A Tabu Search Heuristic for Periodic and Multi-Depot Vehicle Routing Problems

Jean-François Cordeau, Michel Gendreau, Gilbert Laporte

Centre de recherche sur les transports, Université de Montréal, Case postale 6128, succursale ''Centre-ville,'' Montréal, Canada H3C 3J7

Received 15 December 1995; accepted 18 February 1997

Abstract: We propose a tabu search heuristic capable of solving three well-known routing problems: the periodic vehicle routing problem, the periodic traveling salesman problem, and the multi-depot vehicle routing problem. Computational experiments carried out on instances taken from the literature indicate that the proposed method outperforms existing heuristics for all three problems. © *1997 John Wiley & Sons, Inc.* Networks **30:** 105–119, 1997

Keywords: periodic vehicle routing problem; periodic traveling salesman problem; multi-depot vehicle routing problem; tabu search heuristic

1. INTRODUCTION

The purpose of this paper is to present a unified Tabu Search (TS) heuristic for three important variants of the classical Vehicle Routing Problem (VRP): the Periodic Vehicle Routing Problem (PVRP), the Periodic Traveling Salesman Problem (PTSP), and the Multi-Depot Vehicle Routing Problem (MDVRP). Both the PVRP and the MDVRP are generalizations of the VRP, while the PTSP is a special case of the PVRP.

The VRP is defined on a graph $G = (V, A)$, where $V = \{v_0, v_1, \ldots, v_n\}$ 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 the depot at which are based m vehicles of capacity $Q_1, \ldots, Q_m$. Each vertex of $V \setminus \{v_0\}$ corresponds to a city or a customer and has a nonnegative demand q_i, as well as a nonnegative service duration d_i. With A is associated a cost or travel time matrix $C = (c_{ij})$. The VRP consists of designing m vehicle routes on G such that (i) each route starts and ends at the depot; (ii) each customer belongs to exactly one route; (iii) the total demand of the route followed by vehicle k does not exceed Q_k, and its total duration does not exceed a preset value D_k; and (iv) the total duration of all routes is minimized. The VRP is a hard combinatorial problem that has received a great deal of attention in the operations research literature. In the current state of knowledge, it can rarely be solved to optimality for sizes in excess of 50, and it is usually tackled by means of heuristics. See Laporte [26], Fisher [14], and Laporte and Osman [29] for surveys and a bibliography.

The PVRP works with a planning horizon of t days and each customer i specifies a service frequency e_i and a set C_i of allowable combinations of visit days. For example, if $e_i = 2$ and $C_i = \{\{1, 3\}, \{2, 4\}, \{3, 5\}\}$,

Correspondence to: G. Laporte

Contract grant sponsor: Canadian Natural Sciences and Engineering Research Council; contract grant numbers: OGP0038816; OGP0039682

Contract grant sponsor: Quebec Government FCAR programs

CCC 0028-3045/97/020105-15

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 then consists of simultaneously selecting a visit combination for each customer and establishing vehicle routes for each day of the planning horizon, according to the VRP rules. The PVRP arises naturally, e.g., in the areas of grocery distribution [3, 24] and refuse collection [2, 37]. Early heuristics were developed by Beltrami and Bodin [2] and Russell and Igo [37] who adapted methods previously proposed for the VRP. More elaborate algorithms were then described by Christofides and Beasley [8], Tan and Beasley [39], Russell and Gribbin [36], and Gaudioso and Paletta [15]. The best method is probably that proposed recently by Chao et al. [6]. These authors use integer linear programming to assign a visit combination to each customer. They then solve a VRP for each day by means of a modified version of the Clarke and Wright algorithm [23]. Local improvements are then obtained by using the record-to-record approach of Dueck [12] and 2-opt interchanges [30]. Reinitializations are finally performed to diversify the search.

In some contexts such as mail delivery or lawn-care services, side constraints are often nonbinding and can be disregarded. This gives rise to the PTSP, a special case of the PVRP obtained by setting $m = 1$ and $Q_1 = D_1 = \infty$. Heuristics for the PTSP were proposed by Christofides and Beasley [8], Paletta [31] and Chao et al. [7]. Again, the latter approach appears to be the best. It constructs an initial solution by allocating every customer to days so as to minimize the largest number of customers visited on a single day. Combination interchanges are then performed by using Dueck's method, followed by reinitializations.

The MDVRP is defined on a single day but vehicles operate from one of several depots instead of only one, and each vehicle route must start and end at the same depot. Otherwise, the statement of the problem is the same as that of the VRP. As will be seen later, the MDVRP can also be viewed as a special case of the PVRP. Early branch and bound algorithms were proposed by Laporte et al. [27] for the case where C is symmetric and by Laporte et al. [28] for the asymmetric case. The largest problems reported solved to optimality involved 50 customers in the first case and 80 customers in the second case. The first heuristics were proposed by Tillman [40], Tillman and Hering [42], Tillman and Cain [41], Wren and Holliday [43], Gillett and Johnson [18], Gillett and Miller [19], Golden et al. [23], and Raft [32], all using adaptations of standard VRP procedures. Chao et al. [5] proposed a better approach based again on the record-to-record method of Dueck. Finally, Renaud et al. [34] described a tabu search heuristic yielding highly competitive results. It constructs an initial solution by assigning each customer to its nearest depot and solving the resulting VRPs by means of the Improved Petal heuristic [33]. The search heuristic is composed of three phases: fast improvement, intensification, and diversification. In each of these phases, various interchange mechanisms are used to swap customers between routes belonging to the same or to different depots.

In the three problems under consideration, we minimize the total travel cost of vehicles, as do most authors in the field. The number of vehicles used in the solution is bounded above by a given value m each day (or at each depot), but does not affect the value of the solution.

Our aim is to describe a single tabu search heuristic capable of handling the PVRP, the PTSP, and the MDVRP. A main advantage of the approach lies in its simplicity. Its structure is uncomplicated and parameter setting is kept at a minimum. On test problems, the proposed heuristic produces better solutions than do the previous methods in relatively short computing times. The problem formulation is presented in Section 2, followed by the algorithm in Section 3 and computational results in Section 4.

2. FORMULATION

We first formulate the PVRP and then show how the same formulation can be used to describe the PTSP and the MDVRP. This formulation should help better focus the problems under consideration. Since the same arc can be traversed several times over the planning horizon, it is convenient to define the PVRP on a multigraph $G = (V, A)$, where $V = \{v_0, v_1, \ldots, v_n\}$ is the vertex set and $A = \{(v_i, v_j)^{k,\ell}\}$ is the arc set. Here, k and ℓ refer to the vehicle number and the day of visit, respectively. With each arc $(v_i, v_j)^{k,\ell}$ is associated a nonnegative cost $c_{ijk\ell}$ proportional to the travel time of vehicle k from customer i to customer j on day ℓ. To formulate the PVRP as a 0–1 linear program, define binary constants $a_{r\ell}$ equal to 1 if and only if day ℓ belongs to visit combination r. Also, define binary variables $x_{ijk\ell}$ equal to 1 if and only if vehicle k visits customer j immediately after visiting customer i during day ℓ $(i \neq j)$, and y_{ir} equal to 1 if and only if visit combination $r \in C_i$ is assigned to customer i. Assume that $d_0 = 0$ and $q_0 = 0$. The PVRP can then be stated as follows:

$$\text{Minimize} \sum_{i=0}^{n} \sum_{j=0}^{n} \sum_{k=1}^{m} \sum_{\ell=1}^{t} c_{ijk\ell}\, x_{ijk\ell}, \tag{1}$$

subject to

$$\sum_{r \in C_i} y_{ir} = 1 \quad (i = 1, \ldots, n); \tag{2}$$

$$\sum_{j=0}^{n}\sum_{k=1}^{m} x_{ijk\ell} - \sum_{r\in C_i} a_{r\ell}\, y_{ir} = 0 \quad (i = 1, \ldots, n;\ \ell = 1, \ldots, t); \tag{3}$$

$$\sum_{i=0}^{n} x_{ihk\ell} - \sum_{j=0}^{n} x_{hjk\ell} = 0 \quad (h = 0, \ldots, n;\ k = 1, \ldots, m;\ \ell = 1, \ldots, t); \tag{4}$$

$$\sum_{j=1}^{n} x_{0jk\ell} \le 1 \quad (k = 1, \ldots, m;\ \ell = 1, \ldots, t); \tag{5}$$

$$\sum_{i=0}^{n}\sum_{j=0}^{n} q_i\, x_{ijk\ell} \le Q_k \quad (k = 1, \ldots, m;\ \ell = 1, \ldots, t); \tag{6}$$

$$\sum_{i=0}^{n}\sum_{j=0}^{n} (c_{ijk\ell} + d_i)\, x_{ijk\ell} \le D_k \quad (k = 1, \ldots, m;\ \ell = 1, \ldots, t); \tag{7}$$

$$\sum_{v_i\in S}\sum_{v_j\in S} x_{ijk\ell} \le |S| - 1 \quad (k = 1, \ldots, m;\ \ell = 1, \ldots, t;\ S \subseteq V\setminus\{0\};\ |S| \ge 2); \tag{8}$$

$$x_{ijk\ell} \in \{0, 1\} \quad (i = 0, \ldots, n;\ j = 0, \ldots, n;\ k = 1, \ldots, m;\ \ell = 1, \ldots, t); \tag{9}$$

$$y_{ir} \in \{0, 1\} \quad (i = 1, \ldots, n;\ r \in C_i). \tag{10}$$

Constraints (2) mean that one feasible visit combination must be assigned to each customer while constraints (3) guarantee that each customer is visited only on the days corresponding to the assigned combination. The role of constraints (4) is to ensure that when a vehicle arrives at a customer on a given day it also leaves that customer on the same day. Constraints (5) specify that each vehicle is used at most once every day. Limits on vehicle capacity and route duration are imposed through constraints (6) and (7). Finally, constraints (8) are standard subtour elimination constraints.

If the fleet is homogeneous, as is the case in our computational experiments, then $D_k = D$ and $Q_k = Q$ for $k = 1, \ldots, m$. If the travel time between customers i and j is independent of the day of the visit and of the vehicle used, then $c_{ijk\ell} = c_{ij}$ for $k = 1, \ldots, m$ and $\ell = 1, \ldots, t$. If $t = 1$, then the planning horizon is a single day and the PVRP reduces to the VRP. If $m = 1$, $Q_1 = D_1 = \infty$, then the PVRP reduces to the PTSP. Finally, it is possible to formulate the MDVRP using this model by associating depots with days. For this, assume that there are t depots and let $C_i = \{\{1\}, \ldots, \{t\}\}$ for $i = 1, \ldots, n$. Then, define $c_{0ik\ell}$ and $c_{i0k\ell}$ as the travel costs between depot ℓ and customer i, using vehicle k. Note that the periodic MDVRP would require five-index variables and thus cannot be formulated as a special case of the above model.

3. TABU SEARCH ALGORITHM

Tabu search is a local search metaheuristic proposed independently by Glover [20] and Hansen and Jaumard [25]. Briefly, the method explores the solution space by moving at each iteration from a solution s to the best solution in a subset of its neighborhood $N(s)$. Contrary to classical descent methods, the current solution may deteriorate from one iteration to the next. Thus, to avoid cycling, solutions possessing some attributes of recently explored solutions are temporarily declared tabu or forbidden, unless their cost is less than a so-called aspiration level. Some implementations allow for intermediate infeasible solutions. Also, various techniques are often employed to diversify or to intensify the search process. For recent surveys of TS, see Glover and Laguna [21] as well as Glover et al. [22].

A basic ingredient of our TS algorithm is the GENI heuristic used to perform insertions of unrouted customers or removals of customers from their current routes and their reinsertions into different routes. Starting with three arbitrary vertices, GENI gradually constructs a Hamiltonian tour by inserting at each step a vertex v between two of its p closest neighbors v_i and v_j on the current tour. At this time, v_i and v_j are not necessarily consecutive, but after the insertion, the sequence (v_i, v, v_j) will be part of the tour. To perform the insertion, GENI considers $O(p^4)$ 4-opt [30] modifications of the tour around v_i and v_j and it selects the least-cost insertion. Similarly, the reverse of this procedure can be applied to remove a vertex v from a tour. Tests have shown that GENI outperforms classical tour construction heuristics. For further details and computational results on this method, see Gendreau et al. [16].

Our TS algorithm for the PVRP, the PTSP, and the MDVRP is essentially the same, except for the construction of the initial solution. It also applies to the VRP, but specialized algorithms are recommended in this case (see, e.g., Gendreau et al. [17], Taillard [38], Rochat and Taillard [35]). We first describe the initialization procedure used for the PVRP, followed by the modifications necessary for the PTSP and the MDVRP. We then explain the general functioning of the search algorithm, followed by a step-by-step description.

3.1. Construction of an Initial Solution

1. If customers are represented by Euclidean coordinates (as they are in our test problems), sort them in increasing order of the angle that they make with the depot

and an arbitrary radius. If not, use any arbitrary ordering.

2. Assign to each customer i a visit combination randomly chosen in the set C_i.
3. For $\ell = 1, \ldots, t$, do

   (a) Let j be a customer randomly chosen among those closest to the depot.
   (b) Set $k := 1$.
   (c) Using the sequence of customers $j, j+1, \ldots, n, 1, \ldots, j-1$, perform the following steps for every customer i scheduled for visit on day ℓ.

   - If the insertion of customer i in route k on day ℓ would result in the violation of capacity or duration constraints, set $k := \min\{k+1, m\}$.
   - Insert customer i into route k of day ℓ using the GENI heuristic.

Note that at the end of this procedure only route m may be infeasible. For the PTSP, customers are processed according to the sequence determined at Step 3, and each is assigned a visit combination that minimizes the total insertion cost in existing routes. For the MDVRP, each customer is assigned to its nearest depot and the angle is computed with respect to that depot. In both these problems, Step 2 does not apply.

TABLE I. Notations used in the description of the TS algorithm

$B(s)$	Attribute set of solution s
$c(s)$	Routing cost of solution s
$d(s)$	Excess duration of solution s
$f(s)$	Cost of solution s
$g(s)$	Penalized cost of solution s
$M(s)$	A subset of $N(s)$
$N(s)$	Neighborhood of solution s
p	Neighborhood size in GENI
$s, \bar{s}$	Solutions
s^*	Best solution identified
α	Penalty factor for overcapacity
β	Penalty factor for overduration
γ	Factor used to adjust the intensity of the diversification
δ	Parameter used to update α and β
η	Total number of iterations to be performed
θ	Tabu duration
λ	Iteration counter
ρ_{ikl}	Number of times attribute (i, k, l) has been added to the solution
σ_{ikl}	Aspiration level of attribute (i, k, l)
τ_{ikl}	Last iteration for which attribute (i, k, l) is declared tabu

3.2. General Functioning of the Search Algorithm

The initial solution just produced is not necessarily feasible since the number of vehicles available on each day or at each depot is limited. As in TABUROUTE [17], infeasible solutions are allowed during the search and a solution s is evaluated by means of the cost function $f(s) = c(s) + \alpha q(s) + \beta d(s)$, where

$$c(s) = \sum_{i=0}^{n}\sum_{j=0}^{n}\sum_{k=1}^{m}\sum_{\ell=1}^{t} c_{ijk\ell}\, x_{ijk\ell},$$

$$q(s) = \sum_{k=1}^{m}\sum_{\ell=1}^{t}\left[\left(\sum_{i=0}^{n}\sum_{j=0}^{n} q_i\, x_{ijk\ell}\right) - Q_k\right]^+,$$

$$d(s) = \sum_{k=1}^{m}\sum_{\ell=1}^{t}\left[\left(\sum_{i=0}^{n}\sum_{j=0}^{n} (c_{ijk\ell} + d_i)\, x_{ijk\ell}\right) - D_k\right]^+,$$

α, β are positive parameters, and $[x]^+ = \max\{0, x\}$. Thus, if s is feasible, the values of $c(s)$ and $f(s)$ coincide. Otherwise, $f(s)$ includes two penalty terms proportional to the total excess quantity and excess duration of the routes.

With each solution s is associated an attribute set $B(s) = \{(i, k, \ell) : \text{customer } i \text{ is visited by vehicle } k \text{ on day } \ell\}$. The transition from the current solution s to the solution $s' \in N(s)$ can be expressed by the removal and addition of attributes to the set $B(s)$. In fact, the neighborhood $N(s)$ of a solution s is composed of all solutions that can be obtained by performing one of the following transformations:

1. Remove customer i from route k on day ℓ and insert it into another route k'.
2. (a) Replace visit combination r currently assigned to customer i with another combination $r' \in C_i$.
   (b) For $\ell = 1, \ldots, t$, do

   - If $a_{r\ell} = 1$ and $a_{r'\ell} = 0$, remove customer i from its route on day ℓ.
   - If $a_{r\ell} = 0$ and $a_{r'\ell} = 1$, insert customer i into the route on day ℓ minimizing the increase in $f(s)$.

If customer i is removed from route k on day ℓ, reinserting it in that route is forbidden for the next θ iterations. The number of the last iteration for which attribute (i, k, ℓ) is declared tabu is denoted by $\tau_{ik\ell}$. The tabu status of an attribute can, however, be revoked if that would lead to a feasible solution of lesser cost than that of the best solution identified having that attribute. The aspiration

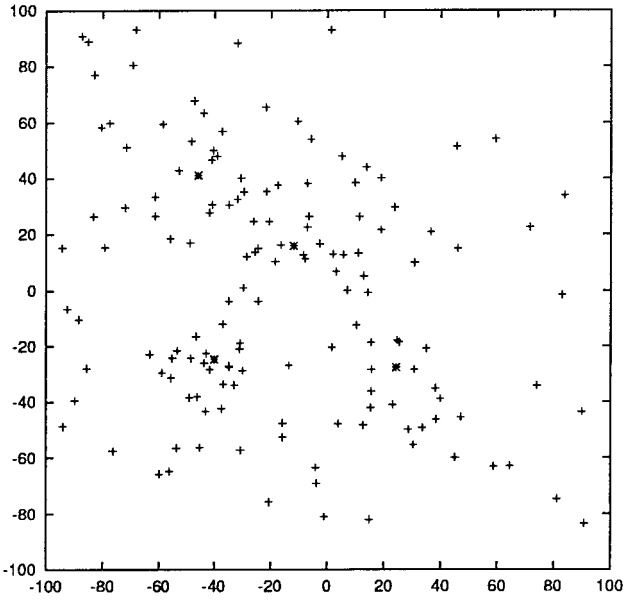

Fig. 1. Locations of vertices generated for instance *c*.

level of an attribute (i, k, ℓ), noted $\sigma_{ik\ell}$, is first set equal to $c(s)$ if (i, k, ℓ) belongs to the attribute set of a feasible initial solution and to ∞ otherwise. Every time a feasible solution s is identified, the aspiration level of each of its attributes (i, k, ℓ) is updated to $\min\{\sigma_{ik\ell}, c(s)\}$.

At each iteration, the subset $M(s) \subseteq N(s)$ consists of all solutions $\bar{s} \in N(s)$ reachable from s without incurring the risk of cycling. These solutions $\bar{s}$ are such that at least one of the attributes that must be added to s to obtain $\bar{s}$ is not tabu or $c(\bar{s})$ is less than the aspiration level of this attribute.

To diversify the search, the cost evaluation of a solution leading to an increase in the value of $f(s)$ is biased so as to penalize the attributes most frequently added to the solution. Let $\rho_{ik\ell}$ be the number of iterations during which the attribute (i, k, ℓ) has been added to the current solution. The penalty term is proportional to the product of $c(s)$, $\rho_{ik\ell}$, and $\sqrt{nmt}$. The use of a square-root factor was first suggested in the context of the VRP by Taillard [38] after extensive numerical experiments. It can be broadly explained as follows: Since there are $O(nmt)$ possible attributes, the frequency of addition into the solution of a given attribute is inversely related to problem size. Using an $\sqrt{nmt}$ factor seems to adequately compensate for problem size. A constant factor γ is used to adjust the intensity of the diversification. Finally, at each iteration, the values of α and β are modified by a factor $1 + \delta > 1$. If the current solution is feasible with respect to quantity (duration), the value of $\alpha(\beta)$ is divided by $1 + \delta$; otherwise, it is multiplied by $1 + \delta$.

The algorithm just described bears some similarities with TABUROUTE [17]: It uses GENI to perform insertions, it allows intermediate infeasible solutions, and it employs a diversification scheme based on a penalized function. However, it differs from TABUROUTE in several fundamental respects: The initial solution is constructed very differently, no ''false starts'' are used to identify a promising solution, the tabu duration is constant, the update rules for α and β are different, there are no periodical reoptimizations, the number of iterations is fixed a priori, aspiration levels are attribute-dependent, candidate vertices for reinsertion are selected differently, and we do not apply an intensification phase.

TABLE II. Characteristics of the 10 randomly generated instances

I	*n*	*m*	*t*	*D*	*Q*
a	48	2	4	500	200
b	96	4	4	480	195
c	144	6	4	460	190
d	192	8	4	440	185
e	240	10	4	420	180
f	288	12	4	400	175
g	72	3	6	500	200
h	144	6	6	475	190
i	216	9	6	450	180
j	288	12	6	425	170

TABLE III. Effect of parameters η and *p*

η	*p*	$c(s^*)$	T^*	*T*
40,000	0	5398.33	18.21	25.56
30,000	2	5264.26	15.34	25.08
20,000	3	5252.05	13.84	26.21
10,000	4	5283.13	16.37	24.83
5000	5	5307.89	13.60	24.65

TABLE IV. Column headings for Tables V–XIII

I	Instance number
n	Number of customers
m	Number of vehicles
t	Number of days (PVRP, PTSP) or depots (MDVRP)
D	Maximal duration of a route
Q	Vehicle capacity
$c(s^*)$	Solution cost using the standard algorithm
T^*	Time required to obtain a solution of cost $c(s^*)$ (in minutes)
T	Total computation time (in minutes)
$c(s^{**})$	Cost of the best solution identified during the sensitivity analyses
%	$100[c(s^*) - c(s^{**})]/c(s^{**})$

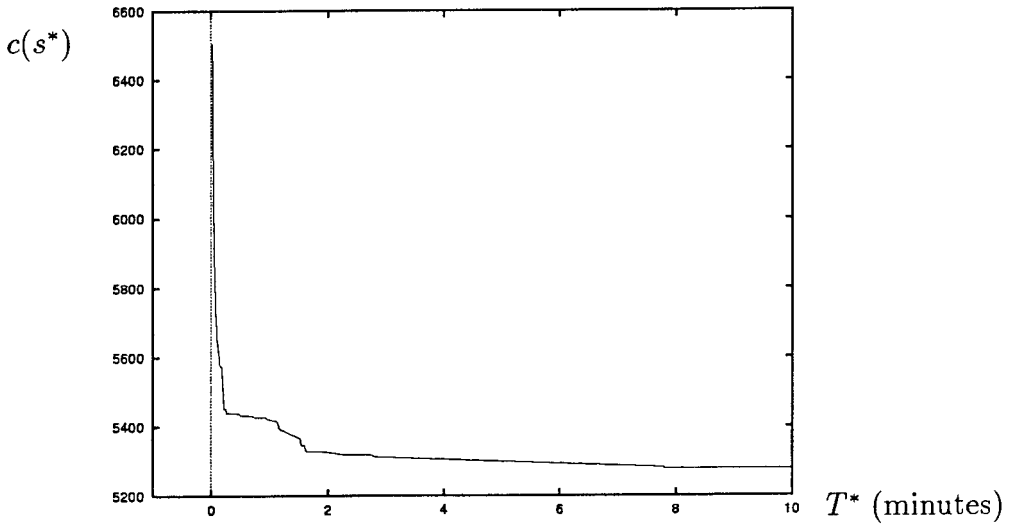

Fig. 2. Cost of best solution identified against running time.

3.3. Step-by-step Description of the Search Algorithm

The following algorithm can be used without modification to solve the PVRP, the PTSP, and the MDVRP. To ease its description, we provide in Table I a summary of the notations. The algorithm starts from a solution s, is initialized by the vector of parameters $(\delta, \gamma, \theta, \eta, p)$, and returns, after execution, the best feasible solution found s^*, if any. Assume that $c(s^*) = \infty$ if no feasible solution has yet been identified:

1. If s is feasible, set $s^* := s$. Set $\alpha := 1$ and $\beta := 1$.
2. For every (i, k, ℓ), do
   (a) Set $\rho_{ik\ell} := 0$ and $\tau_{ik\ell} := 0$.
   (b) If $(i, k, \ell) \in B(s)$ and s is feasible, set $\sigma_{ik\ell} := c(s)$; else, set $\sigma_{ik\ell} := \infty$.
3. For $\lambda = 1, \ldots, \eta$, do
   (a) Set $M(s) := \varnothing$.
   (b) For each $\bar{s} \in N(s)$, do
   - If there exists $(i, k, \ell) \in B(\bar{s}) \backslash B(s)$ such that $\tau_{ik\ell} < \lambda$ or such that $\bar{s}$ is feasible and $c(\bar{s}) < \sigma_{ik\ell}$, set $M(s) := M(s) \cup \{\bar{s}\}$.

   (c) For each $\bar{s} \in M(s)$, do
   - If $f(\bar{s}) \geq f(s)$, set $g(\bar{s}) := f(\bar{s}) + \gamma\sqrt{nmt}c(s) \times \Sigma_{(i,k,\ell)\in B(\bar{s})\backslash B(s)} \rho_{ik\ell}/\lambda$;
     else, set $g(\bar{s}) := f(\bar{s})$.

   (d) Identify a solution $s' \in M(s)$ minimizing $g(\bar{s})$.
   (e) For each attribute $(i, k, \ell) \in B(s) \backslash B(s')$, do

TABLE V. Results on the randomly generated PVRP instances

I	n	m	t	D	Q	$c(s^*)$	T^*	T	$c(s^{**})$	%
a	48	2	4	500	200	2234.23	0.89	3.86	2209.02	1.14
b	96	4	4	480	195	3836.49	8.74	9.79	3799.28	0.98
c	144	6	4	460	190	5277.62	7.82	18.62	5218.13	1.14
d	192	8	4	440	185	6072.67	24.31	28.24	6012.79	1.00
e	240	10	4	420	180	6769.80	35.29	36.62	6769.80	0.00
f	288	12	4	400	175	8462.37	49.65	50.97	8422.64	0.47
g	72	3	6	500	200	5000.90	1.50	9.63	4997.41	0.07
h	144	6	6	475	190	7183.39	22.63	24.05	7094.52	1.25
i	216	9	6	450	180	10507.34	32.10	45.65	10370.45	1.32
j	288	12	6	425	170	13629.25	67.59	71.56	13370.04	1.94

TABLE VI. Results of the randomly generated PTSP instances

I	n	t	$c(s^*)$	T^*	T	$c(s^{**})$	%
a	48	4	2068.46	0.09	3.69	2064.84	0.18
b	96	4	3293.50	9.84	11.83	3207.44	2.68
c	144	4	4106.72	2.21	24.42	4054.12	1.30
d	192	4	4661.97	38.04	41.31	4626.24	0.77
e	240	4	4698.83	59.52	65.66	4678.19	0.44
f	288	4	5699.96	37.83	92.48	5610.74	1.59
g	72	6	4453.15	6.57	9.26	4443.30	0.22
h	144	6	5405.40	19.83	29.72	5393.04	0.23
i	216	6	7469.73	62.56	65.77	7418.61	0.69
j	288	6	8493.74	106.20	113.72	8466.33	0.32

- Remove customer i from route k of day ℓ using the GENI heuristic.
- Set $\tau_{ik\ell} := \lambda + \theta$.

(f) For each attribute $(i, k, \ell) \in B(s')\backslash B(s)$, do

- Insert customer i in route k of day ℓ using the GENI heuristic.
- Set $\rho_{ik\ell} := \rho_{ik\ell} + 1$.

(g) If s' is feasible, do

- If $c(s') < c(s^*)$, set $s^* := s'$.
- For each $(i, k, \ell) \in B(s')$, set $\sigma_{ik\ell} := \min\{\sigma_{ik\ell}, c(s')\}$.

(h) If $q(s') = 0$, set $\alpha := \alpha/(1 + \delta)$; else, set $\alpha := \alpha(1 + \delta)$.

(i) If $d(s') = 0$, set $\beta := \beta/(1 + \delta)$; else, set $\beta := \beta(1 + \delta)$.

4. COMPUTATIONAL RESULTS

We present in this section sensitivity analyses performed on the five user-controlled parameters of the algorithm, as well as computational results on PVRP, PTSP, and MDVRP instances taken from the literature. Although our algorithm applies equally well to directed and undirected instances, all our tests are performed on the latter class. This enables direct comparisons with previous algorithms to be made.

4.1. Sensitivity Analyses

To tune the five parameters δ, γ, θ, η, and p, several new instances were randomly generated using the following procedure. Here, ϕ is a constant equal to 0.05, and n and t are given as input data:

1. Randomly generate t centers in the $[-50, 50]^2$ square according to a continuous uniform distribution.
2. Set $i := 1$.

TABLE VII. Results on the randomly generated MDVRP instances

I	n	m	t	D	Q	$c(s^*)$	T^*	T	$c(s^{**})$	%
a	48	2	4	500	200	861.32	0.15	4.04	861.32	0.00
b	96	4	4	480	195	1314.99	1.70	8.42	1307.61	0.56
c	144	6	4	460	190	1815.62	13.10	14.23	1806.60	0.50
d	192	8	4	440	185	2094.24	8.25	19.30	2072.52	1.05
e	240	10	4	420	180	2408.10	25.05	25.49	2385.77	0.94
f	288	12	4	400	175	2768.13	26.20	33.45	2723.27	1.65
g	72	3	6	500	200	1092.12	1.06	6.86	1089.56	0.23
h	144	6	6	475	190	1676.26	13.14	15.10	1666.60	0.58
i	216	9	6	450	180	2176.79	8.39	24.37	2153.10	1.10
j	288	12	6	425	170	3089.62	25.87	35.08	2921.85	5.74

TABLE VIII. Results on the PVRP instances

Instance					CB	TB	RG	CGW		CGL		
I	n	m	t	Q	$c(s^*)$	$c(s^*)$	$c(s^*)$	$c(s^*)$	T^a	$c(s^*)$	T^*	T^b
1	50	3	2	160	547.4		537.3	**524.6**	1.1	**524.61**	0.91	3.39
2	50	3	5	160	1443.1	1481.3	1355.4	1337.2	6.8	**1330.09**	0.60	4.06
3	50	1	5	160	546.7			**524.6**	0.6	**524.61**	3.18	3.73
4	75	6	5	140	843.9		867.8	860.9	7.7	**837.94**	4.74	5.19
5	75	1	10	140	2187.3	2192.5	2141.3	2089.0	5.4	**2061.36**	7.46	7.48
6	75	1	10	140	938.2			881.1	3.0	**840.30**	6.87	7.84
7	100	4	2	200	839.2		833.6	832.0	5.5	**829.37**	6.60	7.63
8	100	5	5	200	2151.3	2281.8	2108.3	2075.1	13.5	**2054.90**	8.97	10.70
9	100	1	8	200	875.0			829.9	4.6	**829.45**	6.70	10.03
10	100	4	5	200	1674.0	1833.7	1638.5	1633.2	14.7	**1629.96**	5.57	9.68
11	126	4	5	235	847.3	878.5	820.3	*791.3*[c]	205.4	**817.56**	9.72	14.17
12	163	3	5	140			1312.0	**1237.4**	11.9	1239.58	11.51	18.37
13	417	9	7	2000			3638.1	3629.8	33.7	**3602.76**	57.74	59.98
14	20	2	4	20				**954.8**	0.2	**954.81**	0.01	1.15
15	38	2	4	30				**1862.6**	0.5	**1862.63**	0.04	2.58
16	56	2	4	40				**2875.2**	0.3	**2875.24**	0.34	4.28
17	40	4	4	20				1614.4	5.3	**1597.75**	0.07	3.01
18	76	4	4	30				3217.7	11.1	**3159.22**	4.84	6.46
19	112	4	4	40				**4846.5**	60.6	4902.64	9.08	11.90
20	184	4	4	60				**8367.4**	150.5	**8367.40**	6.34	23.44
21	60	6	4	20				2216.1	0.1	**2184.04**	3.12	5.20
22	114	6	4	30				4436.4	13.6	**4307.19**	0.95	11.46
23	168	6	4	40				6769.0	70.0	**6620.50**	5.16	19.58
24	51	3	6	20				3773.0	3.3	**3704.11**	1.50	4.26
25	51	3	6	20				3826.0	0.3	**3781.38**	0.84	4.34
26	51	3	6	20				3834.0	1.1	**3795.32**	0.67	4.26
27	102	6	6	20				23401.6	2.0	**23017.45**	10.36	11.31
28	102	6	6	20				23105.1	2.9	**22569.40**	2.14	11.13
29	102	6	6	20				24248.2	1.1	**24012.92**	9.45	11.22
30	153	9	6	20				80982.1	4.5	**77179.33**	15.01	20.72
31	153	9	6	20				80279.1	5.9	**79382.35**	19.04	20.30
32	153	9	6	20				83838.7	3.4	**80908.95**	12.87	20.62

[a] Time in minutes on a Sun 4/370.
[b] Time in minutes on a Sun Sparcstation 10.
[c] Reported solution is infeasible.

3. While $i \leq n$, do

- Randomly generate a vertex v_i in the $[-100, 100]^2$ square according to a continuous uniform distribution and compute its distance d to the nearest center.
- Let u be a number randomly chosen in the $[0, 1]$ interval according to a continuous uniform distribution. If $u < e^{-\phi d}$, set $i := i + 1$. Otherwise, delete v_i.

The service duration d_i and demand q_i of each customer i are randomly and independently chosen according to a discrete uniform distribution on $[1, 25]$.

This algorithm generates instances featuring clusters of customers around a certain number of centers. Figure 1 illustrates the dispersion of the points for an instance with 144 customers and four centers represented by asterisks. The characteristics of the new instances for the PVRP are summarized in Table II. Instances for the PTSP are generated by setting $m = 1$. One also obtains instances for the MDVRP by multiplying m by $2/t$. In the case of

TABLE IX. Best solution values for the PVRP instances

I	CB	TB	RG	CGW	CGL
1	547.4		537.3	**524.6**	**524.61**
2	1443.1	1481.3	1355.4	**1322.9**	**1322.87**
3	546.7			**524.6**	**524.61**
4	843.9		867.8	840.2	**835.43**
5	2187.3	2192.5	2141.3	2046.2	**2027.99**
6	938.2			847.2	**836.37**
7	839.2		833.6	831.1	**826.14**
8	2151.3	2281.8	2108.3	2042.0	**2034.15**
9	875.0			828.3	**826.14**
10	1674.0	1833.7	1638.5	1611.9	**1595.84**
11	847.3	878.5	820.3	785.7	**779.29**
12			1312.0	1219.6	**1195.88**
13			3638.1	3538.0	**3511.62**
14				**954.8**	**954.81**
15				**1862.6**	**1862.63**
16				**2875.2**	**2875.24**
17				1614.4	**1597.75**
18				3217.7	**3147.24**
19				4846.5	**4834.34**
20				**8367.4**	**8367.40**
21				2216.1	**2184.04**
22				4436.4	**4271.11**
23				6769.0	**6602.59**
24				3773.0	**3687.46**
25				3826.0	**3777.15**
26				3834.0	**3795.33**
27				23401.6	**21956.46**
28				23105.1	**22934.71**
29				24248.2	**22909.36**
30				80982.1	**75016.58**
31				80279.1	**78179.89**
32				83838.7	**80479.20**

the MDVRP, the number of depots is equal to t and the location of depot j corresponds to the location of center j. In the case of the PVRP and the PTSP, the coordinates of the unique depot are equal to the average coordinates of the centers. The values of D and Q were determined experimentally in order to guarantee the feasibility of each instance. The service frequency is set as follows: For the instances generated with $t = 4$, we used $e_i = 4$ for $i = 1, \ldots, n/4$, $e_i = 2$ for $i = n/4 + 1, \ldots, n/2$, and $e_i = 1$ for $i = n/2 + 1, \ldots, n$. For the instances generated with $t = 6$, we used $e_i = 6$ for $i = 1, \ldots, n/4$, $e_i = 3$ for $i = n/4 + 1, \ldots, n/2$, $e_i = 2$ for $i = n/2 + 1, \ldots, 3n/4$, and $e_i = 1$ for $i = 3n/4 + 1, \ldots, n$.

All data sets and solutions referred to in this paper are available in Cordeau et al. [11] and on the Internet by following the instructions presented in the Appendix.

To evaluate various alternatives regarding the definition of the neighborhood and the diversification strategy, a preliminary version of the algorithm was tested on some PVRP instances from the literature. These tests have also enabled the identification of ranges of acceptable values for the different parameters. Based on these experiments and on results from the literature, we have first set $\theta := 7$. We have also set $\delta := 0.1$ to avoid modifying the values of α and β too drastically after each iteration. The value of p has been set equal to 4. Judging from the results obtained by Gendreau et al. [16], this value produces very good solutions within a reasonable amount of computer time. Finally, the value of η was set equal to 10,000. Sensitivity analyses on the parameters were performed sequentially, leaving the remaining parameters unchanged, and using the following order: γ, δ, θ, (η, p). The last two parameters were tested jointly. Our way of setting and testing parameters follows some of the guidelines stated by Barr et al. [1]. First, parameters are determined using instances different from our main set of test problems. Second, parameters remain the same in all tests: They are not tuned for each instance. Sensitivity analyses were carried out to assess the robustness of the various parameter settings. Parameter values were determined by using a sequential process as opposed to a statistical experimental design scheme. No claim is made that our parameter values are the best possible. It is conceivable that a different testing scheme would have produced different results. The algorithm was coded in C and all tests were run on a Sun Sparcstation 10. We now present the results of the analysis on each parameter.

4.1.1. Parameter γ

Having fixed the value of the other parameters, we ran tests on all instances using different values of γ in the interval $[0.001, 0.1]$. Using too small a value of γ does not produce the desired diversification effect and restricts the search to a small subset of solutions close to the initial solution. Using too large a value seems to overly bias the evaluation of the cost and it then becomes hard to distinguish between good and poor solutions. The cost of the solutions produced for a given instance seems rather insensitive to variations of γ in the interval $[0.005, 0.02]$. Repeated runs with different values belonging to that interval show a slight decrease in the variability of results when γ takes a value in the range from 0.01 to 0.02, with the most appropriate value being equal to 0.015. Tests conducted on the randomly generated PTSP and MDVRP instances have led to similar conclusions.

4.1.2. Parameter δ

Having set $\gamma := 0.015$, we then let parameter δ vary in the interval $[0.05, 5]$. The cost of the best solution identified

TABLE X. Results on the PTSP instances

Instance			CB	P	CGW		CGL		
I	n	t	$c(s^*)$	$c(s^*)$	$c(s^*)$	T^a	$c(s^*)$	T^*	T^b
1	50	2	487.9	456.9	442.1	0.03	**439.02**	0.01	4.04
2	50	5	1148.1	1128.2	**1106.7**	1.56	1111.93	0.44	4.57
3	50	5	546.8	524.7	474.0	0.27	**469.69**	2.31	4.45
4	75	2	587.5	577.8	**544.2**	0.76	556.21	6.34	7.33
5	75	5	1461.5	1428.9	1394.0	5.22	**1389.54**	4.35	8.72
6	75	10	948.3	863.1	657.3	0.66	**651.28**	0.76	9.91
7	100	2	712.3	667.6	662.4	1.75	**660.41**	11.34	12.32
8	100	5	1705.5	1665.1	1635.2	12.41	**1634.68**	2.84	14.14
9	100	8	893.7	820.0	735.3	0.69	**734.16**	3.37	13.48
10	100	5	1367.7	1277.6	1248.8	2.68	**1240.01**	4.95	14.13
11	65	4			**491.0**	0.46	**490.97**	0.02	5.60
12	87	4			**664.1**	1.30	**664.10**	0.04	8.88
13	109	4			**830.8**	2.43	**830.80**	0.08	12.46
14	131	4			**994.6**	3.05	**994.60**	0.13	17.63
15	153	4			**1157.1**	7.41	**1157.07**	0.18	24.06
16	48	4			726.8	0.20	**660.12**	0.02	3.88
17	66	4			**776.4**	0.20	**776.43**	0.02	6.18
18	84	4			**873.7**	1.87	**873.73**	0.02	9.17
19	102	4			974.6	1.52	**958.88**	3.65	12.85
20	120	4			1053.6	2.97	**1034.51**	0.06	16.87
21	77	4			1379.1	1.26	**1375.08**	0.47	7.29
22	154	4			4323.6	11.67	**4319.72**	3.96	25.20
23	231	4			8753.3	47.59	**8553.10**	41.26	56.31

[a] Time in minutes on a Sun 4/370.
[b] Time in minutes on a Sun Sparcstation 10.

during the search is not very sensitive to this parameter. However, it seems that using too small a value does not sufficiently influence the search. When the values of α and β are only slightly modified at each iteration, the search cannot reach solutions that would significantly violate the capacity or duration constraints. This reduces the desired effect of the diversification. On the other hand, using too large a value of δ modifies the values of α and β too drastically. As a result, the search is driven toward strongly infeasible solutions, and it also becomes difficult to examine a sequence of several consecutive infeasible solutions. The most appropriate value for δ seems to be around 0.5. This conclusion is also valid for the MDVRP. In the case of the PTSP, the parameter δ has obviously no effect.

4.1.3. Parameter θ

The initial value of $\theta := 7$ seems to suit most instances of each problem, but one also notices that the best value for θ grows slowly with instance size. For a given instance, similar results are obtained with values of θ in the interval $[5 \log_{10} n, 10 \log_{10} n]$. The value of θ has thus been set equal to $[7.5 \log_{10} n]$, where $[x]$ is the integer nearest to x. Variable tabu durations, proposed by Taillard [38] as a means of reducing cycling, and used in TABUROUTE [17], do not seem to improve the search in this application but rather result in increased variability. For example, no improvement in the average cost of solutions produced for a given instance is observed when using tabu tags chosen randomly in the interval $[5 \log_{10} n, 10 \log_{10} n]$. This is explained by the presence of a diversification term in the cost function of nonimproving solutions. Every cycle that appears during the course of the search must contain at least one iteration during which the search goes from the current solution to a solution of larger or equal cost. Now, the cost of the latter solution increases during each repetition of the cycle since the frequency of addition of some of its attributes increases. Eventually, another solution becomes more attractive and is selected—hence, breaking the cycle.

TABLE XI. Best-solution values for the PTSP instances

I	CB	P	CGW	CGL
1	487.9	456.9	437.3	**432.10**
2	1148.1	1128.2	1106.7	**1105.81**
3	546.8	524.7	469.2	**466.71**
4	587.5	577.8	553.1	**550.65**
5	1461.5	1428.9	1394.0	**1382.61**
6	948.3	863.1	655.2	**646.10**
7	712.3	667.6	662.4	**653.13**
8	1705.5	1665.1	1625.7	**1613.42**
9	893.7	820.0	735.3	**721.24**
10	1367.7	1277.6	1248.8	**1239.51**
11			**491.0**	**490.97**
12			**664.1**	**664.10**
13			**830.8**	**830.80**
14			**994.6**	**994.60**
15			**1157.1**	**1157.07**
16			726.8	**660.12**
17			**776.5**	**776.43**
18			**873.7**	**873.73**
19			974.6	**958.52**
20			1053.6	**1034.51**
21			1379.1	**1375.08**
22			4323.6	**4312.31**
23			8753.3	**8494.04**

4.1.4. Parameters η and p

The running time of the algorithm obviously depends on the number of iterations executed, but also on the value of the parameter p which controls the neighborhood size in the GENI heuristic. It is thus convenient to set these two parameters jointly. For a given number of iterations, it seems that a value of p inferior to 2 produces low-quality solutions while a value greater than 4 makes computation times excessive. The best compromise appears to be $p = 3$. Table III presents the cost of the solutions obtained for instance c when using different values of η and p determined in order to use a comparable computing budget. Column $c(s^*)$ indicates the cost of the best solution produced during the search, column T^* gives the time at which that solution was found, and column T gives the total computation time, in minutes, to execute all η iterations. The case $p = 0$ consists of executing only simple insertions and deletions. When the number of iterations is inferior to 10,000, the search is not sufficiently broad to ensure that a good solution will be found. On the other hand, a number of iterations larger than 20,000 does not bring much because a better solution is seldom found after that many iterations have been executed. Figure 2 shows the decrease in the cost of the best solution identified as a function of computation time for instance c of the PVRP, using $p = 3$ and $\eta = 15{,}000$.

4.1.5. Results on the Randomly Generated Instances

The ''standard algorithm'' is obtained by setting $\delta := 0.5$, $\gamma := 0.015$, $\theta := [7.5 \log_{10} n]$, $\eta := 15{,}000$, and $p := 3$. It was executed once on every instance of every problem. The meanings of the various headings used in Tables V–XIII are given in Table IV. Tables V–VII present the results obtained on each instance of the PVRP, the PTSP, and the MDVRP, respectively.

The best known solution s^{**} was identified by performing several runs using different parameter settings. It would therefore make no sense to report computing times associated with these values. These best-solution values do not as such help assess the quality of the algorithm. However, they can be used to assess the robustness of the standard algorithm: When the relative gap between $c(s^*)$ and $c(s^{**})$ is small, this is an indication that our current setting of parameters is good. Also, $c(s^{**})$ may be used as bench marks to test other algorithms. In most cases, the gap between $c(s^*)$ and $c(s^{**})$ is less than 2%. Also, note that the moment at which the best solution is identified is somewhat unrelated to the quality of the solution. In the case of the PTSP, computation time increases more rapidly as a function of the problem size. This is explained by the fact that each tour contains more customers, thus requiring more computer operations to update the neighborhood list.

4.2. Results on Instances from the Literature

The standard algorithm was tested on a variety of instances from the literature.

4.2.1. PVRP Instances

We first conducted tests on a set of 32 PVRP instances whose characteristics are summarized in Table VIII. In all cases, $D = \infty$. Instances 1–10 were introduced by Eilon et al. [13] for the VRP and the details of the adaptation to the PVRP are taken from Christofides and Beasley [8]. The data set for instance 11 was proposed by Russell and Igo [37]. Instance 12 was proposed by Cook and Russell [10] and the details for the PVRP are found in Russell and Gribbin [36]. The data for instance 13 also come from [36]. Instances 14–32 were introduced by Chao et al. [6] and the complete data sets are given in [4]. In the first instance, two vehicles are available on day 1 and three vehicles are available on day 2. To cope with this situation, we have added a fictitious customer, located at the depot, with a demand equal to the capacity of a vehicle, and requiring a visit on day 1. In instance 11, some customers visited daily have a demand that

TABLE XII. Results on the MDVRP instances

Instance					GJ	CGW		RLB		CGL		
I	*n*	*t*	*D*	*Q*	*c*(*s**)	*c*(*s**)	T^a	*c*(*s**)	T^b	*c*(*s**)	*T**	T^b
1	50	4	∞	80	593.2	582.4	1.1	**576.87**	3.2	**576.87**	0.25	3.24
2	50	4	∞	160	486.2	476.6	1.2	476.66	4.8	**473.87**	0.83	3.46
3	75	2	∞	140	652.4	**641.2**	1.8	645.14	5.8	645.15	0.29	5.66
4	100	2	∞	100	1066.7	1026.9	2.2	1016.13	11.4	**1006.66**	5.60	7.79
5	100	2	∞	200	778.9	756.6	2.4	754.20	12.8	**753.34**	1.32	8.21
6	100	3	∞	100	912.2	883.6	2.1	**876.50**	8.4	877.84	4.88	7.65
7	100	4	∞	100	939.5	898.5	4.8	897.86	6.8	**891.95**	7.12	7.71
8	249	2	310	500	4832.0	4511.6	24.1	4500.48	69.4	**4482.44**	19.59	25.43
9	249	3	310	500	4219.7	3950.9	20.9	3969.31	41.2	**3920.85**	6.99	26.73
10	249	4	310	500	3822.0	3815.6	7.2	3720.88	43.0	**3714.65**	16.40	25.50
11	249	5	310	500	3754.1	3733.0	16.7	3670.25	36.4	**3580.84**	17.34	25.91
12	80	2	∞	60		1327.3	2.8	**1318.95**	5.4	**1318.95**	0.01	5.57
13	80	2	200	60		1345.9	0.7	**1318.95**	4.8	**1318.95**	0.01	5.58
14	80	2	180	60		1372.5	1.3	1365.69	2.6	**1360.12**	0.35	5.44
15	160	4	∞	60		2610.3	2.3	2551.46	15.5	**2534.13**	6.59	14.06
16	160	4	200	60		2605.3	6.1	**2572.23**	11.1	**2572.23**	0.77	14.05
17	160	4	180	60		2816.6	6.5	2731.37	5.8	**2720.23**	8.99	13.70
18	240	6	∞	60		3877.4	8.6	3789.96	23.2	**3710.49**	18.27	24.85
19	240	6	200	60		3863.9	22.3	**3827.06**	22.0	**3827.06**	0.17	25.20
20	240	6	180	60		4272.0	14.6	4097.06	10.0	**4058.07**	18.79	24.72
21	360	9	∞	60		5791.5	78.5	5678.50	48.7	**5535.99**	29.77	48.16
22	360	9	200	60		5857.4	132.4	5718.00	33.5	**5716.01**	14.45	48.90
23	360	9	180	60		6494.6	24.4	6145.58	17.3	**6139.73**	21.61	47.86

[a] Time in minutes on a Sun 4/370.
[b] Time in minutes on a Sun Sparcstation 10.

fluctuates every day. To solve this instance, we replaced every such customer with a set of customers having the same location, each requiring a visit on a different day, and having a demand equal to the initial demand for the corresponding day. Also, according to the original data, customer 1 should be visited on day 6. Since this is the only customer requiring a visit that day, Christofides and Beasley [8] modified this instance to allow the delivery scheduled for that day to take place on any other day of the week. We have taken this into account by introducing an additional customer.

Results obtained by Christofides and Beasley (CB) [8], Tan and Beasley (TB) [39], Russell and Gribbin (RG) [36], and Chao et al. (CGW) [6] as well as by our tabu search method (CGL) are reported in Table VIII. As before, we report the time needed to obtain the solution and the total time, in columns *T** and *T*, respectively. The cost of the best solution is indicated in bold characters. These results show that, 24 times out of 32, our method identifies a solution of better quality than those found by existing methods. Among the eight other instances, there are six ties with CGW, and in each of these cases, the best known solution was found. It should be mentioned that the solution reported by Chao et al., for instance 11 is infeasible due to some errors in the data. The coordinates of customers 41–100 are erroneous according to the data printed in Chao [4]. A direct comparison of running times is difficult since the various methods were tested on different computers. Overall, however, it would seem that our tabu search algorithm requires more time than do other methods. It is also conceivable that given the same running time on the same machine other algorithms such as CGW may outperform CGL more frequently. The only way to perform a full comparison would be to run the different codes on the same instances under the same conditions. This comment also applies to the PVRP and the MDVRP. Excluding instance 13 which contains 417 customers and for which the computation time is close to 60 min, the computation time for all instances is quite reasonable and seems to grow linearly with the size of the instance. Using different parameter settings, the algorithm has sometimes led to the identifi-

TABLE XIII. Best-solution values for the MDVRP instances

I	GJ	CGW	RLB	CGL
1	593.2	**576.9**	**576.87**	**576.87**
2	486.2	474.6	**473.53**	**473.53**
3	652.4	**641.2**	**641.19**	**641.19**
4	1066.7	1012.0	1003.87	**1001.59**
5	778.9	756.5	750.26	**750.03**
6	912.2	879.1	**876.50**	**876.50**
7	939.5	893.8	892.58	**885.80**
8	4832.0	4511.6	4485.09	**4437.68**
9	4219.7	3950.9	3937.82	**3900.22**
10	3822.0	3727.1	3669.38	**3663.02**
11	3754.1	3670.2	3648.95	**3554.18**
12		1327.3	**1318.95**	**1318.95**
13		1345.9	**1318.95**	**1318.95**
14		1372.5	1365.69	**1360.12**
15		2610.3	2551.46	**2505.42**
16		2605.3	**2572.23**	**2572.23**
17		2816.6	2731.37	**2709.09**
18		3877.4	3781.04	**3702.85**
19		3863.9	**3827.06**	**3827.06**
20		4272.0	4097.06	**4058.07**
21		5791.5	5656.47	**5474.84**
22		5857.4	5718.00	**5702.16**
23		6494.6	6145.58	**6095.46**

cation of solutions of even better quality. These are reported in Table IX, accompanied by the cost of the best solutions produced by other methods. Most of these solutions were found by increasing the value of η, leaving the other parameters unchanged.

4.2.2. PTSP Instances

The algorithm was also tested on 23 PTSP instances whose characteristics are given in Table X. Instances 1–10 were introduced by Eilon et al. [13] for the VRP and the details of the adaptation for the PTSP are reported in Christofides and Beasley [8]. Instances 11–23 were introduced by Chao et al. [7] and the complete data sets were given in Chao [4].

It is worth noting that Chao et al. added the constraint that at least one customer must be visited each day. We dealt with this additional constraint by associating a large cost to empty tours and forbidding their presence in feasible solutions. The search can thus consider intermediate solutions in which no customer is visited on some of the days, but this is never the case in the final solution. Results obtained by Christofides and Beasley [8] (CB), Paletta [31] (P), and Chao et al. [7] (CGW) as well as by the proposed tabu search method (CGL) are reported in Table X and the best solutions identified by each method are reported in Table XI. As before, our method typically produces better solutions than do alternative algorithms.

4.2.3. MDVRP Instances

Finally, the algorithm was tested on 23 MDVRP instances whose characteristics are summarized in Table XII. Instances 1–7 were extracted from Christofides and Eilon [9]. Instances 8–11 were described in Gillett and Johnson [18]. Instances 12–23 were proposed by Chao et al. [5] and the complete data sets are given in [4].

Results obtained by Gillett and Johnson [18] (GJ), Chao et al. [5], (CGW), and Renaud et al. [34] (RLB) and by the proposed tabu search method (CGL) are reported in Table XII. Our method obtained a solution strictly better than those produced by all other methods 17 times out of 23. Among the six other instances, there are four ties with RLB, and the best known solution was found in each case. Also, in the two cases where our method did not produce the best solution, the deviation from the cost of the best known solution is less than 1%. Computation times used by the two tabu search heuristics are somewhat comparable. The costs of the best solutions found by each method are reported in Table XIII.

5. CONCLUSIONS

We proposed a tabu search algorithm for three variants of the classical vehicle routing problem: the PVRP, the PTSP, and the MDVRP. A first contribution of our work was to show that the MDVRP can be formulated as a special case of the PVRP, and it can thus be solved using the same methodology. A second contribution was to develop a simple tabu search heuristic capable of solving all problems under consideration. Contrary to several tabu search implementations described in the literature, our algorithm contains very few user-controlled parameters. The value of these parameters was set through sensitivity analyses on a number of randomly generated instances. Computational results on PVRP, PTSP, and MDVRP instances taken from the literature indicate that our algorithm outperforms all previously described heuristics for each of the three problems.

This research was partially supported by the Canadian Natural Sciences and Engineering Research Council under grants OGP0038816 and OGP0039682 and by the Quebec Government FCAR program. This support is gratefully acknowledged. Thanks are due to three anonymous referees for their valuable comments.

APPENDIX

Test data and solutions described in this paper can be obtained from the Internet at `ftp.crt.umontreal.ca` via anonymous `ftp`. Use ''anonymous'' as the login name and your complete e-mail address as the password. All data files reside in the directory `pub/users/gilbert/tabu`. You should also check the `README` file in this directory for a more detailed description of the various data files and formats.

REFERENCES

[1] R. S. Barr, B. L. Golden, J. P. Kelly, M. G. C. Resende, and W. R. Stewart, Jr., Designing and reporting on computational experiments with heuristic methods. *J. Heur.* **1** (1995) 9–32.

[2] E. J. Beltrami and L. D. Bodin, Networks and vehicle routing for municipal waste collection. *Networks* **4** (1974) 65–94.

[3] M. W. Carter, J. M. Farvolden, G. Laporte, and J. Xu, Solving an integrated logistics problem arising in grocery distribution. *INFOR* **34** (1996) 290–306.

[4] I. M. Chao, Algorithms and solutions to multi-level vehicle routing problems. PhD Thesis, Applied Mathematics Program, University of Maryland, College Park, MD (1993).

[5] I. M. Chao, B. L. Golden, and E. A. Wasil, A new heuristic for the multi-depot vehicle routing problem that improves upon best-known solutions. *Am. J. Math. Mgmt. Sci.* **13** (1993) 371–406.

[6] I. M. Chao, B. L. Golden, and E. A. Wasil, An improved heuristic for the period vehicle routing problem. *Networks* **26** (1995) 25–44.

[7] I. M. Chao, B. L. Golden, and E. A. Wasil, A new heuristic for the period traveling salesman problem. *Comput. Oper. Res.* **22** (1995) 553–565.

[8] N. Christofides and J. E. Beasley, The period routing problem. *Networks* **14** (1984) 237–256.

[9] N. Christofides and S. Eilon, An algorithm for the vehicle-dispatching problem. *Oper. Res. Q.* **20** (1969) 309–318.

[10] T. M. Cook and R. A. Russell, A simulation and statistical analysis of stochastic vehicle routing with timing constraints. *Decis. Sci.* **9** (1978) 673–687.

[11] J.-F. Cordeau, M. Gendreau, and G. Laporte, A tabu search heuristic for periodic and multi-depot vehicle routing problems. Technical Report 95-75, Center for Research on Transportation, Montréal (1995).

[12] G. Dueck, New optimization heuristics: The great deluge algorithm and the record-to-record travel. *J. Comput. Phys.* **104** (1993) 86–92.

[13] S. Eilon, C. D. T. Watson-Gandy, and N. Christofides, *Distribution Management: Mathematical Modelling and Practical Analysis.* Griffin, London (1971).

[14] M. L. Fisher, Vehicle routing. *Network Routing, Handbooks in Operations Research and Management Science* (M. O. Ball, T. L. Magnanti, C. L. Monma, and G. L. Nemhauser, Eds.). North-Holland, Amsterdam, **8** (1995) 1–33.

[15] M. Gaudioso and G. Paletta, A heuristic for the periodic vehicle routing problem. *Trans. Sci.* **26** (1992) 86–92.

[16] M. Gendreau, A. Hertz, and G. Laporte, New insertion and postoptimization procedures for the traveling salesman problem. *Oper. Res.* **40** (1992) 1086–1094.

[17] M. Gendreau, A. Hertz, and G. Laporte, A tabu search heuristic for the vehicle routing problem. *Mgmt. Sci.* **40** (1994) 1276–1290.

[18] B. E. Gillett and J. G. Johnson, Multi-terminal vehicle-dispatch algorithm. *Omega* **4** (1976) 711–718.

[19] B. E. Gillett and L. R. Miller, A heuristic algorithm for the vehicle-dispatch problem. *Oper. Res.* **21** (1974) 340–349.

[20] F. Glover, Future paths for integer programming and links to artificial intelligence. *Comput. Oper. Res.* **13** (1986) 533–549.

[21] F. Glover and M. Laguna, Tabu search. *Modern Heuristic Techniques for Combinatorial Problems* (C. R. Reeves, Ed.). Halsted Press, New York, (1993) 70–150.

[22] F. Glover, É. D. Taillard, and D. de Werra, A user's guide to tabu search. *Ann. Oper. Res.* **41** (1993) 3–28.

[23] B. L. Golden, T. L. Magnanti, and H. Q. Nguyen, Implementing vehicle routing algorithms. *Networks* **7** (1977) 113–148.

[24] B. L. Golden and E. A. Wasil, Computerized vehicle routing in the soft drink industry. *Oper. Res.* **35** (1987) 6–17.

[25] P. Hansen and B. Jaumard, Algorithms for the maximum satisfiability problem. *Computing* **44** (1990) 279–303.

[26] G. Laporte, The vehicle routing problem: An overview of exact and approximate algorithms. *Eur. J. Oper. Res.* **59** (1992) 345–358.

[27] G. Laporte, Y. Nobert, and D. Arpin, Optimal solutions to capacitated multi-depot vehicle routing problems. *Congress. Num.* **44** (1984) 283–292.

[28] G. Laporte, Y. Nobert, and S. Taillefer, Solving a family of multi-depot vehicle routing and location-routing problems. *Trans. Sci.* **22** (1988) 161–172.

[29] G. Laporte and I. H. Osman, Routing problems: A bibliography. *Ann. Oper. Res.* **61** (1995) 227–262.

[30] S. Lin, Computer solutions of the traveling salesman problem. *Bell Syst. Tech. J.* **44** (1965) 2245–2269.

[31] G. Paletta, A multiperiod traveling salesman problem: Heuristic algorithms. *Comput. Oper. Res.* **19** (1992) 789–795.

[32] O. M. Raft, A modular algorithm for an extended vehicle scheduling problem. *Eur. J. Oper. Res.* **11** (1982) 67–76.

[33] J. Renaud, F. F. Boctor, and G. Laporte, An improved petal heuristic for the vehicle routing problem. *J. Oper. Res. Soc.* **47** (1996) 329–336.

[34] J. Renaud, G. Laporte, and F. F. Boctor, A tabu search

heuristic for the multi-depot vehicle routing problem. *Comput. Oper. Res.* **23** (1996) 229–235.

[35] Y. Rochat and É. D. Taillard, Probabilistic diversification and intensification in local search for vehicle routing. *J. Heurist.* **1** (1995) 147–167.

[36] R. A. Russell and D. Gribbin, A multiphase approach to the period routing problem. *Networks* **21** (1991) 747–765.

[37] R. A. Russell and W. Igo, An assignment routing problem. *Networks* **9** (1979) 1–17.

[38] É. D. Taillard, Parallel iterative search methods for vehicle routing problems. *Networks* **23** (1993) 661–673.

[39] C. C. R. Tan and J. E. Beasley, A heuristic algorithm for the period vehicle routing problem. *Omega* **12** (1984) 497–504.

[40] F. A. Tillman, The multiple terminal delivery problem with probabilistic demands. *Trans. Sci.* **3** (1969) 192–204.

[41] F. A. Tillman and T. M. Cain, An upperbound algorithm for the single and multiple terminal delivery problem. *Mgmt. Sci.* **18** (1972) 664–682.

[42] F. A. Tillman and R. W. Hering, A study of look-ahead procedure for solving the multiterminal delivery problem. *Trans. Res.* **5** (1971) 225–229.

[43] A. Wren and A. Holliday, Computer scheduling of vehicles from one or more depots to a number of delivery points. *Oper. Res. Q.* **23** (1972) 333–344.