Journal of the Operational Research Society (2001) 52, 928–936 ©2001 Operational Research Society Ltd. All rights reserved. 0160-5682/01 $15.00

www.palgrave-journals.com/jors

A unified tabu search heuristic for vehicle routing problems with time windows

J-F Cordeau, G Laporte* and A Mercier

École des Hautes Études Commerciales, Montréal, Québec, Canada

This paper presents a unified tabu search heuristic for the vehicle routing problem with time windows and for two important generalizations: the periodic and the multi-depot vehicle routing problems with time windows. The major benefits of the approach are its speed, simplicity and flexibility. The performance of the heuristic is assessed by comparing it to alternative methods on benchmark instances of the vehicle routing problem with time windows. Computational experiments are also reported on new randomly generated instances for each of the two generalizations.

Keywords: vehicle routing problem; periodic vehicle routing problem; multi-depot vehicle routing problem; time windows; tabu search

Introduction

Vehicle Routing Problems with Time Windows (VRPTWs) arise frequently in a number of contexts such as grocery distribution,[1–4] school bus routing,[5] oil and petroleum delivery,[6,7] armoured car routing,[8] repairmen scheduling,[9] etc. Applications often involve additional constraints or characteristics that may complicate the solution process. This study addresses the VRPTW and two important generalizations of this problem: the *Periodic Vehicle Routing Problem with Time Windows* (PVRPTW) and the *Multi-Depot Vehicle Routing Problem with Time Windows* (MDVRPTW).

The VRPTW is defined on a complete graph $G=(V,A)$, where $V=\{v_0, v_1, \dots, v_n\}$ is the vertex set and $A=\{(v_i, v_j) : v_i, v_j \in V, i \neq j\}$ is the arc set. Vertex v_0 represents a depot at which is based a fleet of m vehicles, and the remaining vertices of V represent customers to be serviced. With each vertex $v_i \in V$ are associated a nonnegative load q_i (with $q_0=0$), a nonnegative service duration d_i (with $d_0=0$), and a time window $[e_i, l_i]$, where e_i and l_i are nonnegative integers. Each arc (v_i, v_j) has an associated nonnegative cost or travel time c_{ij}. The VRPTW consists of designing m vehicle routes on G such that: (i) every route starts and ends at the depot; (ii) every customer belongs to exactly one route; (iii) the total load and duration of route k do not exceed Q_k and D_k, respectively; (iv) the service at customer i begins in the interval $[e_i, l_i]$, and every vehicle leaves the depot and returns to the depot in the interval $[e_0, l_0]$; and (v) the total travel time of all vehicles is minimized. In recent years, several approximate and exact algorithms have been proposed for the VRPTW. Highly efficient metaheuristics have been developed, among others, by Chiang and Russell,[10] Homberger and Gehring,[11] Potvin and Bengio,[12] Potvin *et al*,[13] Rochat and Taillard[14] and Taillard *et al.*[15] An exact algorithm using branching and cutting on solutions obtained through Dantzig–Wolfe decomposition was also recently introduced by Kohl *et al.*[16] For a general overview of time constrained vehicle routing and scheduling problems, see Desaulniers *et al.*[17]

In the PVRPTW, a planning horizon of t days is considered, and each customer i specifies a service frequency f_i as well as a set R_i of allowable combinations of visit days. For example, if $t=5, f_i=2$ and $R_i=\{\{1,3\},\{2,4\},\{3,5\}\}$, then customer i must be visited twice, and these visits should take place on days 1 and 3, on days 2 and 4, or on days 3 and 5. The problem consists of simultaneously selecting an allowable combination of visit days for each customer and establishing vehicle routes for each day of the planning horizon according to the rules of the VRPTW. In the MDVRPTW, several depots with different locations are considered instead of only one. The problem is to simultaneously assign each customer to a depot and to construct routes for each depot according to the VRPTW rules.

By removing time window constraints from the PVRPTW and MDVRPTW, one obtains the well known PVRP and MDVRP for which several approximate algorithms have been proposed. The PVRP was recently studied by Russell and Gribbin,[18] Gaudioso and Paletta,[19] Chao *et al*[20] and Cordeau *et al*,[21] while competitive heuristics for the MDVRP were proposed by Chao *et al*,[22] Renaud *et al*[23] and Cordeau *et al.*[21] By using a general formulation of the PVRP, Cordeau *et al*[21] showed that the MDVRP can be viewed as a special case of the former problem by associating depots with days of the planning period and

**Correspondence: G Laporte, Centre de recherche sur les transports, Pavillon André-Aisenstadt, suite 3520, 2920 Chemin des services, Campus de l'Universite de Montreal, Montreal, Québec, Canada H3T 1J4.*
E-mail: gilbert@crt.umontreal.ca

setting $f_i = 1$ and $R_i = \{\{1\}, \ldots, \{t\}\}$ for $i = 1, \ldots, n$. The authors also proposed a common and highly efficient tabu search heuristic for addressing the PVRP and the MDVRP. This heuristic was later applied to a linen distribution problem in a large hospital.[24]

While both the PVRP and the MDVRP have been studied by several authors, we are not aware of any work addressing the generalization of these problems to include time window constraints. Our primary aim is to provide, for the first time, an algorithm for the PVRPTW and the MDVRPTW. Both problems generalize the VRPTW and are therefore NP-hard.[25] Since only relatively small instances of the VRPTW can be solved optimally within short computing times,[16] we believe that it is appropriate to tackle these problems through a heuristic approach. We will in fact describe a unified tabu search capable of solving the VRPTW, the PVRPTW and the MDVRPTW. The algorithm is an adaptation of the heuristic previously developed by Cordeau *et al*[21] for the PVRP and the MDVRP. What makes this study interesting is that the same algorithmic framework can be used without any major structural or parametric modification for a variety of routing problems, thus demonstrating the flexibility and robustness of the method.

Tabu search algorithm

Our algorithm is based on tabu search,[26] a local search meta-heuristic that explores the solution space by moving at each iteration from the current solution s to the best solution in its neighbourhood $N(s)$. Since the current solution may deteriorate during the search, anti-cycling rules must be implemented. In addition, diversification mechanisms can be used to help the search process explore a broad portion of the solution space. In what follows, we first introduce some notation and we present a general overview of the algorithm. We then provide a step-by-step description of the method for the VRPTW, followed by the modifications necessary for the PVRPTW and the MDVRPTW.

Notation and overview of the algorithm

An important feature of our approach is the possibility of exploring infeasible solutions during the search. Let S denote the set of solutions that satisfy constraints (i) and (ii) defined in the introduction. Each solution $s \in S$ is represented by a set of m routes such that every customer belongs to exactly one route. This solution may, however, violate the maximum load and duration constraints associated with the routes, or the time window constraints associated with the customers and the depot. The time window constraint at customer i is violated if the arrival time a_i of the vehicle is larger than the time window's upper bound l_i. However, arrival before e_i is allowed and the vehicle then incurs a waiting time $w_i = e_i - a_i$.

For a solution $s \in S$, let $c(s)$ denote the total travel time of the routes, and let $q(s)$, $d(s)$ and $w(s)$ denote the total violation of load, duration and time window constraints, respectively. The total violation of load and duration constraints is computed on a route basis with respect to the values Q_k and D_k, whereas the total violation of time window constraints is equal to $\sum_{i=1}^{n}(a_i - l_i)^+$, where $x^+ = \max\{0, x\}$. Solutions are then evaluated using a cost function $f(s) = c(s) + \alpha q(s) + bd(s) + \gamma w(s)$, where α, β and γ are positive parameters. By dynamically adjusting the values of the three parameters, this relaxation mechanism facilitates the exploration of the search space and is particularly useful for tightly constrained instances.

Another important ingredient of our method is the definition of attributes to characterize the solutions of S. With each solution $s \in S$ is associated an attribute set $B(s) = \{(i, k)$: customer i is visited by vehicle $k\}$. The neighbourhood $N(s)$ of a solution s is defined by applying a simple operator that removes an attribute (i, k) from $B(s)$ and replaces it with an attribute (i, k'), where $k \neq k'$. When customer i is removed from route k, the route is simply reconnected by linking the predecessor and successor vertices. Insertion in route k' is then performed between two consecutive vertices so as to minimize the value of $f(s)$. Attributes are also used to a large extent to control tabu tenures and implement a diversification strategy. When customer i is removed from route k, its reinsertion in that route is forbidden for the next θ iterations by assigning a tabu status to the attribute (i, k). Through an aspiration criterion, the tabu status of an attribute can, however, be revoked if that would allow the search process to reach a solution of smaller cost than that of the best solution identified having that attribute.

To diversify the search, any solution $\bar{s} \in N(s)$ such that $f(\bar{s}) \geqslant f(s)$ is penalized by a factor that is proportional to the addition frequency of its attributes and a scaling factor. More precisely, let ρ_{ik} be the number of times attribute (i, k) has been added to the solution during the search process. A penalty $p(\bar{s}) = \lambda c(\bar{s})\sqrt{nm}\sum_{(i,k)\in B(\bar{s})}\rho_{ik}$ is added to $f(\bar{s})$. The scaling factor $c(\bar{s})\sqrt{nm}$ introduces a correction to adjust the penalties with respect to the total solution cost and the size of the problem as measured by the number of possible attributes. Finally, the parameter λ is used to control the intensity of the diversification. These penalties have the effect of driving the search process toward less explored regions of the search space whenever a local optimum is reached. For notational convenience, assume that $p(\bar{s}) = 0$ if $f(\bar{s}) < f(s)$.

Algorithm for the VRPTW

An initial solution is constructed as follows. If customers are represented by Euclidean coordinates, they are first sorted in increasing order of the angle they make with the depot and an arbitrary radius. Otherwise, any arbitrary ordering may be used. At most m routes are then constructed with the following procedure:

1. Randomly choose a customer $j \in \{1, \ldots, n\}$.
2. Set $k := 1$.
3. Using the sequence of customers $j, j+1, \ldots, n, 1, \ldots, j-1$, perform the following steps for every customer i:
   - If the insertion of customer i into route k would result in the violation of load or duration constraints, set $k := \min\{k+1, m\}$.
   - Insert customer i into route k so as to minimize the increase in the total travel time of route k.

In step 3, the insertion of customer i can only be performed between successive customers j_1 and j_2 such that $e_{j_1} \leqslant e_i \leqslant e_{j_2}$, or at the end of the route if no such pair of customers exists. This rule helps satisfy time window constraints although it does not guarantee that a feasible solution will be constructed. At the end of the procedure, routes $1, \ldots, m-1$ satisfy load and duration constraints but may not respect time windows; route m may violate any of the three types of constraints.

The tabu search algorithm starts from the solution produced by the above procedure and chooses, at each iteration, the best non-tabu solution in $N(s)$. After each iteration, the values of the parameters α, β and γ are modified by a factor $1 + \delta$, where $\delta > 0$. If the current solution is feasible with respect to load constraints, the value of α is divided by $1 + \delta$; otherwise, it is multiplied by $1 + \delta$. The same rule applies to β and γ with respect to duration and time window constraints, respectively. This process is repeated for η iterations and the best feasible solution s^* identified during the search is finally post-optimized by applying, to each individual route, a specialized heuristic for the Travelling Salesman Problem with Time Windows.[27] This heuristic is in fact an adaptation of the GENIUS insertion and post-optimization procedure developed for the Travelling Salesman Problem by Gendreau *et al.*[28] The complete algorithm can be summarized as follows:

1. Set $\alpha := 1$, $\beta := 1$ and $\gamma := 1$. If s is feasible, set $s^* := s$ and $c(s^*) := c(s)$; otherwise, set $c(s^*) := \infty$.
2. For $\kappa = 1, \ldots, \eta$, do
   - Choose a solution $\bar{s} \in N(s)$ that minimizes $f(\bar{s}) + p(\bar{s})$ and is not tabu or satisfies its aspiration criteria.
   - If solution $\bar{s}$ is feasible and $c(\bar{s}) < c(s^*)$, set $s^* := \bar{s}$ and $c(s^*) := c(\bar{s})$.
   - Compute $q(\bar{s})$, $d(\bar{s})$ and $w(\bar{s})$ and update α, β and γ accordingly.
   - Set $s := \bar{s}$
3. Apply a post-optimization heuristic to each route of s^*.

Adaptations for the PVRPTW and the MDVRPTW

The adaptation of the heuristic for the PVRPTW and the MDVRPTW is rather straightforward. As mentioned in the introduction, the MDVRPTW can be seen as a special case of the PVRPTW by associating the t depots with the days of a fictitious planning period of length t. The same solution methodology can thus be used for both problems provided the implementation allows for distances and travel times that vary from day to day to reflect the different locations of the depots.

First, the set S is redefined to denote the set of solutions formed by routes that satisfy the frequency f_i and visit combination possibilities R_i of each customer i as well as constraints (i). The attribute set used to implement tabu tenures and the diversification strategy is also redefined as $B(s) = \{(i, k, \ell)$: customer i is visited by vehicle k on day $\ell\}$. A new type of exchange is then added to the definition of the neighbourhood. For every visit combination $r \in R$ and every day $\ell \in \{1, \ldots, t\}$, let $b_{r\ell}$ be a binary constant equal to 1 if and only if day ℓ belongs to visit combination r. The following two transformations are allowed during the search process:

1. Remove customer i from route k of day ℓ and insert it into another route k'.
2. (a) Replace visit day combination r currently assigned to customer i with another combination $r' \in R_i$.
   (b) For $\ell = 1, \ldots, t$, do
      - If $b_{r\ell} = 1$ and $b_{r'\ell} = 0$, remove customer i from its route of day ℓ.
      - If $b_{r\ell} = 0$ and $b_{r'\ell} = 1$, insert customer i into the route of day ℓ minimizing the increase in $f(s)$.

The first transformation is identical to that used for the VRPTW, while the second allows for modifications in the visit combination assigned to a customer.

To construct an initial solution for the PVRPTW, each customer i is first assigned a valid visit combination r randomly chosen in the set R_i. For each day ℓ of the planning period, routes are then constructed by applying the procedure explained in the previous subsection but retaining only the customers scheduled for a visit on day ℓ. To construct an initial solution for the MDVRPTW, each customer i is first assigned to the nearest depot and routes are then constructed for each depot by the same procedure.

Computational experiments

To analyse the behaviour of the tabu search heuristic and to tune its parameters, we first created new test instances by adding randomly generated time windows to existing PVRP and MDVRP instances. We then performed experiments with these instances to characterize the tradeoff between computation time and solution quality. To further assess the performance of the heuristic, we finally performed experiments on Solomon's well-known set of instances for the VRPTW,[29] and compared our results to those obtained by other heuristics. The test data and solutions described in this paper can be obtained directly from the

Internet at http://www.crt.umontreal.ca/~cordeau/tabu/. The algorithm was coded in C and all experiments were performed on a Sun Ultra 2 (300 MHz).

New instances for the PVRPTW and the MDVRPTW

Since we are not aware of any prior work on the PVRPTW or MDVRPTW, we created new instances for each problem by adding time windows of different widths to ten basic PVRP and MDVRP instances previously introduced by Cordeau *et al.*[21] These basic instances were generated randomly with a procedure that creates clusters of customers grouped around a certain number of seed points. Each instance has a different set of customers but the same locations are used for both the PVRP and the MDVRP. In the case of the MDVRP, the locations of the different depots correspond to the seed points used to generate the data. For each problem, we created two groups of ten instances. In group (a), we generated narrow time windows by first choosing a uniform random number e_i in the interval [60, 480] and then choosing a uniform random number l_i in the interval $[e_i + 90,\ e_i + 180]$. In group (b), larger time windows were created by choosing the random numbers e_i and l_i in the intervals [60, 300] and $[e_i + 180,\ e_i + 360]$, respectively. In all instances, the depots have a time window of [0, 1000].

The characteristics of the new instances are summarized in Table 1. For each instance, the value of t represents the number of days in the planning period for the PVRPTW or the number of depots for the MDVRPTW. The values of m_1 and m_2 represent the number of vehicles in the corresponding PVRPTW and MDVRPTW instance, respectively. For example, instance 2a of the PVRPTW has 48 customers, a planning period of 4 days and 6 vehicles available. The corresponding instance of the MDVRPTW has 4 depots and each depot has 3 vehicles available. Instances in group (b) often have a smaller number of vehicles because larger time windows make it easier for the same vehicle to service several customers. Although our approach can handle vehicles of different capacity, all these instances consider a homogeneous fleet where each vehicle has capacity Q and a maximal route duration D. Since vehicles do not have to depart from the depot at time e_0, maximum duration constraints are not equivalent to a time window at the depot. They can be active whenever $D < l_0 - e_0$.

Results on PVRPTW and MDVRPTW instances

In the first step of our experiments, we wanted to determine, through sensitivity analyses, the most appropriate values for the different parameters that control the heuristic. In previous experiments, Cordeau *et al*[21] observed that the best parameter settings for solving the PVRP and the MDVRP were $\delta := 0.5$, $\lambda := 0.015$ and $\theta := [7.5 \log_{10} n]$ where $[x]$ is the integer nearest to x. Starting from these initial values, we successively varied all parameters, either individually or jointly, and observed the impact on solution quality. An interesting finding of our study is that the best parameter values for solving the PVRPTW and MDVRPTW correspond exactly to the best values identified for the former two problems. As a result, these settings were used in all further experiments.

The purpose of the second step was to analyse the tradeoff between CPU time and solution quality. For this end, we first executed the algorithm once on each instance with $\eta = 10^6$ and noted the best solution identified at different points during the execution. Tables 2 and 3 indicate the cost of the best solution found after 10^4, 10^5, and 10^6 iterations. To measure the sensitivity of the heuristic to the characteristics of the initial solution, we then executed the algorithm 10 times on each instances with $\eta = 10^5$ and noted the best solution found during these runs. These values are reported in the sixth column of Tables 2 and 3. We also indicated in parentheses the percentage gap between the cost of these solutions and that of the best solution identified during all experiments. Finally, the second column of these tables reports the CPU time in minutes for performing 10^4 iterations. Since the time needed to compute an initial solution is negligible, the total computing time is directly proportional to the number of iterations. The CPU time for performing 10^5 or 10^6 iterations is thus obtained by multiplying the reported time

Table 1 Characteristics of new instances for the PVRPTW and MDVRPTW

No.	*n*	*t*	m_1	m_2	*D*	*Q*	*No.*	*n*	*t*	m_1	m_2	*D*	*Q*
1a	48	4	3	2	500	200	1b	48	4	3	1	500	200
2a	96	4	6	3	480	195	2b	96	4	6	2	480	195
3a	144	4	9	4	460	190	3b	144	4	9	3	460	190
4a	192	4	12	5	440	185	4b	192	4	12	4	440	185
5a	240	4	15	6	420	180	5b	240	4	15	5	420	180
6a	288	4	18	7	400	175	6b	288	4	18	6	400	175
7a	72	6	5	2	500	200	7b	72	6	4	1	500	200
8a	144	6	10	3	475	190	8b	144	6	8	2	475	190
9a	216	6	15	4	450	180	9b	216	6	12	3	450	180
l0a	288	6	20	5	425	170	l0b	288	6	16	4	425	170

Table 2 Best solutions identified for the PVRPTW

	Time	*10^4*	*10^5*	*10^6*	*10×10^5*	*Best*
1a	0.75	3027.89 (0.67)	3012.99 (0.17)	3007.84 (0.00)	3012.82 (0.17)	3007.84
2a	1.98	5523.07 (3.65)	5454.80 (2.37)	5330.75 (0.05)	5372.58 (0.83)	5328.33
3a	3.40	7711.44 (4.25)	7520.63 (1.67)	7462.13 (0.88)	7470.05 (0.99)	7397.10
4a	5.60	8724.32 (4.15)	8615.30 (2.85)	8380.00 (0.04)	8390.28 (0.16)	8376.95
5a	7.67	9278.18 (3.46)	9134.20 (1.85)	9075.60 (1.20)	9054.14 (0.96)	8967.90
6a	11.38	11 978.90 (2.50)	11 728.86 (0.36)	11 700.84 (0.12)	11 698.97 (0.10)	11 686.91
7a	1.12	7152.45 (2.30)	6996.98 (0.08)	6996.98 (0.08)	6991.54 (0.00)	6991.54
8a	4.94	10 428.95 (3.82)	10 179.26 (1.34)	10 045.05 (0.00)	10 092.34 (0.47)	10 045.05
9a	9.60	14 818.66 (3.66)	14 472.60 (1.24)	14 94.97 (0.00)	14 417.01 (0.85)	14 294.97
l0a	18.03	19 689.12 (5.80)	18 918.14 (1.66)	18 609.72 (0.00)	18 862.65 (1.36)	18 609.72
1b	0.60	2491.40 (7.46)	2381.24 (2.71)	2319.23 (0.04)	2318.37 (0.00)	2318.37
2b	1.65	4447.14 (4.00)	4384.94 (2.54)	4276.13 (0.00)	4315.03 (0.91)	4276.13
3b	3.92	5836.36 (2.36)	5742.92 (0.72)	5702.07 (0.00)	5726.65 (0.43)	5702.07
4b	6.61	7001.34 (3.12)	6853.96 (0.95)	6803.52 (0.20)	6789.73 (0.00)	6789.73
5b	8.44	7532.90 (6.06)	7282.34 (2.53)	7102.36 (0.00)	7172.41 (0.99)	7102.36
6b	11.79	9441.53 (2.85)	9296.64 (1.27)	9180.15 (0.00)	9194.11 (0.15)	9180.15
7b	1.72	5770.03 (2.92)	5646.73 (0.73)	5606.08 (0.00)	5619.51 (0.24)	5606.08
8b	5.18	8359.63 (4.66)	8062.15 (0.93)	7987.64 (0.00)	8040.14 (0.66)	7987.64
9b	10.37	11 815.13 (6.54)	11 415.61 (2.94)	11 089.91 (0.00)	11 127.62 (0.34)	11 089.91
l0b	15.48	14 692.50 (3.41)	14 324.66 (0.82)	14 207.64 (0.00)	14 278.87 (0.50)	14 207.64
Average	6.51	8786.05 (3.88)	8571.25 (1.49)	8458.93 (0.13)	8497.24 (0.51)	8448.32

by 10 or 100. The post-optimizer was not applied in these experiments because it rarely improved the solution.

These results show that, for both the PVRPTW and the MDVRPTW, the average gap between the best known solution and the best solution found after 10^5 iterations is below 1.5%. In addition, this gap is always below 3% except for instance 10b of the MDVRPTW. Most of the best solutions found during the experiments were in fact obtained by letting the algorithm run for 10^6 iterations. It seems that performing shorter but repeated runs with different starting solutions produces final solutions of inferior quality, as measured by the average gap. Since optimal solutions to these problems are not known, we have no precise way of measuring solution quality, but it appears that a reasonable strategy for producing good feasible solutions consists of executing a single run of the algorithm with a total of 10^5 iterations. In our experiments, this requires a computing time that varies from 5 min to a little

Table 3 Best solutions identified for the MDVRPTW

	CPU	*10^4*	*10^5*	*10^6*	*10×10^5*	*Best*
1a	0.55	1104.10 (1.86)	1084.83 (0.08)	1083.98 (0.00)	1083.98 (0.00)	1083.98
2a	1.86	1789.27 (1.49)	1764.65 (0.09)	1763.84 (0.04)	1763.07 (0.00)	1763.07
3a	2.77	2480.90 (3.01)	2441.45 (1.37)	2408.42 (0.00)	2411.58 (0.13)	2408.42
4a	4.08	3053.96 (3.24)	3008.38 (1.70)	2976.11 (0.60)	2985.07 (0.91)	2958.23
5a	5.57	3350.42 (6.90)	3165.69 (1.01)	3165.69 (1.01)	3174.03 (1.28)	3134.04
6a	7.69	4272.32 (9.43)	3944.49 (1.04)	3904.07 (0.00)	3956.01 (1.33)	3904.07
7a	1.01	1436.90 (0.95)	1428.03 (0.33)	1423.35 (0.00)	1427.32 (0.28)	1423.35
8a	2.78	2230.30 (3.72)	2204.58 (2.53)	2163.32 (0.61)	2150.22 (0.00)	2150.22
9a	4.83	2849.62 (0.56)	2849.62 (0.56)	2833.80 (0.00)	2839.39 (0.20)	2833.80
10a	7.03	3886.00 (4.54)	3717.72 (0.01)	3717.22 (0.00)	3726.86 (0.26)	3717.22
1b	0.91	1060.74 (2.84)	1048.44 (1.64)	1031.49 (0.00)	1043.35 (1.15)	1031.49
2b	2.21	1514.58 (0.94)	1514.58 (0.94)	1506.08 (0.37)	1500.48 (0.00)	1500.48
3b	3.45	2050.47 (1.48)	2036.90 (0.81)	2020.58 (0.00)	2022.34 (0.09)	2020.58
4b	5.00	2437.81 (8.46)	2266.76 (0.85)	2247.72 (0.00)	2280.55 (1.46)	2247.72
5b	6.43	2719.48 (8.36)	2583.14 (2.92)	2509.75 (0.00)	2516.19 (0.26)	2509.75
6b	8.29	3071.25 (4.33)	2995.45 (1.75)	2948.01 (0.14)	2949.55 (0.19)	2943.90
7b	1.67	1294.55 (3.56)	1250.09 (0.00)	1250.09 (0.00)	1250.09 (0.00)	1250.09
8b	3.54	1858.34 (2.71)	1823.16 (0.76)	1823.16 (0.76)	1825.53 (0.89)	1809.35
9b	7.78	2371.72 (2.63)	2338.94 (1.21)	2310.92 (0.00)	2331.74 (0.90)	2310.92
10b	8.24	3427.41 (9.44)	3231.68 (3.19)	3137.27 (0.17)	3131.90 (0.00)	3131.90
Average	4.28	2413.01 (4.02)	2334.93 (1.14)	2311.24 (0.19)	2318.46 (0.47)	2306.63

more than 3 h. While this last figure may seem large, it is worth remembering that the longest computing times are obtained for large PVRPTW instances which must normally be solved only once a week. The heuristic can also be used to quickly produce feasible solutions of reasonable quality: by performing only 10^4 iterations, a feasible solution with an average gap below 5% is obtained in less than 20 min for every instance.

Results on VRPTW instances

To obtain a more meaningful comparison of results, we performed experiments on the well known set of 56 VRPTW instances introduced by Solomon.[29] These instances, which all contain 100 customers, are divided into six sets. In sets R1 and R2, customers are randomly and uniformly distributed, whereas in sets C1 and C2, they are clustered. Sets RC1 and RC2 exhibit a mix of uniformly distributed and clustered customers. In the problems of type 1, the depot has a narrow time window which results in few customers being visited in each route. Problems of type 2 have a longer horizon and require only a few vehicles.

In these instances, the number of vehicles is not fixed but should instead be minimized as a primary objective. The secondary objective consists of minimizing the total travel time. Our heuristic was not designed to minimize the number of vehicles. However, as in the work of Taillard *et al*,[15] we decided to set the number m of available vehicles equal to the number of routes of the best solution reported in the literature. Because it may then be impossible to find a feasible solution, this number was increased by one if no feasible solution was found after $\eta/10$ iterations. The algorithm was executed several times on each instance by varying both the starting solution and the number of iterations performed. Tables 4 and 5 compare the best solutions produced by our heuristic (CLM) with those reported by Rochat and Taillard,[14] Chiang and Russell,[10] Taillard *et al*[15] and Homberger and Gehring.[11] In these tables, two asterisks indicate that our approach produced a better solution whereas a single asterisk indicates a tie.

Although several other authors have also worked on the VRPTW, we have chosen to restrict our comparisons to the most recent (and best) heuristics proposed in the literature. In addition, we report only results obtained by methods in which the feasibility and cost of the solutions were computed using a sufficient degree of precision. For example, Kohl *et al*[16] multiplied the real distances by 10 and truncated the resulting values so as to work with integers.

Table 4 Best solutions reported on Solomon's instances (first series)

No.	*RT (1995)*		*CR (1997)*		*TB (1997)*		*HG (1999)*		*CLM (2001)*		
R101	19	1650.80	19	1651.01	19	1650.79	19	1650.80	19	1650.80	*
R102	17	1486.12	17	1489.13	17	1487.60	17	1486.12	17	1488.10	
R103	14	1213.62	13	1299.15	13	1294.24	13	1292.85	13	1299.78	
R104	10	982.01	10	985.98	10	982.72	9	1013.32	10	984.00	
R105	14	1377.11	14	1382.05	14	1377.11	14	1377.11	14	1377.11	*
R106	12	1252.03	12	1255.34	12	1259.71	12	1260.12	12	1253.23	
R107	10	1159.86	10	1124.72	10	1126.69	10	1120.85	10	1113.69	**
R108	9	980.95	9	971.03	9	968.59	9	970.05	9	964.38	**
R109	11	1235.68	12	1013.16	11	1214.54	11	1194.73	11	1199.63	
R110	11	1080.36	10	1174.49	11	1080.36	10	1182.49	10	1125.04	**
R111	10	1129.88	10	1119.48	10	1104.83	10	1099.46	10	1108.90	
R112	10	953.63	10	970.87	10	964.01	9	1003.73	10	957.04	
C101	10	828.94	10	828.94	10	828.94	10	828.94	10	828.94	*
C102	10	828.94	10	828.94	10	828.94	10	828.94	10	828.94	*
C103	10	828.06	10	828.06	10	828.06	10	828.94	10	828.06	*
C104	10	824.78	10	824.78	10	824.78	10	828.94	10	824.78	*
C105	10	828.94	10	828.94	10	828.94	10	828.94	10	828.94	*
C106	10	828.94	10	828.94	10	828.94	10	828.94	10	828.94	*
C107	10	828.94	10	828.94	10	828.94	10	828.94	10	828.94	*
C108	10	828.94	10	828.94	10	828.94	10	828.94	10	828.94	*
C109	10	828.94	10	828.94	10	828.94	10	828.94	10	828.94	*
RC101	15	1623.58	15	1642.82	14	1696.94	14	1697.43	14	1708.80	
RC102	13	1477.54	13	1540.97	12	1554.75	12	1558.07	12	1558.07	
RC103	11	1262.02	11	1302.73	11	1264.27	11	1263.09	11	1262.98	
RC104	10	1135.83	10	1155.07	10	1135.83	10	1138.57	10	1135.48	**
RC105	13	1733.56	13	1735.84	13	1643.38	13	1637.15	13	1644.43	
RC106	12	1384.92	12	1395.37	11	1448.26	11	1432.12	11	1427.13	**
RC107	11	1230.95	11	1235.28	11	1230.54	11	1232.26	11	1231.53	
RC108	10	1170.70	10	1171.40	10	1139.82	10	1147.26	10	1149.79	
Total	332	32 976.57	331	33 071.31	328	33 080.40	325	33 218.04	327	33 095.33	

Table 5 Best solutions reported on Solomon's instances (second series)

No.	*RT (1995)*		*CR (1997)*		*TB (1997)*		*HG (1999)*		*CLM (2001)*		
R201	4	1281.58	4	1272.70	4	1254.80	4	1252.37	4	1253.26	
R202	4	1088.07	3	1264.44	3	1214.28	3	1198.45	3	1197.66	**
R203	3	948.74	3	950.08	3	951.59	3	942.64	3	945.55	
R204	2	869.29	2	855.21	2	941.76	2	854.88	2	849.62	**
R205	3	1063.24	3	1035.60	3	1038.72	3	1013.47	3	1008.52	**
R206	3	912.97	3	917.59	3	932.47	3	913.68	3	913.18	
R207	3	814.78	2	914.39	3	837.20	2	971.34	2	948.23	
R208	2	738.60	2	746.85	2	748.01	2	731.23	2	734.85	
R209	3	944.64	3	935.67	3	959.47	3	910.55	3	916.47	
R210	3	967.50	3	982.60	3	980.90	3	955.39	3	964.22	
R211	2	949.50	2	967.32	2	923.80	2	910.09	2	933.75	
C201	3	591.56	3	591.56	3	591.56	3	591.56	3	591.56	*
C202	3	591.56	3	591.56	3	591.56	3	591.56	3	591.56	*
C203	3	591.17	3	591.17	3	591.17	3	591.17	3	591.17	*
C204	3	590.60	3	603.05	3	590.60	3	590.60	3	590.60	*
C205	3	588.88	3	588.88	3	588.88	3	588.88	3	588.88	*
C206	3	588.49	3	588.49	3	588.49	3	588.49	3	588.49	*
C207	3	588.29	3	588.29	3	588.29	3	588.29	3	588.29	*
C208	3	588.32	3	588.32	3	588.32	3	588.32	3	588.32	*
RC201	4	1438.89	4	1456.33	4	1413.79	4	1415.00	4	1406.94	**
RC202	4	1165.57	3	1532.25	4	1164.25	3	1389.57	3	1407.52	
RC203	3	1079.57	3	1078.73	3	1112.55	3	1060.45	3	1073.39	
RC204	3	806.75	3	849.10	3	831.69	3	799.12	3	806.12	
RC205	4	1333.71	4	1762.12	4	1328.21	4	1302.42	4	1326.83	
RC206	3	1212.64	3	1168.12	3	1158.81	3	1196.12	3	1160.91	
RC207	3	1085.61	3	1133.23	3	1082.32	3	1112.60	3	1062.05	**
RC208	3	833.97	3	872.68	3	847.90	3	848.10	3	832.36	**
Total	83	24 254.49	80	25 426.33	82	24 441.39	80	24 496.34	80	24 460.30	

This may produce solutions that violate some of the time window constraints. In our case, all computations are performed using double precision floating point arithmetic. It should be mentioned that Chiang and Russell[10] used an objective that first minimizes the number of vehicles, followed by total route time and, finally, total distance.

On all instances of sets C1 and C2, our approach produced the best known solution. On the complete set of 56 instances, our approach found better solutions than the other heuristics in 11 cases and solutions of equal cost in 19 cases. Our algorithm is only surpassed by the Homberger and Gehring algorithm[11] which produced 16 best solutions and 18 ties. When the best solution found has a higher cost than that of the best solution reported by other authors, the relative difference exceeds 1% in only five cases. Finally, in only two cases (instances R104 and R112) did our approach fail to find a feasible solution with the number of vehicles equal to the number of routes used in the best known solution. In both cases, however, only Homberger and Gehring[11] reported a solution with this number of vehicles.

Again, the best solutions were found by performing a large number of iterations from a single starting solution. In most cases, however, good solutions were obtained after only 10^5 iterations. The time needed to perform this number of iterations varies from about 20 min for instances of type 1 to 60 min for instances of type 2. Instances with fewer vehicles take longer to solve because more possibilities must be considered when computing insertion costs. We observed that the post-optimizer was rather useful for the second series of Solomon's instances (Table 5) in which the number of vehicles is low and each route contains several customers. Thus, the solutions of these instances can often be improved by performing intra-route exchanges. In contrast, the post-optimizer has little impact when vehicle routes contain few customers, as is the case in Solomons's first series of instances (Table 4).

We finally tested our method on nine real-life instances of the VRPTW. The first two, D417 and E417, originate from a fast food routing application in the southeastern United States.[30] Both instances have 417 customers and are based on Euclidean distances. The instances differ in that E417 has a larger proportion of tight time windows. Table 6 compares the best solutions produced by our heuristic (CLM) with those reported by Rochat and Taillard,[14] Chiang and Russell,[10] Taillard *et al*[15] and Homberger and Gehring.[11]

The best known solutions are again those of Homberger and Gehring.[11] As in the case of Chiang and Russell[10] and Taillard *et al*,[15] our algorithm did not produce feasible solutions with 54 vehicles. However, we did find the best known solutions with 55 vehicles. It is noteworthy that Rochat and Taillard[14] also reported solutions with 55

Table 6 Best solutions reported on first set of real-life instances

No.	*RT (1995)*		*CR (1997)*		*TB (1997)*		*HG (1999)*		*CLM (2001)*	
D417	54	6264.8	55	3455.3	55	3439.8	54	4703	55	3426.2
E417	54	7211.8	55	3796.6	55	3707.1	54	4732	55	3633.6

vehicles for D417 and E417 with a cost of 3467.8 and 3693.2, respectively.

The next seven real-life instances originate from a grocery distributor in Germany. These data are available from ftp://ftp.zpr.uni-koeln.de/pub/VRP/. To our knowledge, no results have been reported in the literature for these instances, although Hamacher[3] reports solution values for modified instances without providing the modified data. These instances contain between 819 and 1035 customers each and are also based on Euclidean distances. Their characteristics are reported in Table 7. In instance 3, which originally contained 910 customers, one customer had a demand in excess of the capacity of a vehicle. This was handled by duplicating the customer and assigning a full load to one vehicle.

Because the number of vehicles is not fixed in these instances, we first ran the algorithm with $\eta = 1000$ to evaluate the number of required vehicles. For each instance, the number of vehicles used in the best solution found during these experiments is indicated in column *m* of Table 7. We then set the number of vehicles equal to this number and ran the algorithm twice on each instance. Using standard parameter settings, 10 000 iterations were performed in the first run and 50 000 iterations were performed in the second run. The results of these experiments are also reported in Table 7. Here, column *Time* indicates the CPU time in minutes required to perform the first run. CPU time for the second run was approximately five times larger. Columns $m(s^*)$ and $c(s^*)$ indicate the number of vehicles used in the best solution found and the cost of that solution. Interestingly, the results show that when the algorithm runs longer, better solutions are usually found both in terms of number of vehicles and of cost. For the largest instance, G6, the CPU time exceeded 16 h when 50 000 iterations were performed. In all instances, however, solutions of very good quality were found after 20 000 or 25 000 iterations. Again, the post-optimizer played an important role in these instances which tend to produce solutions containing several long vehicle routes.

Table 7 Characteristics and results for the second set of real-life instances

				$\eta = 10\,000$		$\eta = 50\,000$	
No.	*n*	*m*	*Time*	*m(s*)*	*c(s*)*	*m(s*)*	*c(s*)*
G1	915	31	185.82	30	1944.10	30	1673.28
G2	819	29	156.63	27	1781.49	28	1463.98
G3	911	34	145.15	33	2046.99	30	1763.53
G4	849	31	144.87	30	1831.17	28	1602.20
G5	974	39	147.21	39	2303.46	39	2137.70
G6	1035	36	202.49	34	2326.48	34	2163.25
G7	984	32	200.82	32	2344.94	29	1910.70

Conclusion

We have proposed a tabu search heuristic for the Vehicle Routing Problem with Time Windows (VRPTW) and two of its generalizations: the Periodic Vehicle Routing Problem with Time Windows (PVRPTW) and the Multi-Depot Vehicle Routing Problem with Time Windows (MDVRPTW). We believe the strengths of the method lie in its simplicity, its efficiency and its robustness. Unlike most alternative approaches, our method is based on a very simple scheme and operates with a limited number of parameters. It generates a single initial solution and applies a very simple exchange procedure for a predetermined number of iterations. Excluding the post-optimizer, the complete C implementation of the algorithm requires less than 1000 lines of code. Another advantage of this heuristic is its speed of execution and memory usage. The program can be run on any computer with minimal resources and requires only a few minutes of computation time before identifying good quality solutions even on large size instances. The proposed algorithm may not be the best available for the VRPTW and it does not always yield a solution with the least possible number of vehicles. However, these weaknesses are largely compensated by the flexibility and the robustness of the approach. By using the same methodology and parameter settings as for the PVRP and MDVRP, we were indeed able to solve three other classes of problems, with provably good solutions in the case of the VRPTW.

Acknowledgements—This work was supported by the École des Hautes Études Commerciales de Montréal, the Social Sciences and Humanities Research Council of Canada, and the Natural Sciences and Engineering Research Council of Canada under grant OGP0039682. Thanks are also due to Hicham Lahlou for programming the post-optimizer, to Robert A. Russell for providing the data for instances D417 and E417, and to the referees for their valuable comments.

References

1 Carter MW, Farvolden JM, Laporte G and Xu J (1996). Solving an integrated logistics problem arising in grocery distribution. *INFOR* **34**: 290–306.
2 Golden BL and Wasil EA (1987). Computerized vehicle routing in the soft drink industry. *Opns Res* **35**: 6–17.
3 Hamacher A (1997). *Baumzerlegungen unter Nebenbedingungen – Ein Clusterverfahren zur Lösung praktischer Vehicle-Routing-Probleme*. *PhD thesis*, Mathematisches Institut, Universität zu Köln.

4 Rochat Y and Semet F (1994). A tabu search approach for delivering pet food and flour in Switzerland. *J Opl Res Soc* **45**: 1233–1246.

5 Desrosiers J *et al* (1986). TRANSCOL: A multi-period school bus routing and scheduling system. *TIMS Stud Mngt Sci* **22**: 47–71.

6 Bell WJ *et al* (1983). Improving the distribution of industrial gases with an on-line computerized routing and scheduling optimizer. *Interfaces* **13**(6): 4–23.

7 Brown G, Ellis C, Graves G and Ronen D (1987). Real-time wide area dispatch of Mobil tank trucks. *Interfaces* **17**(1): 107–120.

8 Assad AA (1988). Modeling and implementation issues in vehicle routing. In: Golden BL and Assad AA (eds). *Vehicle Routing: Methods and Studies*. North-Holland: Amsterdam, pp 7–45.

9 Madsen OBG, Tosti K and Vaelds J (1995). A heuristic method for dispatching repair men. *Ann Opns Res* **61**: 213–226.

10 Chiang WC and Russell RA (1997). A reactive tabu search metaheuristic for the vehicle routing problem with time windows. *INFORMS J Comp* **9**: 417–430.

11 Homberger J and Gehring H (1999). Two evolutionary meta-heuristics for the vehicle routing problem with time windows. *INFOR* **37**: 297–318.

12 Potvin JY and Bengio S (1996). The vehicle routing problem with time windows – part II: genetic search. *INFORMS J Comp* **8**: 165–172.

13 Potvin JY, Kervahut T, Garcia BL and Rousseau JM (1996). The vehicle routing problem with time windows – part I: Tabu search. *INFORMS J Comp* **8**: 158–164.

14 Rochat Y and Taillard ED (1995). Probabilistic diversification and intensification in local search for vehicle routing. *J Heuristics* **1**: 147–167.

15 Taillard ED *et al* (1997). A tabu search heuristic for the vehicle routing problem with soft time windows. *Transport Sci* **31**: 170–186.

16 Kohl N *et al* (1999). 2-path cuts for the vehicle routing problem with time windows. *Transport Sci* **33**: 101–116.

17 Desaulniers G *et al* (1998). A unified framework for deterministic time constrained vehicle routing and crew scheduling problems. In: Crainic T and Laporte G (eds). *Fleet Management and Logistics*. Kluwer: Norwell, MA, pp 57–93.

18 Russell RA and Gribbin D (1991). A multiphase approach to the period routing problem. *Networks* **21**: 747–765.

19 Gaudioso M and Paletta G (1992). A heuristic for the periodic vehicle routing problem. *Transport Sci* **26**: 86–92.

20 Chao IM, Golden BL and Wasil EA (1995). A new heuristic for the period traveling salesman problem. *Comp Oper Res* **22**: 553–565.

21 Cordeau JF, Gendreau M and Laporte G (1997). A tabu search heuristic for periodic and multi-depot vehicle routing problems. *Networks* **30**: 105–119.

22 Chao IM, Golden BL and Wasil EA (1993). A new heuristic for the multi-depot vehicle routing problem that improves upon best-known solutions. *Am J Math Mngt Sci* **13**: 371–406.

23 Renaud J, Boctor FF and Laporte G (1996). An improved petal heuristic for the vehicle routing problem. *J Opl Res Soc* **47**: 329–336.

24 Banerjea-Brodeur M, Cordeau JF, Laporte G and Lasry A (1998). Scheduling linen deliveries in a large hospital. *J Opl Res Soc* **49**: 777–780.

25 Lenstra JK and Rinnooy Kan AHG (1981). Complexity of vehicle routing and scheduling problems. *Networks* **11**: 221–227.

26 Glover F (1986). Future paths for integer programming and links to artificial intelligence. *Comp Oper Res* **13**: 533–549.

27 Gendreau M, Hertz A, Laporte G and Stan M (1998). A generalized insertion heuristic for the traveling salesman problem with time windows. *Opns Res* **43**: 330–335.

28 Gendreau M, Hertz A and Laporte G (1992). New insertion and postoptimization procedures for the traveling salesman problem. *Opns Res* **40**: 1086–1094.

29 Solomon MM (1987). Algorithms for the vehicle routing and scheduling problems with time window constraints. *Opns Res* **35**: 254–265.

30 Russell RA (1995). Hybrid heuristics for the vehicle routing problem with time windows. *Transport Sci* **29**: 156–166.

*Received January 2000;
accepted February 2001 after two revisions*