VEHICLE ROUTING WITH TIME WINDOWS

A. W. J. KOLEN, A. H. G. RINNOOY KAN and H. W. J. M. TRIENEKENS

Erasmus University, Rotterdam, The Netherlands

(Received December 1984; revisions received October 1985, June 1986; accepted August 1986)

In vehicle routing problems with time windows, a fixed fleet of vehicles of limited capacity is available at a depot to serve a set of clients with given demands. Each client must be visited within a given time window. We describe a branch-and-bound method that minimizes the total route length, and present some computational results.

A set of clients with known demands must be served by a fixed fleet of vehicles of limited capacity. All vehicles are located at a common depot. The vehicle routing problem (VRP) is to serve each client exactly once, so as to minimize the total route length.

The time-constrained vehicle routing problem (TCVRP) is an important generalization of the VRP. In the TCVRP, each client has a time window in which he must be served. Such windows arise, for instance, from traffic restrictions in city centers or from the special nature of the goods delivered by the vehicle.

The VRP represents an area of extensive research. The detailed survey compiled by Bodin et al. (1983) contains about 700 references, documenting optimization as well as approximation methods proposed to solve this problem or one of its many variations (including the important dial-a-ride problem). These references also bear witness to the many practical occurrences of the problem and to its economic significance. The TCVRP has received much less attention, in spite of the fact that time window constraints are very common in practice. Christofides, Mingozzi and Toth (1981a) and Baker (1983) proposed optimization methods for the single vehicle case (the time constrained traveling salesman problem). Desrochers and Soumis (1985) addressed the shortest path problem with time windows. Desrosiers, Dumas and Soumis (1984) describe a column generation approach to the time-constrained routing problem. Finally, Desrosiers, Sauve and Soumis (1985) proposed a Lagrangian approach. Heuristics for the VRP have been extended to the TCVRP (Solomon (1983, 1987) and Savelsbergh 1984). The analysis of their worst-case performance suggests that the TCVRP is fundamentally more difficult than the VRP. For example, it is not hard to see that finding a feasible solution to the TCVRP is itself an NP-complete problem (Savelsbergh).

The contribution of this paper is to describe a branch-and-bound method for the TCVRP. It is the first optimization method for this problem and was partially inspired by the optimization method for the VRP developed by Christofides, Mingozzi and Toth (1981c). Among other things, we use a different implementation of the idea of a state space relaxation. Computational experience with the method will be described as well.

1. The Branch-and-Bound Method

Let us suppose that client i for $i = 1, \ldots, n$ has demand q_i that must be satisfied within a time window $[a_i, b_i]$; u_i units of unloading time will be required. (Actually, u_i can be taken to be equal to 0 without loss of generality.) The distance and travel time between clients i and j are given by d_{ij} and t_{ij}, respectively. The depot is indexed by 0; it can be used in the period $[a_0, b_0]$ and houses m identical vehicles, each with an upper bound c on the total load carried. A vehicle route is feasible if it satisfies this capacity constraint as well as all time window constraints; if necessary, the vehicle can wait at a client to ensure feasibility of the next time window on the route. We are interested in a set of feasible routes through which each client is served exactly once, such that the total length of these routes is minimal.

Our description of the branch-and-bound method starts with the branching rule. Each node α in the search tree corresponds to a set $F(\alpha)$ of fixed routes starting and finishing at the depot, a partial route $P(\alpha)$ starting at the depot and a set $C(\alpha)$ of clients that are forbidden to be next on $P(\alpha)$. Initially, of course, $F(\alpha)$ and $C(\alpha)$ are empty and $P(\alpha)$ consists only of the depot.

Subject classification: 627 branch-and-bound, 837 routing with time windows.

0030-364X/87/3502-0266 $01.25

In a branching step, we first select a client j who does not appear in any fixed or partial route and is not forbidden. We do so according to a selection heuristic explained in the following paragraphs. We branch by creating two new nodes α' and α''. In α', the partial route $P(\alpha)$ is extended by j and $C(\alpha') = C(\alpha)$. In α'', $P(\alpha'') = P(\alpha)$ and $C(\alpha'') = C(\alpha) \cup \{j\}$. If $j = 0$, the extended partial route will be added to the set of fixed routes, and a new partial route is started ($P(\alpha) = (0)$, $C(\alpha) = \emptyset$).

In each node α of the search tree, the method will calculate a lower bound on all possible feasible extensions of the partial solution characterized by $F(\alpha)$, $P(\alpha)$ and $C(\alpha)$ by relaxing the condition that each client not yet on a route must be served exactly once. In fact, in the following sections we shall see how to compute the cheapest extension of the current partial solution to a set of routes so that the total load on these routes is equal to $Q = \sum_{i=1}^{n} q_i$ and so that each route has a different last client (i.e., the one visited directly before the return to the depot). In Section 4, we consider the extension that also requires that these routes do not contain 2-loops, i.e., subroutes in which the same client appears twice separated only by a single intermediate one. Larger loops with associated multiple visits to the same client will not be excluded.

Given the nature of the lower bound, the selection heuristic just referred to will be to select next the client with whom the partial route $P(\alpha)$ was extended in the lower bound calculation. If there is no partial route, ($P(\alpha) = \emptyset$), a new one is started by selecting the client who appears most frequently as the first client in the routes computed for the lower bound. Initially and whenever ties occur, preference is given to the client with the largest demand, since this choice reduces the size of the directed graphs appearing in the lower bound computations.

2. Computation of the Lower Bound

We now describe the computation of the lower bound in the root node; we discuss extension to arbitrary nodes at the end of this section.

To compute the lower bound, we construct a directed graph with vertices $v(i, q, k)$, for $i = 0, \ldots, n$, $q = 0, \ldots, Q$, and $k = 0, \ldots, m$, so that each directed path from $v(0, 0, 0)$ to $v(i, q, k)$ will correspond to a set of k routes with total load q, and with different last clients each one belonging to $\{1, \ldots, i\}$. The arc lengths in the directed graph will be defined so that the length of this directed path is equal to the total length of the corresponding routes. The lower bound is then given by taking the minimum over $k = 1, \ldots, m$ of the shortest-path lengths from $v(0, 0, 0)$ to $v(n, Q, k)$.

There are two ways to extend a set of k routes with total load q and last clients from $\{1, \ldots, i\}$ to a set whose last clients come from $\{1, \ldots, i + 1\}$. The first way is not to include client $i + 1$ at all. This approach is represented by a *type I* arc from $v(i, q, k)$ to $v(i + 1, q, k)$ of length 0. The second way is to take client $i + 1$ as the last client on a route of total load q'. This approach is represented by a *type II* arc from $v(i, q, k)$ to $v(i + 1, q + q', k + 1)$ of length $F(i + 1, q')$. Here, $F(i, q)$ is defined in general to be the minimum length of a feasible route with total load q and last client i; we will show how to compute these quantities in the next section.

To discuss the adjustment required in an arbitrary node α of the search tree, let us first assume that $P(\alpha) = \emptyset$. Let k^* be the number of fixed routes (i.e., $k^* = |F(\alpha)|$), let q^* be their total load and I^* be their set of clients (these can be eliminated from the lower bounding problem). The diagraph now has vertices $v(i, q, k)$ for $i = 0, \ldots, n - |I^*|$ (after renumbering), $q = q^*, \ldots, Q$ and $k = k^*, \ldots, m$. The lower bound is given by taking the minimum over $k = k^* + 1, \ldots, m$ of all shortest path lengths from $v(0, q^*, k^*)$ to $v(n - |I^*|, Q, k)$.

If $P(\alpha) \neq \emptyset$, then exactly one of the routes appearing in the lower bound must be an extension of $P(\alpha)$. Let $\bar{F}(i, q)$ be the minimum length of such an extension with total load q and last customer i; again, we will show in Section 3 how to compute $\bar{F}(i, q)$. The digraph is now expanded to contain vertices $v(i, q, k)$ and $\bar{v}(i, q, k)$ for $i = 0, \ldots, n - |I^*|$ (after renumbering); $q = q^*, \ldots, Q$ (q^* is still defined by $F(\alpha)$); $k = k^*, \ldots, m$ and the following arcs:

- type I arcs of length 0 from
  $v(i, q, k)$ to $v(i + 1, q, k)$ and from
  $\bar{v}(i, q, k)$ to $\bar{v}(i + 1, q, k)$;
- type II arcs of length $F(i + 1, q')$ from
  $v(i, q, k)$ to $v(i + 1, q + q', k + 1)$ and from
  $\bar{v}(i, q, k)$ to $\bar{v}(i + 1, q + q', k + 1)$;
- type III arcs of length $\bar{F}(i + 1, q')$ from
  $v(i, q, k)$ to $\bar{v}(i + 1, q + q', k + 1)$.

The latter type of arc corresponds to the addition of the single route extending $P(\alpha)$. The lower bound is given by taking the minimum over $k = k^* + 1, \ldots, m$ of the shortest-path lengths from $v(0, q^*, k^*)$ to $\bar{v}(n - |I^*|, Q, k)$.

3. Computation of the Shortest Routes

Let $p(i, q, T)$ denote a path with total load $q < c$ that is feasible with respect to the time windows and arrives at client i at time T; the clients on this path are not necessarily all different, and each time that a particular client occurs his demand is added to the current load. The shortest of all such paths is denoted by $p_*(i, q, T)$; its length is denoted by $l(i, q, T)$. Then $F(i, q)$, introduced in the previous section, is given by

$$F(i, q) = \min_{p_*(i,q,T)} \{l(i, q, T) + d_{i0} \mid T + u_i + t_{i0} \leq b_0\}. \quad (1)$$

In computing $F(i, q)$, we can actually restrict ourselves to a subset of feasible paths, namely, those without *needless waiting time*. This subset is characterized by the property that, if client j is visited directly after client i, then the arrival times T_j and T_i satisfy

$$T_j = \max\{a_j, T_i + u_i + t_{ij}\}. \quad (2)$$

It is easy to see that any path whose client arrival times T_i respect the time windows can be transformed in a path without needless waiting time, with client arrival times $T_i' \leq T_i$. Hence, from now on we assume that all paths $p(i, q, T)$ have this property.

Let us call $p(i(+), q(+), T(+))$ an extension of $p(i, q, T)$ if $i(+) \neq i$ and

$$q(+) = q + q_{i(+)} \leq c, \quad (3)$$

$$T(+) = \max\{a_{i(+)}, T + u_i + t_{ii(+)}\}, \quad (4)$$

$$T(+) + u_{i(+)} \leq b_{i(+)}. \quad (5)$$

In order to calculate $F(i, q)$, we shall construct a directed graph with vertices $w(i, q, T)$, so that the length of the shortest path from $w(0, 0, 0)$ to $w(i, q, T)$ is equal to $l(i, q, T)$.

This shortest path will be calculated by a labeling method, similar to Dijkstra's method, which will have to be described in some detail. In each iteration of this algorithm, we have a set of vertices $w(i, q, T)$ with a permanent or tentative label $L(i, q, T)$. A permanent label $L(i, q, T)$ will be equal to $l(i, q, T)$. A tentative label $L(i, q, T)$ will always have the property that it is equal to $\min\{l(i(-), q(-), T(-)) + d_{i(-)i}\}$, where the minimum is taken over all $p(i(-), q(-), T(-))$ for which $l(i(-), q(-), T(-))$ has been established and for which $p(i, q, T)$ is an extension. (It is equal to $+\infty$ if no such path $p(i(-), q(-), T(-))$ exists.)

In each iteration, the overall smallest tentative label, say $L(i, q, T)$, will be made permanent. We then consider all extensions $p(i(+), q(+), T(+))$ of $p(i, q, T)$, update $L(i(+), q(+), T(+))$ by

$$L(i(+), q(+), T(+)) := \min\{L(i(+), q(+), T(+)), l(i, q, t) + d_{ii(+)}\} \quad (6)$$

and proceed to the next iteration. It is easy to see that this procedure maintains the previous mentioned property of the label.

Note that throughout this scheme the vertices $w(i, q, T)$ can be created as we go along; the arc $(w(i, q, T), w(i(+), q(+), T(+)))$ of length $d_{ii(+)}$ is implicitly created in (6).

Since we are interested only in small $l(i, q, T)$ values (cf. (1)), extensions of $p(i, q, T)$ can be eliminated if, for some $T^* \leq T$, we have that $l(i, q, T^*) \leq l(i, q, T)$. The validity of this dominance rule can be easily verified by observing that, for every extension of $p(i, q, T)$, we can construct one of $p(i, q, T^*)$ of at least the same quality.

To calculate $\bar{F}(i, q)$, let us assume that the given partial $P(\alpha)$ route has total load q^* and arrives at client i^* at time T^*. We now simply start our calculations at $w(i^*, q^*, T^*)$ instead of at $w(0, 0, 0)$ by giving this vertex a permanent label equal to the length of $P(\alpha)$ and excluding clients in $C(\alpha)$.

4. Improvements in the Lower Bound

As in Christofides, Mingozzi and Toth (1981c), our lower bound can be improved substantially by considering only routes without 2-loops. Let $G(i, q)$ denote the minimum length of such a route with total load q and last client i. In this section, we shall show how $G(i, q)$ can be computed by an extension of the labeling algorithm. The adaptation to $\bar{G}(i, q)$ is similar to the one sketched in the last paragraph of the previous section.

Unlike the situation in Section 3, we can no longer claim that if $p_*(i(+), q(+), T(+))$ is an extension of a path $p_*(i, q, T)$, then the latter path must be optimal as well; the reason is that the second to last client on $p_*(i, q, T)$ (i.e., the client directly preceding i) can be equal to $i(+)$. Consequently, for each vertex, in addition to the true shortest-path length, which we will continue to denote by $l(i, q, T)$, we shall also have to know the length of the shortest path subject to the additional constraint that its second to last client is different from the second to last client on $p_*(i, q, T)$. This latter client will be denoted by $s(i, q, T)$.

Before describing a labeling procedure to find $G(i, q)$, let us first redefine an extension of $p(i, q, T)$ to be a path $p(i(+), q(+), T(+))$ for which (3), (4) and (5) hold and, in addition, the second to last

client on $p(i, q, T)$ is not equal to $i(+)$. In the labeling procedure, three labels will be associated to each vertex $w(i, q, T)$: $L(i, q, T)$, $M(i, q, T)$ and $S(i, q, T)$. Each can be permanent or tentative. Permanent labels $L(i, q, T)$ and $S(i, q, T)$ will be equal to $l(i, q, T)$ and $s(i, q, T)$ respectively. A permanent label $M(i, q, T)$ will be denoted by $m(i, q, T)$ and will correspond to the shortest-path length subject to its second to last customer being different from $s(i, q, T)$. A tentative label $L(i, q, T)$ will have the property that it is equal to the minimum of

$$\min\{l(i(-), q(-), T(-)) + d_{i(-)i}\}$$

and

$$\min\{m(i(-), q(-), T(-)) + d_{i(-)i}\}.$$

The latter minima are taken over all those paths $p(i(-), q(-), T(-))$ for which $l(i(-), q(-), T(-))$ and $m(i(-), q(-), T(-))$, respectively, have been established and for which $p(i, q, T)$ is an extension. If the minimum is attained by $(i(-), q(-), T(-))$, then $S(i, q, T)$ will be equal to $i(-)$. A tentative label $M(i, q, T)$ satisfies the same property as $L(i, q, T)$ under the additional constraint that its second to last customer is not equal to $S(i, q, T)$.

These properties again justify our turning the smallest tentative L, M-label into a permanent one in every iteration. In case of ties, we prefer L-labels to M-labels. To ensure that they continue to hold from one iteration to the next, however, requires a much more complicated updating schema.

We first consider the case that $L(i, q, T)$ (and simultaneously $S(i, q, T)$) are made permanent. If $p(i(+), q(+), T(+))$ is an extension of $p_*(i, q, T)$ (i.e., $i(+) \neq s(i, q, T)$), then we have to update the label $L(i(+), q(+), T(+))$ (see Equation 6) by

$$L(i(+), q(+), T(+)) := \min\{L(i(+), q(+), T(+)), l(i, q, T) + d_{ii(+)}\}. \quad (7)$$

There are two possibilities to be considered.

(a) If the minimum in the expression (7) is attained by $L(i(+), q(+), T(+))$, then it is still possible that $M(i(+), q(+), T(+))$ can be improved, provided that $S(i(+), q(+), T(+)) \neq i$:

$$M(i(+), q(+), T(+)) := \min\{M(i(+), q(+), T(+)), l(i, q, T) + d_{ii(+)}\}. \quad (8)$$

(b) If the minimum in the expression (7) is attained by $l(i, q, T) + d_{ii(+)}$ and $S(i(+), q(+), T(+)) \neq i$, then the old value of $L(i(+), q(+), T(+))$ will be the new value of $M(i(+), q(+), T(+))$ and $S(i(+), q(+), T(+)) := i$.

We now consider the case that $M(i, q, T)$ is made permanent. (Note that, since we prefer L-labels to M-labels and $M(i, q, T) \geq L(i, q, T)$, this situation implies that $L(i, q, T)$ and $S(i, q, T)$ have been made permanent earlier.) Let $p(i(+), q(+), T(+))$ be an extension of the path corresponding to $m(i, q, T)$ (i.e., the second to last customer $j(-)$ of the path corresponding to $m(i, q, T)$ is not equal to $i(+)$; it will turn out that we do not have to know $j(-)$).

We claim that an update is required only if $i(+) = s(i, q, T)$. For if $i(+) \neq s(i, q, T)$, then the path $p(i(+), q(+), T(+))$ is also an extension of $p_*(i, q, T)$, so that a possible update of $L(i(+), q(+), T(+))$ has been considered already. And an update of the tentative label $M(i(+), q(+), T(+))$ would be conceivable only if $S(i(+), q(+), T(+)) \neq i$; but in this case, (8) in conjunction with $m(i, q, T) \geq l(i, q, T)$ implies that again such an update will not be required.

If $i(+) = s(i, q, T)$, then, since $j(-) \neq i(+)$, the label $L(i(+), q(+), T(+))$ must be updated:

$$L(i(+), q(+), T(+)) := \min\{L(i(+), q(+), T(+)), m(i, q, T) + d_{ii(+)}\}, \quad (9)$$

and the same two possibilities arise again.

(a) If the minimum in the expression (9) is attained by $L(i(+), q(+), T(+))$, then, provided that the label $S(i(+), q(+), T(+)) \neq i$, $M(i(+), q(+), T(+))$ must be updated:

$$M(i(+), q(+), T(+)) := \min\{M(i(+), q(+), T(+), m(i, q, T) + d_{ii(+)}\}. \quad (10)$$

(b) If the minimum in the expression (9) is attained by $m(i, q, T) + d_{ii(+)}$ and $S(i(+), q(+), T(+)) \neq i$, then the old value of $L(i(+), q(+), T(+))$ will be the new value of $M(i(+), q(+), T(+))$ and $S(i(+), q(+), T(+)) := i$.

As in Section 3, various dominance rules can be used to speed up the calculations.

- If $m(i, q, T) \geq l(i, q, T^*)$ $(T \geq T^*)$ and $s(i, q, T) \neq s(i, q, T^*)$, then extensions of the path corresponding to $m(i, q, T)$ need not be considered (these extensions are relevant only if the next client is $s(i, q, T)$, in which case they are dominated by extensions of $p_*(i, q, T^*)$);
- If $l(i, q, T) \geq l(i, q, T^*)$ $(T \geq T^*)$ and $s(i, q, T) = s(i, q, T^*)$, then any extension of $p_*(i, q, T)$ is dominated by one of $p_*(i, q, T^*)$; if $s(i, q, T) \neq$

$s(i, q, T^*)$, then the only next client on $p_*(i, q, T)$ that need be considered is $s(i, q, T^*)$;

- If $m(i, q, T) \geq m(i, q, T^*)$ $(T \geq T^*)$, then any extension of the path corresponding to $m(i, q, T)$ is dominated by one corresponding to $m(i, q, T^*)$ or to $l(i, q, T^*)$);
- If $l(i, q, T) \geq m(i, q, T^*)$ $(T \geq T^*)$, then for similar reasons no extensions of $p_*(i, q, T)$ need be considered.

These dominance rules can be incorporated in the shortest-path calculations in an obvious fashion.

5. A Penalty Procedure

The optimal solution of the TCVRP is invariant under the transformation

$$d_{ij}^* = d_{ij} + r_i + r_j \qquad (i, j = 0, \ldots, n; r_0 = 0), \tag{11}$$

in that this transformation simply adds the constant term $2\sum_{i=0}^{n} r_i$ to the objective function. We can make use of this property by interpretating the r_i as penalties, depending on the degree δ_i of client i, i.e., twice the number of times that he is visited. If δ_i is larger (smaller) than 2, then a visit to client i should be made less (more) attractive. In iteration $p+1$, $r_i^{(p+1)}$ is chosen according to

$$r_i^{(p+1)} = r_i^{(p)} + \beta^{(p+1)} \cdot \frac{z_u - z^{(p)}}{(\sum_{j=1}^{n} (\delta_j^{(p)} - 2)^2)^{1/2}} \cdot (\delta_i^{(p)} - 2). \tag{12}$$

In this expression z_u is an upper bound on the optimal solution value and $z^{(p)}$ is the lower bound obtained in iteration p. In our algorithm we fix the number of steps of the penalty procedure in the root of the branch-and-bound tree as well as in all other nodes. Several pairs of parameters have been tested. Section 7 describes our computational experience.

Formula (12) is inspired by the subgradient step familiar from Lagrangian relaxation (see Fisher 1981). In accordance with empirical experience from that area, $\beta^{(p+1)}$ is chosen to satisfy $\beta^{(p+1)} - 2\beta^{(p)} + \beta^{(p-1)} = c$ for some is a constant c, $\beta^{(1)}$ is empirically chosen to be 0.2 in the original node of the branch-and-bound tree and 0.02 in all other nodes, and $\beta^{(s)} := 0$ where s is the given number of iterations of the penalty procedure.

In our algorithm we use the fact that distances are nonnegative. If some of the d_{ij}^* given by (11) are negative, then we adjust the corresponding penalties by

$$r_i := r_i - \tfrac{1}{2} \max\{\min_{j,k}\{d_{jk}^*\}, \min_j\{d_{j0}^*, d_{0j}^*\}\}. \tag{13}$$

Use of the penalty procedure greatly improved the lower bound.

6. An Upper Bound

As proved by Savelsbergh, finding a feasible solution to the TCVRP is itself an NP-complete problem. Hence, unlike the VRP, we should not expect a simple heuristic to produce feasible solutions that yield upper bounds on the optimal solution value. However, the following simple insertion rule was successful on all our test problems. Clients are inserted according to increasing value of

$$q_1(b_i - a_i) - q_2 q_i, \tag{14}$$

(q_1, q_2 constant) so that clients with small time windows and large demands get the highest priority. Starting with m empty routes, we determine for each successive client the amounts added to traveled distance and waiting time if this client is inserted in position k on route j. We would like both quantities,

Table I
Test Problems for the Computational Experiments

Problem	No. of Clients	No. of Time Windows	No. of Vehicles	Capacity of One Vehicle	Total Demand	Optimal Solution
A	10	2	4	24	93	244.200
B	6	5	3	3	6	119.000
C	10	3 + 1	3	30	87	365.735
D	12	12 + 1	4	50	182	314.000
E	14	11 + 1	3	45	126	291.109
F	15	15 + 1	3	50	135	303.196
G	10	7 + 1	3	4	10	656.707
H	12	12	3	15	38	911.121
I	14	14	4	20	77	873.351

Table II
Influence of the Penalty Procedure

(p_1, p_2)	50, 12	25, 6	10,3	10,10	10,20	100,12	200,12
A:	0.22[b]	0.13	0.44	1.16	1.43	0.43	1.30
	3[c]	3	37	21	15	3	3
	244.125[d]	244.134	242.201	242.201	242.201	244.162	244.186
244.200[a]							
B:	0.34	0.18	0.21	0.30	0.46	0.41	0.59
	41	25	55	41	41	41	55
	116.995	116.993	116.822	116.822	116.822	116.998	116.999
119.000							
C:	0.32	0.14	0.10	0.11	0.11	1.04	2.00
	1	1	3	3	3	1	1
	365.735	365.735	365.624	365.624	365.624	365.735	365.735
365.735							
D:	1.10	0.39	s.o[e]	s.o	s.o	2.16	4.24
	1	1				1	1
	314.000	314.000	311.933	311.933	311.933	314.000	314.000
314.000							
E:	0.33	0.45	s.o	s.o	s.o	1.36	4.37
	1	1				1	1
	291.109	291.109	284.454	284.454	284.454	291.109	291.109
291.109							
F:	2.05	1.11	s.o	s.o	s.o	4.14	8.00
	1	1				1	1
	303.196	303.196	297.007	297.007	297.007	303.196	303.196
303.196							
G:	1.43	0.53	1.18	1.14	1.22	1.49	2.26
	31	15	67	23	9	21	13
	647.399	647.478	644.387	644.387	644.387	647.483	647.565
303.196							
H:	3.45	2.48	2.10	4.10	8.53	4.33	5.58
	21	37	37	41	75	19	35
	898.571	894.315	876.895	876.895	876.895	897.872	898.913
911.121							
I:	0.58	s.o	s.o	s.o	s.o	2.04	3.55
	1					1	1
	873.351	872.282	715.494	715.494	715.494	873.351	873.351
873.351							

[a] Optimal solution value.
[b] CPU time in minutes (VAX 11/785).
[c] Number of nodes in the branch-and-bound tree.
[d] Initial lower bound.
[e] Stack overflow.

D_{jk} and W_{jk} respectively, to be small. Hence, we choose the (j, k) combination that minimizes

$$q_3 D_{jk} + q_4 W_{jk}, \tag{15}$$

(q_3, q_4 constant).

The constants q_1, q_2, q_3 and q_4 are set and reset interactively by the programmer. If the heuristic fails, then a trivial upper bound is, for instance, given by

$$m \cdot \max_i\{d_{i0}\} + \sum_{j=1}^{n} \max_k\{d_{kj}\}. \tag{16}$$

7. Computational Results

We investigated the computational performance of the branch-and-bound algorithm just described, using nine test problems that ranged from $n = 6$ to $n = 15$. In each problem, the travel time between two clients is proportional to the distance between them.

Problem A was derived from Christofides, Mingozzi and Toth (1981c) by adding time windows for two of the clients. These time windows were chosen so as to make the optimal solution of the problem without

time constraints infeasible in the time constrained problem. Problems B, C, D, E, F are extracted from examples in Eilon, Watson-Gandy and Christofides (1971); again we added time windows for some or all of the clients in the previously described manner.

The last three problems were random test problems. Using a two-dimensional uniform distribution, we generated points in the plane. The distance between two clients was defined to be the euclidean distance in the plane. We then constructed a few arbitrary routes feasible with respect to the capacity constraints, in which each client was visited exactly once. Finally, we constructed the time windows in such a way that these routes would be feasible in the time-constrained problem. While solving these random test problems, we discovered that these routes were not the optimal ones.

Table I contains the specific data for each test problem. A "+1" in the column with the number of time windows means that the depot also had a time window. Except for the first three problems, none of the problems could be solved without time windows, due to stack overflow during the computation of the lower bound. All test problems are available on request.

In Table II we show the performance of the algorithm excluding 2-loops for different pairs of parameters $p1$ and $p2$ for the penalty procedure; $p1$ corresponds to the number of iterations during the computation of the lower bound in the root of the branch-and-bound tree, and $p2$ corresponds to the number of iterations in all other nodes. The first number in the table indicates the CPU time in minutes on the VAX 11/785 using the VAX-Pascal compiler with optimization but also with run-time checks during execution. The second number is the number of nodes in the branch-and-bound tree generated by the algorithm, and the last number indicates the value of the initial lower bound found.

Table III specifies the performance of the algorithm with and without 2-loops for $p1 = 25$ and $p2 = 6$; the value with 2-loops allowed is given between parentheses. As can be seen from the table, computations for the last five problems could not be completed when 2-loops were allowed.

These and other experiments revealed that the relative width and number of time windows were the problem parameters that had the most significant influence on the running time of the algorithm. As the time windows (including the one for the depot) become larger, or as the number of time windows becomes smaller, the number of feasible q-routes increases with a corresponding increase in running time of the algorithm and a decrease in the quality of the lower bounds. Neither the number of vehicles m nor their capacity c had anywhere near the same influence on the computational results.

Table III
Influence of 2-Loop Elimination

	Optimal Solution Value	Heuristic	Initial Lowerbound	CPU Time	No. of Nodes
A	244.200	275.100	244.134 (243.930)[a]	0.15 (2.00)[a]	3 (3)[a]
B	119.000	119.000	116.993 (109.877)	0.12 (0.25)	25 (85)
C	365.735	469.628	365.735 (361.008)	0.18 (0.31)	1 (59)
D	314.000	372.000	314.000 (314.000)	0.36 (0.43)	1 (1)
E	291.109	312.574	291.109 (s.o)[b]	0.56 (s.o)	1
F	303.196	430.239	303.196 (s.o)	1.32 (s.o)	1
G	656.707	725.495	647.478 (592.569)	0.44 (>6.00)	15
H	911.121	962.704	894.315) (753.447)	2.33 (>5.00)	37 >22
I	873.351	1129.607	872.281 (818.758)	1.05 (>6.00)	5 >3

[a] The performance of the algorithm with 2-loops allowed.
[b] Stack overflow.

Acknowledgment

The authors gratefully acknowledge constructive suggestions from Jacques Desrosiers. Professor Rinnooy Kan was partially supported by a NATO Senior Scientist Fellowship.

References

BAKER, E. K. 1983. An Exact Algorithm for the Time-Constrained Traveling Salesman Problem. *Opns. Res.* **31,** 938–945

BODIN, L., B. GOLDEN, A. ASSAD AND M. BALL. 1983. Routing and Scheduling of Vehicles and Crews: The State of the Art. *Comput. Opns. Res.* **10,** 69–211.

CHRISTOFIDES, N., A. MINGOZZI AND P. TOTH. 1981a. An Algorithm for the Time Constrained Travelling Salesman Problem. Technical Report, Imperial College, London.

CHRISTOFIDES, N., A. MINGOZZI AND P. TOTH. 1981b. State-Space Relaxation Procedures for the Computation of Bounds to Routing Problems. *Networks* **11,** 145–164.

CHRISTOFIDES, N., A. MINGOZZI AND P. TOTH. 1981c. Exact Algorithms for the Vehicle Routing Problem. *Math. Program.* **20,** 255–282.

DESROCHERS, M., AND F. SOUMIS. 1985. A Generalized Permanent Labelling Algorithm for the Shortest Path Problem with Time Windows. Technical Report 394A, Centre de Recherche sur les Transports, Montréal.

DESROSIERS, J., Y. DUMAS AND F. SOUMIS. 1984. The Multiple Vehicles Many to Many Routing Problem with Time Windows. Technical Report 684-13, Ecole des Hautes Etudes Commerciales, Montréal.

DESROSIERS, J., M. SAUVE AND F. SOUMIS. 1985. Lagrangean Methods for Solving the Minimum Fleet Size m-TSP with Time Windows. Technical Report 396, Centre de Recherche sur les Transports, Montréal.

EILON, S., C. WATSON-GANDY AND N. CHRISTOFIDES. 1971. *Distribution Management: Mathematical Modelling and Practical Analysis.* Griffin, London.

FISHER, M. L. 1981. Langrangian Relaxation Methods for Solving Integer Programming Problems. *Mgmt. Sci.* **27,** 1–18.

SAVELSBERGH, M. 1984. Local Search in Routing Problems with Time Windows. Report 05-R 8409, Center for Mathematics and Computer Science, Amsterdam.

SOLOMON, M. 1983. On the Worst-Case Performance of Some Heuristics for the Vehicle Routing and Scheduling Problem with Time Window Constraints. Report 83-05-03, Department of Decision Sciences, The Wharton School, University of Pennsylvania.

SOLOMON, M. 1987. Algorithms for the Vehicle Routing and Scheduling Problems with Time Window Constraints. *Opns. Res.* **35,** 254–265.