A Tabu Search Heuristic for the Vehicle Routing Problem

Michel Gendreau • Alain Hertz • Gilbert Laporte
Centre de recherche sur les transports Université de Montréal, C.P. 6128, succursale A, Montréal, Québec, Canada H3C 3J7
Département de mathématiques, École Polytechnique Fédérale de Lausanne, Ecublens, CH-1015 Lausanne, Switzerland

The purpose of this paper is to describe TABUROUTE, a new tabu search heuristic for the vehicle routing problem with capacity and route length restrictions. The algorithm considers a sequence of adjacent solutions obtained by repeatedly removing a vertex from its current route and reinserting it into another route. This is done by means of a generalized insertion procedure previously developed by the authors. During the course of the algorithm, infeasible solutions are allowed. Numerical tests on a set of benchmark problems indicate that tabu search outperforms the best existing heuristics, and TABUROUTE often produces the best known solutions.
(*Vehicle Routing Problem; Tabu Search; Generalized Insertion*)

1. Introduction

The purpose of this paper is to present TABUROUTE, a new heuristic for the following version of the *Vehicle Routing Problem* (VRP). Let $G = (V, A)$ be a directed graph where $V = \{v_0, v_1, \ldots, v_n\}$ is a vertex set, and $A = \{(v_i, v_j): i \neq j\}$ is an arc set. Vertex v_0 denotes a *depot* at which m identical vehicles are based, and the remaining vertices of V represent *cities*. The value of m is either fixed at some constant, or bounded above by $\bar{m}$. With every arc (v_i, v_j) is associated a nonnegative distance c_{ij}. (For the sake of simplicity, the terms "distances," "travel times," and "travel costs" will be used interchangeably.) The VRP consists of designing a set of least cost vehicle routes in such a way that

(a) every route starts and ends at the depot;

(b) every city of $V \setminus \{v_0\}$ is visited exactly once by exactly one vehicle, and

(c) some side constraints are satisfied.

We consider the following side constraints:

(d) With every city is associated a nonnegative demand q_i ($q_0 = 0$). The total demand of any vehicle route may not exceed the vehicle capacity Q.

(e) Every city v_i requires a service time δ_i ($\delta_0 = 0$). The total length of any route (travel plus service times) may not exceed a preset bound L.

In our version of the problem, vehicles bear no fixed cost, and their number is a decision variable.

The VRP lies at the heart of distribution management and has been extensively studied over the last three decades or so. (See the surveys by Christofides, Mingozzi, and Toth 1979, Bodin, Golden, Assad, and Ball 1983, Christofides 1985, Laporte and Nobert 1987, Golden and Assad 1988, and Laporte 1992). The VRP is a hard combinatorial problem, and to this day only relatively small VRP instances can be solved to optimality. Interesting exceptions are the problems solved to optimality by Fisher (1989), using minimum k-trees. We are mostly interested here in heuristic algorithms. Extending the scheme proposed by Christofides (1985), these algorithms can be broadly classified into four types: (1) *Constructive algorithms* (see, e.g., Clarke and Wright 1964, Mole and Jameson 1976, Desrochers and Verhoog 1989, Altinkemer and Gavish 1991); (2) *Two-phase algorithms* (see, e.g., Gillett and Miller 1974,

0025-1909/94/4010/1276$01.25

Christofides, Mingozzi, and Toth 1979, Fisher and Jaikumar 1981, Toth 1984); (3) *Incomplete optimization algorithms* (see, e.g., Christofides, Mingozzi, and Toth 1979); (4) *Improvement methods* (see, e.g., Stewart and Golden 1984; Harche and Raghavan 1991).

Metaheuristics such as *simulated annealing* and *tabu search* can be viewed as improvement methods. These are search schemes in which successive neighbors of a solution are examined, and the objective is allowed to deteriorate in order to avoid local minima. As a rule, these algorithms are designed to be open-ended and their running time, which can sometimes be quite large, is not a polynomial function of the size of the input data. Using an analogy with a material annealing process used in mechanics (Metropolis et al. 1953, Kirkpatrick, Gelatt, and Vecchi 1983), simulated annealing ensures that the probability of attaining a worse solution tends to zero as the number of iterations grows. Such a method was applied to the VRP by Osman (1991, 1993). *Tabu search* was proposed by Glover (1977) (see Glover 1989, 1990 and Glover, Taillard, and de Werra 1993 for recent overviews). Here, successive "neighbors" of a solution are examined and the best is selected. To prevent cycling, solutions that were recently examined are forbidden and inserted in a constantly updated *tabu list*. We are aware of a number of VRP algorithms based on this approach. One of the first attempts to apply tabu search to the VRP is due to Willard (1989). Here, the problem is first transformed into a TSP by replication of the depot, and the search is restricted to neighbor solutions that can be reached by means of 2-opt or 3-opt interchanges while satisfying the VRP constraints. In Pureza and França (1991), the search proceeds from one solution to the next by swapping vertices between two routes. Osman (1991, 1993) uses a combination of 2-opt moves, vertex reassignments to different routes, and vertex interchanges between routes. Another algorithm was developed by Semet and Taillard (1993) for the solution of a real-life VRP containing several features, and different from the version considered in this paper. Here the basic tabu move consists of moving a city from its current route into an alternative route. Finally, Taillard (1992) partitions the vertex set into clusters separately through vertex moves from one route to another. Clusters are updated throughout the algorithm. Note that in all these algorithms, a feasible solution is never allowed to become infeasible with respect to side constraints.

Our purpose is to describe a new tabu search procedure for the VRP. It differs from the implementations just described in several fundamental aspects. Our results show that the proposed algorithm is highly competitive on a set of benchmark problems. The remainder of this paper is organized as follows. The algorithm is presented in §2 and the computational results in §3. This is followed by the conclusion, in §4. We also provide, in an appendix, the best solutions obtained by our algorithm on the test problems.

2. Algorithm

This section contains a description of TABUROUTE followed by some comments. We use the following notation. A solution is a set S of m routes $R_1, \ldots, R_m$ where $m \in [1, \bar{m}]$, $R_r = (v_0, v_{r_1}, v_{r_2}, \ldots, v_0)$, and each vertex v_i $(i \geq 1)$ belongs to exactly one route. These routes may be feasible or infeasible with respect to the capacity and length constraints. For convenience, we write $v_i \in R_r$ if v_i is a component of R_r, and $(v_i, v_j) \in R_r$ if v_i and v_j are two consecutive vertices of R_r. With any feasible solution S, we associate the objective function

$$F_1(S) = \sum_r \sum_{(v_i,v_j)\in R_r} c_{ij}.$$

Also, with any solution S (feasible or not), we associate the objective

$$F_2(S) = F_1(S) + \alpha \sum_r \left[\left(\sum_{v_i\in R_r} q_i\right) - Q\right]^+ + \beta \sum_r \left[\left(\sum_{(v_i,v_j)\in R_r} c_{ij} + \sum_{v_i\in R_r} \delta_i\right) - L\right]^+,$$

where $[x]^+ = \max(0, x)$ and α, β are two positive parameters. If the solution is feasible, $F_1(S)$ and $F_2(S)$ coincide; otherwise, $F_2(S)$ incorporates two penalty terms for excess vehicle capacity and excess route duration. At any step of the algorithm, F_1^* and F_2^* denote respectively the lowest value of $F_1(S)$ and $F_2(S)$ so far encountered. Also, S^* is the best known feasible solution and $\tilde{S}^*$, the best known solution (feasible or not).

We first describe procedure SEARCH (P), central to TABUROUTE. This procedure attempts to improve upon a given solution S, using tabu search. It calls GENI and US, two heuristics developed by the authors for the TSP (Gendreau, Hertz, and Laporte 1992). The first, GENI, is a generalized insertion routine. It is less myopic but more powerful than standard insertion procedures in that a vertex may be inserted only into a route containing one of its closest neighbors, and every insertion is executed simultaneously with a local reoptimization of the current tour. US is a post-optimization procedure that successively removes and reinserts every vertex, using GENI. Again, US has produced highly satisfactory results on the TSP, better than Or-opt, for example. The combination of GENI and US yields a powerful two-phase heuristic for the TSP. SEARCH is governed by a vector of parameters

$$P = (W, q, p_1, p_2, \theta_{\min}, \theta_{\max}, g, h, n_{\max})$$

defined as follows:

W: a nonempty subset of $V \setminus \{v_0\}$ containing vertices that are allowed to be moved from their current route;

q: number of vertices of W that are candidate for reinsertion into another route;

p_1: the route in which vertex v is reinserted must contain at least one of its p_1 nearest neighbors;

p_2: neighborhood size used in GENI;

$\theta_{\min}$, $\theta_{\max}$: bounds on the number of iterations for which a move is declared tabu;

g: a scaling factor used to define an artificial objective function value;

h: the frequency at which updates of α and β are considered;

$n_{\max}$: maximum number of iterations during which the last step of the procedure is allowed to run without any improvement in the objective function.

PROCEDURE SEARCH (P)

Step 0 (*Initialization*). Set the iteration count $t := 1$; no move is tabu.

Step 1 (*Vertex selection*). Consider solution S and randomly select q cities from W.

Step 2 (*Evaluation of all candidate moves*). **Repeat the following procedure for all selected vertices v.**

Consider all potential moves of v from its current route R_r into another route R_s containing no city (if $m < \bar{m}$), or at least one of the p_1 nearest neighbors of v. Repeat the following operations for all candidate moves:

(a) Remove v from R_r and compute its insertion cost into R_s, using the GENI algorithm with parameter p_2, and determine the corresponding S'.

(b) If the move is tabu, it is disregarded unless S' is feasible and $F_1(S') < F_1^*$, or S' is infeasible and $F_2(S') < F_2^*$.

(c) Otherwise, S' is assigned a value $F(S')$ equal to $F_2(S')$ if $F_2(S') < F_2(S)$, or to $F_2(S') + \Delta_{\max}\sqrt{m}gf_v$ otherwise, where $\Delta_{\max}$ is the largest observed absolute difference between the values of $F_2(S)$ obtained at two successive iterations, and f_v is the number of times vertex v has been moved, divided by t.

Step 3 (*Identification of best move*). The candidate move yielding the least value of F and solution $\bar{S}$ is identified.

Step 4 (*Next solution*). The move identified in Step 3 is not necessarily implemented. It may indeed be advantageous to attempt to improve S by applying to each individual route of S the US post-optimization procedure described in Gendreau, Hertz, and Laporte (1992). Solution S is set equal to $\bar{S}$, unless the following three conditions are satisfied: (a) $F_2(\bar{S}) > F_2(S)$; (b) S is feasible; (c) US has not been used at iteration $t - 1$; in such a case S is obtained by applying the US post-optimization process.

Step 5 (*Update*). If the US procedure has not been used in Step 4 and vertex v has been moved from route R_r to route R_s ($s \neq r$), reinserting v into R_r is declared tabu until iteration $t + \theta$, where θ is an integer randomly selected in $[\theta_{\min}, \theta_{\max}]$. Set $t := t + 1$, update F_1^*, F_2^*, S^*, $\tilde{S}^*$, $\Delta_{\max}$, m and f_v.

Step 6 (*Penalty adjustment*). If t is a multiple of h, adjust α and β as follows. Check whether all previous h solutions were feasible with respect to vehicle capacity. If so, set $\alpha := \alpha/2$; if they were all infeasible, set $\alpha := 2\alpha$. Similarly, if all previous h solutions were feasible with respect to route length, set $\beta := \beta/2$; if they were all infeasible, set $\beta := 2\beta$.

Step 7 (*Termination check*). If F_1^* and F_2^* have not decreased for the last $n_{\max}$ iterations, stop. Otherwise, go to step 1. □

The main algorithm can now be described. At first, several tentative initial solutions are generated, SEARCH is applied to each of them for a limited number of iterations, and the most promising solution is selected as a starting point for TABUROUTE. Procedure SEARCH is then called twice with different values P_1

and P_2 of the parameters. The first call usually brings the most significant improvement to the initial solution, while the second call intensifies the search locally by concentrating on specific subsets of cities of the best known feasible solution if any, or of the best known infeasible solution otherwise.

ALGORITHM TABUROUTE

Step 0 (*Initialization*). Set $\alpha := \beta := 1$ and $F_1^* := \infty$. If vertices are described by two-dimensional coordinates, relabel them according to the angle they make with the depot and a horizontal line.

Step 1 (*First solution*). Repeat the following operations λ times, where λ is an input parameter.

(a) Randomly select a city v_i.

(b) Using the vertex sequence

$$(v_0, v_i, v_{i+1}, \ldots, v_n, v_1, \ldots, v_{i-1}),$$

construct a tour on all vertices by means of the GENIUS heuristic for the TSP (Gendreau, Hertz, and Laporte 1992).

(c) Starting with v_0, create at most $\bar{m}$ vehicle routes by following the tour: the first vehicle contains all cities starting from the first city on the tour and up to, but excluding, the first city v_i whose inclusion in the route would cause a violation of the capacity or maximal length constraint; this process is then repeated, starting from v_i, and until all cities have been included into routes (the solution is then feasible), or until $\bar{m} - 1$ vehicles have been used, in which case all remaining cities are assigned to vehicle $\bar{m}$ (the solution may then be infeasible). Let S be the solution (feasible or not) obtained through this process. Update F_1^*, F_2^*, S^* and $\tilde{S}^*$.

(d) Call SEARCH (P_1).

(e) If $F_1^* < \infty$, set $S := S^*$; otherwise, set $S := \tilde{S}^*$.

Step 2 (*Solution improvement*). Call SEARCH (P_2). If $F_1^* < \infty$, set $S := S^*$; otherwise, set $S := \tilde{S}^*$.

Step 3 (*Intensification*). Call SEARCH (P_3). If $F_1^* < \infty$, S^* is the best known feasible solution; otherwise, no feasible solution has been found.

Stop. □

We now comment on the choice of parameters used in SEARCH and TABUROUTE, and on a number of algorithmic aspects. As far as parameters are concerned, we have selected them independently of the test problems, relying as much as possible on theoretical considerations and on the experience developed by other researchers in the field of tabu search. In a limited number of cases where no firm basis existed for choosing the parameters, we have selected reasonable a priori values, and sensitivity analyses were then conducted on all test problems.

Step 2c of SEARCH contains a *diversification* strategy. Following Glover (1989), vertices that have been moved frequently are penalized by adding to the objective function of the candidate solution a term proportional to the absolute frequency of movement of the vertex v currently being moved. Taillard (1992) suggests using a constant equal to the product of three factors: (a) $\Delta_{\max}$, a factor equal to the absolute difference value between two successive values of the objective function, (b) the square root of the neighborhood size (shown later to be proportional to the number m of routes), (c) a scaling factor g equal to 0.01 in our implementation. As a rule, using too large a value of g lessens the likelihood of obtaining a good solution. Too low a value does not produce the desired diversification effect because the algorithm does not move away from the current solution. Post-optimality tests show that the algorithm is quite insensitive to g as long as it remains in the interval [0.005, 0.02].

The variable tabu list length (θ) used in Step 5 of SEARCH was also inspired from Taillard's work (1991). After extensive experiments on the application of tabu search to the quadratic assignment problem, this author concludes that the probability of obtaining a global optimum is increased in the case of a variable list length. Our implementation of random duration tabus differs from that proposed by Taillard since no tabu list is actually used. Instead, each move individually receives a random duration *tabu tag* denoted θ: this limits the amount of bookkeeping required and, as a result, the speed of the algorithm is increased. In the current implementation, we use $\theta_{\min} = 5$ and $\theta_{\max} = 10$, as suggested by Glover and Laguna (1993) for "simple dynamic tabu term rules."

The idea used in Step 6 of SEARCH of updating α and β during the course of the algorithm could also be applied to other contexts where penalty terms are added to the objective function. All too often, choosing an appropriate coefficient value is difficult, and a wrong choice can have an adverse impact on the performance of the algorithm. Here penalty coefficients are doubled if the $h = 10$ previous solutions were infeasible and halved if the $h = 10$ previous solutions were feasible. With this rule, we quickly arrive at values of α and β that produce a mix of feasible and infeasible solutions. We found the algorithm is not very sensitive to the value of h. Thus, solutions produced with $h = 5$ or 20 are at most 1% worse than those obtained with $h = 10$. In this type of algorithm, obtaining infeasible solutions is important since this helps moving out of local optima. Hertz (1992) uses this idea in the context of a course scheduling algorithm.

We now comment on algorithm TABUROUTE. The value currently used for λ, the number of tentative initial solutions, is equal to $[\sqrt{n}/2]$. Post-optimality tests indicate that it pays to use a value of λ greater than 1 and as large as $[\sqrt{n}/2]$, because the algorithm is then less likely to start on the wrong track. Values of λ larger than $[\sqrt{n}/2]$ were also tested, but as a rule the extra computational effort required is not justified by the quality of the results.

The idea of using a tour construction heuristic, as in Step 1c of TABUROUTE, has already been used by a number of researchers (see, e.g., Beasley 1983 and Haimovich and Rinnooy Kan 1985). Our implementation is different in that we resort to the more powerful GENIUS algorithm to obtain an initial tour. When the number of available vehicles is unbounded (i.e., $\bar{m} = n$), the initial solution is always feasible. However, for smaller values of $\bar{m}$, feasibility at this stage is not guaranteed, because the problem of finding a feasible solution to the capacity constrained VRP is a bin packing problem and is therefore NP-complete (Garey and Johnson 1979). Comparisons were made with a simplified version of the algorithm using random starting solutions. More precisely, for each solution 50 routes were initialized with a randomly selected seed, and the remaining vertices were then arbitrarily inserted into the existing routes. Results show that the final solution values obtained using the procedure described in Step 1 of TABUROUTE are approximately 1% better than those obtained from randomly generated routes.

We now discuss the choice of parameters W, q, p_1, p_2, and $n_{\max}$ in the various calls to SEARCH (P). Parameter W defines the subset of cities that can be moved into different routes in procedure SEARCH. This parameter is always equal to $V \setminus \{v_0\}$, except in the intensification step of TABUROUTE (Step 3), where W is defined as the $\lfloor |V|/2 \rfloor$ vertices v with the largest f_v; these vertices have often been moved and are therefore likely to yield a solution improvement if moved. In Step 3, the value of q is equal to $|W|$. In other words, all vertices that are allowed to move are candidates for reinsertion. In P_1 and P_2, however, it would be prohibitive to consider so many reinsertions. Here, q is chosen to ensure a sufficiently high probability of selecting at least one vertex from each route. This probability is $P(q, m) = S(q, m)m!/m^q$ (assuming the number of cities in each route is sufficiently large), where $S(q, m)$ is a Stirling number of the second kind (Riordan 1958). The most appropriate value of q depends on m; as long as $m \leq 30$, taking $q = 5m$ ensures that $P(q, m) \geq 0.9$. Parameter p_2 corresponds to the neighborhood size in GENI. Extensive tests performed by Gendreau, Hertz, and Laporte (1992) indicate that taking $p_2 = 5$ ensures that a near-optimal TSP solution will be found relatively quickly; this is the value used in P_1, P_2, and P_3. The algorithm is quite sensitive to the value of this parameter. Using $p_2 \leq 4$ tends to produce low quality solutions; in contrast, when $p_2 \geq 6$, running times become excessive.

Parameter p_1 is equal to max (k, p_2), where k is the number of cities in the route containing the vertex v currently being moved. This value of p_1 ensures that at least one potential move will relocate v into a different route. Finally, the value of $n_{\max}$ is equal to n in P_1, P_3, and to $50n$ in P_2, as the most important part of the search is executed in Step 2. The running time of the algorithm is linearly related to the value of this parameter in P_2. If $n_{\max}$ is too low, some good solutions will be missed. If it is too high, there is a risk that the algorithm will run a long time without improvement. Sensitivity analyses performed on all test problems suggest $50n$ is a good compromise.

3. Computational Results

TABUROUTE was tested on the fourteen test problems described in Christofides, Mingozzi, and Toth (1979). These problems contain between 50 and 199 cities in addition to the depot. Problems 1–5 and 11–12 have capacity restrictions only. Problems 6–10 are the same as 1–5, except that they have a route length constraint as well; problems 13–14 are also the same as 11–12, with a route length constraint. In problems 1–10, cities are randomly generated in the plane, while in problems 11–14, they appear in clusters. All computations were executed with distances rounded up or down after four decimals. The final solutions were evaluated with real distances, and the objective value was then rounded up or down after two decimals.

Comparisons were made between TABUROUTE and other heuristic algorithms for which computational results have already been published for the same problems:

- CW: the Clarke and Wright (1964) savings algorithm;
- MJ: the Mole and Jameson (1976) generalized savings algorithm;
- AG: the Altinkemer and Gavish (1991) PSA-T algorithm;
- DV: the Desrochers and Verhoog (1989) MBSA algorithm;
- GM: the Gillett and Miller (1974) SWEEP algorithm;
- CMT1: the Christofides, Mingozzi, and Toth (1979) two-phase algorithm;
- FJ: the Fisher and Jaikumar (1981) two-phase algorithm;
- CMT2: the Christofides, Mingozzi, and Toth (1979) incomplete tree search algorithm;
- OSA: Osman's (1993) simulated annealing algorithm;
- PF: the Pureza and França (1991) tabu search algorithm;
- OTS: Osman's (1993) tabu search algorithm;
- T: Taillard's (1992) tabu search algorithm.

Solution values for these algorithms are reported in Table 1. These values are extracted from the respective references except for CW, MJ, and GM, which are taken from Christofides, Mingozzi, and Toth (1979).

We report two sets of figures for TABUROUTE. The "standard" column contains results for a *single pass* of TABUROUTE, using the parameters described in §2. However, in the course of performing the various sensitivity analyses, we did on occasions produce better solutions; the corresponding local optima are reported in column "best." Asterisks correspond to the best verifiable solutions obtained with real c_{ij}s. The full solutions for the "best" column are reported in the appendix.

These results show that all "classical" algorithms (CW to CMT2 in Table 1) are clearly dominated by simulated annealing and tabu search, as far as solution values are concerned. TABUROUTE is highly competitive and generally produces the best known solutions. However, when analyzing results, care must be taken to make equitable comparisons. Thus, "TABUROUTE standard" executes a single pass with a priori parameters, while for other columns (e.g., AG, OSA, OTS, T, and "TABUROUTE best") the algorithm was run for several variants, and the best solution was selected. Similarly, parameters in some algorithms are undefined in the original article, and the rule for selecting seed points in FJ is not well specified. Another problem arises from the type of distances that were used. It must first be said that we did not generally possess the full solutions produced by the other algorithms, but only their value. This poses a number of difficulties. It is obvious that rounding or truncating must have occurred in the final solution value, on the individual route costs, or on the distances themselves since the reported optima are often integer while the original distances are real. As a result, the integer values reported in Table 1 may underestimate the true value to some extent. To our knowledge, only the columns OSA, OTS, T, and TABUROUTE correspond to verifiable solutions obtained with real c_{ij}s. The effect of rounding and truncating is best illustrated on problem 1. When this problem is solved with real c_{ij}s, a feasible solution of cost 524.61 is obtained. Recently, Hadjiconstantinou and Christofides (1993) have proved this result is optimal. Using rounded costs, TABUROUTE produces a solution of value 521, again a proven optimum (Cornuejols and Harche 1993). The same value is given by Fisher (1989) without specification of the rounding convention employed, and by other authors who worked with rounded distances (Harche and Raghavan 1991, Noon, Mittenthal, and Pillai 1991). Using truncated costs, we obtain an objective value of 508 with TABUROUTE. Another difficulty arises in problems with very tight route length

Table 1 **Comparison of TABUROUTE with Twelve Alternative Heuristics**

Problem	n	CW	MJ	AG	DV	GM	CMT1	FJ	CMT2	OSA	PF	OTS	T	TABUROUTE	
														Standard	Best
1	50	585	575	556	586	532	547	524	534	528	536	524.61*	524.61*	524.61*	524.61*
2	75	900	910	855	885	874	883	857	871	838.62	842	844	835.32*	835.77	835.32*
3	100	886	882	860	889	851	851	833	851	829.18	851	835	828.98	829.45	826.14*
4	150	1204	1259	1085	1133	1079	1093	1014	1064	1058	1081	1044.35	1029.64*	1036.16	1031.07
5	199	1540	1545	1351	1424	1389	1418	1420	1386	1378	—	1334.55	1300.89*	1322.65	1311.35
6	50	619	599	577	593	560	565	560	560	555.43*	560	555.43*	555.43*	555.43*	555.43*
7	75	976	969	939	963	933	969	916	924	909.68*	929	911	909.68*	913.23	909.68*
8	100	973	999	913	914	888	915	885	885	866.75	887	866.75	865.94*	865.94*	865.94*
9	150	1426	1289	1210	1292	1230	1245	1230	1217	1164.12	1227	1184	1164.24	1177.76	1162.89*
10	199	1800	1770	1464	1559	1518	1508	1518	1509	1417.85	—	1417.85	1403.21*	1418.51	1404.75
11	120	1079	1100	1047	1058	1266	1066	—	1092	1176	1049	1042.11*	1073.05	1073.47	1042.11*
12	100	831	879	834	828	937	827	824	816	826	826	819.59	819.56*	819.56*	819.56*
13	120	1634	1590	1551	1562	1770	1612	—	1608	1545.98	1631	1547	1550.15	1573.81	1545.93*
14	100	877	883	874	882	949	876	848	878	890	866	866.37*	866.37*	866.37*	866.37*

* Asterisks correspond to best verifiable solutions obtained with real c_{ij}s.

constraints, where some routes can be infeasible for TABUROUTE, but feasible when rounding or truncating occurs. We are aware of such cases where TABUROUTE would have achieved a much better value had we considered legal some routes with a length exceeding L by less than one unit. This type of problem has already been reported by Mole (1983) in relation with a vehicle scheduling algorithm by Cheshire, Malleson, and Naccache (1982). We also compared TABUROUTE with algorithms known to have been tested with truncated distances (Toth 1984) or with rounded distances (Harche and Raghavan 1991, Noon, Mittenthal, and Pillai 1991) by using the same type of distance. In each case, TABUROUTE produced better or identical results on all 14 problems.

The methodological problems just raised make direct computation time comparisons difficult, particularly when an unspecified number of passes of the same algorithm were executed with different parameters, or when inordinate computing times were allowed. In addition, at least one algorithm (T) uses parallel computing. By and large, metastrategies such as simulated annealing and tabu search require higher computation times than classical heuristics, but given the vast improvements in solution quality, we feel the extra computational effort is well justified. We report in Table 2 the computation times in minutes on a Silicon Graphics workstation, 36 MHz, 5.7Mflops, for the standard version of TABUROUTE. More specifically, we show the times required to compute the λ initial solutions, to reach the best encountered solution, and to terminate the algorithm. These results show that the relationship between problem size and computation time is not monotonous, and the moment at which the best solution is identified is quite unpredictable. Thus, in problems 4 and 5, it is encountered toward the end of the search process, while in problem 12, it is discovered at an early stage, during the initial trials.

4. Conclusion

We have described in this paper a new tabu search algorithm for the VRP. Results obtained on a series of benchmark problems indicate clearly that tabu search outperforms the best existing heuristics, and TABUROUTE often produces the best known solutions. By nature, tabu search is a metaheuristic that must be tai-

Table 2 Computation Times for the Standard Version of TABUROUTE

		Computation Times in Minutes		
Problem	Size	For the λ Initial Trials	To Obtain the Best Solution	Total
1	50	0.6	1.4	6.0
2	75	2.6	39.2	53.8
3	100	3.1	6.8	18.4
4	150	7.4	54.5	58.8
5	199	15.9	83.8	90.9
6	50	1.1	7.8	13.5
7	75	3.2	31.8	54.6
8	100	3.9	5.9	25.6
9	150	11.9	21.3	71.0
10	199	21.4	44.1	99.8
11	120	3.0	11.9	22.2
12	100	3.5	1.7	16.0
13	120	10.3	34.8	59.2
14	100	8.2	29.7	65.7

lored to the shape of the particular problem at hand. We attribute a large part of the success of TABUROUTE to at least two main implementation devices. One is the fact that we allow infeasible solutions through penalty terms in the objective function, thus reducing the likelihood of local minima. A second important ingredient of our method is the use of GENI to execute the insertions. Not only does this help produce better tours, but as a result, the solution is periodically perturbed and thus the risk of being trapped in a local optimum is again reduced. Finally, one major advantage of the proposed algorithm lies in its flexibility. It can be executed from any starting solution (feasible or not); it can also be adapted to contexts where the number of vehicles is fixed or bounded, where vehicles have different characteristics, etc. Also, additional features can easily be handled, such as assigning particular cities to specific vehicles, using several depots, allowing for primary and secondary routes, and so on.[1]

[1] This work was partially supported by the Canadian Natural Sciences and Engineering Research Council under grants OGP0038816, OGP0039682, and OGP0105384. The second author also benefitted from an NSERC International Fellowship. Thanks are also due to the referees for their valuable comments.

Appendix. Best Solutions Obtained with Taburoute (real distances)

The "Time" column shows travel times only. To obtain total route durations, service times must be added, where applicable.

Problem 1

Number of Cities							Route							$Q = 160$ Load	Time
10	0	38	9	30	34	50	16	21	29	2	11	0		159	99.33
11	0	32	1	22	20	35	36	3	28	31	26	8	0	149	118.52
9	0	27	48	23	7	43	24	25	14	6	0			152	98.45
9	0	18	13	41	40	19	42	17	4	47	0			157	109.06
11	0	12	37	44	15	45	33	39	10	49	5	46	0	160	99.25
$n = 50$															524.61

Problem 2

Number of Cities	Route												$Q = 140$ Load	Time
10	0	51	3	44	50	18	55	25	31	72	12	0	138	119.32
5	0	17	40	32	9	39	0						126	57.01
6	0	7	53	11	10	58	26	0					139	71.90
6	0	38	65	66	59	14	35	0					135	93.03
9	0	34	46	8	19	54	13	57	15	29	0		140	82.69
9	0	5	37	20	70	60	71	36	47	48	0		135	94.14
7	0	30	74	21	69	61	28	2	0				138	88.71
9	0	73	1	43	41	42	64	22	62	68	0		136	94.63
8	0	6	33	63	23	56	24	49	16	0			140	92.69
6	0	67	52	27	45	4	75	0					137	41.22
$n = 75$														835.32

Problem 3

Number of Cities	Route																	$Q = 200$ Load	Time
5	0	94	95	97	87	13	0											108	40.91
12	0	21	72	75	56	39	67	23	41	22	74	73	40	0				194	106.06
16	0	92	98	37	100	91	16	86	38	44	14	42	43	15	57	2	58	0	198	126.66
14	0	50	33	81	51	9	71	65	35	34	78	79	3	77	76	0			199	118.79
14	0	52	7	82	48	19	11	64	49	36	47	46	8	83	18	0			199	138.79
12	0	28	12	80	68	29	24	54	55	25	4	26	53	0					165	98.25
14	0	31	88	62	10	63	90	32	66	20	30	70	1	69	27	0			199	113.93
13	0	89	60	5	84	45	17	61	85	93	59	99	96	6	0				196	82.73
$n = 100$																			826.14

Problem 4

Number of Cities	Route																			$Q = 200$ Load	Time
13	0	138	48	112	7	61	114	99	43	86	97	69	23	57	0					196	120.33
15	0	32	1	120	80	28	31	82	140	113	26	8	60	81	27	46	0			199	88.22
11	0	11	100	2	83	131	20	59	3	101	51	77	0							195	74.55
16	0	119	22	70	116	121	115	36	85	35	84	128	29	129	53	127	126	0		200	120.62
13	0	78	16	118	130	50	21	79	74	34	104	9	62	38	0					192	75.74
13	0	90	10	54	106	73	117	89	39	75	105	30	49	76	0					200	110.25
18	0	137	44	107	65	93	92	42	64	88	40	94	19	141	150	148	142	147	17	0	197	119.46
14	0	63	37	52	15	45	91	72	33	125	124	122	123	71	5	0					197	72.73
10	0	144	145	109	87	135	143	4	149	146	47	0									200	51.96
11	0	110	18	55	134	67	13	136	41	66	111	56	0								199	83.96
13	0	139	68	133	14	58	25	95	96	24	98	132	6	102	0						199	90.86
3	0	103	108	12	0																61	22.38
$n = 150$																					1031.07

Problem 5

Number of Cities	Route																	Q = 200 Load	Time
13	0	158	184	190	41	90	143	89	137	142	114	156	93	86	0			200	103.02
14	0	175	176	8	102	178	78	19	70	128	123	13	83	153	111	0		195	81.39
15	0	66	196	191	1	136	197	199	43	42	68	113	91	141	22	186	0	184	76.61
13	0	57	189	131	80	10	77	165	38	119	129	169	50	168	0			197	97.41
11	0	95	97	161	9	110	25	56	118	72	147	181	0					200	81.82
11	0	26	100	150	108	69	180	132	7	51	149	60	0					196	68.90
13	0	157	2	101	28	64	94	140	121	82	173	21	172	139	0			200	65.43
7	0	112	194	193	33	96	61	6	0									170	35.67
12	0	16	67	159	182	49	74	144	145	24	107	63	117	0				193	79.65
13	0	81	71	52	11	170	164	85	134	84	14	133	177	35	0			197	112.60
11	0	54	30	48	47	155	36	122	174	171	120	152	0					197	73.46
11	0	125	98	45	58	27	179	99	167	65	46	34	0					197	53.48
3	0	127	87	4	0													61	20.80
13	0	29	79	15	154	124	20	166	138	37	88	103	5	59	0			199	96.16
14	0	106	73	18	146	135	92	148	163	31	162	75	39	109	12	0		200	127.50
14	0	188	104	183	23	116	62	185	115	160	192	198	53	195	105	0		200	83.89
11	0	126	17	76	40	130	187	32	44	55	3	151	0					200	53.57
n = 199																			1311.35

Problem 6

Number of Cities	Route											Q = 160 Load	L = 200 δ = 10 Time
10	0	32	11	16	29	21	50	34	30	9	38	0	141	95.33
10	0	12	37	44	15	45	33	39	10	49	5	0	155	99.12
8	0	14	25	13	41	40	19	42	17	0			131	109.94
9	0	6	23	24	43	7	26	8	48	27	0		133	100.64
9	0	2	20	35	36	3	28	31	22	1	0		137	108.08
4	0	18	4	47	46	0							80	42.33
n = 50														555.43

Problem 7

Number of Cities	Route										$Q = 140$ Load	$L = 160$ $\delta = 10$ Time
7	0	17	40	44	3	24	49	16	0		132	76.84
7	0	51	63	23	56	41	43	33	0		115	85.24
7	0	62	22	64	42	1	73	6	0		112	89.92
7	0	7	35	14	59	19	8	46	0		138	81.36
5	0	53	11	66	65	38	0				129	77.16
6	0	32	50	18	55	25	9	0			113	92.97
8	0	67	34	52	54	13	57	15	27	0	135	77.54
8	0	45	29	5	37	36	47	48	75	0	140	73.82
6	0	4	20	70	60	71	69	0			87	98.76
7	0	30	74	21	61	28	2	68	0		140	74.38
7	0	26	58	10	31	39	72	12	0		123	81.69
$n = 75$												909.68

Problem 8

Number of Cities	Route															$Q = 200$ Load	$L = 230$ $\delta = 10$ Time
13	0	53	40	21	73	72	74	75	22	41	15	57	2	58	0	157	83.10
9	0	54	55	25	39	67	23	56	4	26	0					153	107.08
11	0	99	61	16	86	38	44	14	43	42	87	13	0			191	111.40
12	0	94	95	97	92	98	37	100	91	85	93	59	96	0		199	59.35
11	0	18	82	48	47	36	49	64	11	19	7	52	0			178	117.55
11	0	27	69	70	30	32	90	63	10	62	88	31	0			155	90.12
10	0	89	60	83	8	46	45	17	84	5	6	0				93	89.16
11	0	50	33	81	9	35	71	65	66	20	51	1	0			163	117.93
12	0	12	80	68	24	29	34	78	79	3	77	76	28	0		169	90.26
$n = 100$																	865.94

Problem 9

Number of Cities								Route								$Q = 200$ Load	$L = 200$ $\delta = 10$ Time
11	0	138	12	109	134	24	25	55	130	54	149	26	0			154	82.39
10	0	5	84	17	113	86	140	38	14	100	95	0				173	99.74
9	0	51	103	71	136	65	66	20	122	1	0					119	108.57
13	0	28	80	150	68	121	29	129	79	3	77	116	76	111	0	188	69.64
12	0	52	106	7	123	19	107	11	62	148	88	127	27	0		169	74.60
11	0	18	82	124	46	45	125	8	114	83	60	118	0			168	88.80
10	0	50	102	33	81	120	9	135	35	34	78	0				166	91.04
12	0	53	40	21	73	74	133	22	41	145	115	2	58	0		142	64.60
10	0	105	110	4	139	39	67	23	56	75	72	0				193	96.13
12	0	31	10	108	90	32	131	128	30	70	101	69	132	0		168	79.99
11	0	13	117	97	42	142	43	15	57	144	87	137	0			129	78.33
9	0	146	89	147	6	96	104	99	94	112	0					133	42.01
8	0	48	47	36	143	49	64	63	126	0						135	112.65
12	0	92	37	98	91	119	44	141	16	61	85	93	59	0		198	74.40
$n = 150$																	1162.89

Problem 10

Number of Cities								Route									$Q = 200$ Load	$L = 200$ $\delta = 10$ Time
11	0	60	26	100	71	119	38	165	77	10	129	50	0				181	86.75
9	0	150	52	11	170	164	85	134	84	108	0						145	104.63
11	0	149	51	7	132	180	69	14	133	177	35	175	0				194	77.80
12	0	98	29	103	5	88	37	124	154	15	79	153	45	0			173	75.56
12	0	83	13	123	128	70	19	78	178	102	8	176	46	0			172	77.52
11	0	127	34	65	167	99	179	27	58	111	87	4	0				194	48.67
12	0	33	193	194	186	22	141	91	142	114	156	93	86	0			177	79.98
12	0	120	172	21	173	174	82	121	140	94	64	28	101	0			17	70.15
10	0	171	47	155	36	122	166	20	138	59	48	0					194	99.43
11	0	158	43	190	41	143	89	137	113	68	42	199	0				160	89.71
14	0	61	105	195	53	198	192	184	197	136	1	191	196	66	112	0	200	51.64
11	0	96	104	23	160	115	90	185	62	116	183	188	0				198	85.47
12	0	117	63	107	24	145	144	74	49	182	159	67	16	0			193	79.65
10	0	72	118	148	92	135	146	18	73	147	181	0					181	99.44
13	0	95	151	3	55	44	106	32	187	130	12	168	81	126	0		176	57.41
10	0	17	40	9	110	163	25	56	161	97	76	0					187	96.71
10	0	169	80	131	31	162	75	189	57	109	39	0					139	90.07
8	0	152	54	125	30	139	2	157	6	0							146	35.17
$n = 199$																		1404.75

Problem 11

Number of Cities	Route																			Q = 200 Load	Time
16	0	100	53	55	58	56	60	63	66	64	62	61	65	59	57	54	52	0		199	213.63
21	0	109	21	20	23	26	28	32	35	29	36	34	31	30	33	27	24	22	25		
	19	16	17	0																197	207.94
16	0	40	43	45	48	51	50	49	47	46	44	41	42	39	38	37	95	0		200	199.63
16	0	106	73	76	68	77	79	80	78	72	75	74	71	70	69	67	107	0		199	144.43
17	0	120	105	102	101	99	104	103	116	98	110	115	97	94	96	93	92	87	0	193	74.56
16	0	88	2	1	3	4	5	6	7	9	10	11	15	14	13	12	8	0		199	134.96
18	0	82	111	86	85	89	91	90	114	18	118	108	83	113	117	84	112	81	119		
	0																			188	66.96
n = 120																					1042.11

Problem 12

Number of Cities	Route															Q = 200 Load	Time
10	0	91	89	88	85	84	82	83	86	87	90	0					170	76.07
14	0	81	78	76	71	70	73	77	79	80	72	61	64	68	69	0	200	137.02
6	0	67	65	63	74	62	66	0									150	43.59
8	0	57	55	54	53	56	58	60	59	0							200	101.88
11	0	75	1	2	4	6	9	11	8	7	3	5	0				170	56.17
9	0	10	12	14	16	15	19	18	17	13	0						200	96.04
11	0	21	22	23	26	28	30	29	27	25	24	20	0				170	50.80
9	0	34	36	39	38	37	35	31	33	32	0						200	97.23
9	0	99	100	97	93	92	94	95	96	98	0						190	95.94
13	0	43	42	41	40	44	45	46	48	51	50	52	49	47	0		160	64.81
n = 100																		819.56

Problem 13

Number of Cities	Route														Q = 200 Load	L = 720 δ = 50 Time
12	0	113	83	2	1	3	4	5	6	109	114	90	91	0		138	113.85
12	0	18	118	108	8	12	13	14	15	11	10	9	7	0		153	117.70
10	0	21	20	26	23	25	24	22	19	16	17	0				123	170.53
10	0	29	32	35	36	34	33	30	27	31	28	0				61	195.35
10	0	38	39	42	47	50	49	46	44	41	37	0				115	183.65
10	0	40	43	45	48	51	65	61	57	54	52	0				143	218.37
10	0	53	55	58	56	60	63	66	64	62	59	0				125	207.08
11	0	73	71	74	72	75	78	80	79	77	76	68	0			141	136.49
12	0	120	70	69	67	98	110	115	97	94	93	96	95	0		144	118.66
13	0	88	82	111	86	87	92	89	85	112	84	117	81	119	0	146	45.80
10	0	102	101	99	100	116	103	104	107	106	105	0				86	38.45
n = 120																	1545.93

Problem 14

Number of Cities	Route												Q = 200 Load	L = 1040 δ = 90 Time
8	0	57	59	60	58	56	53	54	55	0			200	101.88
9	0	32	33	31	35	37	38	39	36	34	0		200	97.23
10	0	21	22	24	25	27	29	30	28	26	23	0	160	49.41
9	0	10	12	14	16	15	19	18	17	13	0		200	96.04
10	0	5	3	7	8	11	9	6	4	2	75	0	160	56.17
10	0	98	96	95	94	92	93	97	100	99	1	0	200	96.70
10	0	90	87	86	83	82	84	85	88	89	91	0	170	76.07
10	0	63	80	79	77	73	70	71	76	78	81	0	200	128.04
9	0	20	49	52	50	51	48	45	46	47	0		110	61.56
5	0	41	40	44	42	43	0						60	45.47
10	0	67	65	62	74	72	61	64	68	66	69	0	150	57.79
n = 100														866.37

References

Altinkemer, K. and B. Gavish, "Parallel Savings Based Heuristic for the Delivery Problem," *Oper. Res.*, 39 (1991), 456–469.

Beasley, J. E., "Route First–Cluster Second Methods for Vehicle Routing," *Omega*, 11 (1983), 403–408.

Bodin, L. D., B. L. Golden, A. A. Assad, and M. O. Ball, "Routing and Scheduling of Vehicles and Crews. The State of the Art," *Computers & Oper. Res.*, 10 (1983), 69–211.

Cheshire, I. M., A. M. Malleson, and P. F. Naccache, "A Dual Heuristic for Vehicle Scheduling," *J. Operational Res. Soc.*, 33 (1982), 51–61.

Christofides, N., "Vehicle Routing," *The Traveling Salesman Problem. A Guided Tour of Combinatorial Optimization*, E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, and D. B. Shmoys (Eds.), Wiley, Chichester, 1985, 431–448.

——, A. Mingozzi, and P. Toth, "The Vehicle Routing Problem," *Combinatorial Optimization*, N. Christofides, A. Mingozzi, P. Toth, and C. Sandi (Eds.), Wiley, Chichester, 1979, 315–338.

Clarke, G. and J. W. Wright, "Scheduling of Vehicles from a Central Depot to a Number of Delivery Points," *Oper. Res.*, 12 (1964), 568–581.

Cornuejols, G. and F. Harche, "Polyhedral Study of the Capacitated Vehicle Routing Problem," *Math. Prog.*, 60 (1993), 21–52.

Desrochers, M. and T. W. Verhoog, "A Matching Based Savings Algorithm for the Vehicle Routing Problem," Cahier du GERAD G-89-04. École des Hautes Études Commerciales de Montréal, 1989.

Fisher, M. L., "Optimal Solution of Vehicle Routing Problems and Minimum k-Trees," Report 89-12-13. Decision Sciences Department, The Wharton School, Philadelphia, PA, 1989.

—— and R. Jaikumar, "A Generalized Assignment Heuristic for Vehicle Routing," *Networks*, 11 (1981), 109–124.

Garey, M. R. and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freeman, San Francisco, 1979.

Gendreau, M., A. Hertz, and G. Laporte, "New Insertion and Post-Optimization Procedures for the Traveling Salesman Problem," *Oper. Res.*, 40 (1992), 1086–1094.

Gillett, B. and L. Miller, "A Heuristic Algorithm for the Vehicle Dispatch Problem," *Oper. Res.*, 22 (1974), 340–349.

Glover, F., "Heuristic for Integer Programming Using Surrogate Constraints," *Decision Sciences*, 8 (1977), 156–166.

——, "Tabu Search, Part I," *ORSA J. Computing*, 1 (1989), 190–206.

——, "Tabu Search, Part II," *ORSA J. Computing*, 2 (1990), 4–32.

—— and M. Laguna, "Tabu Search," *Modern Heuristic Techniques for Combinatorial Problems*, C. Reeves (Ed.), Blackwell Scientific Publications, Oxford, 1993, 70–150.

——, E. Taillard, and D. de Werra, "A User's Guide to Tabu Search," *Annals of Operations Research*, 41 (1993), 3–28.

Golden, B. L. and A. A. Assad, "*Vehicle Routing: Methods and Studies*," North-Holland, Amsterdam, 1988.

Hadjiconstantinou, E. and N. Christofides, "An Optimal Procedure for Solving Basic Vehicle Routing Problems," presented at the 35th Annual Conference of the Canadian Operational Research Society, Halifax, Canada, 1993.

Haimovich, M. and A. H. G. Rinnooy Kan, "Bounds and Heuristics for Capacitated Routing Problems," *Math. Oper. Res.*, 10 (1985), 527–542.

Harche, F. and P. Raghavan, "A Generalized Exchange Heuristic for the Capacitated Vehicle Problem," Working Paper, Stern School of Business, New York University, 1991.

Hertz, A., "Finding a Feasible Course Schedule Using Tabu Search," *Discrete Applied Math.*, 35 (1992), 255–270.

Kirkpatrick, S., Gelatt, C. D. Jr., and M. P. Vecchi, "Optimization by Simulated Annealing," *Science*, 220 (1983), 671–680.

Laporte, G., "The Vehicle Routing Problem: An Overview of Exact and Approximate Algorithms," *European J. Operational Res.*, 59 (1992), 345–358.

—— and Y. Nobert, "Exact Algorithms for the Vehicle Routing Problem," *Surveys in Combinatorial Optimization*, S. Martello, G. Laporte, M. Minoux and C. Ribeiro (Eds.), North-Holland, Amsterdam, 1987, 147–184.

Metropolis, N., A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, "Equation of State Calculations by Fast Computing Machines," *J. Chemical Physics*, 21 (1953), 1087–1091.

Mole, R. H., "The Curse of Unintended Rounding Error: A Case from the Vehicle Scheduling Literature," *J. Operational Res. Society*, 34 (1983), 607–613.

—— and S. R. Jameson, "A Sequential Route-Building Algorithm Employing a Generalised Savings Criterion," *Operational Res. Quarterly*, 27 (1976), 503–511.

Noon, C. E., J. Mittenthal, and R. Pillai, "A TSSP+1 Decomposition Approach for the Capacity-Constrained Vehicle Routing Problem," Working Paper, Management Science Program, The University of Tennessee, Knoxville, TN, 1991.

Or, I., "Traveling Salesman-Type Combinatorial Optimization Problems and Their Relation to the Logistics of Regional Blood Banking," Ph.D. Dissertation, Northwestern University, Evanston, IL, 1976.

Osman, I. H., "Metastrategy Simulated Annealing and Tabu Search Algorithms for Combinatorial Optimization Problems," Ph.D. Dissertation, The Management School, Imperial College, London, 1991.

——, "Metastrategy Simulated Annealing and Tabu Search Algorithms for the Vehicle Routing Problem," *Annals of Oper. Res.*, 41 (1993), 421–451.

Pureza, V. M. and P. M. França, "Vehicle Routing Problems via Tabu Search Metaheuristic," Publication CRT-747, Centre de recherche sur les transports, Montréal, 1991.

Riordan, J., *An Introduction to Combinatorial Analysis*, Wiley, New York, 1958.

Semet, F. and E. Taillard, "Solving Real-Life Vehicle Routing Problems Efficiently Using Taboo Search," *Annals of Oper. Res.*, 41 (1993), 469–488.

Stewart, W. R. Jr. and B. L. Golden, "A Lagrangian Relaxation Heuristic for Vehicle Routing," *European J. Operational Res.*, 15 (1984), 84–88.

Taillard, E., "Robust Taboo Search for the Quadratic Assignment Problem," *Parallel Computing*, 17 (1991), 433–445.

——, "Parallel Iterative Search Methods for Vehicle Routing Problems." Working Paper ORWP 92/03, Département de Mathématiques, École Polytechnique Fédérale de Lausanne, Switzerland, 1992.

Toth, P., "Heuristic Algorithms for the Vehicle Routing Problem," presented at the Workshop on Routing Problems, Hamburg, 1984.

Willard, J. A. G., "Vehicle Routing Using r-Optimal Tabu Search," M.Sc. Dissertation. The Management School, Imperial College, London, 1989.

Accepted by Awi Federgruen; received July 15, 1991. This paper has been with the authors $7\frac{1}{2}$ months for 2 revisions.