ALGORITHMS FOR THE VEHICLE ROUTING AND SCHEDULING PROBLEMS WITH TIME WINDOW CONSTRAINTS

MARIUS M. SOLOMON

Northeastern University, Boston, Massachusetts

(Received February 1984; revisions received October 1984; March, October 1985; accepted December 1985)

This paper considers the design and analysis of algorithms for vehicle routing and scheduling problems with time window constraints. Given the intrinsic difficulty of this problem class, approximation methods seem to offer the most promise for practical size problems. After describing a variety of heuristics, we conduct an extensive computational study of their performance. The problem set includes routing and scheduling environments that differ in terms of the type of data used to generate the problems, the percentage of customers with time windows, their tightness and positioning, and the scheduling horizon. We found that several heuristics performed well in different problem environments; in particular an insertion-type heuristic consistently gave very good results.

A key element of many distribution systems is the routing and scheduling of vehicles through a set of customers requiring service.

The vehicle routing problem (VRP) involves the design of a set of minimum-cost vehicle routes, originating and terminating at a central depot, for a fleet of vehicles that services a set of customers with known demands. Each customer is serviced exactly once and, furthermore, all the customers must be assigned to vehicles without exceeding vehicle capacities (see Bodin et al. 1983 for a comprehensive survey). In the vehicle routing and scheduling problem with time window constraints (VRSPTW), these issues must be addressed under the added complexity of allowable delivery times, or time windows, stemming from the fact that some customers impose delivery deadlines and earliest-delivery-time constraints.

In the presence of time windows, the total routing and scheduling costs include not only the total travel distance and time costs considered for routing problems, but also the cost of waiting time incurred when a vehicle arrives too early at a customer location or when the vehicle is loaded or unloaded.

The VRSPTW has emerged as an important area for progress in handling realistic complications and generalizations of the basic routing model (Schrage 1981, Bodin et al.). Time windows arise naturally in problems faced by business organizations that work on fixed time schedules. Specific examples include bank deliveries, postal deliveries, industrial refuse collection, dial-a-ride service, and school bus routing and scheduling. So far, this type of constraint has been handled in an ad hoc manner, mostly by manual adjustments to routing-based schedules.

While the spatial problem of routing vehicles has been intensively studied in the literature (Bodin et al.), very little work has been done on the VRSPTW, which encompasses both spatial and temporal aspects of vehicle movements. Almost all approaches to the routing problem suffer from the limitation that they do not consider time window constraints. The existing literature on the problem with time windows has dealt mainly with case studies (Pullen and Webb 1967, Knight and Hofer 1968, and Cook and Russell 1978). The latter authors report computational experience with a k-optimal improvement heuristic, M-TOUR (Russell 1977) for an actual problem involving 163 customers, 4 vehicles and only 15% time-constrained customers. The computational requirements of Russell's method are, however, very large (1.27 minutes of IBM 370/168 CPU time).

The rest of the literature has been directed primarily at special structures. Christofides, Mingozzi and Toth (1981) discuss state space relaxations for dynamic programming approaches to the traveling salesman problem with time windows, while Baker (1983), and Baker and Rushinek (1982) present a branch-and-bound algorithm for a new, time-oriented formulation of the problem. Some progress has been made in the multiperiod routing problem (Federgruen and Lageweg 1980, Fisher et al. 1982). In this problem, the time windows are full days, and a service activity

Subject classification: 632 heuristic programming, 831 transportation, 483 distance algorithms.

0030-364X/87/3502-0254 $01.25

must occur on a specified number of days of the planning horizon.

Other classes of problems with time windows have received increased attention lately: the dial-a-ride problem (Psaraftis 1983, Sexton and Bodin 1985a,b) and the school-bus scheduling problem. Orloff (1976) presents a matching-based heuristic, while Swersey and Ballard (1982) discuss an optimal approach to this problem with the time window discretized. Desrosiers, Soumis and Derochers (1983a,b) develop exact methods for this problem. One algorithm uses a column generation approach in which the columns are generated by using a shortest-path-with-time-windows algorithm (Desrosiers, Pelletier and Soumis 1982). Two other branch-and-bound algorithms involve relaxations of the time-window-related constraints.

Problems with time windows are from a computational complexity perspective quite difficult. Since the (VRP) is NP-hard, by restriction, the VRSPTW is NP-hard. Furthermore, Savelsbergh (1984) has recently shown that even finding a feasible solution to the VRSPTW when the number of vehicles is fixed is itself a NP-complete problem. Therefore, the development of heuristic algorithms for this problem class is of primary interest.

This paper is concerned with the design and analysis of tour-building algorithms for the VRSPTW. All the algorithms presented are extensions of known VRP heuristics. The novelty of our approach consists in incorporating not only the distance but also the time dimension in the heuristic process. The extensive modifications to the original algorithms have produced more robust and flexible methods that can accommodate time window constraints.

1. Heuristics

Tour-building algorithms for the VRSPTW can be divided into sequential and parallel methods. Sequential procedures construct one route at a time until all customers are scheduled. Parallel procedures are characterized by the construction of a number of routes simultaneously. The routes are either allowed to form freely or their number is fixed a priori. In the light of Savelsbergh's results (1984), we will focus on free-routing type of parallel procedures and on sequential methods.

Before describing the heuristics, let us first state the assumptions and introduce the concepts that will be needed throughout the paper. For notational simplicity, we will assume a homogeneous vehicle fleet.

The service at a customer, say i, $i = 1, \ldots, n$, involving pickup and/or delivery of goods or services for s_i units of time, can begin at time b_i, within a *time window* defined by the earliest time e_i and the latest time l_i that customer i will permit the start of service. Hence, if a vehicle travels directly from customer i to customer j and arrives too early at j, it will wait, that is, $b_j = \max\{e_j, b_i + s_i + t_{ij}\}$, where t_{ij} is the direct travel time between i and j. Note that the times b_i for $i = 1, \ldots, n$, at which services begin are decision variables.

We assume that the cost of direct travel from customer i to customer j, is given by $c_{ij} = \rho_1 d_{ij} + \rho_2(b_j - b_i)$, $\rho_1 \geq 0$, $\rho_2 \geq 0$, as defined by the direct distance d_{ij} from customer i to customer j. For example, if $\rho_1 = 0$ and $\rho_2 = 1$, one seeks to minimize total schedule time.

We also assume that the number of vehicles used is free, i.e., the fleet size is determined simultaneously, using the best set of routes and schedules. The vehicles leave the depot, denoted by node 0, at time e_0, at the earliest and must return to the depot by time l_0, at the latest. Note that vehicle departure times from the depot are decision variables.

Solomon (1983) presents an MIP formulation for the VRSPTW. It is based on a model for the basic routing problem given by Fisher and Jaikumar (1981), and on a generalization of the Miller, Tucker and Zemlin (1960) subtour elimination constraints that is similar to the generalization used by Desrosiers, Soumis and Desrochers (1983a) in the context of the traveling salesman problem with time windows.

Of primary importance to the effectiveness and efficiency of heuristics for this problem is the way in which the time window constraints are incorporated in the solution process. Since we will concentrate on route-building procedures, let us first examine the necessary and sufficient conditions for time feasibility when inserting a customer, say u, between the customers i_{p-1}, and i_p, $1 \leq p \leq m$, on a partially constructed feasible route, $(i_0, i_1, i_2, \ldots, i_m)$, $i_0 = i_m = 0$, for which the times to begin service, b_{i_r} for $0 \leq r \leq m$, are known.

We will assume that initially each vehicle leaves the depot at the earliest possible time, e_0. After the complete vehicle schedules have been created, we can adjust the depot departure time separately for each vehicle to eliminate any unnecessary waiting time.

Denote by $b_{i_p}^{new}$ the new time when service at customer i_p, begins, given the insertion of customer u. Also, let w_{i_r} be the waiting time at i_r for $p \leq r \leq m$. If we assume that the triangle inequality holds both for travel distances and times, this insertion defines a

push forward in the schedule at i_p:

$$PF_{i_p} = b_{i_p}^{new} - b_{i_p} \geq 0$$

Furthermore,

$$PF_{i_{r+1}} = \max\{0, PF_{i_r} - w_{i_r i_{r+1}}\}, \quad p \leq r \leq m-1.$$

Similarly, one can also define a push backward in the schedule at i_r for $r = 0, \ldots, p-1$. However, this concept is not appealing, since the current schedule can be pushed backward only if there is no waiting time in the schedule up to i_{p-1} and the vehicle leaves the depot later than e_0. Note that by initially assuming that each vehicle leaves the depot at e_0, we can take advantage of the maximum possible push forward.

If $PF_{i_p} > 0$, some of the customers i_r, $p \leq r \leq m$, could become infeasible. It is easy to see that we should examine these customers sequentially for time feasibility until we find some customer, say i_r with $r < m$, for which $PF_{i_r} = 0$, or i_r is time infeasible, or, in the worst case all the customers i_r, $p \leq r \leq m$ are examined. We have just proved:

Lemma 1.1. *The necessary and sufficient conditions for time feasibility when inserting a customer, say u, between i_{p-1} and i_p, $1 \leq p \leq m$, on a partially constructed feasible route $(i_0, i_1, i_2, \ldots, i_m)$, $i_0 = i_m = 0$ are*

$$b_u \leq l_u, \quad \text{and} \quad b_{i_r} + PF_{i_r} \leq l_{i_r}, \; p \leq r \leq m.$$

Note that if we assume non-euclidean travel distances and times, then it is possible that $PF_{i_p} < 0$, which leaves all the customers time feasible.

Note also that since $i_m = 0$, the use of Lemma 1.1 will ensure that any customer that does not permit the vehicle to return to the depot within the scheduled time will not be added to the partial route.

Let us now describe several heuristic methods for the solution of VRSPTW.

1.1. Savings Heuristics

Our initial approach to solving VRSPTW is to extend the savings heuristic originally proposed for the routing problem by Clarke and Wright (1964). This procedure begins with n distinct routes in which each customer is served by a dedicated vehicle. The parallel version of this tour-building heuristic is characterized by the addition at every iteration of a link of distinct, partially formed routes between two end customers, guided by a measure of cost savings given by

$$sav_{ij} = d_{i0} + d_{0j} - \mu d_{ij}, \quad \mu \geq 0.$$

For example, when $\mu = 1$, sav_{ij} is the savings in distance that results from servicing customers i and j on one route as opposed to servicing them individually, directly from the depot.

Due to the existence of time windows, we now must account for route orientation. Two partial routes with end customers i and j, respectively, have compatible orientations if i is first (last), and j is last (first), i.e., the admissible links are from the last customer (l) on one route to the first customer (f) on another. Furthermore, in addition to taking into account vehicle capacity constraints, we must check time window constraints for violation at every step in the heuristic process. Lemma 1.1 can be easily adapted to the case of links of the form (l, f). As in Lemma 1.1, we simply use the push forward generated at f to efficiently test time feasibility.

We implemented the parallel savings method using list processing and heapsort structures, as proposed by Golden, Magnanti and Nguyen (1977).

As presented, the savings heuristic could find it profitable to join two customers very close in distance but far apart in time. Such links introduce extended periods of waiting time, which can have a high opportunity cost since the vehicle could be servicing other customers instead of, for example, waiting for a customer to open. To account for both the spatial and temporal closeness of customers, we propose limiting the waiting time when joining two customers: letting w_f^{new} be the resulting waiting time at f if l and f were to be joined, and W be a parameter, i.e., if $w_f^{new} > W$, then we do not use the link (l, f).

1.2. A Time-Oriented, Nearest-Neighbor Heuristic

The second heuristic we consider belongs to the class of sequential, tour-building algorithms.

The nearest-neighbor heuristic starts every route by finding the unrouted customer "closest" (in terms of a measure to be described later) to the depot. At every subsequent iteration, the heuristic searches for the customer "closest" to the last customer added to the route. This search is performed among all the customers who can feasibly (with respect to time windows, vehicle arrival time at the depot, and capacity constraints) be added to the end of the emerging route. A new route is started any time the search fails, unless there are no more customers to schedule.

The metric used in this approach tries to account for both geographical and temporal closeness of customers. Let the last customer on the current partial route be customer i, and let j denote any unrouted customer that could be visited next. Then the metric used, c_{ij}, measures the direct distance between the two customers, d_{ij}, the time difference between the completion of service at i and the beginning of service at

j, T_{ij}, and the urgency of delivery to customer j, v_{ij}, as expressed by the time remaining until the vehicle's last possible service start.

Formally,

$$T_{ij} = b_j - (b_i + s_i),$$

$$v_{ij} = l_j - (b_i + s_i + t_{ij}),$$

and

$$c_{ij} = \delta_1 d_{ij} + \delta_2 T_{ij} + \delta_3 v_{ij},$$

is defined by weights satisfying $\delta_1 + \delta_2 + \delta_3 = 1$, $\delta_1 \geq 0$, $\delta_2 \geq 0$, $\delta_3 \geq 0$.

1.3. Insertion Heuristics

This class of sequential, tour-building heuristics initializes every route using one of several criteria to be described later (see Section 3.1). After initializing the current route, the method uses two criteria, $c_1(i, u, j)$ and $c_2(i, u, j)$, at every iteration to insert a new customer u into the current partial route, between two adjacent customers i and j on the route.

Let $(i_0, i_1, i_2, \ldots, i_m)$ be the current route, with $i_0 = i_m = 0$. For each unrouted customer, we first compute its best feasible insertion place in the emerging route as

$$c_1(i(u), u, j(u)) = \min[c_1(i_{p-1}, u, i_p)], \quad p = 1, \ldots, m.$$

As noted previously, inserting u between i_{p-1} and i_p could potentially alter all the times to begin service at customers $(i_p, \ldots, i_m)$. In our computational tests, we have found our time feasibility conditions, i.e., Lemma 1.1, to be much faster than the explicit testing of time feasibility at each customer. Next, the best customer u to be inserted in the route is selected as the one for which

$$c_2(i(u^*), u^*, j(u^*)) = \text{optimum}[c_2(i(u), u, j(u))],$$

u unrouted and feasible.

Customer u^* is then inserted in the route between $i(u^*)$ and $j(u^*)$. When no more customer with feasible insertions can be found, the method starts a new route, unless it has already routed all customers.

We now consider three more specific approaches based on the general insertion criteria just described.

i) $c_{11}(i, u, j) = d_{iu} + d_{uj} - \mu d_{ij}$, $\mu \geq 0$;
$c_{12}(i, u, j) = b_{j_u} - b_j$,

where b_{j_u} is the new time for service to begin at customer j, given that u is on the route;

$$c_1(i, u, j) = \alpha_1 c_{11}(i, u, j) + \alpha_2 c_{12}(i, u, j), \quad \alpha_1 + \alpha_2 = 1;$$

$\alpha_1 \geq 0$, $\alpha_2 \geq 0$;

$$c_2(i, u, j) = \lambda d_{0u} - c_1(i, u, j), \quad \lambda \geq 0.$$

This type of insertion heuristics tries to maximize the benefit derived from servicing a customer on the partial route being constructed rather than on a direct route. For example, when $\mu = \alpha_1 = \lambda = 1$ and $\alpha_2 = 0$, $c_2(i, u, j)$ is the savings in distance from servicing customer u on the same route with customers i and j, as opposed to individual, direct service. The best feasible insertion place for an unrouted customer is the one that minimizes the weighted combination of its distance and time insertion, i.e., the one that minimizes a measure of the extra distance and extra time required to visit the customer.

Clearly, different values of μ and λ lead to different possible criteria for selecting the customer for insertion and its best insertion spot. For example, if $\alpha_2 = 0$ and $c_{0u} = d_{0u}$, we obtain the Mole and Jameson (1976) approach introduced for the routing problem. Therefore, the class of heuristics described in this discussion, which considers insertion costs on both distance and time dimensions, is a generalization of Mole and Jameson's approach.

The second type of insertion heuristics aims to select customers whose insertion costs minimize a measure of total route distance and time.

ii) $c_1(i, u, j)$ is defined as before;
$c_2(i, u, j) = \beta_1 R_d(u) + \beta_2 R_t(u)$, $\beta_1 + \beta_2 = 1$, $\beta_1 \geq 0$, $\beta_2 > 0$,

where $R_d(u)$ and $R_t(u)$ are the total route distance and time of the current partial route, if u is inserted.

In our third approach, the temporal aspect of the criterion used for insertion also accounts for the urgency of servicing a customer.

iii) $c_{11}(i, u, j)$ and $c_{12}(i, u, j)$ are defined as before;
$c_{13}(i, u, j) = l_u - b_u$;
$c_1(i, u, j) = \alpha_1 c_{11}(i, u, j) + \alpha_2 c_{12}(i, u, j) + \alpha_3 c_{13}(i, u, j)$,

where

$$\alpha_1 + \alpha_2 + \alpha_3 = 1, \quad \alpha_1 \geq 0, \quad \alpha_2 \geq 0, \quad \alpha_3 \geq 0;$$
$c_2(i, u, j) = c_1(i, u, j)$.

It is easy to see that, in fact, this class of heuristics is a generalization of the time-oriented, nearest-neighbor heuristic, in that we allow insertion of an unrouted customer in any feasible location between a pair of

customers on the route, rather than only at the end of the route.

In all the approaches presented, the insertion of unrounted customers is guided by both geographical and temporal criteria. As a consequence, we expect that the waiting time in the schedules produced by these heuristics will be significantly lower than that produced by distance-driven criteria.

1.4. A Time-Oriented Sweep Heuristic

This heuristic can be viewed as a member of a broad class of approximation methods that decompose the VRSPTW into a clustering stage and a scheduling stage. In the first phase, we assign customers to vehicles as in the original sweep heuristic (Gillett and Miller 1974). In the second phase, we create a one-vehicle schedule for the customers in this sector, using a tour-building heuristic. The insertion heuristic of type (i) was used in our computer implementation. Due to the time window constraints, some customers in this cluster could remain unscheduled.

After eliminating scheduled customers from further consideration, we repeat the clustering-scheduling process. To preserve geographical cohesiveness, one might consider different seed selection criteria for the next cluster. We use a simple rule that bisects the sector just considered and, assuming a counterclockwise sweep in the more counterclockwise half-sector lets the seed for a new cluster be the unscheduled customer with the smallest angle formed by the ray from the depot through that customer and the bisector.

The intuition for partitioning the unscheduled customers in the sector into two subsets is that the customers in the more clockwise half-sector will be relatively far away from the new cluster. By inserting these customers at a later stage, we hope to generate a better schedule.

We then repeat the process until all customers have been scheduled.

In order to evaluate the computational capabilities of the heuristics presented, we have developed a set of test problems. This approach is necessary given that no benchmark problem set is available in the literature for vehicle routing and scheduling problems with time windows.

2. Development of the Problem Sets

We generated six sets of problems; the actual data are available from the author. The design of these test problems highlights several factors that can affect the behavior of routing and scheduling heuristics. These factors include: geographical data; the number of customers serviced by a vehicle; and time window characteristics such as percentage of time-constrained customers, and tightness and positioning of the time windows.

The data used for the customer coordinates and demands are based on the data for some of the problems from the standard set of routing test problems given in Christofides, Mingozzi and Toth (1979). The geographical data are randomly generated by a random uniform distribution (denote the corresponding problem sets by R1 and R2), clustered (denote the corresponding problem sets by C1 and C2), and semi-clustered (denote the corresponding problem sets by RC1 and RC2). By a semiclustered problem, we mean one that contains a mix of randomly generated data and clusters. Problem sets R1, C1, and RC1 have a short scheduling horizon. The length of route-time constraint acts as a capacity constraint which, together with the vehicle capacity constraints, allows only a few customers to be serviced by the same vehicle. In contrast, the sets R2, C2 and RC2 have a long scheduling horizon; this characteristic, coupled with large vehicle capacities, permits many customers to be serviced by the same vehicle.

Given certain geographical and demand data, we created the VRSPTW test problems by generating time windows of various widths for different percentages of customers. In terms of time window density, that is, the percentage of customers with time windows, we created problems with 25, 50, 75 and 100% time windows.

We now present a method we designed for the random generation of time window constraints. This method was used for the development of the problem sets R1, R2, RC1, RC2. First, we select the percentage of customers to receive time windows, say f, $0 < f \leq 1$. We then generate n random numbers from the random uniform distribution over the interval (0, 1) and sort them. This approach creates a random permutation of customers, $(i_1, \ldots, i_n)$, with i_j being the position after the sort of the jth random number generated. Finally, we assign time windows to the customers $i_1, \ldots, i_{n_1}$, with n_1 chosen to be the integer that most closely approximates fn. The time windows have a randomly generated center and width. We choose the center of the time window for customer i_j for $j = 1, \ldots, n_1$ as being a uniformly distributed, randomly generated number in the interval $(e_0 + t_{0,i_j}, l_0 - t_{i_j,0} - s_{i_j})$. To create the time window's width for i_j, we generate half the width as a normally distributed random number.

We used a somewhat different method for the clustered problems, C1 and C2. We first run a 3-opt routine (Lin 1965) on each cluster to create routes and then produce schedules by selecting an orientation for each cluster. The time window constraints are generated by choosing the center as the arrival time at each customer; the width and density are derived as before.

Given the design method, problem sets C1 and C2 are composed of structured problems in the sense that the customers appear in clusters and the time windows are positioned around the arrival times at customers. This approach permits the identification of a very good, possibly optimal, cluster-by-cluster solution which, in turn, provides an additional means of evaluating heuristic performance.

It is worth mentioning that the best solution we found for C1 requires 10 vehicles, for a total schedule of 9,829 units, a distance of 829 units and no waiting time. For C2, the best schedule we found requires a fleet of 3 vehicles, a total of 9,591 units of time, 591 units of distance and no waiting time.

All the test problems are 100-customer euclidean problems. This problem size is not limiting for the methods presented, since much larger problems could be solved. Travel times between customers are taken to equal the corresponding distances. Furthermore, a homogeneous fleet is assumed.

3. Computational Study

3.1. Computational Results

Earlier computational experiments for the VRP (Christofides, Mingozzi and Toth 1979) indicate that the sweep algorithm performed much better than either the savings or the insertion heuristics on a number of randomly generated problems. For several structured problems, the reverse was true. Also, the insertion heuristic had a slightly better overall performance than the savings method. All the algorithms used 2-optimal refining procedures to improve the routes.

To analyze the behavior of the VRSPTW heuristics described in Section 1, we programmed them in FORTRAN and performed computational tests on the problem classes described in the last section. In obtaining the computational results, we did not use any k-optimal improvement procedures.

Solution quality is measured in terms of the minimum number of vehicles, minimum schedule time, minimum distance, and minimum waiting time in that order, i.e., we use a lexicographic ordering of the solutions. Hence, a schedule with, for example, fewer vehicles and a higher total schedule time will be better than one utilizing more vehicles but having a lower total schedule time.

In the tables that follow, for each heuristic, the numbers on the left-hand side of the column headed "Average solution values and CPU time" are the total schedule time, total distance and total waiting time, respectively, of the best of several runs using different parameter values and initialization criteria, averaged over the respective problem set. The total schedule time is the sum of the total travel time, total service time, and total waiting time, respectively. The numbers on the right-hand side of this column are the number of routes of the best solution and the total CPU time for the different runs, averaged over the respective problem set. The solution times are in seconds on the DEC-10, and include the time to read the data and compute intercustomer distances and times. The column headed "Percent deviation from the best average solution value" presents the deviation, in percentages, of each heuristic's average solution value from the best average solution value, on each solution dimension. The specific computational results are available from the author.

In obtaining the computational results reported, we used the following parameters:

Savings. The results are the best of two runs, with $\mu = 1, 0.2$.

Savings with Waiting-Time Limit. On R1, the computational results are the best of four runs: $\mu = 1, 0.2$, and the waiting time limited to 30 and 60 units, respectively. On C1, two additional runs were made with the waiting time limited to 120 units.

Insertion-Criterion (i). The results are the best of eight runs. The parameters used, $(\mu, \lambda, \alpha_1, \alpha_2)$, are: (1, 1, 1, 0), (1, 2, 1, 0), (1, 1, 0, 1), and (1, 2, 0, 1). Two initialization criteria were tested:

(a) the farthest unrouted customer, and

(b) the unrounted customer with the earliest deadline.

Insertion-Criterion (ii). Eight runs were performed for this heuristic. The parameters used, $(\mu, \alpha_1, \alpha_2, \beta_1, \beta_2)$, are (1, 0.5, 0.5, 0.5, 0.5), (1, 1, 0, 0.5, 0.5), and (1, 0, 1, 1, 0). Three initialization criteria were used in conjunction with the first two sets of parameters: criteria (a) and (b), and

(c) the unrouted customer with the minimum equally weighted combination of direct route-time and distance.

Only criteria (a) and (b) were used with the third set of parameters.

Insertion-Criterion (iii). The results here are the best of ten runs with $(\mu, \alpha_1, \alpha_2, \alpha_3) = (1, 0.5, 0.5, 0)$,

(1, 0.4, 0.4, 0.2), and (1, 0, 1, 0). The initialization criteria (a), (b), and (c) were used with the first set of parameters. In addition to the previous three criteria, we used the actual heuristic criterion for initialization in conjunction with the second set of parameters. For the third set of parameters, criteria (a), (b) and the actual heuristic criterion were used for initialization.

Sweep. The results for this heuristic are the best of eight runs with the parameters used for the insertion heuristic with criterion (i). The first seed was always chosen as the first customer encountered in the first quadrant in a counterclockwise sweep.

Nearest Neighbor. The parameters used for this heuristic, (δ_1, δ_2, δ_3), are: (0.4, 0.4, 0.2), (0, 1, 0), (0.5, 0.5, 0), and (0.3, 0.3, 0.4).

Tables I–VI compare the methods on each problem set, for solution quality and running time.

With respect to solution quality, heuristic I1 performed the best. Its behavior was very stable across all problem environments, obtaining the best solution for 27 out of the 56 problems tested; even when it did not beat the other methods, its deviation from the best solution was quite small. Additional support for these results is provided by further comparing, on C1 and on C2, this heuristic's values with the best known solution (see Section 2). I1 was consistently close to the best known total schedule time on both problem sets; on C1, its smallest deviation was 0.3%, its largest deviation was 8.3%, with an average of 2.8%, while on C2, its deviation was within 5% in seven out of the eight problems tested.

The insertion-based sweep strategy was found to be very good for problems with many customers per vehicle. This heuristic obtained the best solution in 13 out of the 27 cases. This method performed very well on the clustered problem sets C1 and C2, where it was superior to the other heuristics in 4 and 7 cases, respectively. When compared to the best known solution for these problem sets, its largest total schedule time deviation was only 9.7% on C1, and 3.8% on C2. While it did not behave well on R1 and RC1, we believe its behavior on these problems could be improved by the consideration of each of the customers as initial seeds.

The insertion heuristics I2 and I3 dominated the waiting time dimension for problems with a short

Table I
Comparison of the Algorithms on R1

Algorithm	Average Solution Values[a] and CPU Time[b]		Percent Deviation from Best Average Solution Value		No. of Problems on Which Method Found Best Solution
Savings (SAV)	3116.3	16.6	15.6%	22%	0
	1498.9		4.3%		
	617.4	2.4	138.6%		
Savings, waiting time limit (SWT)	2925.2	15.1	8.5%	11%	0
	1517.2		5.6%		
	408.1	4.7	57.7%		
Insertion criterion i (I1)	2695.5	13.6	0.0%	0%	11
	1436.7		0.0%		
	258.8	24.7	0.0%		
Insertion criterion ii (I2)	2888.1	14.5	7.1%	6.6%	0
	1638.7		14.1%		
	249.4	25.5	−3.6%		
Insertion criterion iii (I3)	2855.1	14.1	5.9%	3.7%	1
	1651.7		15.0%		
	203.3	31.7	−21.4%		
Nearest Neighbor (NN)	2968.7	14.5	10.1%	6.6%	0
	1600.1		11.4%		
	368.0	8.9	42.2%		
Sweep (S)	2817.4	14.6	4.5%	7.3%	0
	1499.7		4.4%		
	317.7	18.2	22.8%		

[a] Schedule time Number of routes
Distance
Waiting time CPU time.
[b] CPU time in seconds on the DEC-10.

Table II
Comparison of the Algorithms on C1

Algorithm	Average Solution Values[a] and CPU time[b]		Percent Deviation from Best Average Solution Value		No. of Problems on Which Method Found Best Solution
SAV	11125.7	11.7	10.1%	17%	0
	976.2		2.5%		
	1149.4	2.3	654.7%		
SWT	10469.1	10.7	3.6%	7%	5
	987.4		3.7%		
	481.7	6.9	216.2%		
I1	10104.2	10.0	0.0%	0%	0
	951.9		0.0%		
	152.3	25.3	0.0%		
I2	10174.3	10.1	.7%	1%	2
	1049.8		10.3%		
	124.5	25.3	−18.2%		
I3	10174.9	10.0	0.7%	0%	2
	1103.3		15.9%		
	71.5	31.1	−53.0%		
NN	10472.5	10.2	3.6%	2%	0
	1171.2		23.0%		
	301.3	8.4	97.8%		
S	10133.8	10.0	0.3%	0%	4
	940.8		−1.2%		
	193.0	13.5	26.7%		

[a] Schedule time Number of routes
Distance
Waiting time CPU time.
[b] CPU time in seconds on the DEC-10.

Table III
Comparison of the Algorithms on RC1

Algorithm	Average Solution Values[a] and CPU Time[b]		Percent Deviation from Best Average Solution Value		No. of Problems on Which Method Found the Best Solution
I1	2775.0	13.5	0.0%	0.0%	7
	1596.5		0.0%		
	178.5	23.8	0.0%		
I2	3029.5	14.2	9.2%	5.2%	0
	1874.4		17.4%		
	155.1	24.4	−13.1%		
I3	3014.0	14.0	8.6%	3.7%	1
	1849.7		15.8%		
	164.2	30.3	−8.0%		
NN	3057.2	14.2	10.2%	5.2%	0
	1800.0		12.7%		
	257.2	8.9	44.1		
S	3094.5	14.9	11.5%	10.4%	0
	1804.5		13.0%		
	290.0	18.4	62.5%		

[a] Schedule time Number of routes
Distance
Waiting time CPU time.
[b] CPU time in seconds on the DEC-10.

Table IV
Comparison of the Algorithms on R2

Algorithm	Average Solution Values[a] and CPU Time[b]		Percent Deviation from Best Average Solution Value		No. of Problems on Which Method Found Best Solution
I1	2578.1	3.3	−0.5%	3.1%	4
	1402.4		−3.2%		
	175.6	62.6	24.2%		
I2	2645.8	3.3	2.1%	3.1%	0
	1470.7		1.5%		
	175.1	71.1	23.8%		
I3	2676.4	3.4	3.3%	6.2%	0
	1474.6		1.8%		
	201.7	81.7	42.6%		
NN	2719.6	3.4	5.0%	6.2%	3
	1472.3		1.6%		
	247.4	7.7	75.0%		
S	2590.1	3.2	0.0%	0.0%	4
	1448.6		0.0%		
	141.4	40.5	0.0%		

[a] Schedule time Number of routes
Distance
Waiting time CPU time
[b] CPU time in seconds on the DEC-10.

Table V
Comparison of the Algorithms on C2

Algorithm	Average Solution Values[a] and CPU Time[b]		Percent Deviation from Best Average Solution Value		No. of Problems on Which Method Found Best Solution
I1	9921.4	3.1	1.7%	3.3%	1
	692.7		−2.7%		
	228.6	43.0	426.7%		
I2	10151.4	3.4	4.1%	13.3%	1
	921.5		29.4%		
	229.9	44.5	429.7%		
I3	10118.5	3.5	3.7%	16.7%	1
	1072.7		50.7%		
	45.7	52.9	5.3%		
NN	10785.9	3.5	10.6%	16.7%	1
	963.1		35.3%		
	822.7	7.7	1795.6%		
S	9755.2	3.0	0.0%	0.0%	7
	711.9		0.0%		
	43.4	18.5	0.0%		

[a] Schedule time Number of routes
Distance
Waiting time CPU time
[b] CPU time in seconds on the DEC-10.

scheduling horizon. Otherwise, they were rarely able to obtain the best solution. It seems that these heuristics will be successful for problems with idle time as the primary objective.

The time-oriented, nearest-neighbor method also had limited success; this occurred in the long scheduling horizon problems that permit many customers to be serviced by the same vehicle.

Table VI
Comparison of the Algorithms on RC2

Algorithm	Average Solution Values[a] and CPU Time[b]		Percent Deviation from Best Average Solution Value		No. of Problems on Which Method Found Best Solution
I1	2955.4	3.9	0.0%	0.0%	4
	1682.1		0.0%		
	273.2	51.7	0.0%		
I2	3128.4	4.1	5.8%	5.1%	0
	1797.6		6.9%		
	330.7	54.0	21.0%		
I3	3149.1	4.0	6.5%	2.6%	1
	1816.4		8.0%		
	332.7	63.4	21.8%		
NN	3120.0	3.9	5.6%	0.0%	1
	1754.7		4.3%		
	365.2	7.7	33.7%		
S	3007.9	4.0	1.8%	2.6%	2
	1735.7		3.2%		
	272.1	31.6	−0.4%		

[a] Schedule time Number of routes
Distance
Waiting time CPU time
[b] CPU time in seconds on the DEC-10.

The savings heuristic with waiting-time limit, while a considerable improvement over the original savings method, did not perform well in general. After we examined its behavior on R1 and C1, preliminary computational experiments on the rest of the problem sets indicated that this heuristic will, in general, require more vehicles than the number utilized by the other heuristics. Therefore, we did not examine its behavior any further. Nevertheless, given that it found the best solutions to 5 problems of C1, it might conceivably be used with success in a sweep-type heuristic.

In terms of computation time, all the heuristics seem to be very efficient. Algorithms SWT and SAV were the fastest. Our results indicate that the efficiency of the insertion methods increased with the increase in the percentage of time window constraints and their tightness. This result is due to a lower number of feasible insertions possible in such problems.

3.2. Parametric Analysis

Most of the heuristics presented are parameterized. It is thus of interest to know whether there are any relationships between the parameter values and the initialization criteria that produced the best solution values, and the corresponding problem structures. Given our computational results, we will focus on the heuristic I1.

We begin by examining the effect of distance insertion ($\alpha_1 = 1$, $\alpha_2 = 0$) versus time insertion ($\alpha_1 = 0$, $\alpha_2 = 1$). For the problem sets involving long scheduling horizons, time insertion proved clearly superior to distance insertion. It was used in obtaining the best solution to 21 out of 27 problems. The few exceptions, two for each problem set, where distance insertion performed best, were problems with less than 100% time window density, or the least tight problems.

For the problem sets with a short scheduling horizon, time insertion proved useful for most of the tight and high density problems of R1 and RC1. It was used for 5 problems of RC1, and 4 problems of R1.

Not surprisingly, time insertion was used for the less constrained problems of C1, 4 problems, since it was able to provide more intelligence than distance insertion in directing the heuristic toward the optimal solution.

In conclusion, time insertion proved superior to distance insertion, especially in problems involving many customers per vehicle, and/or high density and tight time windows. Distance insertion, by emphasizing the geographical component, can lead to higher total schedule time from accumulated waiting time and, possibly, additional vehicles. We should note that we observed the same pattern for the insertion-based-sweep strategy.

Examining now the choice of parameters λ, we find that $\lambda = 2$ was used in obtaining the best solution to

35 problems, while $\lambda = 1$ was used in only 21 cases. On R1, $\lambda = 2$ was utilized for 9 out of 12 problems, while on RC2, it was used for 7 out of 8. Five out of 9 C1 problems used $\lambda = 2$. This parameter value was also used for 5 out of 8 RC1 problems, 5 out of 11 R2 problems, and 4 out of 8 C2 problems.

The initialization criterion used for a heuristic can have a significant impact on its behavior. We used two initialization criteria for I1: the farthest unrouted customer and the unrouted customer with the earliest deadline. The former criterion proved successful for the problem sets with short scheduling horizons. It was used in obtaining 23 out of the 29 best solutions to these problems: all R1 problems except one, all RC1 problems, and 4 C1 problems. In contrast, the latter criterion was better on the long scheduling horizon problems in which, given that only a few vehicles were used, it was of primary importance to service customers with early due dates at the beginning of the scheduling horizon so no additional vehicles would be necessary. In these experiments, the earliest deadline criterion was used in 16 out of the 27 best solutions: 6 problems of R2, 6 problems of C2, and 4 problems of RC2.

Note that most of these problems are characterized by 100% density and/or tight time windows. This initialization method was quite successful on the structured problems since it was able to guide the heuristic search toward the optimal solution.

4. Conclusions

We have presented the development of heuristics and test problem sets and have reported our computational experiments for VRSPTW. Our results indicate that the insertion heuristic I1 proved to be very successful. A parametric analysis of this heuristic revealed the importance of time driven insertions for heavily time-constrained problems, particularly when many customers are to be scheduled for each vehicle. In the latter case, it seems that a very effective strategy is to embed this heuristic in a sweep-type approach.

The excellent performance of the insertion heuristic I1 can be explained if we realize that, while routing problems seems to be driven by the assignment-of-customers-to-vehicles component—as indicated by the success of the Fisher and Jaikumar generalized assignment heuristic—the sequencing aspect of the problem seems to drive routing problems dominated by time windows. It is this aspect of the problem that the insertion heuristic I1 captures so well.

We believe that our computational study provides some definite guidelines concerning the relative effectiveness of the various tour-building VRSPTW heuristics suggested. Furthermore, given the wide variety of routing and scheduling environments used in our study, we consider the relative performance of these heuristics on the test problems indicative of their relative performance in general.

Based on our study, we recommend the use of the insertion heuristic I1, possibly embedded in a hybrid sweep-insertion approach, to obtain excellent initial VRSPTW solutions in a reasonable amount of computing time. Given its very stable behavior, we believe that this heuristic will perform very well on practical problems. Further support for this conclusion is provided by the recent work of Jaw et al. (1984) that showed the effectiveness of an insertion-based procedure in a time-window constrained real dial-a-ride environment.

Acknowledgment

The author has benefited from fruitful discussions with Marshall Fisher, Ramchandran Jaikumar, Monique Guignard-Spielberg, Pradeep Kedia and Anand Desai. He also wishes to thank the three anonymous referees for their useful comments. The computer time for this project was provided by the Wharton Computer Center of the University of Pennsylvania.

References

BAKER, E. 1983. An Exact Algorithm for the Time-Constrained Traveling Salesman Problem. *Opns. Res.* **31**, 938–945.

BAKER, E., AND S. RUSHINEK. 1982. Large Scale Implementation of a Time Oriented Vehicle Scheduling Model. U.S. Department of Transportation, Urban Mass Transit Administration.

BODIN, L., B. GOLDEN, A. ASSAD AND M. BALL. 1983. Routing and Scheduling of Vehicles and Crews: The State of the Art. *Comput. Opns. Res.* **10**, 62–212.

CHRISTOFIDES, N., A. MINGOZZI AND P. TOTH. 1979. The Vehicle Routing Problem. In *Combinatorial Optimizations*, N. Christofides, R. Mingozzi, P. Toth, and C. Sandi (eds.). John Wiley & Sons, New York.

CHRISTOFIDES, N., A. MINGOZZI AND P. TOTH. 1981. State Space Relaxation Procedures for the Computation of Bounds to Routing Problems. *Networks* **11**, 145–164.

CLARKE, G., AND W. WRIGHT. 1964. Scheduling of Vehicles from a Central Depot to A Number of Delivery Points. *Opns. Res.* **12**, 568–581.

COOK T., AND R. RUSSELL. 1978. A Simulation and Statistical Analysis of Stochastic Vehicle Routing with Timing Constraints. *Decision Sci.* **9,** 673–687.

DESROSIERS, J., P. PELLETIER, AND F. SOUMIS. 1982. Shortest Path with Time Windows. Working Paper 81-21, Ecole des Hautes Etudes Commerciales, Montreal.

DESROSIERS, J., F. SOUMIS AND M. DESROCHERS. 1983a. Routing With Time Windows by Column Generation. Working Paper 277, Centre de Reserche sur les Transports, University of Montreal.

DESROSIERS, J., F. SOUMIS AND M. DESROCHERS. 1983b. Routing with Time Windows: Synthesis. Working Paper G-83-05, Ecole des Hautes Etudes Commerciales, Montreal.

FEDERGRUEN, A., AND B. J. LAGEWEG. 1980. *Hierarchical Distribution Modelling with Routing Costs.* Mathematical Centre, Amsterdam.

FISHER, M., AND R. JAIKUMAR. 1981. A Generalized Assignment Heuristic for Vehicle Routing. *Networks* **11,** 109–124.

FISHER, M., A. GREENFIELD, R. JAIKUMAR AND P. KEDIA. 1982. Real-Time Scheduling of a Bulk Delivery Fleet: Practical Application of Lagrangian Relaxation. Working Paper 82-10-11, Department of Decision Sciences, University of Pennsylvania.

GILLET, B., AND L. MILLER. 1974. A Heuristic Algorithm For the Vehicle Dispatching Problem. *Opns. Res.* **22,** 340–349.

GOLDEN, B., T. MAGNANTI AND H. NGUYEN. 1977. Implementing Vehicle Routing Algorithms. *Networks* **7,** 113–148.

JAW, J., A. ODONI, H. PSARAFTIS AND N. WILSON. 1984. A Heuristic Algorithm for the Multi-Vehicle Advance Request Dial-A-Ride Problem with Time Windows. Working Paper MIT-UMTA-83-L.

KNIGHT, K., AND J. HOFER. 1968. Vehicle Scheduling with Timed and Connected Calls: A Case Study. *Opnl. Res. Quart.* **19,** 299–310.

LIN, S. 1965. Computer Solutions of the Traveling Salesman Problem. *Bell System Tech. J.* **44,** 2245–2269.

MILLER, C., A. TUCKER AND R. ZEMLIN. 1960. Integer Programming Formulations and Traveling Salesman Problems. *J. Assoc. Comput. Mach.* **7,** 326–329.

MOLE, R., AND S. JAMESON. 1976. A Sequential Route-Building Algorithm Employing a Generalized Savings Criterion. *Opnl. Res. Quart.* **27,** 503–511.

ORLOFF, C. 1976. Route Constrained Fleet Scheduling. *Trans. Sci.* **10,** 149–168.

PSARAFTIS, H. 1983. An Exact Algorithm for the Single Vehicle Many-to-Many Dial-A-Ride Problem with Time Windows. *Trans. Sci.* **17,** 351–358.

PULLEN, H., AND M. WEBB. 1967. A Computer Application to a Transport Scheduling Problem. *Comput. J.* **10,** 10–13.

RUSSELL, R. 1977. An Effective Heuristic for the M-Tour Traveling Salesman Problem with Some Side Conditions. *Opns. Res.* **25,** 517–524.

SAVELSBERGH, M. 1984. Private Communication by Alexander Rinnooy Kan.

SCHRAGE, L. 1981. Formulation and Structure of More Complex/Realistic Routing and Scheduling Problems. *Networks* **11,** 229–232.

SEXTON, T., AND L. BODIN. 1985a. Optimizing Single Vehicle Many-to-Many Operations with Desired Delivery Times: I. Scheduling. *Trans. Sci.* **19,** 378–410.

SEXTON, T., AND L. BODIN. 1985b. Optimizing Single Vehicle Many-to-Many Operations with Desired Delivery Times: II. Routing. *Trans. Sci.* **19,** 411–435.

SOLOMON, M. 1983. Vehicle Routing and Scheduling with Time Window Constraints: Models and Algorithms. Ph.D. Dissertation, Dept. of Decision Sciences, University of Pennsylvania.

SWERSEY, A., AND W. BALLARD. 1982. Scheduling School Buses. Working Paper, Yale School of Organization and Management.