Simulated annealing metaheuristics for the vehicle routing problem with time windows

Wen-Chyuan Chiang and Robert A. Russell
Department of Quantitative Methods, University of Tulsa, Tulsa, OK 74014, USA

This paper develops simulated annealing metaheuristics for the vehicle routing and scheduling problem with time window constraints. Two different neighborhood structures, the λ-interchange mechanism of Osman and the k-node interchange process of Christofides and Beasley, are implemented. The enhancement of the annealing process with a short-term memory function via a tabu list is examined as a basis for improving the metaheuristic approach. Computational results on test problems from the literature as well as large-scale real-world problems are reported. The metaheuristics achieve solutions that compare favorably with previously reported results.

Keywords: Metaheuristics, simulated annealing, vehicle routing problems with time windows.

1. Introduction

This paper presents simulated annealing metaheuristics for the vehicle routing problem with time windows (VRPTW). The VRPTW has recently been the subject of intensive research and can be used to model many real-world problems. Applications of the VRPTW include bank deliveries, postal deliveries, industrial refuse collection, national franchise restaurant deliveries, school bus routing, and security patrol services.

The objective of the VRPTW is to design a set of minimum-cost vehicle routes originating and terminating at a central depot. The routes must serve a set of customers with known demands. Each customer is to be serviced (pick-up or delivery but not both) once during the planning horizon and customers must be assigned to vehicles without exceeding vehicle capacities. In the VRPTW, some or all of the customers must be serviced during allowable delivery times or time windows. The time windows are treated as hard constraints in that service after the upper limit of the time window is infeasible and arrival before the lower limit of the time window results in a wait before service can begin. The objective has multiple criteria in that the goal is to minimize not only the number of vehicles

required to service all customers, but also the total travel time and total travel distance incurred by the fleet of vehicles.

Previous work on the VRPTW has included both optimization algorithms and heuristic approaches. The NP-completeness of the VRPTW has centered research focus on tour construction and tour improvement heuristics. Solomon's insertion heuristics [32] remain the benchmark for construction type approaches. More recently, Potvin and Rousseau [28] and Russell [30] have developed parallel construction procedures for the VRPTW. Other construction procedures include the greedy randomized adaptive search procedure of Kontoravdis and Bard [20]. Interesting surveys by Solomon and Desrosiers [33] and Desrochers et al. [7] provide more information on construction approaches. Tour improvement procedures are represented by branch exchange heuristics such as the k-opt heuristic described by Russell [29], Savelsbergh [31], and the Or-opt heuristic of Or [26]. Other local search procedures are described by Solomon et al. [34].

Recently, Desrochers et al. [6] have extended the maximum problem size that can be solved optimally. Using a column generation approach, they have been able to solve seven of the fifty-six Solomon test problems [32]. The problems consist of $n = 100$ customers. Fisher et al. [9] have developed a k-tree relaxation approach that generated optimal solutions to two of the Solomon test problems. However, because of its exponential nature and the fact that the branch and bound procedure needs to be tuned for a specific problem, it is unlikely that these optimization procedures will yield optimal solutions for the general case of $n = 100$ or larger problems.

Heuristic approaches are required for the solution of most real-world vehicle routing applications. To improve upon the limitations of local search procedures, researchers are turning to artificial intelligence (AI) approaches to find better solutions. Thangiah et al. [36] developed the GIDEON system, which uses genetic algorithms in a cluster-first route-second approach to the VRPTW. Osman [27] has developed metastrategies including simulated annealing and tabu search for the vehicle routing problem (VRP). Thangiah et al. [37] have developed hybrid genetic, simulated annealing, and tabu search methods for the VRPTW. The solution quality obtained with these metaheuristics for the VRP and VRPTW has often led to new best known solutions to problems from the literature.

In this paper, we develop simulated annealing metaheuristics for the VRPTW. The metaheuristics are applied to the parallel construction approach of Russell [30] that incorporates improvement procedures during the construction process. Two different types of neighborhoods are compared as a basis for the solution process. The λ-interchange mechanism of Osman [27] and the k-node interchange process of Christofides and Beasley [5] are compared on test problems from the literature as well as large-scale real-world VRPTWs. Lastly, the enhancement of the annealing process with a short-term memory function via a tabu list is examined as a basis for improving the metaheuristic approach.

2. A parallel construction approach

The two-phase approach to the VRPTW presented in this paper can be described as a parallel construction procedure with a simulated annealing tour improvement heuristic. This approach differs from standard construction/improvement methods in that the improvement process is invoked periodically during the construction process. Russell [30] found that applying an improvement procedure during route construction helped to improve solutions to the VRPTW when using a local search method. The local search method employed the Christofides and Beasley [5] neighborhood scheme, as described in section 3.3. Russell invoked global route improvement procedures after every $0.1n$ to $0.16n$ customers have been inserted on routes, where n is the total number of customers. In this paper, the simulated annealing tour improvement heuristic is invoked periodically during the parallel route construction process. It is also applied at the end of the construction process after all n customers have been inserted on routes.

The parallel construction procedure is based on the insertion heuristic of Solomon [32], but differs from that sequential approach in that a specified number of routes are constructed in parallel rather than one at a time. Potvin and Rousseau [27] have independently developed a parallel construction heuristic. The parallel approach developed here differs from the Potvin and Rousseau approach mainly in the selection of seed points, customer insertion criteria, and handling of infeasible solutions.

To describe the parallel construction heuristic, we assume that there are n customers to be serviced. For each customer i, let

q_i = demand for pick-up or delivery for customer i,
s_i = required service time for customer i,
e_i = earliest time service can begin at customer i,
l_i = latest time service can begin at customer i,
t_{ij} = travel time between two customers i and j,
d_{ij} = distance between two customers i and j,
b_i = current time when service can begin at customer i,
b_{ij} = time when service can begin at customer j given that customer i is inserted immediately before customer j in the route,
Q_v = capacity of vehicle v, $v = 1, 2, \ldots, V$.

The calculation for b_{ij} is

$$b_{ij} = \max\{e_j, b_i + s_i + t_{ij}\}.$$

Arrival at customer j before e_j will require a wait time $w_j = e_j - (b_i + s_i + t_{ij})$. Arrival after time l_j is infeasible. Thus, the time window constraints are treated as hard constraints.

The parallel construction heuristic requires an initial estimate, V, of the number of vehicles required. This estimate can be obtained from the number of existing routes or a heuristic such as Solomon's sequential insertion heuristic. We assume that the number of vehicles required is free to be determined by the solution process.

Given V, the parallel insertion heuristic requires n iterations, assigning one customer per iteration to the best available route. Routes are initialized by using the seed point generation scheme of Fisher and Jaikumar [8]. These seed points represent fictitious customers on a route and are deleted as soon as one real customer is added to the route. Customers are selected for route insertion in a particular order. Three ordering rules are used to help achieve time window feasibility during the construction phase. Rule one for selecting the next customer is based on the smallest early time window parameter e_j. Rule two is based on the tightness of the time window as calculated by $100\ (l_j - e_j) - d_{0j}$, where d_{0j} denotes the distance from the depot to customer j. The number 100 is an arbitrary weight used to emphasize the tightness of the time window relative to the distance from the depot. Rule three is based on the largest value of d_{0j}.

The best insertion location for a given customer on a specific route is determined by the route distance and travel time increase, as described by Solomon [32]. Consider the measures c_1 and c_2 for inserting customer j between customers i and k on route r:

$$c_1(i, j, k) = d_{ij} + d_{jk} - d_{ik}, \tag{1}$$

$$c_2(i, j, k) = b_{jk} - b_k. \tag{2}$$

The measures c_1 and c_2 calculate the distance increase and local schedule time increase, respectively, of inserting j between i and k. Let

$$c_3(r) = \alpha_1 c_1(i, j, k) + \alpha_2 c_2(i, j, k), \quad \alpha_1 + \alpha_2 = 1. \tag{3}$$

The criterion $c_3(r)$ is used to select the best insertion location for customer j on route r. Thus, customer j is inserted on route r for which

$$r = \arg\min\{c_3(v);\ v = 1, 2, \ldots, V\}.$$

The solution of the parallel insertion heuristic represents the best of six passes. The parameters used are $(\alpha_1 = 1,\ \alpha_2 = 0)$ and $(\alpha_1 = 0,\ \alpha_2 = 1)$. The two parameters together with three customer ordering rules generate six passes in each run to determine the best solution.

The initial choice of fleet size V can yield an infeasible solution with some customers unrouted. In this instance, the final step of the construction procedure is to apply Solomon's insertion heuristic to the set of unrouted customers. The resulting W routes are included to yield a final solution with $R = V + W$ routes. In problems

where the initial choice V yields a feasible solution, i.e. $R = V$, another iteration of the construction procedure is attempted with $R = V - 1$ routes in order to minimize the number of routes required in the final solution.

3. Simulated annealing improvement heuristics

Simulated Annealing (SA) is an algorithmic approach to solving combinatorial optimization problems. The relatively recent work of Kirkpatrick et al. [19] and Cerny [3] stimulated considerable interest in the method. The name of the algorithm derives from an analogy between solving optimization problems and simulating the annealing of solids as proposed by Metropolis et al. [25]. Simulated annealing can be viewed as an enhanced version of the iterative improvement method, in which an initial solution is repeatedly improved by making small local changes until no such changes yield a better solution. Simulated annealing randomizes the local search procedure and in some instances allows for solution changes that worsen the solution. This constitutes an attempt to reduce the probability of becoming trapped in a locally suboptimal solution. In nature it is a randomization algorithm and can be asymptotically viewed as an optimization procedure. In any practical implementation, however, it behaves like an approximation algorithm. Promising computational results have suggested that the simulated annealing method is an efficient approach for solving difficult combinatorial problems (e.g. traveling salesman problem, Bonomi and Lutton [2], quadratic assignment problem, Wilhelm and Ward [40], machine scheduling problem, Matsuo et al. [23,24], graph partitioning and graph coloring, Johnson et al. [17,18], and network design, Friesz et al. [10]).

In this section, we describe how the simulated annealing method is used to solve the VRPTW problem. The following general description applies to any neighborhood scheme $N(S)$.

3.1. SIMULATED ANNEALING PROCEDURE

Step 1. Obtain an initial feasible configuration (solution) S for the VRPTW using the parallel construction heuristic.

Step 2. Set the cooling parameters including the initial temperature T, the cooling ratio r, and the epoch length *Len*.

Step 3. 3.1. For $1 \leq i \leq Len$ do

3.1.1. Pick a random neighbor $S' \in N(S)$.

3.1.2. Let $\Delta = \text{Cost}(S') - \text{Cost}(S)$.

3.1.3. If $\Delta \leq 0$, then set $S = S'$.

3.1.4. If $\Delta > 0$, then set $S = S'$ with probability $e^{-\Delta/T}$.

3.2. Set $T = rT$.

Step 4. Return S.

In general, the choices the designer of a simulated annealing heuristic has to make can be classified into two classes, namely problem specific and generic. The terms that must be specified are as follows.

3.2. PARAMETERS USED IN SIMULATED ANNEALING

Problem specific	*Generic*
1. Configuration	1. Initial temperatures
2. Neighborhood of a configuration	2. Cooling ratio
3. Cost of a configuration	3. Epoch length
4. Initial configuration	4. Frozen system

Simulated annealing can be modeled as an algorithm which, given a neighborhood structure, constantly attempts to transform the current configuration into one of its neighbors. Let us now describe the various problem-specific terms that appear for the VRPTW. The *configuration* is given by the number of vehicle routes V and an assignment of the customers to the candidate routes without violation of time window constraints, and can be represented by $S = \{R_1,\ldots,R_p,\ldots,R_V\}$, where R_p is the set of customers serviced by route p. The *neighborhood* of a configuration consists of those configurations that result from perturbing the initial one by the shift of customers from one route to another or the interchange of the location of two customers. In this paper, we consider two neighborhood schemes. The first is a modified version of the k-node interchange mechanism of Christofides and Beasley [5] and the second is based on the λ-interchange mechanism of Osman [27].

3.3. NEIGHBORHOOD $N1$

To describe the $N1$ neighborhood, consider a customer i on route p and the L ($L = 0$, 1, or 2) customers that follow customer i on route p. Let the set $M1$ be defined by customer i and the L customers that follow i on route p. Define the set $M2$ as the set of two customers not on route p that are close to the customers in set $M1$. More precisely, let the points α and β be the two customers on route p that immediately precede and succeed the set $M1$, respectively. The set $M2$ is generated by finding the two customers not on route p whose insertion cost using the Euclidean distance metric between α and β is minimal, i.e. minimize$_{j \notin R_p} d(\alpha, j) + d(j, \beta)$. The sets $M1$ and $M2$ define the size of the neighborhood in the search process. For $L = 1$, the sets $M1$ and $M2$ each consist of two customers; $M1 \cup M2$ consists of four customers. In this case, the $N1(S)$ neighborhood of a given solution S is the set of all neighboring solutions achievable by deleting the four customers in $M1 \cup M2$ and re-inserting them on all possible routes.

For each customer i, let $U_i = M1 \cup M2$. Let $\mathcal{U}$ denote the family of all subsets U_i, $i = 1, 2,\ldots,n$. The simulated annealing heuristic proceeds at iteration k by selecting the four customers in subset U_i and evaluating possible route insertions. A complete neighborhood search would involve V^4 possible route assignment combinations. This is computationally intractable if the number of routes is large, say 15. However, a preprocessing step is used to determine the best two routes for each point in U_i. The insertion criterion to determine the best two routes per customer is the c_3 criterion in equation (3). The preprocessing step reduces the maximum number of route assignment combinations to $2^4 = 16$ for each subset U_i. Thus, the $N1$ neighborhood search process involves a partial search of a relatively large neighborhood; the partial search attempts to examine only the more promising moves within the neighborhood.

The interchange proceeds by sequentially evaluating the U_i, $i = 1, 2,\ldots,n$. Each possible route assignment is examined for capacity constraints, time window constraints, and route duration constraints. The candidate solution $S' \in N1(S)$ is accepted as the new solution if $\Delta = C(S') - C(S) \leq 0$. If $\Delta > 0$, then S' is accepted with probability $e^{-\Delta/T}$. The heuristic invokes a first-improvement strategy in that it immediately implements any improved solutions found. The criterion for an improved solution, i.e. *the cost of a configuration*, is

$$C(S) = \beta_1 R + \beta_2 T + \beta_3 D; \quad \beta_1 \gg \beta_2 > \beta_3,$$

where

R = total number of vehicles needed,
D = total travel distance,
T = total schedule time of all routes.

The cost of a configuration is the weighted sum of the number of vehicles, the total travel time, and the total travel distance by the vehicles. Note that the number of vehicles required, R, can be different from V, the initial fleet size specification. Here, the total number of vehicles is given the heaviest weight for the multiple-objective function. This is due to the fact that the vehicles, the number of drivers required, and maintenance often incur greater costs than the other factors. Initially, the U_i are calculated in order of the customers on $R_1, R_2,\ldots, R_V$, etc. Subsequently, the U_i are examined in a random order.

3.4. NEIGHBORHOOD $N2$

The second neighborhood scheme is to use Osman's [27] λ-interchange mechanism, which is an ordered search method that examines all possible combinations of pairs of routes for exchange. Suppose that permutation σ is the order of route indices for a given solution $S = \{R_1,\ldots, R_s,\ldots, R_t,\ldots, R_V\}$, then all possible combinations of pairs of routes (R_s, R_t) would be examined without repetition in the order as follows:

$$
\begin{aligned}
&(R_{\sigma(1)}, R_{\sigma(2)}), \ldots, (R_{\sigma(1)}, R_{\sigma(V)}),\\
&(R_{\sigma(2)}, R_{\sigma(3)}), \ldots, (R_{\sigma(2)}, R_{\sigma(V)}),\\
&\quad\vdots\\
&(R_{\sigma(V-2)}, R_{\sigma(V-1)}), (R_{\sigma(V-2)}, R_{\sigma(V)}),\\
&(R_{\sigma(V-1)}, R_{\sigma(V)}).
\end{aligned}
$$

To generate neighboring solutions, we use (0, 1), (1, 0), and (1, 1) operators to represent the shift or exchange process. Here, (0, 1) and (1, 0) denote the shift of one customer from one route (R_s) to another (R_t) and (1, 1) denotes the exchange of customers between the selected pair of routes (R_s, R_t). The customers of the selected pair of routes (R_s, R_t) would be searched completely for possible improvement. A first-improvement strategy is used. Note that the next solution is accepted in the following fashion. Choose a candidate solution in the ordered search $S' \in N2(S)$, where S is the previous solution. Compute the differences in objective values, i.e. $\Delta = C(S') - C(S)$. If $\Delta \leq 0$, then accept S' as the current solution. Otherwise, if $\Delta > 0$, then accept S' with probability $e^{-\Delta/T}$. A total number of $V(V-1)/2$ pairs of routes would be generated and examined in this manner. Note that a similar neighborhood structure was also implemented by Connolly [4] for the quadratic assignment problem.

The *initial configuration* is obtained by a construction of routes and assignment of customers to those routes as discussed above.

3.5. GENERIC PARAMETERS OF SIMULATED ANNEALING

There are four generic parameters which must be specified, namely the initial temperature T_0, the final value of the temperature, the epoch length, and a rule specifying how the temperature is reduced. A choice of these parameters is referred to as the cooling schedule.

The initial temperature T_0 is chosen in such a way that a cost increasing move would be accepted in the first stage of the annealing process with probability P_0, which in our program implementation is denoted as PINIT. The value of PINIT is set at 0.05. Next, a number of moves are performed randomly and the average cost increase $\overline{\Delta C}$ is calculated. Note that at this point we are interested in uphill moves only. Then, the value of T_0 is calculated according to the formula $e^{-\overline{\Delta C}/T} \approx P_0$. The number of moves attempted to calculate the $\overline{\Delta C}$ is a fraction of the neighborhood size of the initial configuration, which is denoted as SAMPLE in our program and its value is 0.05.

The simulated annealing algorithm can be modeled as a process which, given a neighborhood structure, attempts to transform the current configuration into one of its neighbors. This process can be modeled as a Markov chain. For each temperature value, we get one such Markov chain. The length of each Markov chain,

which is referred to as the epoch length *Len* is taken to be equal to a percentage of the total neighborhood size. This percentage is denoted as LENPERCENT in our implementation and its value is 0.95. This epoch length *Len* should be large enough to achieve a good solution. It should not be overly dependent on the temperature, but we have to take into account the fact that at low temperatures the number of moves actually accepted is rather small. Otherwise, we would have unnecessarily long Markov chains in the final stages of the annealing schedule.

The temperature should be decreased in such a way that we shall not have to employ very long Markov chains. We have used a method described by van Laarhoven and Aarts [39], which appears to be the most common in the current literature. This frequently used decrease rule specifies that: $T_k = rT_{k-1}$, $k = 1, 2, \ldots$, where r is a constant less than (but close to) 1, which is termed RATIO in our program. Typical values for r lie between 0.85 and 0.95. Our experimentation verifies the behavior we had expected. Higher cooling ratio values correspond to slower cooling schedules and therefore more reduction steps are required for the algorithm to stop. A value of $\approx 95\%$ produces the best results.

The last detail that must be specified is the stopping criterion, which will specify when the system is frozen. Obviously, the execution of the algorithm can be discontinued if the expected improvement in the configuration cost is rather small. In our version of the simulated annealing algorithm, the execution is terminated if either of two conditions is satisfied. The first condition specifies that the execution is stopped if the optimal configuration found so far remains unchanged for a number of temperature reduction stages. This number is denoted as TIME in our implementation and set at $2n$. The second condition specifies that the execution will halt if the number of accepted moves drops below a certain point. Specifically, our algorithm will stop if the number of accepted moves is less than a fraction of the total number of iterations. This fraction is called MINPERCENT. We observed that as the value of this fraction becomes smaller, the quality of the solution dramatically increases and a greater number of steps are executed for the system to reach lower temperatures. The solutions we report are for a fraction value of 1%.

The above parameter values were obtained through fine tuning. In general, the performance of the SA algorithms is sensitive to the above cooling schedule. Our computational results show that the above parameter setting constitutes an excellent cooling schedule.

4. A tabu list enhanced hybrid simulated annealing algorithm

The simulated annealing algorithm is considered as a memoryless heuristic in the sense that the past search information is not retained. However, this information might be valuable in helping to find a good solution (see Glover [11 – 14], Glover and Laguna [15], Glover et al. [16], Osman [27], Laguna and Glover [22], and Taillard [35]). The hybrid simulated annealing algorithm incorporates a tabu list

which keeps track of a set of moves that we would want to exclude for the time being. The purpose of this tabu list is to avoid cycling in the algorithm. For example, if the search hits a local minimum, the best move in the next iteration will result in an increase in cost. If the old solution is not excluded from the search, there is a good probability that the algorithm will return to the previous selection since it reduces the cost. Thus, by making the move to the previous selection a tabu move, we avoid cycling the algorithm around the local minimum. To enhance the search capability, we let the tabu list size vary dynamically between a lower limit of $\lceil n/15 \rceil$ and an upper limit of $\lceil n/5 \rceil$. The tabu status of a solution can be overridden with the use of certain aspiration criteria. Our aspiration criterion is the best cost found so far. The tabu list and the aspiration criteria help to avoid becoming trapped at a local optimal solution.

The tabu-list-enhanced simulated annealing algorithm uses Osman's [27] λ-interchange neigborhood scheme. Let us define TABU(i, r), $i = 1,\ldots,n$, $r = 1,\ldots,R$, as the tabu list, where n = total number of customers, R = number of routes. TABU(i, r) records the tabu status of customer i in route r, after being moved to another route, i.e., if the TABU(i, r) = m, then customer i cannot return to route i in the next m iterations. At each iteration, m is reduced by 1. For example, in the case of operators (0, 1) or (1, 0), the TABU list is shown as follows:

	Route 1	2	3	...	R
Customer 1		m			
2					
⋮				...	
n					

Immediately after customer 1 in route 2 moves to another route, TABU(1, 2) = m, where m is the tabu list size. That is, customer 1 cannot be allowed to return to route 2 for the next m iterations. For the case of operator (1, 1), consider the exchange of customer 1 in route 3 and customer 7 in route 2. For this exchange, TABU(1, 3) = TABU(7, 2) = m. That is, customer 1 is not allowed to return to route 3 and customer 7 is not allowed to return to route 2 in the next m operations.

Hence, for operators (0, 1) or (1, 0), if i switches from r_1 to r_2 and if TABU(i, r_2) > 0, then the move is tabu, otherwise it is not tabu. For operator (1, 1), if customer i in route r_1 is exchanged with customer j in route r_2, and if TABU(i, r) > 0 or TABU(j, r_1) > 0, then the move is tabu, otherwise it is not.

Let us use the (1, 1) operator for illustration. Define $\Delta C(S(i,j)) = C(S'(i,j)) - C(S(i,j))$ as the difference in the objective function value after the exchange of customers i and j in different routes. The exchange of i and j is considered as a move and it can be classified as a tabu or a non-tabu move. Then this information can be used for the decision to accept or reject the current candidate solution as the next solution depending upon the following conditions. If customer i in route r_1 and j in route r_2 are not in the tabu list and if $\Delta C(S(i,j)) \leq 0$, then the candidate solution would be accepted as the next solution with probability 1. If customers i in route r_1 and j in route r_2 are not in the tabu list and if $\Delta C(S(i,j)) > 0$, then the candidate solution would be accepted with probability $\exp(-\Delta C(S(i,j))/T)$. If customers i in route r_1 and j in route r_2 are in the tabu list, and $\Delta C(S(i,j)) \leq 0$ and Curcost < Mincost, then the tabu status would be overridden by the aspiration criterion. Here, our aspiration criterion is the objective function value of the best solution found so far. That is, if the current solution cost is less than the minimum cost found so far, we will disregard the tabu status and accept the current solution. If customers i in route r_1 and j in route r_2 are in the tabu list, but $\Delta C(S(i,j)) \leq 0$ and Curcost $\geq$ Mincost, then the current solution would be accepted with probability 1. Note that this is due to simulated annealing. Combining the above conditions, i.e., when $\Delta C(S(i,j)) \leq 0$, the current solution is accepted with probability 1. If (i, r_1) and (j, r_2) are both tabu and $\Delta C(S(i,j)) > 0$, then the current solution is accepted with a further reduced probability equal to $\exp(-\Delta C(S(i,j))/T)\,[(1-k1)(1-k2)]$, where

$$k1 = \begin{cases} k_1 \in [0.2, 0.5] & \text{if } (i, r_1) \text{ is tabu,} \\ 0 & \text{otherwise,} \end{cases}$$

$$k2 = \begin{cases} k_2 \in [0.2, 0.5] & \text{if } (j, r_2) \text{ is tabu,} \\ 0 & \text{otherwise;} \end{cases}$$

thus, $k1$ and $k2$ are between 0.2 and 0.5 if the associated moves (i, r_1) and (j, r_2) are tabu. Usually, fine tuning is required to obtain a better value. From our experimentation, the value of 0.3 seems to generate an effective result. Although we accept worse solutions with positive probabilities, the purpose is to drive the search out of local optima and to a new frontier where the potential global minimum might lie. The procedure for the acceptance check can be summarized as in table 1.

The shift operators (0, 1) and (1, 0) are implemented in a similar fashion.

5. Computational results

In order to evaluate the effectiveness of the simulated annealing metaheuristics on the VRPTW, we performed computational tests based on Solomon's standard set of problems from the literature as well as large-scale real-world problems. The

Table 1

Summary of tabu conditions check and acceptance probabilities.

Move is tabu?	Condition	Acceptance
No	$\Delta C(S(i,j)) \leq 0$	Accept with probability 1
No	$\Delta C(S(i,j)) > 0$	Accept with probability $\exp(-\Delta C(S(i,j))/T)$
Yes	$\Delta C(S(i,j)) \leq 0$	Accept with probability 1 (due to simulated annealing and aspiration criterion)
Yes	$\Delta C(S(i,j)) > 0$	Accept with probability $\exp(-\Delta C(S(i,j))/T)\ [(1-k1)\ (1-k2)]$

metaheuristics were coded in FORTRAN, and compiled using Microsoft FORTRAN Powerstation and run on a 486DX2/66 personal computer.

The Solomon test problems consist of 100 customers with Euclidean distance. The 56 test problems are grouped into six problem types. The geographical distribution of customers in problem sets R1 and R2 is randomly generated according to a uniform distribution. Problem sets C1 and C2 have clustered routes based on solutions to a VRP. Sets RC1 and RC2 are semi-clustered with a mix of randomly distributed and clustered customers. Sets R1, C1, and RC1 have a short scheduling horizon and allow fewer customers per route. Problem sets R2, C2, and RC2 have a longer scheduling horizon and allow a larger number of customers per route. A more complete description of the test problems can be found in Solomon [32].

The computational experiments also include four other test problems based on two real-world problems. The first problem set is the 249 customer Roadway data set reported by Baker and Schaffer [1] and was obtained from Bruce Golden. The data represent one day's deliveries for a food distributor located in the mid-Atlantic region of the United States. The travel time function is a piecewise linear function that allows vehicle speed to vary as a function of distance traveled. Two test problems, D249 and E249, are generated by setting the vehicle capacity to 50 and 100 customers, respectively. Two other test problems, D417 and E417, were reported by Russell [30] and pertain to a national fast food restaurant chain. These two problems are based on a regional application in the southeastern United States. The actual application used great circle distances and a complex travel time function. The company's manually derived solution required 56 vehicle routes for problem D417. The two test problems (like the Solomon problems) are based on Euclidean distance for simplicity. Problems D417 and E417 differ in that E417 has a higher proportion of tight time windows. These new test problems are available from the authors.

The test problems provide an opportunity to compare both the solution quality and rate of convergence of the various simulated annealing approaches. The general simulated annealing metaheuristic has been shown to converge to optimality with probability 1, yet is often slow in converging to an optimal solution. A wise choice in the neighborhood structure will generally help to reduce the convergence time.

The speed of the convergence for the $N1$, $N2$, and tabu-enhanced neighborhood structures can be seen for two specific examples in figures 1 and 2. The graphs show the multiple-objective solution values during the final improvement phase after all $n = 100$ customers have been assigned to tours during the construction process. The initial decreases in solution quality can be partly explained by the relatively high initial temperature, which allows the acceptance of a higher percentage of worse solutions.

In figures 1(a), (b), and (c), it is clear that the $N2$ and $N2$-tabu neighborhoods converge faster than the $N1$ neighborhood in terms of iterations required (1200 versus 5500) to achieve the final solution. For problem RC104 in figures 1(a), (b), and (c), the $N1$ neighborhood requires approximately 1150 iterations, whereas the $N2$ and $N2$-tabu neighborhoods require only 600 and 200 iterations, respectively. The ordered search of the $N2$ neighborhood seems to find an improved solution faster than the $N1$ neighborhood, which samples 2 points on a given route plus 2 other points that are close but not on the same route. We hypothesize that the $N2$-tabu approach (with the help of a short-term memory tabu list) sometimes converges faster because the search procedure is able to avoid cycling, which simulated annealing is not always able to do.

Table 2 shows the computational results on each of the six Solomon problem sets. The results reported include the mean number of vehicles required, mean schedule time for all routes, mean total travel distance, and mean CPU time on a 486DX2/66 PC. All performance measures of the simulated annealing heuristics are based on real number calculations; no integer truncation was applied. The mean performance of the $N1$, $N2$, and $N2$-tabu metaheuristics are compared to three other approaches to the VRPTW that have recently been reported.

On the Solomon test problems, it appears that the $N1$ modified neighborhood of Christofides and Beasley [5] yields slightly better results than the $N2$ neighborhood of Osman [27]. A total of 4 fewer vehicle routes are required using the $N1$ neighborhood on all 56 Solomon test problems. The $N2$ neighborhood, however, did yield slightly better results on the clustered C1 problem set and the RC1 set when enhanced by a tabu list.

Table 3 shows the detailed results on four large-scale real-world routing problems. On the large-scale problems, the differences are amplified. The $N1$ neighborhood achieved slightly better results on the 240-node problems, where there are only 4 or 5 vehicle routes. However, on the larger 417-node problems with 55 vehicle routes, the $N2$ neighborhood and λ-interchange mechanism of Osman [27] yields dramatically superior results. We hypothesize that the $N1$ neighborhood is

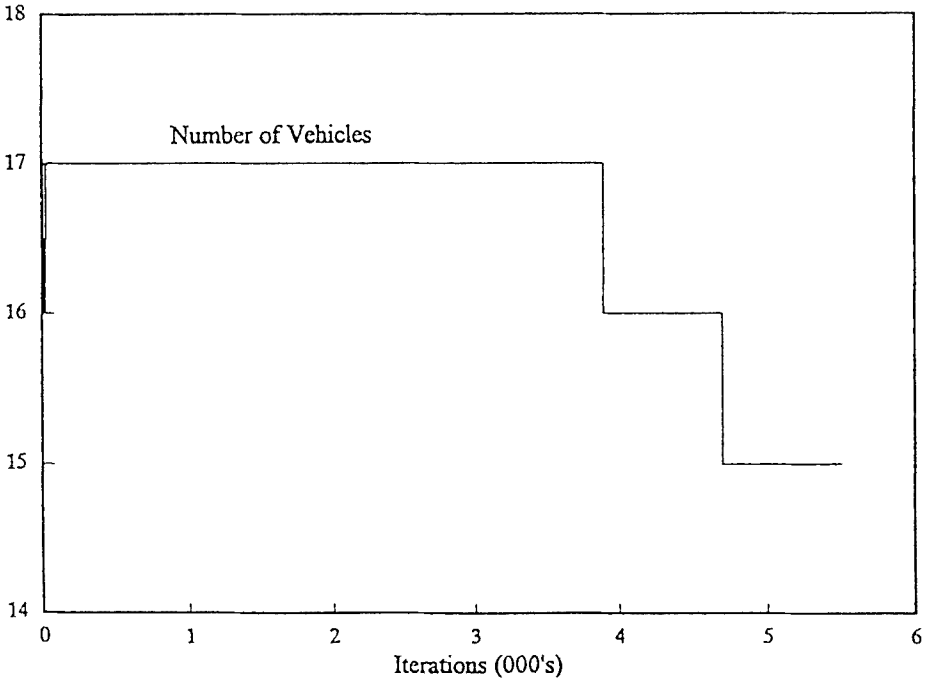
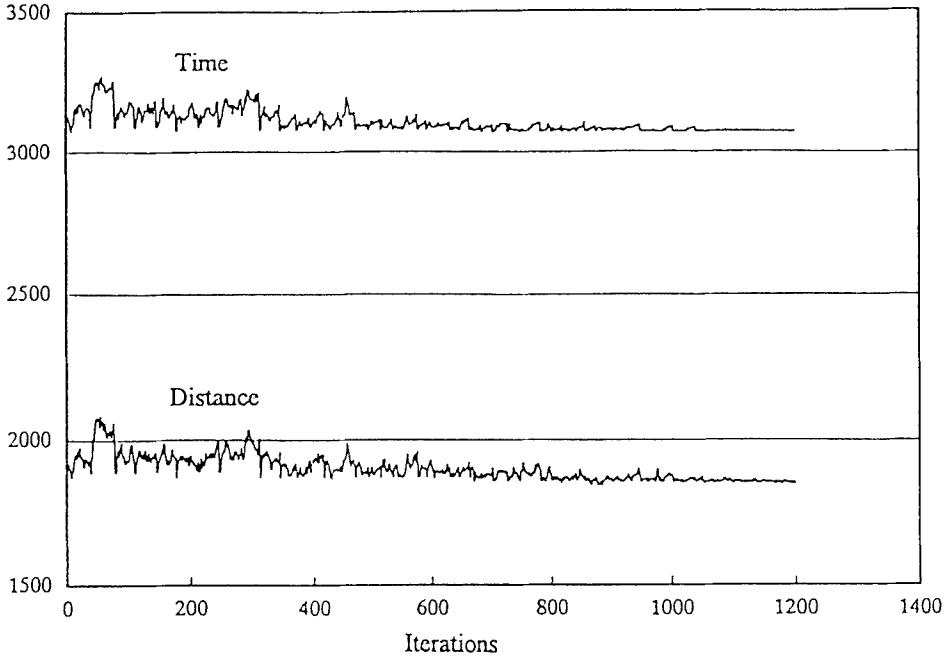


Figure 1(a). Heuristic performance – $N1$ neighborhood, problem RC101.



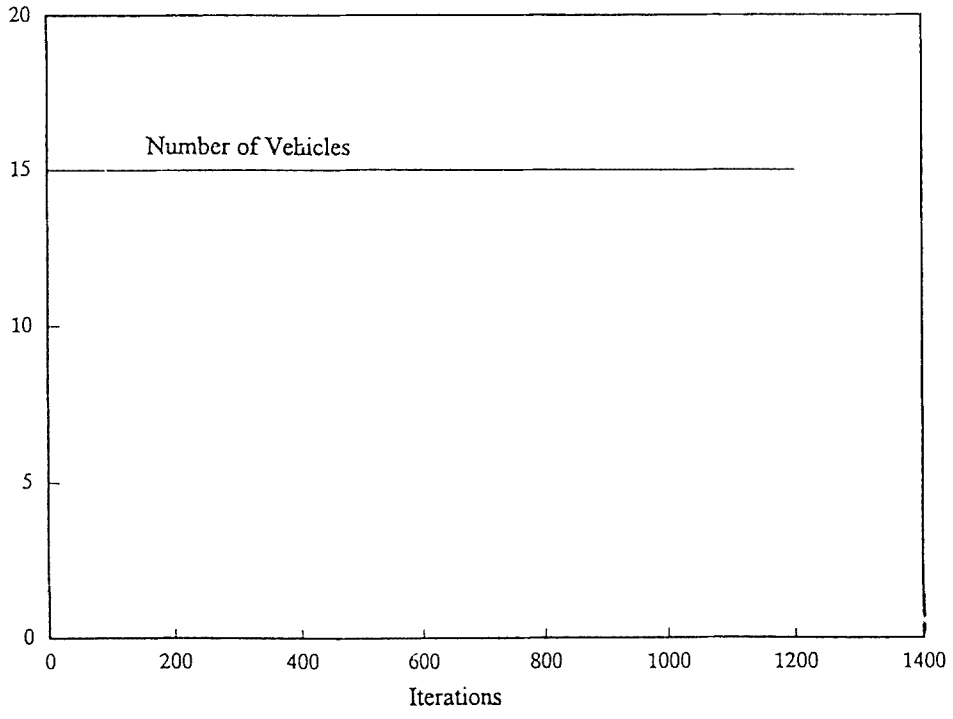


Figure 1(b). Heuristic performance – $N2$ neighborhood, problem RC101.

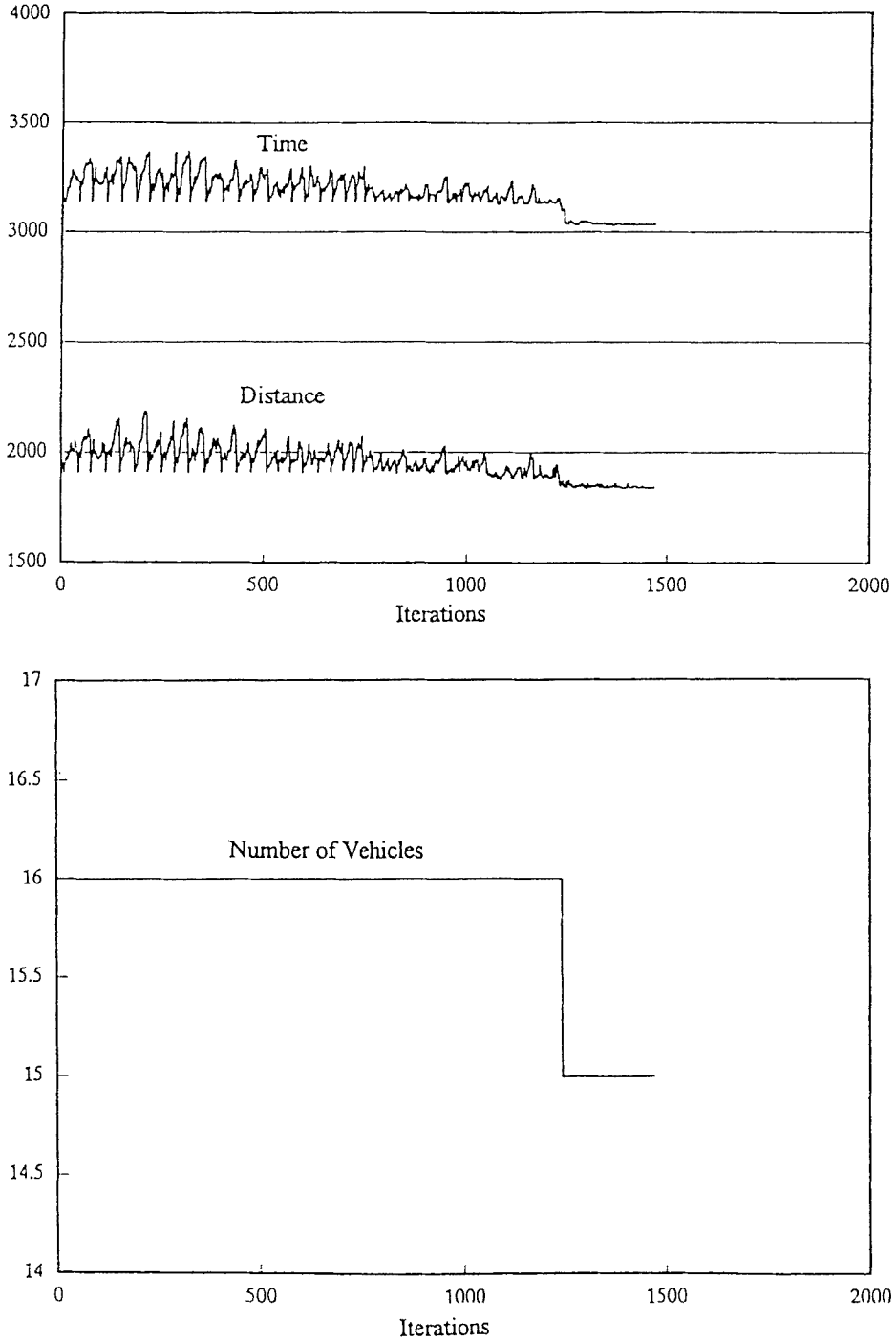
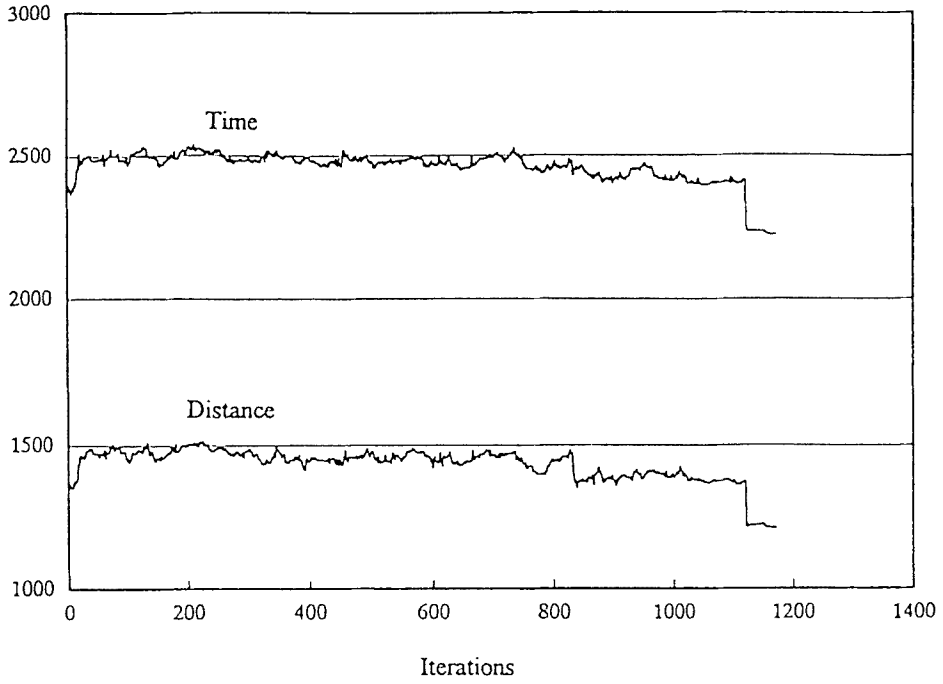


Figure 1(c). Heuristic performance – $N2$-tabu neighborhood, problem RC101.



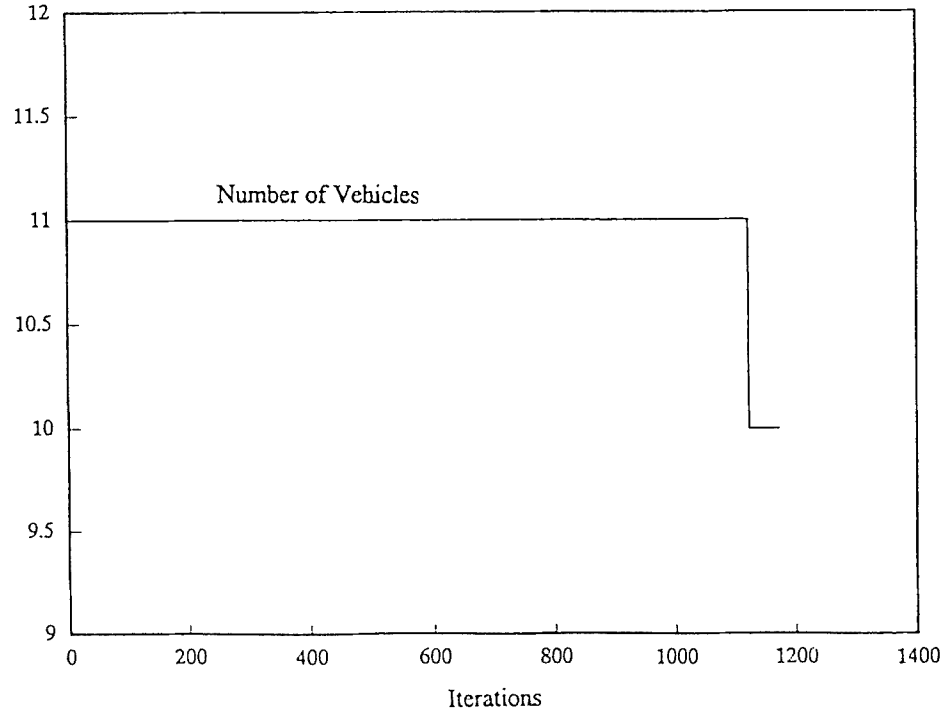


Figure 2(a). Heuristic performance – $N1$ neighborhood, problem RC104.

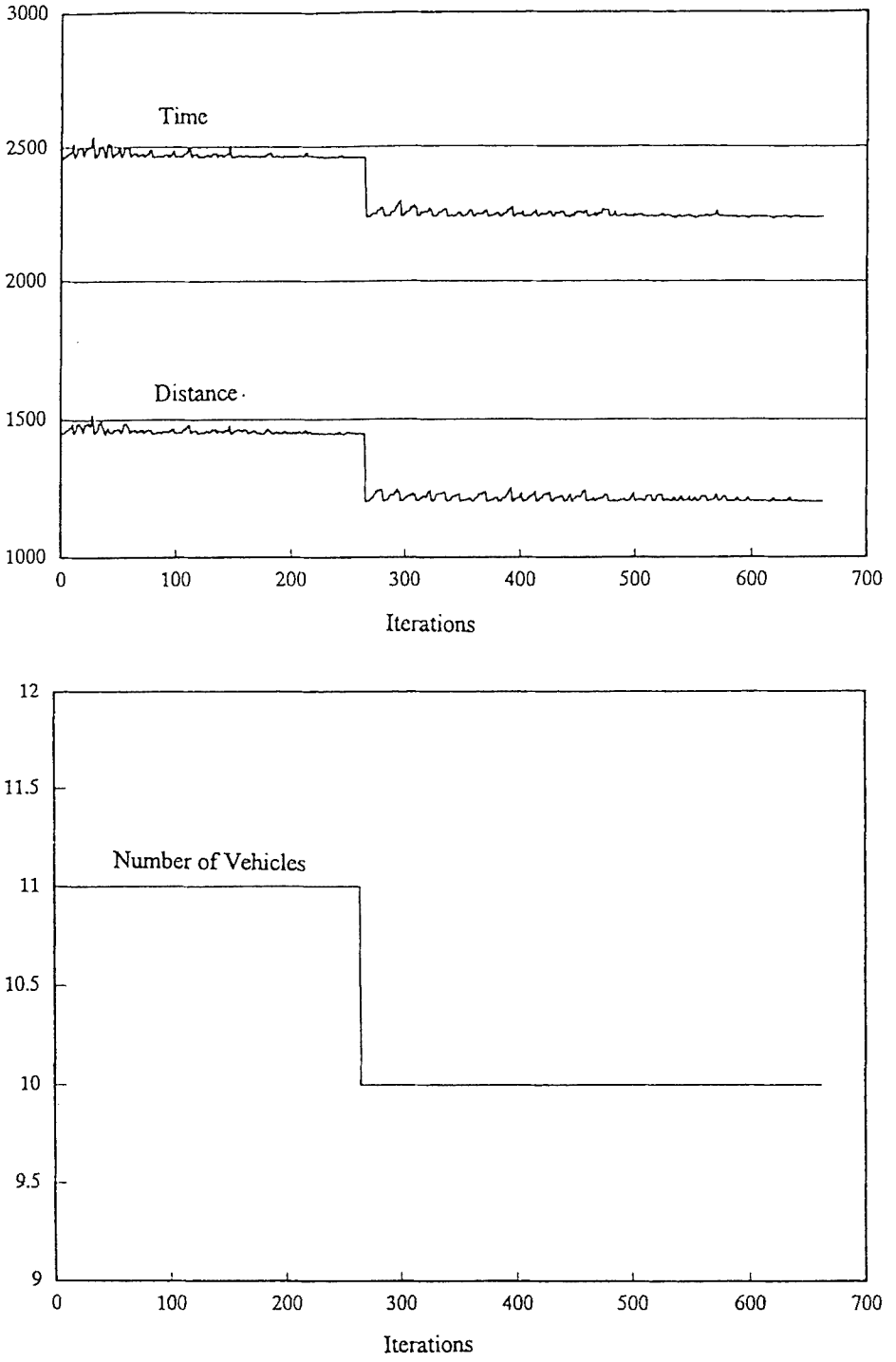


Figure 2(b). Heuristic performance – $N2$ neighborhood, problem RC104.

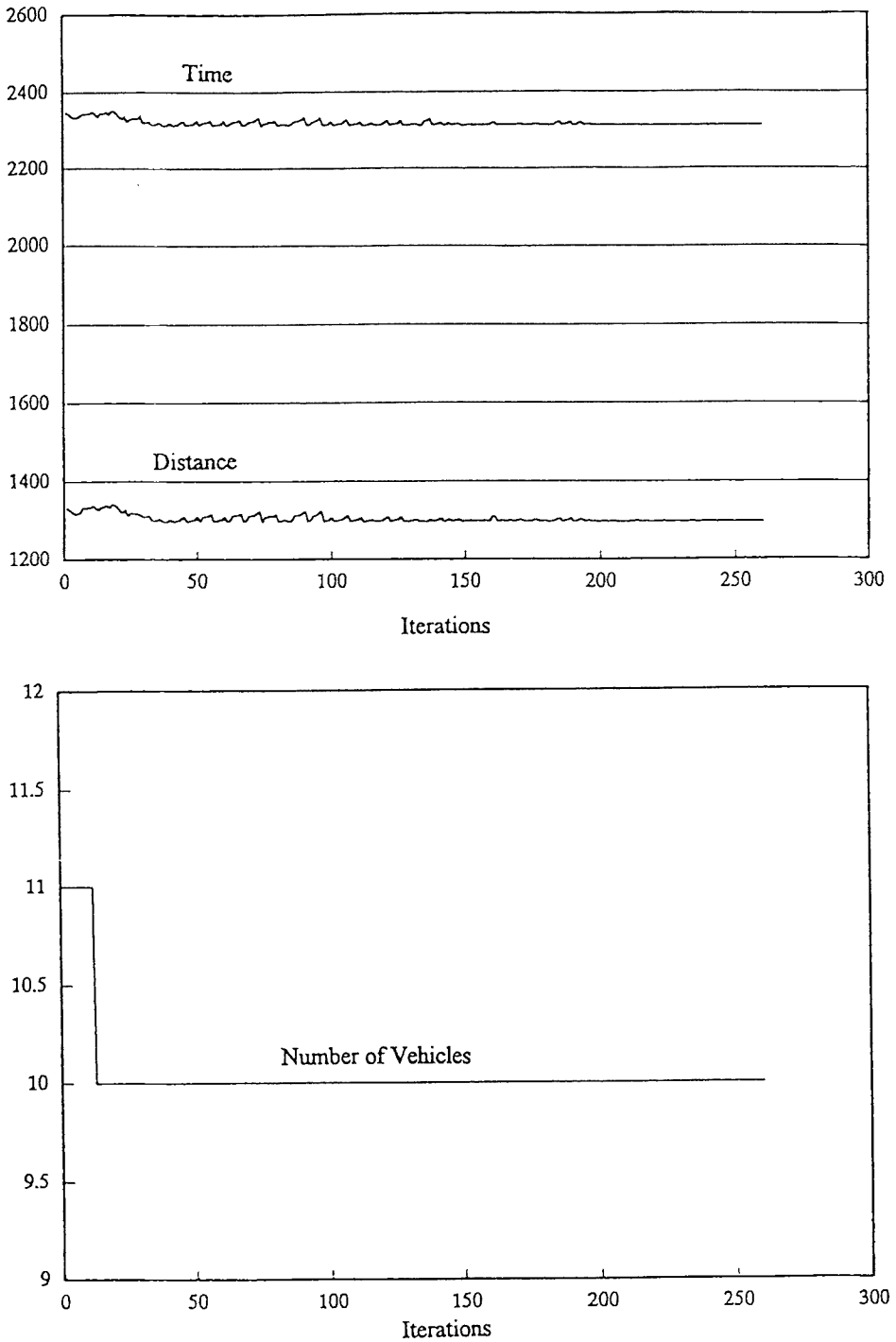


Figure 2(c). Heuristic performance – $N2$-tabu neighborhood, problem RC104.

Table 2

Comparison of results on Solomon test problems. First row: mean number of vehicles required; second row: mean schedule time; third row: mean total travel distance; fourth row: mean CPU time.

Problem set	*N*1	*N*2	*N*2-tabu	Potvin and Rousseau	Thompson and Psaraftis (SP3)	GIDEON
R1	12.5	12.66	12.58	13.3	13.08	12.8
	2444.9	2447.3	2434.10	2696.0	2484	–
	1308.82	1306.3	1300.4	–	1367	1300
	2.86*	2.41	2.42	14.7	1.08	–
R2	2.91	3.0	3.0	3.1	3.09	3.2
	2258.25	2286.1	2290.2	2513.3	2333	–
	1166.42	1147.4	1163.1	–	1299	1125
	3.93	2.40	2.35	31.4	4.33	–
C1	10.0	10.0	10.0	10.7	10.0	10.0
	9929.1	9962.7	9962.3	10610.3	9965	–
	909.8	932.6	930.8	–	939	892
	2.19	2.27	2.24	14.28	0.52	–
C2	3.0	3.0	3.0	3.4	3.0	3.0
	9703.6	9686.1	9695.1	10477.6	9649	–
	684.1	666.3	666.3	–	648	749
	2.46	1.14	1.17	17.3	1.18	–
RC1	12.38	12.5	12.38	13.4	13.0	12.5
	2562.2	2581.1	2558.7	2877.9	2598	–
	1473.9	1481.5	1474.1	–	1534	1474
	2.57	1.95	1.98	14.8	1.02	–
RC2	3.38	3.38	3.38	3.6	3.71	3.4
	2572.7	2617.5	2593.8	2807.4	2706	–
	1401.5	1450.9	1393.7	–	1672	1411
	3.58	2.19	2.04	23.81	2.33	–

*CPU minutes on a 486-66PC.

Table 3

Results on large-scale problems.

	*N*1				*N*2				*N*2-tabu			
Problem	No.	Time	Dist.	CPU	No.	Time	Dist.	CPU	No.	Time	Dist.	CPU
D249	4	3216.16	492.44	8.07	4	3281.87	530.6	2.31	4	3278.80	529.33	2.30
E249	5	3330.26	546.34	6.65	5	3410.55	610.9	8.58	5	3360.94	563.57	9.08
D417	55	50030.70	5711.01	14.5	55	46791.43	4234.99	24.01	55	46871.02	4232.39	31.98
E417	56	55520.78	5749.56	11.4	55	52427.77	4397.49	21.29	55	52592.61	4589.61	22.25

better for problems with fewer than 20–25 routes. The preprocessing procedure which limits the number of potential routes for each point in U_i to 2 is a disadvantage for large problems with as many as 55 routes. For this reason, the tabu enhancement was applied only to the $N2$ neighborhood search process.

From the computational results, it appears that the tabu list enhancement yields modest benefits. On the 56 Solomon test problems, the tabu enhancement obtained better aggregate results on 4 of the 6 problem sets as compared to the unenhanced $N2$ neighborhood search. It was able to eliminate 2 vehicle routes out of the 426 required of the $N2$ approach. The use of a tabu list improved the schedule time and distance traveled on the 249-node problem, but failed to improve either of the 417-node problems.

Table 2 compares the mean performance measures in terms of vehicles, schedule time, distance, and CPU time for the three simulated annealing heuristics and three heuristics previously reported in the literature. The simulated annealing heuristics obtain the best mean solutions for four of the six data sets: these include R1, R2, RC1, and RC2. On the clustered data set C1, the GIDEON approach achieved the same number of vehicles, but slightly smaller average total distance. The cyclic transfer algorithm of Thompson and Psaraftis [38] achieved the same number of vehicles and slightly less total distance on the clustered data set C2. We should also note that the Koskosidis et al. [21] optimization-based heuristic for the soft time window problem achieved the minimum number of routes (10) and the minimum distance of 828.94 on all 8 problems in the C1 data set.

The simulated annealing metaheuristics give higher priority to total schedule time than total distance. In several instances, it is possible to generate shorter travel distances at the expense of total schedule time. This was not attempted in solving for the results of table 2. However, some competing heuristics do not report total schedule time.

A comparative advantage of the simulated annealing heuristics is the minimization of vehicles required, or fleet size. On the 56 Solomon test problems, the $N1$, $N2$, and $N2$-tabu heuristics required 422, 426, and 424 vehicles, respectively. The Potvin and Rousseau parallel construction approach, the cyclic transfer algorithm, and the genetic algorithm GIDEON, required 444, 439, and 430 vehicles, respectively.

It is difficult to compare computational requirements across different computing platforms. However, we can observe that the simulated annealing heuristic required an average of approximately 2.4 minutes to solve each of the R1 problems, Potvin and Rousseau required 14 minutes on an IBM-PC compatible, Thompson and Psaraftis required approximately 1 minute on a 12 MHz IBM PC-AT clone, and the GIDEON system typically required less than 2 minutes on a SOLBOURNE 5/802.

The best performance benchmark for the metaheuristics is a direct comparison to known optimal solutions. Problems R101, R102, C101, C102, C106, C107, and C108 have known optimal solutions, as shown in table 4. (Note that the optimal

Table 4

Simulated annealing compared to optimal solutions.

Optimal solution			Simulated annealing solution	
Problem	No.	Distance	No.	Distance
R101	18	1607	19	1706
R102	17	1434	17	1608
C101	10	827.3	10	827.3
C102	10	827.3	10	1080
C106	10	827.3	10	827.3
C107	10	827.3	10	827.3
C108	10	827.3	10	827.3
Deviation from optimality:			1.17%	7.3%

distances reported by Desrochers et al. [6] are derived by using integer truncation. Thus, the reported distance of 827.3 increases to 828.94 with real number calculations. We used integer truncation only for direct comparison in table 4.) For these 7 test problems, 85 vehicle routes are required. The total travel distance is 7177.5. All three metaheuristics achieved the minimum number of vehicle routes in 6 of these 7 test problems. In problem R101, one more route was required than the optimal number of 18. The total number of vehicle routes required was only 1.17% greater than the optimal for all three simulated annealing metaheuristics. The total travel distance is the third priority goal in the objective function. The three metaheuristics, $N1$, $N2$, and $N2$-tabu required total travel distances of 7711.93, 8003.41, 7974.45, respectively. These are 7.3%, 11.5%, and 11.1% greater than the optimal. Thus, all three metaheuristics achieved near-optimal solutions to problems for which the optimal solution is known.

6. Conclusion

The vehicle routing problem with time windows is a very useful and application-intensive, but notoriously hard problem. This paper presented simulated annealing metaheuristics for the vehicle routing problem with time windows. Three simulated annealing algorithms have been implemented with two different neighborhood structures, an enhancement using a short-term memory tabu list, and improvement procedures invoked during the route construction process. The parameters were fine-tuned to the best levels. Computational experiments suggest that the simulated annealing implementations were able to obtain very good results in reasonable CPU time using a 486DX2/66 personal computer, overcoming the slow convergence of

general simulated annealing algorithms. On four of the six data sets, the simulated annealing metaheuristics generated better mean results than those previously reported in the literature.

The choice of neighborhood structure appears to have a significant effect on algorithm performance. The k-node ($N1$) modified neighborhood of Christofides and Beasley appears to work best on problems with fewer than 25 routes. The λ-interchange mechanism ($N2$) of Osman converges more rapidly in terms of both iterations required and CPU time. It also yielded significantly better results on large-scale problems with a large number (55) of vehicle routes.

Future research will examine other metaheuristics, such as tabu search, in the solution of the VRPTW. Experimental results suggest that metaheuristics are the most appropriate tools to solve problems of such computational complexity, particularly for large-scale problems.

Acknowledgements

The authors are grateful to their graduate assistant, Mr. Xiaofeng Ye, for his dedicated programming effort, and to the referees for their helpful comments.

References

[1] E. Baker and J. Schaffer, Solution improvement heuristics for the vehicle routing and scheduling problem with time window constraints, Amer. J. Math. Manag. Sci. 6(1986)261–300.

[2] E. Bonomi and J.L. Lutton, The asymptotic behavior of quadratic sum assignment problems: A statistical mechanics approach, Euro. J. Oper. Res. 26(1986)295–300.

[3] V. Cerny, Thermodynamic approach to the traveling salesman problem, J. Optim. Theory Appl. 45(1985)41–51.

[4] D.T. Connolly, An improved annealing scheme for the QAP, Euro. J. Oper. Res. 46(1990) 93–100.

[5] N. Christofides and J. Beasley, The period routing problem, Networks 14(1984)237–246.

[6] M. Desrochers, J. Desrosiers and M. Solomon, A new optimization algorithm for the vehicle routing problem with time windows, Oper. Res. 40(1992)342–354.

[7] M. Desrochers, J.K. Lenstra, J.K. Savelsberg and F. Soumis, Vehicle routing with time windows: Optimization and approximation, in: *Vehicle Routing: Methods and Studies*, ed. B.L. Golden and A.A. Assad (North-Holland, Amsterdam, 1988) pp. 65–84.

[8] M. Fisher and R. Jaikumar, A generalized assignment heuristic for vehicle routing, Networks 11(1981)109–124.

[9] M. Fisher, K.O. Jörnsten and O.B.G. Madsen, Vehicle routing with time windows, Research Report 4C/1991, IMSOR, Technical University of Denmark, DK-2800 Lyngby, Denmark (1991).

[10] T. Friesz, H.-J. Cho, N. Mehta, R. Tobin and G. Anandalingam, A simulated annealing approach to the network design problem with variational inequality constraints, Transp. Sci. 26(1992) 18–26.

[11] F. Glover, Future paths for integer programming and links to artificial intelligence, Comp. Oper. Res. 5(1986)533–549.

[12] F. Glover, Tabu search, Part 1, ORSA J. Comp. 1(1990)190–206.

[13] F. Glover, Tabu search: A tutorial, Interfaces 20(1990)74–94.

[14] F. Glover, Tabu search, Part 2, ORSA J. Comp. 2(1991)4–32.
[15] F. Glover and M.M. Laguna, Tabu search, in: *Modern Heuristic Techniques for Combinatorial Problems*, ed. C.R. Reeves (1993) pp. 70–150.
[16] F. Glover, E. Taillard and D. de Werra, A user's guide to tabu search, Ann. Oper. Res. 40(1993) 3–30.
[17] D.S. Johnson, C.R. Aragon, L.A. McGeoch and C. Schevon, Optimization by simulated annealing: An experimental evaluation: Part I, Graph partitioning, Oper. Res. 37(1989)865–892.
[18] D.S. Johnson, C.R. Aragon, L.A. McGeoch and C. Schevon, Optimization by simulated annealing: An experimental evaluation: Part II, Graph coloring and number partitioning, Oper. Res. 39 (1991)378–406.
[19] S. Kirkpatrick, D.D. Gelatt and M.P. Vecchi, Optimization by simulated annealing, Science 220(1983)671–680.
[20] G. Kontoravdis and J. Bard, Improved heuristics for the vehicle routing problems with time windows, Working Paper, Operations Research Group, The University of Texas at Austin (1992).
[21] Y.A. Koskosidis, W.B. Powell and M.M. Solomon, An optimization-based heuristic for vehicle routing and scheduling with soft time window constraints, Transp. Sci. 26(1992)69–85.
[22] M.M. Laguna and F. Glover, Bandwidth packing: A tabu search approach, Manag. Sci. 39 (1993)492–500.
[23] H. Matsuo, C. Suh and R. Sullivan, A controlled search simulated annealing method for the single machine weighted tardiness problems, Ann. Oper. Res. 21(1989)85–108.
[24] H. Matsuo, C. Suh and R. Sullivan, A controlled search simulated annealing method for the general job shop scheduling problem, Working Paper, Graduate School of Business, University of Texas at Austin (1987).
[25] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller and E. Teller, Equation of state calculation by fast computing machines, J. Chem. Phys. 21(1953)1087–1091.
[26] I. Or, Traveling salesman-type combinatorial problems and their relation to the logistics of blood banking, Ph.D. Thesis, Department of Industrial Engineering and Management Sciences, Northwestern University (1976).
[27] I.H. Osman, Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problems, Ann. Oper. Res. 40(1993)421–452.
[28] J.Y. Potvin and J.M. Rousseau, A parallel route building algorithm for the vehicle routing and scheduling problem with time windows, Euro. J. Oper. Res. 66(1993)331–340.
[29] R. Russell, An effective heuristic for the *M*-tour traveling salesman problem with some side conditions, Oper. Res. 25(1977)517–524.
[30] R. Russell, Hybrid heuristics for the vehicle routing problem with time windows, Transp. Sci. 29(1995)156–166.
[31] M. Savelsbergh, Local search in routing problems with time windows, Ann. Oper. Res. 4(1985) 285–305.
[32] M. Solomon, Algorithms for the vehicle routing and cheduling problem with time window constraints, Oper. Res. 35(1987)254–265.
[33] M. Solomon and J. Desrosiers, Time window constrained routing and scheduling problems, Transp. Sci. 22(1988)1–13.
[34] 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*, ed. B.L. Golden and A.A. Assad (North-Holland, Amsterdam, 1985) pp. 85–105.
[35] E. Taillard, Robust tabu search for the quadratic assignment problem, Parallel Comp. 17(1991) 443–455.
[36] S.R. Thangiah, K.E. Nygard and P.L. Juell, GIDEON: A genetic algorithm system for vehicle routing problems with time windows, *Proc. 7th IEEE Conf. on Artificial Intelligence Applications*, Miami, FL (1991).

[37] S.R. Thangiah, I.H. Osman and T. Sun, Hybrid genetic algorithms, simulated annealing and tabu search methods for the vehicle routing problem with time windows (1994).
[38] P.M. Thompson and H. Psaraftis, Cyclic transfer algorithms for multi-vehicle routing and scheduling problems, Oper. Res. 41(1993)935–946.
[39] P.J.M. van Laarhoven and E.H. Aarts, *Simulated Annealing: Theory and Applications* (Reidel, Dordrecht, 1987).
[40] M.R. Wilhelm and T.L. Ward, Solving quadratic assignment problems by simulated annealing, IEEE Trans. 19(1987)107–119.