective values differ by a multiple of  $T$ . However,  $T$  is set to a large value, so that such an occurrence is unlikely. The tabu tenure is equal to the number of iterations divided by 2. It is worth noting that the objective value is integer because real distances are transformed into integers (see Section 4.2).
- d. *Diversification.* Dynamic diversification is incorporated into the tabu search by penalizing CROSS exchanges that are frequently performed during the search. Let  $fr_e$  be such a frequency for a given exchange  $e$ , let  $fr_{\max}$  be the maximal frequency observed and let  $iter$ ,  $n$  and  $m$  be the

current iteration number, the number of customers and the number of routes, respectively. If  $x$  is a random value uniformly chosen in the interval  $[0.0, 0.5]$  and  $\Delta_{iter}^{\max}$  is the maximal absolute difference observed between the objective values of two consecutive solutions up to iteration  $iter$ , then the exchange  $e$  is penalized by:

$$x \Delta_{iter}^{\max} fr_e / fr_{\max}$$

The  $fr_{\max}$  value normalizes the frequencies since they tend to be lower (higher) for larger (smaller) neighborhoods. The  $\Delta_{iter}^{\max}$  component adjusts the penalty to the magnitude of the moves.

- e. *Reordering of each route.* The customers within each individual route are reordered when an overall best solution is found. This reordering is based on Solomon's I1 insertion heuristic (Solomon, 1987). It works as follows for a given route. First, all customers on this route are unrouted. Then, the farthest unrouted customer from the depot is selected as a seed customer (i.e., the vehicle starts from the depot, services the seed customer and comes back to the depot). The remaining unrouted customers are then inserted one by one into the new route. The next customer to be inserted is the one that maximizes a generalized savings measure (since the classical savings (CLARKE and WRIGHT, 1964) can be obtained by setting the parameters of Solomon's heuristic to appropriate values). The selected customer is then inserted at the location that minimizes a weighted sum of detour in space and time. Solomon's heuristic is applied  $R$  times with different parameter settings, and the best route is selected at the end. This strategy can be seen as a form of intensification in the neighborhood of an elite solution. Note that the original route is restored if the new route does not service all customers (due to the time window at the depot) or if the new route is worse than the original one.

### 3.5 Postprocessing of Each Individual Route

A specialized heuristic developed for the Traveling Salesman Problem with time windows (GENDREAU et al., 1995) was applied to each individual route of the final solution produced by our algorithm. This method is an adaptation of the GENIUS heuristic, originally devised for the Traveling Salesman Problem (GENDREAU et al., 1992). This heuristic only slightly improved the total distance of 10 solutions in Solomon's test set (the improvement was less than 1% in all cases but one), but it ran for only a few seconds and three additional best known solutions were obtained (see Section 4.2).

#### 4. COMPUTATIONAL RESULTS

ALTHOUGH OUR METHOD was designed for the VRPSTW, it has been tested on Solomon's set of VRPHTWs (Solomon, 1987). This choice comes from a need to compare our method with competing approaches on standardized test problems. In the following, these test problems are first introduced. Then, the best solutions produced by our algorithm are reported. Some figures about the improvement to solution quality with increasing computation times are also provided. Finally, the impact of the approximation techniques of Section 2 on solution quality and computation time is examined.

##### 4.1 Test Problems

Our algorithm was tested on standard problems found in Solomon (1987). In these 100-customer Euclidean problems, the travel times are equivalent to the corresponding Euclidean distances. The customer locations are distributed within a  $[0, 100]^2$  square. Six different sets of problems are defined, namely C1, C2, R1, R2, RC1, and RC2. The customers are uniformly distributed in the problems of type R and clustered in groups in the problems of type C. These characteristics are mixed in the problems of type RC. Furthermore, the time window is narrow at the central depot for the problems of type 1, so that only a few customers can be serviced on each route. Conversely, this time window is large for the problems of type 2, so that many customers can be serviced on the same route. Finally, a fixed service time is found at each customer location (i.e., time for unloading the goods). This service time is set at 10 time units per customer for the problems of type R and RC, and 90 time units per customer for the problems of type C.

##### 4.2 Numerical Results on Solomon's Test Set

The experiments reported in this section were performed on a SUN Sparc 10 workstation (50 Mhz). The real Euclidean distances between customers were rounded up or down at the third decimal place, and then transformed into integers by multiplying them by  $10^3$ . In this way, precision problems were avoided during the calculations. However, the feasibility of the best solutions produced by our algorithm was later checked using real, double-precision distances. Our results were obtained with the parameter settings presented in Table I. Note that no D&R took place on the problems of type 2, because only a few routes are needed to service all customers. Hence, each tabu search was applied to the entire set of routes in this case.

The best solutions produced by our algorithm are

TABLE I

*Parameter Settings for the Tabu Search Heuristic*

---

Objective Function
lateness penalty coefficient: $\alpha = 100$ .
Stopping Criterion
Three identical best known solutions visited during the search or 100 calls to the adaptive memory
Initialization
Number of initial solutions: $I = 20$
Adaptive Memory
Size: $M = 30$ solutions
Decomposition/Reconstruction (D&R)
Number of D&Rs: $W = 6$
Maximum number of routes in each subproblem: $U = 8$
Tabu Search
Number of iterations: $A = 30, B = 3$ ;
thus 30, 40, 50, 60, 70 and 80 iterations for $DR = 1, 2, 3, 4, 5$ and 6, respectively, for a total of 330 iterations
Neighborhood
maximum length of route segments: $L = 5$ or 7
number of values for constructing the approximation function $g$ , at each customer $i$ : $Z = 6$
Length of tabu list: $T = 100,000$
Tabu tenure: number of iterations/2.
Diversification: random number $x \in [0.0, 0.5]$
Number of best approximate neighboring solutions: $P = 15$
Number of reorderings on each route: $R = 20$

---

reported in Tables II and III for all problems in Solomon's test set (based on real, double-precision distances), using the format: number of routes/total distance. The fleet size  $m$  was set to the number of routes of the best solution reported in the literature for each problem. A feasible solution to the VRPHTW was produced in each case, with the exception of problem R101 (where one additional route is needed). On problems RC102 and RC106, the algorithm saved an additional route. In the tables, our best solutions are compared with the best solutions previously reported in the literature. A single asterisk \* means a tie with the best published solution and a double asterisk \*\* means that the best published solution has been improved. Overall, our algorithm has improved 17 and tied 20 best known solutions on the 56 test problems.

Different transformations of the distance matrix or final solutions were applied by the authors referred to in Tables II and III. In Desrochers et al. (1992), the distances were truncated at the first decimal place; in Thangiah et al. (1994), the final solution value was rounded up or down to the nearest integer; in Rochat and Taillard (1995) and Potvin and Bengio (1996), real double-precision distances were used. One consequence is that slightly different objective values can be associated with the same solution (in particular, for the problems in class C1). Another consequence is that solutions found on truncated or rounded distances are not necessarily

TABLE II  
Best Results on Problems of Type 1

Problem	Best published solution	Reference	Our best solution
R101	18/1607.7	(Desrochers et al., 1992)	19/1650.79
R102	17/1434.0	(Desrochers et al., 1992)	17/1487.60
R103	13/1207	(Thangiah et al., 1994)	13/1294.24
R104	10/982.01	(Rochat and Taillard, 1995)	10/982.72
R105	14/1377.11	(Rochat and Taillard, 1995)	14/1377.11 *
R106	12/1252.03	(Rochat and Taillard, 1995)	12/1259.71
R107	10/1159.85	(Rochat and Taillard, 1995)	10/1126.69 **
R108	09/980.95	(Rochat and Taillard, 1995)	09/968.59 **
R109	11/1235.68	(Rochat and Taillard, 1995)	11/1214.54 **
R110	11/1080.36	(Rochat and Taillard, 1995)	11/1080.36 *
R111	10/1129.88	(Rochat and Taillard, 1995)	10/1104.83 **
R112	10/953.63	(Rochat and Taillard, 1995)	10/964.01
C101	10/827.3	(Desrochers et al., 1992)	10/828.94 *
C102	10/827.3	(Desrochers et al., 1992)	10/828.94 *
C103	10/828.06	(Rochat and Taillard, 1995)	10/828.06 *
C104	10/824.78	(Rochat and Taillard, 1995)	10/824.78 *
C105	10/828.94	(Potvin and Bengio, 1993)	10/828.94 *
C106	10/827.3	(Desrochers et al., 1992)	10/828.94 *
C107	10/827.3	(Desrochers et al., 1992)	10/828.94 *
C108	10/827.3	(Desrochers et al., 1992)	10/828.94 *
C109	10/828.94	(Potvin and Bengio, 1993)	10/828.94 *
RC101	14/1669	(Thangiah et al., 1994)	14/1696.94
RC102	13/1477.54	(Rochat and Taillard, 1995)	12/1554.75 **
RC103	11/1110	(Thangiah et al., 1994)	11/1264.27
RC104	10/1135.83	(Rochat and Taillard, 1995)	10/1135.83 *
RC105	13/1733.56	(Rochat and Taillard, 1995)	13/1643.38 **
RC106	12/1384.92	(Rochat and Taillard, 1995)	11/1448.26 **
RC107	11/1230.95	(Rochat and Taillard, 1995)	11/1230.54 **
RC108	10/1170.70	(Rochat and Taillard, 1995)	10/1139.82 **

\*Tie with the best published solution.

\*\*The best published solution has been improved.

feasible when real distances are used (unless the final solutions are checked for feasibility with real, double-precision distances, as we did).

Table IV compares the averages of the best solutions, as produced by our method and by other methods reported in the literature, for each set of problems. The two numbers in each entry are the average number of notes and average distance, respectively. Note that the previous best averages were all produced by Rochat and Taillard (1995) and that five additional routes were saved by our method over Rochat and Taillard (1995) on the 56 test problems.

A hierarchical objective function is typically associated with the VRPHTW. That is, the number of routes is first minimized and, for the same number of routes, the total distance is minimized. An exception is found in Barnes and Carlton (1995) and Carlton (1995), where the objective is simply to minimize the total distance. Accordingly, the number of routes reported in the work of Barnes and Carlton is typically larger than the numbers presented in Table IV.

In order to compare our method with the reactive tabu search of Barnes and Carlton, the fleet size  $m$  was set to the numbers reported in Appendix B of Carlton (1995) for each problem. The results are summarized in Table V. The first column is the average number of routes. The second column *Reactive Tabu (best)* contains two numbers. The first one is the average of the best distances reported in Appendix B of Carlton (1995), when the parameter settings of the reactive tabu search are adjusted to each problem. The second number is the average computation time of the best runs on a IBM 6000 RISC workstation (in seconds). The third column *Adaptive Tabu (avg.)* is the average distance produced by our algorithm over five different runs, when each run is given the same amount of computation time as the best run of the reactive tabu search. Here, the processing power of their machine was assumed to be similar to ours, which is true only for lower end IBM RISC 6000 workstations. Finally, the last column *Adaptive Tabu (best)* is the average of the best solutions produced by our algorithm in the course of multiple experiments.

TABLE III  
Best Results on Problems of Type 2

Problem	Best published solution	Reference	Our best solution
R201	4/1281.58	(Rochat and Taillard, 1995)	4/1254.80 **
R202	3/1530.49	(Potvin and Bengio, 1993)	3/1214.28 **
R203	3/948.74	(Rochat and Taillard, 1995)	3/951.59
R204	2/869.29	(Rochat and Taillard, 1995)	2/941.76
R205	3/1063.24	(Rochat and Taillard, 1995)	3/1038.72 **
R206	3/833	(Thangiah et al., 1994)	3/932.47
R207	3/814.78	(Rochat and Taillard, 1995)	3/837.20
R208	2/738.60	(Rochat and Taillard, 1995)	2/748.01
R209	3/855	(Thangiah et al., 1994)	3/959.47
R210	3/967.50	(Rochat and Taillard, 1995)	3/980.90
R211	2/949.50	(Rochat and Taillard, 1995)	2/923.80 **
C201	3/591.56	(Potvin and Bengio, 1993)	3/591.56 *
C202	3/591.56	(Potvin and Bengio, 1993)	3/591.56 *
C203	3/591.17	(Rochat and Taillard, 1995)	3/591.17 *
C204	3/590.60	(Potvin and Bengio, 1993)	3/590.60 *
C205	3/588.88	(Potvin and Bengio, 1993)	3/588.88 *
C206	3/588.49	(Potvin and Bengio, 1993)	3/588.49 *
C207	3/588.29	(Rochat and Taillard, 1995)	3/588.29 *
C208	3/588.32	(Rochat and Taillard, 1995)	3/588.32 *
RC201	4/1294	(Thangiah et al., 1994)	4/1413.79
RC202	4/1165.57	(Rochat and Taillard, 1995)	4/1164.25 **
RC203	3/1079.57	(Rochat and Taillard, 1995)	3/1112.55
RC204	3/806.75	(Rochat and Taillard, 1995)	3/831.69
RC205	4/1333.71	(Rochat and Taillard, 1995)	4/1328.21 **
RC206	3/1212.64	(Rochat and Taillard, 1995)	3/1158.81 **
RC207	3/1085.61	(Rochat and Taillard, 1995)	3/1082.32 **
RC208	3/833.97	(Rochat and Taillard, 1995)	3/847.90

\* Tie with the best published solution.

\*\* Improvement over the best published solution.

First, it should be pointed out that the comparison between *Adaptive Tabu* (*avg.*) and *Reactive Tabu* (*best*) is quite unfair. Namely, the computation time associated with the reactive tabu search should be the sum of all runs leading to the best solution, not only the computation time of the best run. In spite of

this enormous handicap, the *average* solutions produced by our algorithm are competitive with the *best* solutions produced by the reactive tabu search, for the “same” amount of computation time. Furthermore, our best solutions are significantly better than those reported in Appendix B of Carlton (1995). The

TABLE IV  
Best Averages for Each Problem Type

	CR*	PB*	TH*	RT*	Our method
R1	12.42 1289.95	12.58 1296.80	12.33 1238	12.25 1208.50	12.17 1209.35
C1	10.00 885.86	10.00 838.01	10.00 832	10.00 828.38	10.00 828.38
RC1	12.38 1455.82	12.13 1446.20	12.00 1284	11.88 1377.39	11.50 1389.22
R2	2.91 1135.14	3.00 1117.70	3.00 1005	2.91 961.72	2.82 980.27
C2	3.00 658.88	3.00 589.93	3.00 650	3.00 589.86	3.00 589.86
RC2	3.38 1361.14	3.38 1360.57	3.38 1229	3.38 1119.59	3.38 1117.44

\*CR = Chiang and Russell (1993); PB = Potvin and Bengio (1996); RT = Rochat and Taillard (1995) and TH = Thangiah et al. (1994).

TABLE V  
Comparison with the Reactive Tabu Search in Carlton (1995)

	Avg. Number of routes	Reactive Tabu (best)	Adaptive Tabu (avg.)	Adaptive Tabu (best)
R1	13.91	1221.2 568.0	1226.3	1195.9
C1	10.00	831.9 273.5	840.0	828.4
RC1	13.25	1408.3 564.8	1399.0	1358.6
R2	3.09	1009.1 988.9	1056.4	994.8
C2	3.25	616.5 408.9	610.5	590.7
RC2	3.38	1303.3 750.5	1288.8	1117.4

reactive tabu search of Barnes and Carlton is thus able to produce good solutions in relatively low computation times. However, due to its simple neighborhood structure, it reaches its “peak” much earlier than our method.

Finally, our tabu search heuristic was applied on two larger problem instances with 417 customers. These problems are known as D417 and E417 and are reported in Russell (1995). A feasible solution with 55 vehicles was found in both cases, with a total distance of 3439.8 for D417 and 3707.1 for E417. Total distances of 3467.8 and 3693.2 for D417 and E417, respectively, are reported in Rochat and Taillard (1995) with the same number of vehicles.

Hence, a better solution was found on problem D417. Solutions with 54 vehicles are also reported in Rochat and Taillard (1995), but the total distance then increases to 6264.8 for D417 and to 7211.8 for E417. Our algorithm did not find feasible solutions with 54 vehicles.

#### 4.3 Solution Quality Versus Computation Time

The results reported in Tables II and III are the best solutions produced by our algorithm during the course of multiple experiments. Accordingly, no meaningful computation times can be provided. However, Table VI illustrates the improvement to solution quality with increasing computation times, using  $L = 7$ . These numbers were obtained over five independent runs. For each problem set, we show the average computation time in seconds after 20, 50, and 100 calls to the adaptive memory (including all preprocessing that takes place before the search actually starts), the average solution value of the best run (*Minimum*), the average solution value of the worst run (*Maximum*) and the overall average (*Average*). The first number is the average number of routes and the second number is the average distance. In these experiments, the number of routes was set to  $m + 1$ , where  $m$  is the minimum value reported in the literature for each problem. By introducing an additional route, feasible solutions to the VRPHTW were quickly produced by our algorithm, so that all numbers in Table VI refer to solutions that satisfy the time window constraint at each

TABLE VI  
Solution Quality Versus CPU Time (in seconds) for our Algorithm

Problem type	CPU time	Minimum		Maximum		Average	
R1	2296	12.58	1222.24	12.67	1248.28	12.64	1233.88
	6887	12.25	1221.89	12.58	1237.50	12.39	1230.48
	13774	12.25	1216.70	12.50	1229.71	12.33	1220.35
C1	2926	10.00	829.22	10.00	834.10	10.00	830.41
	7315	10.00	828.50	10.00	828.77	10.00	828.59
	14630	10.00	828.45	10.00	828.45	10.00	828.45
RC1	1877	11.88	1385.69	12.25	1424.28	12.08	1404.59
	5632	11.88	1373.11	12.13	1400.84	12.00	1387.01
	11264	11.88	1367.51	12.00	1400.12	11.90	1381.31
R2	3372	3.00	1025.24	3.00	1067.87	3.00	1046.56
	10116	3.00	1011.04	3.00	1048.25	3.00	1029.65
	20232	3.00	995.38	3.00	1031.33	3.00	1013.35
C2	3275	3.00	590.37	3.00	596.85	3.00	592.75
	8187	3.00	590.37	3.00	593.58	3.00	591.14
	16375	3.00	590.30	3.00	592.57	3.00	590.91
RC2	1933	3.38	1204.17	3.38	1289.81	3.38	1248.34
	5798	3.38	1186.57	3.38	1263.12	3.38	1220.28
	11596	3.38	1165.62	3.38	1239.14	3.38	1198.63

TABLE VII  
*Solution Quality Versus CPU Time (in seconds) for Rochat and Taillard (1995)*

Problem type	CPU time	RT	
R1	450	12.83	1208.43
	1300	12.58	1202.31
	2700	12.58	1197.42
C1	540	10.00	832.59
	1600	10.00	829.01
	3200	10.00	828.45
RC1	430	12.75	1381.33
	1300	12.50	1368.03
	2600	12.38	1369.48
R2	1600	3.18	999.63
	4900	3.09	969.29
	9800	3.09	954.36
C2	1200	3.00	595.38
	3600	3.00	590.32
	7200	3.00	590.32
RC2	1300	3.62	1207.37
	3900	3.62	1155.47
	7800	3.62	1139.79

customer location. Note that the number of routes can decrease during the search if an empty route is produced by a CROSS exchange.

In the case of C1 and C2, the growth of the route segments was not stopped when a monotonous degradation of the objective value was observed over three consecutive iterations (see Section 2.3.2). On these well-structured problems, the method quickly converges to good local optima and comes back to these same optima in the remainder of the run, thus causing an early stop (e.g., same best solution found three times). By allowing a more thorough exploration of the neighborhood of the current solution, better solutions are found before the stopping criterion is met.

In Table VII, the results in Rochat and Taillard (1995) (RT) for a single run of their method on a Silicon Graphics Indigo (100 Mhz) are shown. By

comparing the solution values reported in Table VI under "Average" with those reported in Table VII for similar computation times, we observe that our method found an average of 12.08 routes on RC1 after 1877 seconds, compared with 12.38 routes after 2600 seconds with RT. Furthermore, RT did not find the averages of 3.00 and 3.38 routes on problem sets R2 and RC2, after 9800 and 7800 seconds, respectively. Conversely, RT found 12.58 routes on set R1 after 2700 seconds, compared with an average of 12.64 routes after 2296 seconds with our method.

In general, it is quite difficult to compare different problem-solving methodologies with respect to their computational requirements, because of specific hardware and implementation languages. Furthermore, our method carries the additional computational burden associated with soft time windows. Disregarding such issues, the competing approaches CR, PB, and TH in Table IV are less computationally expensive than our method (although they do not provide the same solution quality). Each run of CR requires 1–6 minutes of computation time, depending on the problem, on a 486 DX2/66 PC (Chiang and Russell, 1993). The computation times reported in Thangiah et al. (1994) (TH) vary from 1–20 minutes on a NeXT 68040, whereas those reported in Potvin and Bengio (1996) (PB) vary from 10–40 minutes on a SUN Sparc 10 workstation. Note also that the results reported in Table IV are the best solutions obtained over many experiments with different variants or parameter settings.

#### 4.4 CROSS Exchanges Versus 2-opt\* and Or-opt

To evaluate the benefits associated with the CROSS neighborhood, Table VIII shows results obtained on problem set R1 when this neighborhood is replaced by the 2-opt\* or the Or-opt neighborhood within the tabu search heuristic. The format of this table is the same as Table VI. It shows the average computation time in seconds after 20, 50, and 100

TABLE VIII  
*Comparison Between the CROSS, 2-opt\* and Or-opt Neighborhoods on Problem set R1*

	CPU time	Minimum		Maximum		Average	
CROSS	2296	12.58	1222.24	12.67	1248.28	12.64	1233.88
	6887	12.25	1221.89	12.58	1227.50	12.39	1230.48
	13774	12.25	1216.70	12.50	1219.71	12.33	1220.35
2-opt*	1259	12.67	1272.81	13.17	1340.13	12.88	1304.03
	3147	12.58	1268.75	13.00	1320.97	12.73	1293.97
	6294	12.58	1268.75	12.75	1309.82	12.58	1288.92
Or-opt	1857	12.67	1274.63	13.08	1326.80	12.85	1301.14
	4643	12.58	1261.30	12.83	1315.61	12.72	1288.93
	9286	12.58	1243.60	12.67	1303.40	12.62	1275.36

TABLE IX  
Approximate Versus Exact Solutions at the First Local Minimum

	R1	C1	RC1	R2	C2	RC2
Initial	13.67 1523.51	10.00 1609.56	16.50 1903.89	4.27 1347.01	3.00 813.09	3.38 1648.19
Approx						
$L = 5$	13.08 1343.15 7.8	10.00 1106.43 10.89	13.75 1530.31 8.25	3.36 1169.84 50.8	3.00 669.47 9.38	3.38 1513.69 34.00
$L = 7$	13.00 1317.41 12.4	10.00 1080.29 19.2	13.63 1486.89 16.1	3.36 1175.22 61.36	3.00 669.21 11.25	3.38 1490.28 41.63
$L = 10$	13.00 1295.02 18.42	10.00 971.77 32.56	13.63 1494.32 20.38	3.36 1170.76 75.90	3.00 686.82 15.63	3.38 1422.11 77.38
$L = 20$	13.00 1290.9 20.0	10.00 961.30 36.67	13.38 1488.58 20.50	3.36 1157.52 185.73	3.00 694.30 37.63	3.38 1395.66 169.25
$L = \infty$	13.00 1285.13 34.4	10.00 924.55 55.44	13.38 1444.16 36.13	3.36 1046.79 1201.64	3.00 620.33 283.63	3.38 1360.77 758.00
Exact						
$L = 7$	13.08 1294.30 99.83	10.00 1019.10 102.67	13.50 1468.67 90.25	3.36 1106.50 424.909	3.00 632.72 74.63	3.38 1333.20 224.88
$L = \infty$	13.00 1280.18 223.25	10.00 959.11 248.89	13.13 1436.74 200.88	3.36 1024.79 4118.55	3.00 600.31 1002.00	3.38 1275.73 3306.62

calls to the adaptive memory, the average solution value of the best run (over five different runs), the average solution value of the worst run, and the overall average. In the last three columns, the first number is the average number of routes and the second number is the average distance.

As we can see, it is more computationally expensive to generate the CROSS neighborhood than the 2-opt\* and Or-opt neighborhoods. However, by comparing results for similar computation times (e.g., the third line of 2-opt\* and Or-opt with the second line of CROSS), the CROSS neighborhood clearly leads to better solutions. Note that the same trends are observed on the other problem sets.

#### 4.5 Benefits of Approximations

This section evaluates the usefulness of the techniques proposed in Section 2 for approximating the objective value. Table IX shows the average solution value (number of routes, distance, computation time in seconds, in this order) at the first local minimum for a typical run on each problem type, using either an exact or approximate evaluation. Different values of parameter  $L$  were also tested. That is, the neighborhood of the current solution was progressively reduced by decreasing  $L$  from  $\infty$  to 5. Note that when  $L = \infty$ , the restriction on the monotonous degradation of the objective value over three consec-

utive iterations was removed. Hence, the entire neighborhood of the current solution was examined. *Approx* uses the approximation to evaluate each new solution, and *Exact* evaluates each new solution exactly. The row "Initial" in Table IX refers to the value of the starting solution.

A comparison between *Approx*,  $L = \infty$  and *Exact*,  $L = \infty$  shows that the approximation generates solutions that are competitive with those obtained through the exact evaluation in a fraction of the computation time. The performance of the approximation degrades slightly on the large routes found in the problems of type 2 (in particular RC2). By comparing *Exact* and *Approx* when the value of parameter  $L$  is reduced from  $L = \infty$  to  $L = 7$ , we also note that solution quality does not degrade as much when the exact evaluation is used. However, *Exact* with  $L = 7$  is still much more computationally expensive than *Approx* with  $L = 7$ .

We observed finally that the setting of parameter  $L$ , as well as the use of *Approx* or *Exact*, have little impact on the number of iterations needed to reach the first local minimum.

#### 5. CONCLUSION

THIS PAPER HAS introduced a new neighborhood structure (based on CROSS exchanges), which is

exploited within a tabu search heuristic. As the search progresses, good routes are stored in an adaptive memory and new starting solutions are produced by selecting and combining routes found in this memory. That is, new solutions are created from routes associated with different solutions visited during the search (in a manner reminiscent of the recombination or crossover operator of genetic algorithms). This problem-solving methodology has produced many best-known solutions on Solomon's test set (Solomon, 1987) and seems to be fairly robust along the different classes of problems found in this test set.

The CROSS exchanges are a useful generalization of the 2-opt\* (Potvin and Rousseau, 1995) and Or-opt (Or, 1976) exchanges and may be applied to other types of vehicle routing problems as well. Note also that our problem-solving methodology is readily amenable to parallelization. For example, many different starting solutions can be constructed from the adaptive memory and assigned to different tabu search processes running in parallel. Furthermore, each tabu search process can be parallelized by exploiting the D&R procedure that partitions a problem into smaller subproblems. With such parallelization, it is possible to envision the application of our methodology in a dynamic setting where each service request is dispatched soon after it is received.

#### ACKNOWLEDGMENTS

THIS WORK WAS partially supported by the Canadian Natural Sciences and Engineering Research Council (NSERC) and by the Quebec Fonds pour la Formation de Chercheurs et l'Aide à la Recherche (FCAR). The first author was postdoctoral fellow at the Centre de recherche sur les transports (CRT) of Montreal University when the paper was written. He benefitted from an NSERC International Postdoctoral fellowship and was also partially funded by Strategic Grant STR0149269. This support is gratefully acknowledged.

#### REFERENCES

- E. K. BAKER AND J. R. SCHAFFER, "Solution Improvement Heuristics for the Vehicle Routing and Scheduling Problem with Time Window Constraints," *Amer. J. Math. Management Sci.* **6**, 261-300 (1988).
- J. W. BARNES AND W. B. CARLTON, "Solving the Vehicle Routing Problem with Time Windows using Reactive Tabu Search," presented at the Fall 1995 INFORMS Conference, New Orleans, LA, 1995.
- J. L. BLANTON AND R. L. WAINWRIGHT, "Multiple Vehicle Routing with Time and Capacity Constraints using Genetic Algorithms," in *Proceedings of the 5th International Conference on Genetic Algorithms*, S. Forrest (ed.), pp 452-459, Morgan Kaufmann, San Mateo, CA, 1993.
- J. BRAMEL AND D. SIMCHI-LEVI, Probabilistic Analyses and Practical Algorithms for the Vehicle Routing Problem with Time Windows, Working Paper, Graduate School of Business, Columbia University, 1993.
- J. BRAMEL, C. L. LI, AND D. SIMCHI-LEVI, "Probabilistic Analysis of a Vehicle Routing Problem with Time Windows," *Amer. J. Math. Management Sci.* **13**, 267-322 (1993).
- W. B. CARLTON, A Tabu Search Approach to the General Vehicle Routing Problem, Ph.D. Dissertation, The University of Texas at Austin, Austin, TX, 1995.
- W. C. CHIANG AND R. A. RUSSELL, Hybrid Heuristics for the Vehicle Routing Problem with Time Windows, Working Paper, Department of Quantitative Methods, University of Tulsa, Tulsa, OK, 1993.
- G. CLARKE AND J. WRIGHT, "Scheduling of Vehicles from a Central Depot to a Number of Delivery Points," *Opns. Res.* **12**, 568-581 (1964).
- U. DERIGS AND G. GRABENBAUER, "INTIME—A New Heuristic Approach to the Vehicle Routing Problem with Time Windows with a Bakery Fleet Case," *Amer. J. Math. Management Sci.* **13**, 249-266 (1993).
- M. DESROCHERS, J. DESROSIERS, AND M. M. SOLOMON, "A New Optimization Algorithm for the Vehicle Routing Problem with Time Windows," *Opns. Res.* **40**, 342-354 (1992).
- M. DESROCHERS, J. K. LENSTRA, M. W. P. SAVELSBERGH, AND F. SOUMIS, "Vehicle Routing with Time Windows: Optimization and Approximation," in *Vehicle Routing: Methods and Studies*, B. L. Golden and A. A. Assad (eds.), North-Holland, Amsterdam, 65-84, 1988.
- J. DESROSIERS, Y. DUMAS, M. M. SOLOMON, AND F. SOUMIS, "Time Constrained Routing and Scheduling," in *Handbooks in Operations Research and Management Science*, Vol. 8. *Network Routing*, M. O. Ball, T. L. Magnanti, C. L. Monma and G. L. Nemhauser (eds.), North-Holland, Amsterdam, 35-139, 1995.
- C. DUHAMEL, B. L. GARCIA, AND J. Y. POTVIN, "Accélération du Calcul des Fonctions objectif pour les Problèmes de tournées de véhicules avec Fenêtres de temps," Technical Report CRT-95-04, Centre de recherche sur les transports, Université de Montréal, Montréal, Canada, 1995.
- R. FAHRION AND M. WREDE, "On a Principle of Chain-Exchange for Vehicle Routing Problems (1-VRP)," *J. Oper. Res. Soc.* **41**, 821-827 (1990).
- M. GENDREAU, A. HERTZ, AND G. LAPORTE, "New Insertion and Postoptimization Procedures for the Traveling Salesman Problem," *Opns. Res.* **40**, 1086-1094 (1992).
- M. GENDREAU, A. HERTZ, G. LAPORTE, AND M. STAN, "A Generalized Insertion Heuristic for the Traveling Salesman Problem with Time Windows," to appear in *Opns. Res.* (1997).
- F. GLOVER, "Tabu Search—Part I," *ORSA J. Comput.* **1**, 190-206 (1989).



- F. GLOVER, "Tabu Search—Part II," *ORSA J. Comput.* **2**, 4–32 (1990).
- J. H. HOLLAND, *Adaptation in Natural and Artificial Systems*, The University of Michigan Press, Ann Arbor, 1975.
- A. KOLEN, A. H. G. RINNOOY KAN, AND H. TRIENEKESN, "Vehicle Routing with Time Windows," *Opns. Res.* **35**, 266–273 (1987).
- G. KONTORAVDIS AND J. BARD, "A GRASP for the Vehicle Routing Problem with Time Windows," *ORSA J. Comput.* **7**, 10–23 (1995).
- Y. A. KOSKOSIDIS, W. B. POWELL, AND M. M. SOLOMON, "An Optimization-Based Heuristic for Vehicle Routing and Scheduling with Soft Time Window Constraints," *Trans. Sci.* **26**, 69–85 (1992).
- S. LIN, "Computer Solutions of the Traveling Salesman Problem," *Bell System Tech. J.* **44**, 2245–2269 (1965).
- I. OR, Traveling Salesman-type Combinatorial Problems and Their Relation to the Logistics of Blood Banking, Ph.D. dissertation, Department of Industrial Engineering and Management Science, Northwestern University, Evanston, IL, 1976.
- I. H. OSMAN, "Metastrategy Simulated Annealing and Tabu Search Algorithms for the Vehicle Routing Problem," *Ann. Opns. Res.* **41**, 421–451 (1993).
- J. Y. POTVIN, T. KERVAHUT, B. L. GARCIA, AND J. M. ROUSSEAU, "The Vehicle Routing Problem with Time Windows—Part I: Tabu Search," *INFORMS J. Comput.* **8**, 158–164 (1996).
- J. Y. POTVIN AND S. BENGIO, "The Vehicle Routing Problem with Time Windows—Part II: Genetic Search," *INFORMS J. Comput.* **8**, 165–172 (1996).
- J. Y. POTVIN AND J. M. ROUSSEAU, "A Parallel Route Building Algorithm for the Vehicle Routing and Scheduling Problem with Time Windows," *Eur. J. Oper. Res.* **66**, 331–340 (1993).
- J. Y. POTVIN AND J. M. ROUSSEAU, "An Exchange Heuristic for Routing Problems with Time Windows," *J. Oper. Res. Soc.* **46**, 1433–1446 (1995).
- Y. ROCHAT AND E. TAILLARD, "Probabilistic Diversification and Intensification in Local Search for Vehicle Routing," *J. Heuristics* **1**, 147–167 (1995).
- R. A. RUSSELL, "Hybrid Heuristics for the Vehicle Routing Problem with Time Windows," *Trans. Sci.* **29**, 156–166 (1995).
- M. W. P. SAVELSBERGH, "Local Search in Routing Problems with Time Windows," *Ann. Opns. Res.* **4**, 285–305 (1986).
- M. W. P. SAVELSBERGH, "An Efficient Implementation of Local Search Algorithms for Constrained Routing Problems," *Eur. J. Oper. Res.* **47**, 75–85 (1990).
- M. W. P. SAVELSBERGH, "The Vehicle Routing Problem with Time Windows: Minimizing Route Duration," *ORSA J. Comput.* **4**, 146–154 (1992).
- F. SEMET AND E. TAILLARD, "Solving Real-Life Vehicle Routing Problems Efficiently using Tabu Search," *Ann. Opns. Res.* **41**, 469–488 (1993).
- M. M. SOLOMON, "On the Worst-Case Performance of Some Heuristics for the Vehicle Routing and Scheduling Problem with Time Window Constraints," *Networks* **16**, 161–174 (1986).
- M. M. SOLOMON, "Algorithms for the Vehicle Routing and Scheduling Problems with Time Window Constraints," *Opns. Res.* **35**, 254–265 (1987).
- M. M. SOLOMON, E. K. BAKER, AND J. R. SCHAFFER, "Vehicle Routing and Scheduling Problems with Time Window Constraints: Efficient Implementations of Solution Improvement Procedures," in *Vehicle Routing: Methods and Studies*, B. L. Golden and A. A. Assad (eds.), North-Holland, Amsterdam, 85–105, 1988.
- M. M. SOLOMON AND J. DESROSIERS, "Time Window Constrained Routing and Scheduling Problems," *Trans. Sci.* **22**, 1–13 (1988).
- E. TAILLARD, "Parallel Iterative Search Methods for Vehicle Routing Problems," *Networks* **23**, 661–673 (1993).
- S. R. THANGIAH, Vehicle Routing with Time Windows using Genetic Algorithms, Technical Report SRU-CpSc-TR-93-23, Computer Science Department, Slippery Rock University, Slippery Rock, PA, 1993.
- S. R. THANGIAH, K. NYGARD AND P. JUELL, "GIDEON: A Genetic Algorithm System for Vehicle Routing with Time Windows," in *Proceedings of the 7th IEEE Conference on Artificial Intelligence Applications*, pp. 422–425, IEEE Press, Miami, FL, 1991.
- S. R. THANGIAH, I. H. OSMAN, AND T. SUN, Hybrid Genetic Algorithms, Simulated Annealing and Tabu Search Methods for Vehicle Routing Problems with Time Windows, Technical Report UKC/OR94/4, Institute of Mathematics & Statistics, University of Kent, Canterbury, UK, 1994.
- P. THOMPSON AND H. PSARAFTIS, "Cyclic Transfer Algorithms for Multi-Vehicle Routing and Scheduling Problems," *Opns. Res.* **41**, 935–946 (1993).

(Received: November 1995; revisions received: April 1996, May 1996; accepted: May 1996)