Evolving Schedule Graphs for the Vehicle Routing Problem with Time Windows

H. Timucin Ozdemir
Dept. of Electrical Engg. and Computer Science
Syracuse University
Syracuse, NY 13244-4100, U.S.A.
htozdemi@ecs.syr.edu

Chilukuri K. Mohan
Dept. of Electrical Engg. and Computer Science
2-177 CST, Syracuse University
Syracuse, NY 13244-4100, U.S.A.
ckmohan@syr.edu

Abstract- The Vehicle Routing Problem with Time Windows is a very important problem in the transportation industry since it occurs frequently in everyday practice, e.g., in scheduling bank deliveries. Many heuristic algorithms have been proposed for this NP-hard problem. This paper reports the successful application of *GrEVeRT*, an evolutionary algorithm based on a directed acyclic graph model. On well-known benchmark instances of VRPTW, we obtain better results than those reported by other researchers using genetic algorithms.

1 Introduction

The *Vehicle Routing Problem with Time Windows(VRPTW)* is a constrained version of the Vehicle Routing Problem (VRP) wherein each customer must be served within a specified time window. Instances of VRPTW occur frequently in several distribution systems, e.g., in scheduling bank deliveries and postal deliveries. Even slight improvements in solution quality result in large cost savings. However, the VRPTW problem is NP-hard [SD88], and finding a solution with fixed fleet size is an NP-complete problem [S87]. Therefore, current studies of this problem attempt to apply heuristic techniques to obtain suboptimal solutions.

Following the terminology used in other research papers addressing VRP, the term *route* is henceforth used to refer to a cycle of customer locations that may be visited by a vehicle. It is assumed that each vehicle must return to the depot where it starts, and its complete schedule may include one or more such routes. Current VRPTW heuristics can be categorized as follows:

- *Sequential route construction* algorithms build the route for each vehicle, one after another, using decision functions for the selection of
  (a) the first customer in the new route;
  (b) the next customer to insert into the route; and
  (c) the place to insert the next customer.
- *Parallel route construction* algorithms build the routes of all vehicles in parallel, using a precomputed estimate of the number of vehicles, and the decision functions mentioned above.
- *Cluster first-route second* approaches attempt to cluster customers using location and time values, and then build a route for each cluster using some heuristic technique. Repair operations are often needed, since the clusters obtained may not yield feasible routes.

Among early well-known results for VRPTW, Solomon [S87] proposed a set of sequential route construction heuristics and a cluster first-route second algorithm based on the Gilett & Miller Sweep heuristic [CMT79] for VRP.

Thangiah [T95] proposed the Push-Forward Insertion Heuristic (PFIH), a sequential route construction heuristic that selects a customer far from the depot, with an early deadline, and near the last customer of the previously constructed route (if there is one). PFIH uses the *insertion cost* function, a linear combination of the traveled distance and time values, to determine the insertion position and the candidate customer.

Two attempts to apply evolutionary algorithms to VRPTW problems have been reported recently. Thangiah [T95] applied a Genetic Algorithm (GA) to cluster customers, and then used PFIH to build a route for each cluster. Berger [BSB98] defined specialized genetic operators to build solutions. In this paper, we show that better results can be obtained by using GrEVeRT, a new evolutionary algorithm to solve VRPTW, starting from a constraint handling perspective. The main feature of our approach is the use of a directed acyclic graph representation called the *schedule graph* to model the problem.

Section 2 describes our evolutionary algorithm in detail. Section 3 presents the results obtained on the benchmark problems defined by Solomon [S87]. In Section 4, we summarize our findings.

2 GrEVeRT

GrEVeRT is a **G**raph-based **E**volutionary algorithm for the **Ve**hicle **R**outing Problem with **T**ime Windows. GrEVeRT is a steady state algorithm that uses a schedule graph representation to handle some problem constraints, described below in greater detail.

0-7803-6375-2/00/$10.00 ©2000 IEEE.

2.1 Problem Constraints

Evolutionary Algorithms have been successfully applied to many NP-hard problems. When the problem has hard constraints, some candidate solutions are infeasible. This problem is generally addressed in three ways: (a) avoiding infeasible solutions during the search process; (b) penalizing infeasible solutions; or (c) extracting feasible solutions after carrying out the search using populations that may contain infeasible candidate solutions. Repairing infeasible candidate solutions may incur significant computational expense, but omitting them from the search process may leave the search space disconnected, preventing satisfactory optima from being reached. Computationally efficient search can be carried out by choosing a representation that implicitly excludes infeasible candidate solutions, without hindering the search process from visiting different parts of the search space.

Table 1: Notation

N	=	number of customers
D_{max}	=	maximum travel distance for a route
Q_{max}	=	maximum capacity of a vehicle
R_{max}	=	maximum travel time for a route
c_{ij}	=	cost of traveling from customer i to customer j
d_{ij}	=	Euclidean distance from customer i to customer j
t_{ij}	=	travel time between customer i and j
q_i	=	maximum demand of customer i
s_i	=	service time of customer i
τ_i^{beg}	=	earliest arrival time to customer i
τ_i^{fin}	=	latest arrival time to customer i
c_j	=	j^{th} customer
c_0	=	depot
v_j	=	j^{th} vertex (customer) in the graph
w_i	=	waiting time before servicing customer i
w_{v_{i-1},v_i}	=	the waiting time between consecutive customers represented by v_{i-1} and v_i on the route
a_i	=	arrival time to customer i
a_{v_i}	=	arrival time to customer represented by node v_i in the graph
λ_{v_i}	=	latest time to arrive at customer represented by node v_i in the graph
K	=	number of vehicles
V_k	=	route k
R_k	=	total travel time for the route k
D_k	=	total traveled distance for the route k
q_{ik}	=	total demand served by vehicle k until (and including) customer i
r_{ik}	=	total travel time of vehicle k until (and including) customer i

The constraints in most transportation problems are of two kinds:

1. *Static constraints* are always enforced, irrespective of the composition of the current solution. For example, time conflicts may prevent one customer from being served after another customer. A time-conflict occurs between customer a and customer b when it is not possible to serve customer b in its time window after customer a.

2. *Dynamic constraints* are enforced based on the composition of the current solution. For example, whether one customer can be served after another may depend on the total demand.

Our algorithm uses a directed graph representation that embeds some static constraints. This is done for VRPTW by using a compatibility relationship between customers, ensuring the absence of time-conflict when one customer is served after another.

$vehicle_1$	:	$v_0, v_1, v_3, v_5, v_{10}, v_{12}, v_0, v_7, v_9, v_{11}, v_0$
$vehicle_2$	:	$v_0, v_{20}, v_{23}, v_{33}, v_{36}, v_{40}, v_0$
...		
$vehicle_k$	:	$v_0, v_4, v_8, v_0, v_{21}, v_{22}, v_0, v_3, v_{31}, v_0$

Table 2: Chromosome representation. Each line represents the schedule of one vehicle and each sequence demarcated with v_0 represents a route

2.2 Representation

Each individual in the population represents a set of vehicle schedules. Each schedule contains a set of routes, respecting the capacity and time window constraints. Each route is a directed path on the graph starting from a depot (v_0) and returning to the same depot (see Table 2).

In the schedule graph, each customer is represented as a node v_i, and an edge e_{ij} exists between nodes v_i and v_j when

- $\tau_i^{\mathsf{beg}} + s_i + t_{ij} \leq \tau_j^{\mathsf{fin}}$, or
- $\tau_i^{\mathsf{fin}} + s_i + t_{ij} \leq \tau_j^{\mathsf{fin}}$.

GrEVeRT finds a set of distinct paths on the DAG where each path starts and ends at the same depot (v_0), the total demand on each route does not exceed the vehicle capacity (Q_{max}), and each edge respects the time window requirements based on the arrival time. Note that an edge (defined based on the first condition) from v_i to v_j may not be feasible if $\tau_i^{\mathsf{beg}} + s_i + t_{ij} < \tau_j^{\mathsf{fin}} < \tau_i^{\mathsf{fin}} + s_i + t_{ij}$ since the time of arrival (a_i) at i^{th} customer is a decision variable and $\tau_i^{\mathsf{beg}} \leq a_i \leq \tau_i^{\mathsf{fin}}$.

If $R = \{v_0, v_1, .., v_m\}$ is a feasible route for vehicle k, with $v_0 = v_m$, then the arrival time (a_{v_i}) to customer v_i is:

$$a_{v_i} = \begin{cases} \tau_{v_i}^{\mathsf{beg}} & \text{if } i = 0 \\ \max(a_{v_{i-1}} + s_{v_{i-1}} + t_{v_{i-1},v_i}, \tau_{v_i}^{\mathsf{beg}}) & \text{otherwise} \end{cases}$$

Since the vehicle may wait until the earliest-service time ($\tau_{v_i}^{\mathsf{beg}}$),the waiting time ($w_{v_{i-1},v_i}$) between consecutive customers v_{i-1} and v_i on the route is 0 if $a_{v_{i-1}} + s_{v_{i-1}} + t_{v_{i-1},v_i} \geq \tau_{v_i}^{\mathsf{beg}}$, and $\tau_{v_i}^{\mathsf{beg}} - (a_{v_{i-1}} + s_{v_{i-1}} + t_{v_{i-1},v_i})$ otherwise.

The latest time (λ_{v_i}) to arrive at customer v_i is $\tau_{v_i}^{\mathsf{fin}}$ if v_i is the last node on the route, and $\min(\tau_{v_i}^{\mathsf{fin}}, \lambda_{v_{i+1}} - (t_{v_i,v_{i+1}} + s_{v_i}))$ otherwise.

A customer u can be inserted between customer i and j on the route if $q_u \leq Q_{max} - \sum_{v_\ell \in R} q_{v_\ell}$, $a_i + s_i + t_{iu} \leq \tau_u^{\mathsf{fin}}$, and

$$\max(a_i + s_i + t_{iu}, \tau_u^{\mathsf{beg}}) + s_u + t_{uj} \leq \lambda_j.$$

Insertion necessitates updating the total served demand, total waiting time, and total traveled distance.

2.3 Initialization

A random initialization procedure is applied, in which customers are successively added to a current route. New routes are introduced if customers remain that cannot be added to existing routes without violating feasibility constraints. Greedy versions of this procedure select customers using heuristics based on criteria such as distance, waiting time, service time value, and demand.

2.4 Fitness Function

We experimented with several fitness functions, based on the number of vehicles used, the total distance traveled and the total waiting time incurred. The results reported in this study used the reciprocal of the sum of the total distance traveled by all the vehicles as fitness function (to be be maximized).

2.5 Point-based operators

One and two point crossover operations are used in many evolutionary algorithms. In this section, modified versions called *1PR* and *2PR* are presented, applied to the directed path representation with VRPTW constraints. Both operations are applied between two distinct routes, say R^i and R^j. Let $R^i = \{v_0^i, v_1^i, ..., v_k^i\}$, $R^j = \{v_0^j, v_1^j, ..., v_l^j\}$ and $Q_p^r = \sum_{h=0}^{r} q_{v_h^p}$.

- Let $(v_n^i \rightarrow v_{n+1}^i)$ and $(v_m^j \rightarrow v_{m+1}^j)$ be two edges under consideration. If there exist edges $(v_n^i \rightarrow v_{m+1}^j)$ and $(v_m^j \rightarrow v_{n+1}^i)$, then 1PR tests for the following conditions:
  1. $a_{v_n^i} + s_{v_n^i} + t_{v_n^i,v_{m+1}^j} \leq \lambda_{v_{m+1}^j}$
  2. $a_{v_m^j} + s_{v_m^j} + t_{v_m^j,v_{n+1}^i} \leq \lambda_{v_{n+1}^i}$
  3. Capacity constraints are not violated by interchanging the edges.

  If all these conditions are satisfied, then two offspring routes are produced,
  - $R_o^i = \{v_0^i, v_1^i, .., v_n^i, v_{m+1}^j, .., v_l^j\}$, and
  - $R_o^j = \{v_0^j, v_1^j, ., v_m^j, v_{n+1}^i, .., v_k^i\}$.

  Checking the constraints guarantees that there is no time conflict nor capacity violation. Times are shifted to the left as much as possible, so that the resulting routes will be tightly scheduled.
- 2PR operation works in a similar manner, except that it requires two sets of edges on routes. Let $(v_n^i \rightarrow v_{n+1}^i)$, $(v_{n'}^i \rightarrow v_{n'+1}^i)$, $(v_m^j \rightarrow v_{m+1}^j)$, and $(v_{m'}^j \rightarrow v_{m'+1}^j)$ be the edges under consideration. 2PR is applied if edges $(v_n^i \rightarrow v_{m+1}^j)$, $(v_{n'}^i \rightarrow v_{m'+1}^j)$, $(v_m^j \rightarrow v_{n+1}^i)$, and $(v_{m'}^j \rightarrow v_{n'+1}^i)$, exist and satisfy the following conditions:
  1. $a_{v_n^i} + s_{v_n^i} + t_{v_n^i,v_{m+1}^j} \leq a_{v_{m+1}^j}$
  2. $a_{v_m^j} + s_{v_m^j} + t_{v_m^j,v_{n+1}^i} \leq a_{v_{n+1}^i}$
  3. $a_{v_{n'}^i} + s_{v_{n'}^i} + t_{v_{n'}^i,v_{m'+1}^j} \leq a_{v_{m'+1}^j}$
  4. $a_{v_{m'}^j} + s_{v_{m'}^j} + t_{v_{m'}^j,v_{n'+1}^i} \leq a_{v_{n'+1}^i}$
  5. Capacity constraints are not violated by interchanging the edges.

  2PR operation allows the exchange of two subpaths between two routes, and implements the same functionality as the CROSS operator for the Tabu Search algorithm proposed in [BGGPT95].

Three variations of these operators are defined by accepting (a) the first possible pair; (b) the first improving pair; or (c) the best improving pair. The scheme (a) applies the operation to the first applicable point. The scheme (b) can be called as the *first improvement* since it applies the operation to the first point where the result is better than the input pairs. The scheme (c) can be called the *best improvement* since it applies the operation at the position where the result improves the most.

1PR-mutation and *2PR-mutation* are versions of this operator where both parents are identical. No patching up or restoration is required since each node (customer) is covered exactly once. But offspring will have to be repaired when 1PR and 2PR are used for recombining routes from different parents. A restoration step follows these operators, deleting duplicate visits to the same customer, and then applying the procedure described in Section 2.3.

	R1	C1	RC1	R2	C2	RC2
Number of Problems	12	9	8	11	8	8
Vehicle Capacity	200	200	200	1000	700	1000
Optimal number of vehicles without TW	10	10		2		2
Service Time	10	90	10		90	10
Maximum Route Time	230	1236	240	1000	3390	960
Window Size	short	short	short	long	long	long
Customer Distribution	random	clustered	semi-clustered	random	clustered	semi-clustered

Table 3: Characteristics of Solomon's VRPTW Benchmark Problems

2.6 Other operators

We have also defined and explored several other operators, described below.

- In *Set Based Crossover* (SeBaX), offspring inherit vehicle duties common to both parents. The rest of the paths from the parents are split up into subpaths that do not contain customers already covered in the offspring. The resulting subpaths are randomly introduced into the offspring, followed by another splitting phase. This process is repeated until a complete and feasible offspring is obtained. If needed, another final patchup step may be executed.
- *Common Edge Preserving Crossover* (CEPX) also places vehicle duties common to both parents in an offspring, but the solution is patched up without reference to the parents.
- *Time Based 1PX* (TB1PX) selects a time point (t_p) and splits the vehicle duties in each parent using this time value. Each offspring contains vehicle duties before t_p from one parent, and vehicle duties after t_p from the other parent. A greedy version of this operator can also be defined.
- *Time Based 2PX* (TB2PX) selects two time points (t_0 and t_1) and splits the vehicle duties in each parent by using these time values. Each offspring contains vehicle duties between t_0 and t_1 from one parent and extremal segments from the other parent.

2.7 Local Search

We experimented with different local search approaches using the 1PR and 2PR operators and node insertion heuristics. A local neighborhood is explored by iterative application of 1PR and 2PR operators to the vehicle schedules of an individual. This process is continued until computational limits are exceeded or no fitness improvement is observed over many iterations of the algorithm.

Another iterative improvement approach is to examine all possible 1PR (or 2PR) moves between the routes and explore the first or best improvement. A simple greedy approach can be used to select non-conflicting moves.

Local search can also be based on a node insertion heuristic. This procedure traces through two routes and attempts to insert a node from the second route to the first.

2.8 Replacement Strategy

After each operation, our algorithm applies roulette-wheel selection between parents and children based on their scaled fitness values, subject to the following restrictions that enforce some elitism:

- A child of higher fitness than both parents is selected to replace the least fitness parent.
- The current best solution cannot be replaced, except by an offspring of higher fitness.

2.9 Diversity

Since we use small population sizes, premature convergence is a potential problem for the evolutionary algorithm. To reinstate population diversity, we invoke a random restart procedure when the difference between the average and the best fitness value drops below some small threshold (ϵ). Most individuals in the population (except the best) are then replaced by new randomly generated individuals.

3 Experiments

Experiments were performed, and results shown in the tables in this section compare our new evolutionary algorithm with competing genetic algorithms for Solomon's VRPTW benchmark problems [T95, BSB98]. We also evaluate results obtained with our algorithm using various recombination operators.

	C1	C2	R1	R2	RC1	RC2
[T95]	892.11	749.12	1300.25	1124.72	1474.12	1411.12
[BSB98]	**834.61**	**594.24**	1261.57	1030.00	1441.35	1284.25
SeBIX	835.46	605.83	**1229.92**	**943.49**	**1428.53**	**1107.70**
SeBaX	857.04	715.61	1246.69	1030.78	1437.27	1185.88
CEPX	914.96	785.19	1281.12	1270.04	1484.88	1334.02
1PX	863.74	672.90	1247.03	1038.15	1453.70	1187.27
2PX	880.01	714.66	1241.09	1047.55	1454.02	1199.42
TB1PX	898.48	699.62	1279.32	1177.51	1492.94	1306.73
TB2PX	866.59	713.63	1268.97	1140.63	1467.76	1280.73

Table 4: Results for different problem classes: The average of total traveled distance, using different algorithms

Solomon [SD88] generated 56 VRPTW test problem from Christofides VRP benchmark problems. Each problem set associates a time windows 25%, 50%, 75% and 100% of the customers (see Table 3). C1 and C2 problems have customers clustered in terms of their location. C2 problems have vehicles with larger capacity. C2 data set is obtained by relocating the customers in C1 to define a structured problem with three clusters. R1 and R2 problems have customers randomly distributed based on their location. R2 data set is a modification of R1 to allow for servicing many customers by one vehicle. RC1 is obtained by using R1 and C1. Problems with short time windows have vehicle with small capacity, and hence cannot serve many customers.

In all experiments, we use a steady state evolutionary algorithm that applies nondeterministic tournament selection with linear fitness scaling to select the parents. The replacement strategy performs deterministic tournament selection between parents and their offspring. When the evolution process stagnates, a restart procedure is initiated to increase population diversity, replacing 90% of the population by new randomly generated chromosomes. We use 1PR-mutation with first improvement, a population size of 20, recombination probability 0.5, mutation probability 0.4, with computations terminated after 3000 generations.

In Tables 5-10, first two columns gives the results reported by [T95, BSB98]. Note that [BSB98] only used the single precision therefore in Table 5 we assumed that the results are the similar. SeBIX refers to the GA with SeBaX settings but it also applies Solomon I1 heuristic in the local search with the node insertion.

In Table 4, [BSB98] presents slightly better results than SeBIX for clustered problems, C1 and C2. SeBIX finds the best results for R1, R2, RC1, and RC2 problem sets. In Table 12, SeBIX provides the most number of best results between these algorithms.

4 Conclusions

We have proposed a new evolutionary algorithm for the VRPTW problems, using a schedule graph representation that embeds static constraints of the problem. Experiments show that our algorithm produces better results than the heuristic approaches. In two instances, our algorithm led to results better than those previously known. In particular, we obtain better results than genetic algorithms previously applied to this problem [BSB98, T95], who have reported better results than Solomon's heuristics.

Comparisons in this paper have focused on other GAs. However, "Tabu Search" algorithms are reported to have produced even better results [XK96, KX99, RT95]. This suggests directions for future work, exploring the incorporation of additional local search operators into the evolutionary algorithm, as well as considering the application of Tabu Search to the results obtained using evolutionary algorithms.

Bibliography

[BGGPT95] Badeau, P. and Gendreau, M. and Guertin, F. and Potvin, J-Y. and Taillard, E. (1995). "A Parallel Tabu Search Heuristic for the Vehicle Routing Problem with Time Windows," Technical Report, CRT-95-84, Montreal (Quebec).

[BSB98] Berger, J. and Salois, M. and Begin, R. (1998). "A Hybrid Genetic Algorithm for the Vehicle Routing Problem with Time Windows," in *Advances in Artificial Intelligence*, Springer LNAI 1218:114-127.

[CMT79] Christofides, N. and Mingozzi, A. and Toth, P. (1979). "The Vehicle Routing Problem," in N. Christofides *et al.* (Eds.), *Combinatorial Optimization*, Wiley, pp.315-338.

[KX99] Kelly, J.P. and Xu, J. (1999). "A Set-Partitioning-Based Heuristic for the Vehicle Routing Problem," *INFORMS Journal on Computing*, 11(2):161-172.

[RT95] Rochat, Y. and Taillard, E.D. (1995). "Probabilistic Diversification and Intensifications in

	[T95]	[BSB98]	SeBIX	SeBaX	CEPX	1PR	2PR	TB1PX	TB2PX
C101	833.00	**828.90**	**828.93**	**828.93**	**828.93**	**828.93**	**828.93**	**828.93**	**828.93**
C102	832.00	837.29	**828.93**	**828.93**	850.11	841.08	930.88	893.77	839.46
C103	873.00	848.40	**828.06**	910.54	941.58	908.08	898.06	979.40	888.35
C104	904.00	**852.40**	888.57	943.21	1011.46	949.32	1038.42	983.85	989.94
C105	874.00	**828.90**	**828.93**	**828.93**	**828.93**	**828.93**	**828.93**	**828.93**	**828.93**
C106	902.00	**828.90**	**828.93**	**828.93**	835.26	**828.93**	830.33	837.14	832.55
C107	926.00	**828.90**	**828.93**	**828.93**	864.17	**828.93**	**828.93**	**828.93**	**828.93**
C108	928.00	**828.90**	**828.93**	829.37	1062.03	831.54	857.82	917.57	847.93
C109	957.00	**828.90**	**828.93**	885.62	1012.15	927.94	877.75	987.78	914.30

Table 5: Results for C1 problems: Total traveled distance, using different algorithms

	[T95]	[BSB98]	SeBIX	SeBaX	CEPX	1PR	2PR	TB1PX	TB2PX
C201	753.00	**591.59**	**591.55**	**591.55**	**591.55**	**591.55**	**591.55**	**591.55**	**591.55**
C202	756.00	**591.59**	**591.55**	649.31	748.79	708.73	696.97	807.76	728.03
C203	855.00	**600.20**	628.83	852.23	802.28	767.44	806.19	749.83	844.63
C204	803.00	**616.59**	650.09	843.03	1135.89	858.03	846.36	932.04	864.34
C205	667.00	**588.90**	591.04	636.86	708.78	591.55	639.61	591.04	608.12
C206	694.00	**588.50**	610.45	727.52	687.72	625.27	700.05	622.82	697.96
C207	730.00	**588.29**	592.51	711.26	858.35	649.08	726.54	654.55	674.86
C208	735.00	**588.29**	590.65	713.09	748.18	591.55	710.03	647.38	699.50

Table 6: Results for C2 problems: Total traveled distance, using different algorithms

	[T95]	[BSB98]	SeBIX	SeBaX	CEPX	1PR	2PR	TB1PX	TB2PX
R101	1700.00	1688.40	**1643.18**	1661.63	1657.36	1644.88	1651.96	1648.81	1669.43
R102	1549.00	1567.09	1528.30	1495.23	1501.73	1494.23	**1478.22**	1533.86	1512.75
R103	1319.00	1310.19	1268.60	**1237.01**	1283.07	1247.01	1241.83	1283.03	1300.26
R104	1090.00	1081.09	**1013.45**	1086.46	1104.16	1109.47	1095.76	1082.93	1102.11
R105	1448.00	1406.69	**1371.52**	1401.19	1418.56	1398.20	1406.53	1415.81	1441.53
R106	1363.00	1367.59	1289.91	1279.43	**1262.20**	1300.90	1266.17	1297.40	1322.84
R107	1187.00	1145.80	**1125.34**	1128.83	1244.57	1153.90	1160.28	1185.50	1163.06
R108	1048.00	**1002.59**	1008.70	1047.96	1093.57	1042.55	1014.05	1087.28	1063.69
R109	1345.00	1230.50	1203.58	**1188.30**	1285.43	1220.31	1250.17	1299.45	1246.52
R110	1234.00	**1133.50**	1145.52	1188.01	1246.23	1144.93	1161.81	1255.77	1210.20
R111	1238.00	1174.30	**1115.08**	1178.63	1184.11	1146.30	1140.96	1165.80	1157.19
R112	1082.00	1031.09	1045.83	1067.65	1092.45	1061.66	**1025.29**	1096.28	1038.06

Table 7: Results for R1 problems: Total traveled distance, using different algorithms

	[T95]	[BSB98]	SeBIX	SeBaX	CEPX	1PR	2PR	TB1PX	TB2PX
R201	1478.00	1448.50	1244.02	**1219.81**	1405.43	1267.59	1296.11	1436.98	1379.58
R202	1279.00	1248.30	**1139.02**	1224.88	1296.02	1164.01	1148.25	1148.03	1159.90
R203	1167.00	1075.59	**927.02**	1025.73	1322.45	984.91	1043.95	1201.46	1064.90
R204	909.00	821.20	**782.18**	918.71	1194.67	973.66	906.67	996.46	867.85
R205	1274.00	1162.59	**984.67**	1065.30	1333.86	1118.25	1130.29	1234.70	1362.48
R206	1098.00	1056.40	**918.71**	1076.93	1305.91	1067.20	1077.85	1116.25	1054.72
R207	1015.00	891.90	**844.66**	1011.52	1258.41	993.34	955.47	1059.14	1018.40
R208	826.00	761.90	**742.99**	825.48	922.67	829.94	890.63	1151.16	979.28
R209	1159.00	996.00	**928.99**	986.12	1435.77	993.73	1086.59	1242.56	1274.92
R210	1269.00	1047.50	**1012.98**	1021.76	1325.81	1042.18	1045.38	1236.13	1247.76
R211	898.00	**820.20**	853.20	962.38	1169.51	984.84	941.86	1129.72	1137.13

Table 8: Results for R2 problems: Total traveled distance, using different algorithms

	[T95]	[BSB98]	SeBIX	SeBaX	CEPX	1PR	2PR	TB1PX	TB2PX
RC101	1767.00	1696.69	1694.92	1671.27	1665.77	1695.25	**1665.51**	1688.68	1682.61
RC102	1569.00	1638.69	1539.14	**1522.41**	1599.93	1566.34	1537.69	1551.71	1540.56
RC103	**1328.00**	1392.09	1355.93	1387.29	1377.83	1389.41	1391.76	1434.29	1415.15
RC104	1263.00	1238.69	**1224.47**	1273.31	1351.18	1274.65	1337.81	1335.18	1283.93
RC105	1612.00	1652.90	**1574.20**	1594.25	1648.97	1640.45	1581.15	1655.64	1579.30
RC106	1608.00	**1416.80**	1475.42	1441.85	1510.15	1470.76	1431.04	1504.81	1517.43
RC107	1396.00	**1303.00**	1358.98	1323.04	1382.07	1333.34	1393.20	1420.86	1426.66
RC108	1250.00	**1191.90**	1205.16	1284.70	1343.09	1259.41	1294.03	1352.35	1296.48

Table 9: Results for RC1 problems: Total traveled distance, using different algorithms

	[T95]	[BSB98]	SeBIX	SeBaX	CEPX	1PR	2PR	TB1PX	TB2PX
RC201	1823.00	1616.09	1406.58	1399.08	1582.60	**1365.18**	1409.67	1465.93	1470.48
RC202	1459.00	1380.69	**1222.08**	1228.31	1388.43	1238.21	1334.06	1454.63	1341.84
RC203	1323.00	1222.30	1101.56	1112.26	1233.06	1098.81	**1080.91**	1271.93	1105.85
RC204	1021.00	903.29	**869.81**	973.63	1085.05	1036.00	1064.95	1169.92	1019.52
RC205	1594.00	1465.19	**1217.68**	1339.13	1432.91	1314.18	1313.02	1465.32	1479.06
RC206	1530.00	1215.30	**1143.18**	1187.10	1381.75	1239.84	1288.27	1324.70	1335.18
RC207	1501.00	1510.40	**1062.68**	1167.48	1315.26	1186.79	1142.90	1257.47	1326.34
RC208	1038.00	960.70	**838.04**	1080.06	1253.13	1019.19	961.56	1043.91	1167.56

Table 10: Results for RC2 problems: Total traveled distance, using different algorithms

	[T95]	[BSB98]	SeBIX	SeBaX	CEPX	1PR	2PR	TB1PX	TB2PX
C1	7.30	**0.38**	0.47	3.05	10.02	3.86	5.79	8.04	4.19
C2	25.99	**0.00**	1.92	20.29	31.84	13.05	20.14	17.51	19.93
R1	7.05	3.62	**1.09**	2.70	5.75	2.75	2.18	5.50	4.48
R2	19.57	9.19	**0.54**	10.56	36.65	11.43	12.33	26.93	22.30
RC1	5.02	2.56	**1.81**	2.64	6.16	3.67	3.99	6.78	4.91
RC2	27.81	16.46	**0.61**	8.79	22.52	8.96	9.70	19.71	17.42
Avrg	15.45	4.94	**1.07**	8.00	17.32	4.51	9.02	14.57	12.2

Table 11: Average of percentage deviation for all problems from the least total traveled distance found by the most successful algorithm

	[T95]	[BSB98]	SeBIX	SeBaX	CEPX	1PR	2PR	TB1PX	TB2PX
C1	0	7	**8**	5	2	4	3	3	3
C2	0	**8**	2	1	1	1	1	1	1
R1	0	2	**5**	2	1	0	2	0	0
R2	0	1	**9**	1	0	0	0	0	0
RC1	1	**3**	2	1	0	0	1	0	0
RC2	0	0	**6**	0	0	1	1	0	0
Total	1	21	**32**	11	4	6	8	4	4

Table 12: Results for all problems: Number of times each algorithm succeeded in finding the schedule with least total traveled distance

Local search for Vehicle Routing," *Journal of Heuristics*, 1:147-167.

[S87] Solomon, M.M. (1987). "Algorithms for the Vehicle Routing and Scheduling Problems with Time Window Constraints," *Operations Research*, 35(2):254-265.

[SD88] Solomon, M. M and Desrosiers, J. (1988). "Time Window Constrained Routing and Scheduling Problems," *Transportation Science*, 22(1):1-13.

[TLG95] Taillard, E. and Laporte, G. and Gendreau, M. (1995). "Vehicle Routing with Multiple use of vehicles," CRT Technical Report, CRT-95-19, Montreal, Quebec.

[T95] Thangiah, S.R. (1995). "Vehicle Routing with Time Windows using Genetic Algorithms," in L. Chambers (Ed.), *Application Handbook of Genetic Algorithms: New Frontiers*, CRC Press, II:253-277.

[XK96] Xu, J. and Kelly, J.P. (1996). "A Network Flow-Based Tabu Search Heuristic for the Vehicle Routing Problem," *Transportation Science*, 30(4):379-392.