Discrete Applied Mathematics 18 (1987) 185–197
North-Holland

GENERALIZED TRAVELLING SALESMAN PROBLEM THROUGH n SETS OF NODES: THE ASYMMETRICAL CASE*

Gilbert LAPORTE, Hélène MERCURE
Ecole des Hautes Etudes Commerciales de Montréal, 5255 avenue Decelles, Montréal, Québec H3T 1V6, Canada

Yves NOBERT
Département des Sciences administratives, Université du Québec à Montréal, 1495, rue Saint-Denis, Montréal, Québec H3C 3P8, Canada

Received 6 March 1987

This paper presents an exact algorithm for a generalized version of the Travelling Salesman Problem which consists of finding the shortest Hamiltonian circuit through n clusters of nodes, in the case where the distance matrix is asymmetrical. The problem is formulated as an integer linear program. The program is then relaxed and solved by a branch and bound algorithm. Computational results are reported for problems involving up to 100 nodes and 8 clusters.

1. Introduction

In Operations Research, the travelling salesman problem (TSP) lies at the heart of several distribution management problems and has thus received much attention in the last thirty years. The fact that the TSP and many of its extensions are NP-complete [2] means that in practice, exact solutions will be obtained for only modest problem sizes. Much effort has been devoted recently to the derivation of optimal solutions for problems of realistic dimensions and characteristics. This paper describes an exact solution method for a TSP extension encountered in several practical situations. We first provide a formal statement of the TSP and of the extension considered.

Consider a graph $G=(N,A,C)$ where $N=\{1,\ldots,r\}$ is a set of nodes or cities, $A=N^2$ is a set of arcs and $C=(c_{ij})$ is a matrix of distances associated with A. C is symmetrical if and only if $c_{ij}=c_{ji}$ for all $i,j\in N$; it satisfies the triangle inequality if and only if $c_{ik}+c_{kj}\geq c_{ij}$ for all $i,j,k\in N$. For the sake of convenience, problems in which C satisfies the triangle inequality will be referred to as *Euclidean problems*.

The *travelling salesman problem* consists of determining the shortest route passing through each node once and only once.

*The authors are grateful to the Canadian Natural Sciences and Engineering Research Council (grants A4747 and A5486) and to the Quebec Government (F.C.A.C. grant 80EQ0428) for their financial support. Thanks are also due to Giorgio Carpaneto and Paolo Toth who made their TSP code available to the authors.

In the *generalized travelling salesman problem* (GTSP), it is assumed that N is the union of n clusters or sets of nodes S_l. The GTSP consists of determining the shortest route passing through each cluster at least once and through each node at most once. If $|S_l| = 1$ for $l = 1, \ldots, n$, the GTSP reduces to the TSP.

The GTSP was first mentioned in the O.R. literature in the late sixties and early seventies in relation to the optimal sequencing of computer files [4] and the scheduling of clients through welfare agencies [11]. Lately, two other applications came to the attention of the authors. Both are related to the optimal routing of vans used to empty post boxes in an urban setting.

The first of these applications was described by Bovet [1]. By performing statistical analyses of mail volumes, it is possible to determine where approximately to locate post boxes in a city in order to provide a good service to customers. These locations need not be exact: the error margin may be a street block or two and, in particular, if it is decided to locate a post box at a street intersection, one does not, a priori, specify on which side of the street the post box is to be located. The precise locations are determined in a second stage, while establishing routes for the postal vans: the problem to be solved is then a GTSP where S_l represents the set of potential locations for post box l ($l = 1, \ldots, n-1$) and S_n corresponds to the sorting office.

In the second application (due to Rousseau [10]), it is assumed that all post boxes locations are known. The problem consists uniquely of establishing vehicle routes under certain assumptions: one restrictive condition is that van drivers never cross a street on foot in order to empty a post box. It is necessary, in the course of the algorithm, to duplicate post boxes located at street intersections. Consider, for example, the street grid depicted by Fig. 1. This example shows two post boxes, both indicated by triangles at locations 1 and 2. The van used to empty box 1 may come from the West or from the South. These two possibilities are represented by squares 1a and 1b. The two squares correspond to the same box and to the same location. If the van comes from the West, the shortest distance from box 1 to box 2 is equal to $c_{1a,2a}$; otherwise it is equal to $c_{1b,2b}$. It can be seen from the drawing that

$$c_{1a,2a} > c_{1b,2b}$$

Thus, the distance between two boxes i and j cannot be uniquely defined by a single number c_{ij}: it depends on which box preceded i. Standard routing algorithms requiring uniquely defined c_{ij}'s cannot be directly applied to this case. However, the problem can easily be transformed into a GTSP where each post box i ($i = 1, \ldots, n-1$) is represented by a cluster S_i:

$$S_i = \begin{cases} \{i_a, i_b\} & \text{if } i \text{ is located at a street intersection,} \\ \{i\} & \text{otherwise} \end{cases}$$

and $S_n = \{r\}$ where r is the sorting office.

In [6], the authors consider another TSP extension, the 'TSP with specified nodes' (STSP), which constitutes a special case of the GTSP. The STSP is defined

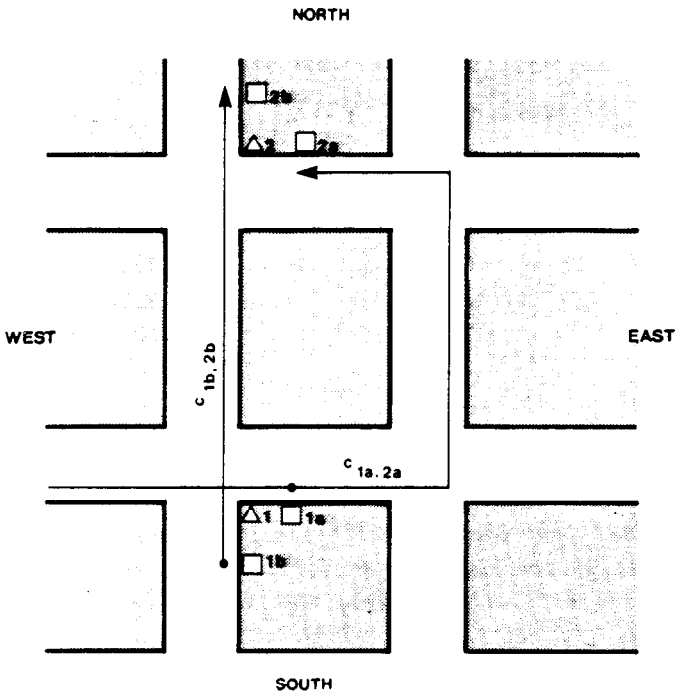

Fig. 1. Street grid for the second post boxes problem.

as follows. Let $K=\{1,\ldots,m\}$ where $m\leq r$. The STSP consists of establishing the shortest Hamiltonian circuit going through each node of K exactly once and through each node of $N-K$ at most once. This problem may be viewed as a GTSP by defining

$$S_1 = \{1\} \cup (N-K),$$
$$S_i = \{i\} \qquad (i \in K-\{1\})$$

and by replacing each c_{1j} by $c_{1j}-M$ $(j\in N-\{1\};\ M>\max_{i,j} c_{ij})$ in order to force node 1 into the optimal solution.

Note that the STSP is only of interest if C does not satisfy the triangle inequality for otherwise, it reduces to a TSP on K [6].

Early algorithms [4, 11, 12] for the GTSP were based on dynamic programming and only applied to Euclidean problems. These algorithms were also severely hampered by the amount of computer memory required to solve even small size problems. In [5], Laporte and Nobert described an integer linear programming approach suitable for Euclidean or non-Euclidean problems having a symmetrical distance matrix. The approach was quite efficient: it succeeded in solving problems much larger than would have been possible by either complete enumeration or dynamic programming. The largest problems solved contained 50 cities and 10 clusters. More recently, the authors tackled asymmetrical Euclidean GTSP's [7]. The problem was

first formulated as an integer linear program (ILP), a relaxation of which was completely unimodular. The GTSP was then solved by means of a branch and bound algorithm in which all subproblems had a network structure. In some versions of the algorithm, Lagrangean relaxation was used to increase the value of the objective function for the subproblems. Overall, the results were similar to those obtained in the symmetrical case.

This paper presents a new algorithm for the asymmetrical case. As in [5] and in [7], the problem is first formulated as an ILP. However, the relaxation is different. Also, it is no longer necessary to assume that C satisfies the triangle inequality. As will be seen, the results are quite impressive: problems involving up to one hundred cities are solved to optimality.

2. Formulations

In the following formulations, x_{ij} is a binary variable indicating the presence ($x_{ij}=1$) or the absence ($x_{ij}=0$) of an arc between node i and node j ($i \neq j$) in the optimal solution. Similarly, y_i and y_i' are binary variables equal to the flow leaving node i and to the flow entering node i respectively in the optimal solution. Thus both these variables are equal to 1 if node i belongs to the optimal solution; they are equal to zero otherwise.

A first formulation is given by

(P1) minimize $z = \sum_{\substack{i,j \in N \\ i \neq j}} c_{ij} x_{ij}$

subject to

$$\sum_{\substack{j \in N \\ i \neq j}} x_{ji} = y_i' \qquad (i \in N), \tag{1}$$

$$\sum_{\substack{j \in N \\ i \neq j}} x_{ij} = y_i \qquad (i \in N), \tag{2}$$

$$\sum_{i \in S_l} y_i = \sum_{i \in S_l} y_i' \geq 1 \qquad (l = 1, \ldots, n), \tag{3}$$

$$y_i = y_i' \qquad (i \in N), \tag{4}$$

$$\sum_{\substack{i,j \in T \\ i \neq j}} x_{ij} \leq |T| - 1 \qquad (T \subseteq N \text{ and } T \cap S_l = \emptyset \text{ for at least one but not all } l), \tag{5}$$

$$0 \leq x_{ij} \leq 1 \qquad (i, j \in N; i \neq j), \tag{6}$$

$$0 \leq y_i \leq 1 \qquad (i \in N), \tag{7}$$

$$0 \leq y_i' \leq 1 \qquad (i \in N), \tag{8}$$

$$x_{ij}, y_i, y_i' \text{ integer} \qquad (i, j \in N;\ i \neq j). \tag{9}$$

In this formulation, constraints (1) and (2) express the x_{ij}'s in terms of the inward flow y_i' and of the outward flow y_i. Constraints (3) state that each cluster is visited at least once. Constraints (4) correspond to flow conservation equations at the nodes; constraints (5) are subtour elimination constraints: they prohibit the formation of subtours including nodes from some, but not all clusters. Finally, constraints (6) to (9) are upper bound and integrality conditions on the variables.

For Euclidean problems, it is fairly straightforward to prove [5, p. 64] that there exists an optimal solution containing only one node from each cluster. Hence, in this case,

(i) all variables x_{ij} for which i and j belong to the same cluster can be eliminated;

(ii) constraints (3) can be replaced by (3′):

$$\sum_{i \in S_l} y_i = \sum_{i \in S_l} y_i' = 1 \qquad (l = 1, \ldots, n). \tag{3′}$$

Relaxing constraints (4) and (5) yields a network flow problem in which, of course, constraints (9) are no longer required. This relaxation can be solved by means of a branch and bound algorithm where the relaxed constraints are introduced into subproblems by branching on variables. This was the approach used in [7].

Reverting to (P1) for general C matrices, one observes that a simpler formulation can be obtained if

(i) variables y_i' are dropped;

(ii) constraints (4) and (8) are eliminated;

(iii) constraints (1) are replaced by

$$\sum_{\substack{j \in N \\ i \neq j}} x_{ji} = y_i \qquad (i \in N) \tag{1′}$$

Then, it may be more convenient to replace each y_i variable by $1 - x_{ii}$ where $x_{ii} = 0$ if node i belongs to the optimal solution and to 1 otherwise. So expressed, (P1) becomes

(P2) minimize $z = \sum_{\substack{i,j \in N \\ i \neq j}} c_{ij} x_{ij}$

subject to

$$\sum_{i \in N} x_{ik} = \sum_{j \in N} x_{kj} = 1 \quad (k \in N), \tag{10}$$

$$\sum_{i \in S_l} x_{ii} \leq |S_l| - 1 \qquad (l = 1, \ldots, n), \tag{11}$$

$$\sum_{\substack{i,j \in T \\ i \neq j}} x_{ij} \leq |T| - 1 \quad (T \subseteq N \text{ and } T \cap S_l = \emptyset \text{ for at least one but not all } l), \tag{12}$$

$$x_{ij} = 0 \text{ or } 1 \qquad (i, j \in N). \tag{13}$$

The structure of (P2) is rather similar to that of the asymmetrical travelling salesman problem (see for example [2]). If constraints (11) and (12) are relaxed, the

resulting problem is an assignment problem and the integrality conditions become irrelevant. Constraints (11) and (12) though different in meaning, possess the same mathematical structure and can therefore be treated similarly in a branch and bound algorithm. Finally, note that this formulation is valid whether C satisfies the triangle inequality or not. In the first case, variables x_{ij} correspondinng to arcs linking two nodes of the same cluster can be eliminated. Also, constraints (11) can be replaced by

$$\sum_{i \in S_l} x_{ii} = |S_l| - 1 \qquad (l = 1, \ldots, n). \tag{11'}$$

3. Algorithm

The proposed algorithm to solve (P2) was constructed by extending or otherwise modifying the Carpaneto and Toth branch and bound algorithm for the asymmetrical TSP [2]. The main changes occur in

(i) the initial elimination of variables;

(ii) the partitioning rule in the case of Euclidean problems;

(iii) the computation of lower bounds on the value of the optimal solution;

(iv) the nature of the problem solved at each node of the search tree: the subproblems are assignment problems instead of modified assignment problems as in [4]. (In the GTSP, variables x_{ii} may take the value 1.)

The following notation will be used in the description of the algorithm.

z^*: the cost of the best feasible solution so far identified,
z_h: the value of the objective function of the assignment problem at node h of the search tree,
$\underline{z}_h$: a lower bound on z_h,
IA_h: the set of included arcs in subproblem h,
EA_h: the set of excluded arcs in subproblem h,
IN_h: the set of included nodes in subproblem h,
EN_h: the set of excluded nodes in subproblem h.

The algorithm can be broken down into the following steps.

Step 0 (Initial feasible solution).

(i) (Euclidean problems only). Eliminate arc (i,j) if $i,j \in S_l$ $(l = 1, \ldots, n)$.

(ii) (Euclidean problems only). Consider a cluster S_l. Let $i,j \in S_l$ be such that $c_{ik} \leq c_{jk}$ and $c_{ki} \leq c_{kj}$ for all $k \notin S_l$. Then there exists an optimal solution not including node j which can therefore be eliminated.

(iii) Construct a first tour containing one node from each cluster according to a nearest neighbour rule.

(iv) Improve the tour obtained in the previous step by means of a 2-opt procedure [9].

(v) (Non-Euclidean problems only). Consider in turn all nodes not belonging to the tour obtained in (iv). Attempt to insert each of these nodes in each possible position of the tour. Execute an insertion whenever it reduces the overall length of the tour.

(vi) Let z^* be the length of the last tour obtained.

Step 1 (Node 1 of the search tree). Set $IA_1 = EA_1 = IN_1 = EN_1 = \phi$ and solve the assignment problem associated with constraints (10) and $x_{ij} \geq 0$ for $i, j \in N$. If $z_1 = z^*$, or if the solution is feasible for the GTSP, terminate. Otherwise, insert node 1 in the queue.

Step 2 (Node selection). If the queue is empty, terminate. Otherwise, select the next node (node h) on which to branch: branching is always done on the pending node having the smallest z_h.

Step 3 (Branching). The solution found at node h is illegal and must be eliminated. A solution may be illegal for one or both of the following reasons:

(i) at least one cluster is not visited, i.e., there exists a cluster S_l for which $x_{ii} = 1$ for all $i \in S_l$;

(ii) there exist illegal subtours, i.e., subtours containing nodes from some but not all clusters.

If all clusters are visited, the algorithm proceeds to Step 4. Otherwise, consider all unvisited clusters and select the cluster S_l for which $|S_l - EN_h|$ is minimized. Let $S_l - EN_h = \{i_1, \ldots, i_p\}$. Subproblems g are then created from subproblem h by updating the sets of included and excluded nodes:

$$EN_g = \begin{cases} EN_h & (g = 1), \\ EN_h \cup \{i_u : u = 1, \ldots, g-1\} & (g = 2, \ldots, p), \end{cases}$$

$$IN_g = IN_h \cup \{i_g\} \qquad (g = 1, \ldots, p).$$

In Euclidean problems, it is valid to exclude all nodes of $S_l = \{i_g\}$ once i_g is included (see constraints (11′)). Therefore IN_g can be defined as above and EN_g as

$$EN_g = EN_h \cup (S_l - \{i_g\}) \qquad (g = 1, \ldots, p).$$

This is clearly stronger than the rule used in the general case. In practice, including a node i in IN_g is done by forcing variable x_{ii} to 0 (by increasing its cost) in subproblem g; similarly, excluding a node i (i.e., including it in EN_g) is done by forcing variable x_{ii} to 1 in subproblem g. Skip Step 4.

Step 4 (Branching). All clusters are visited, but the solution contains illegal subtours. The algorithm creates descendant subproblems from node h by branching on an illegal subtour. The subtour having the least number of arcs not already included in IA_h is selected for branching. Let $\{(i_1, j_1), \ldots, (i_q, j_q)\}$ be the set of arcs belonging to this subtour but not to IA_h (in the same order as they appear on the subtour).

Then

$$\mathrm{IA}_g = \begin{cases} \mathrm{IA}_h & (g=1), \\ \mathrm{IA}_h \cup \{(i_u, j_u) : u = 1, \ldots, g-1\} & (g=2, \ldots, q), \end{cases}$$

$$\mathrm{EA}_g = \mathrm{EA}_h \cup \{(i_g, j_g)\} \qquad (g=1, \ldots, q).$$

For each subproblem g, execute Steps 5 to 8. Then go to Step 2.

Step 5 (Bounding). Compute a lower bound $\underline{z}_g$ on z_g according to Step 3 of [2]. If $\underline{z}_g < z^*$, proceed to Step 6. Otherwise, consider the next g and repeat Step 5.

Step 6 (Assignment problem solution). Solve the assignment problem associated with node g. This assignment problem is constrained by the sets IN_g, EN_g, IA_g and EA_g. If $z_g \geq z^*$, consider the next g and proceed to Step 5.

Step 7 (Alternative lower bound). Compute a new lower bound $\underline{z}'_g$ on the value of the optimal GTSP solution associated with subproblem g. The validity of the bound rests on the fact that this solution will consist of a Hamiltonian circuit through every node of IN_g and through at least one node from each cluster. The bound is obtained by modifying a procedure developed by Christofides for the TSP [3]:

(i) Remove all nodes belonging to EN_g and extract from C the distance matrix $D=(d_{ij})$ associated with $(N-\mathrm{EN}_g)^2$.

(ii) *Contract* every cluster S_l having an empty intersection with IN_g into a single node i: if two clusters S_{l_1} and S_{l_2} are contracted into nodes i_{l_1} and i_{l_2} respectively, then the distance between these two nodes is equal to

$$d_{i_{l_1} i_{l_2}} = \min_{u \in S_{l_1}, v \in S_{l_2}} \{d_{uv}\}.$$

Let L be the set of indices of all contracted clusters.

(iii) Let $N_g = \mathrm{IN}_g \cup \{i_l : l \in L\}$ and D_g be the distance matrix associated with $(N_g)^2$. Set $\underline{z}'_g = 0$.

(iv) *Compress* D_g. This operation consists of tranforming D_g into an Euclidean distance matrix. This is done by replacing every d_{ij} for which $d_{ij} > d_{ik} + d_{kj}$ for some k by $\min_k (d_{ik} + d_{kj})$. The process is repeated until D_g is Euclidean.

(v) Solve the assignment problem associated with D_g. Let v^* be the value of its optimal solution. Let $\underline{z}'_g = \underline{z}'_g + v^*$ and D_g be the reduced distance matrix.

(vi) Contract the set of nodes associated with every subtour into a single node.

(vii) Repeat (iv), (v) and (vi) until D_g is a 1×1 matrix.

(viii) $\underline{z}'_g$ is a valid lower bound on the value of the TSP solution on N_g (see [3, p. 1049]) and hence, on the value of the optimal GTSP solution associated with subproblem g. If $\underline{z}'_g \geq z^*$, consider the next g and go to Step 5.

Step 8 (Feasibility check). Check whether the current solution is feasible, i.e., that all clusters are visited and that there is only one Hamiltonian circuit. If the solution is feasible, set $z^* = z_g$ and store the tour. If $z^* = z_h$, go to Step 2. Otherwise, insert node g in the queue. Consider the next g and go to Step 5.

4. Computational results

The algorithm described in Section 3 was tested on several randomly generated Euclidean and non-Euclidean problems ranging from 20 to 100 cities. The number of clusters varied from 3 to 13. In the case of Euclidean problems, $2r$ points $P_1, P_2, \ldots, P_r, Q_1, \ldots, Q_r$ were first generated according to a uniform distribution on $[0,100]^2$. The c_{ij}'s were then defined by

$$c_{ij} = \|P_i - P_j\| \quad \text{and} \quad c_{ji} = \|Q_i - Q_j\|.$$

These distances were rounded off to the nearest integer and C was compressed as in Step 7 of the algorithm. For non-Euclidean problems, the c_{ij}'s were generated from a uniform distribution on $[0,100]$ and rounded off to the nearest integer.

As in [5,7], the clusters were disjoint and cities were distributed as uniformly as possible among the clusters, on a random basis. This means that in Euclidean problems, the allocation of cities to clusters has no geometrical or physical significance.

For each type of distance matrix, 5 problems were attempted for every combination of r and n. Tables 1 and 2 report average results for problems which could be solved within 500 seconds.

The meanings of the table headings are as follows:

r:	number of citics.
n:	number of clusters.
Succ:	number of successful problems out of 5.
% Arcs:	percentage of arcs eliminated; these can be arcs linking two nodes within the same clusters or arcs eliminated through the dominance test in Step 0(ii).
Nodes:	number of nodes of the search tree for which a lower bound z_h (see Step 4) was computed.
Queue:	number of these nodes inserted in the queue (nodes having a value of z_h less than z^* at the time of their generation).
Desc:	number of nodes in the queue which were later examined (number of nodes having descendants).
Sets:	number of nodes inserted in the queue because the solution of the relaxed problem contained unvisited clusters.
Subtours:	number of nodes inserted in the queue because the solution of the relaxed problem contained illegal subtours.
Time:	number of CPU seconds (University of Montreal CYBER 173 computer).

The results indicate that the algorithm was quite successful. Problems involving up to 100 nodes and 8 clusters were solved to optimality. This compares favourably with 50 nodes in the symmetrical case [5] and 40 nodes in the previous study on the asymmetrical case [7]. The dynamic programming approach first proposed for the

Table 1. Computational results for Euclidean problems

r	n	Succ	% Arcs	Nodes	Queue	Desc	Sets	Subtours	Time
20	3	5	72.0	5.4	1.2	1.2	0.6	0.6	0.096
	5	5	31.1	33.4	15.0	13.6	7.6	7.4	0.730
	8	5	16.4	53.4	32.2	24.6	23.6	8.6	1.566
	11	5	8.3	124.2	74.2	69.0	51.0	23.2	4.046
30	3	5	77.9	6.6	1.2	0.8	1.2	0.0	0.309
	6	5	28.5	157.0	63.8	47.6	29.8	34.0	6.750
	9	5	14.4	465.4	306.2	181.0	155.0	151.2	32.503
	11	5	12.3	517.0	248.0	218.4	214.4	33.6	36.542
40	3	5	83.5	5.8	0.8	0.6	0.8	0.0	0.390
	6	5	28.1	88.4	24.0	19.6	20.8	3.2	7.040
	7	5	22.0	180.2	92.2	36.6	73.4	18.8	19.479
	9	5	17.4	265.4	172.8	67.0	144.0	28.4	37.057
	11	5	12.9	1080.2	567.16	384.2	464.2	103.4	123.261
50	3	5	81.1	3.0	0.8	0.6	0.8	0.0	0.910
	6	5	29.7	41.0	13.8	7.8	12.4	1.4	7.120
	8	5	22.1	384.8	154.6	88.6	136.6	18.0	59.172
	9	5	20.1	506.6	235.2	118.8	231.6	3.6	100.552
	11	5	14.7	395.8	277.0	93.4	274.0	3.0	127.797
60	3	5	87.2	12.0	1.2	0.8	1.2	0.0	1.724
	6	5	36.0	128.0	25.6	18.2	23.0	2.6	21.240
	8	5	23.7	317.0	90.8	63.4	79.8	11.0	82.006
	9	5	22.1	554.6	182.6	125.8	159.4	23.2	147.299
	11	1	16.7	1257.0	484.0	342.0	366.0	118.0	262.278
70	3	5	90.7	17.8	1.8	1.2	1.6	0.2	3.667
	6	5	32.1	368.6	140.0	51.4	82.8	57.2	125.606
	8	5	24.0	573.2	193.8	107.6	158.2	35.6	195.876
	11	1	16.5	519.0	459.0	77.0	435.0	24.0	419.483
	13	1	13.9	785.0	293.0	199.0	255.0	38.0	389.338
80	3	5	91.3	4.4	1.2	0.6	1.2	0.0	2.570
	6	5	33.2	46.8	10.6	4.2	9.4	1.2	22.726
	8	4	25.7	437.0	123.3	63.0	119.0	4.3	233.616
	11	2	18.7	220.0	145.0	31.0	140.0	5.0	223.230
90	3	5	92.5	12.6	1.0	1.0	0.6	0.4	5.303
	6	5	37.4	208.0	42.6	24.6	34.4	8.2	100.797
	8	3	27.8	372.0	115.3	55.0	94.0	21.3	269.064
100	3	5	93.6	2.4	0.6	0.4	0.6	0.0	5.043
	6	5	35.5	74.4	38.6	5.6	35.6	3.0	129.147
	8	2	26.7	358.5	109.0	62.5	77.0	32.0	275.080

GTSP [4, 11, 12] could only handle Euclidean problems and the values of r and n had to be relatively small ($r \geq 50$ and $n \leq 5$ for $r > 30$, see [4] for details).

For all values of r, the difficulty of the problem increases with the number of clus-

Table 2. Computational results for non-Euclidean problems

r	*n*	Succ	Nodes	Queue	Desc	Sets	Subtours	Time
20	3	5	5.6	2.0	1.4	0.6	1.4	0.161
	5	5	11.0	5.0	2.8	3.8	1.2	0.463
	8	5	26.6	16.4	10.2	11.8	4.6	1.314
	11	5	31.4	19.6	16.2	17.0	2.6	1.935
30	3	5	15.4	2.6	2.0	1.4	1.2	0.397
	6	5	40.4	19.4	9.6	15.0	4.4	2.652
	9	5	91.2	54.0	28.6	44.8	9.2	7.904
	11	5	87.4	57.8	32.4	51.6	6.2	8.953
40	3	5	1.8	0.4	0.4	0.4	0.0	0.193
	6	5	35.6	17.0	6.4	10.4	6.6	3.850
	7	5	71.4	33.0	14.2	26.4	6.6	8.573
	9	5	103.2	50.0	26.2	40.4	9.6	13.502
	11	5	118.2	70.4	33.6	68.2	2.2	21.660
50	3	5	10.6	4.2	0.8	0.8	3.4	1.202
	6	5	73.6	24.2	12.4	16.8	7.4	10.688
	8	5	254.2	106.4	47.4	88.6	17.8	48.904
	9	5	342.4	174.8	71.2	141.2	33.6	77.710
	11	4	316.8	123.0	82.5	118.3	4.8	72.578
60	3	5	4.8	0.6	0.6	0.6	0.0	0.621
	6	5	99.2	25.0	12.6	22.6	2.4	21.598
	8	5	152.4	66.6	23.6	57.0	9.6	55.735
	9	5	204.6	115.8	27.2	103.6	12.2	94.895
	11	3	452.3	224.7	89.3	213.0	11.7	179.581
70	3	5	10.8	0.8	0.8	0.8	0.0	1.485
	6	5	96.6	25.6	12.6	21.2	4.4	30.012
	8	5	140.8	63.2	18.8	56.8	6.4	72.105
	11	4	857.8	270.8	162.0	257.0	13.8	325.591
	13	1	678.0	222.0	148.0	220.0	2.0	309.746
80	3	5	14.4	2.0	0.8	0.8	1.2	2.641
	6	5	89.8	20.2	8.8	16.6	3.6	34.340
	8	5	288.0	94.0	35.0	86.0	8.0	158.039
90	3	5	10.8	2.0	0.4	0.4	1.6	3.788
	6	5	111.2	26.8	9.0	21.8	5.0	57.634
	8	4	184.8	67.0	19.8	60.5	6.5	158.277
100	3	5	38.2	5.2	2.0	1.2	4.0	9.072
	6	5	171.0	46.0	12.8	35.4	10.6	140.679
	8	2	222.5	91.5	19.5	83.5	8.0	277.795

ters. This was also the case in [5, 7]. Euclidean problems were in general more difficult to solve than non-Euclidean problems of the same size, in spite of the fact that in the Euclidean case, a large proportion of the initial variables could be eliminated (see column % Arcs in Table 1). This phenomenon was also observed in [5, 7] and

Table 3. Comparative results for Euclidean and non-Euclidean problems

		Euclidean problems			Non-Euclidean problems		
r	n	Nodes	Desc	Time	Nodes	Desc	Time
30	3	24.6	5.4	0.780	15.4	2.0	0.397
	6	69.6	15.4	5.425	40.4	9.6	2.652
	9	139.2	43.6	12.904	91.2	28.6	7.904
	11	163.4	58.4	19.278	87.4	32.4	8.953
40	3	3.8	0.6	0.254	1.8	0.4	0.193
	6	108.8	20.2	11.735	35.6	6.4	3.850
	7	147.6	31.6	18.312	71.4	14.2	8.573
	9	114.4	30.8	17.692	103.2	26.2	13.502
	11	257.4	73.2	61.838	118.2	33.6	21.660

in other TSP extensions studied by the authors. Non-Euclidean problems tend to have feasible solutions whose costs have a larger variance, leading to more dominance and earlier fathoming in the search tree. This behaviour was investigated further by solving test problems for Euclidean and non-Euclidean problems generated from the same data. Non-Euclidean problems are the 30 and 40 city problems of Table 2. Euclidean problems were obtained by compressing the distance matrices of these non-Euclidean problems. Five problems were solved for every combination of r and n. In each case, all problems were succesful. The results show that Euclidean problems are about twice as difficult as non-Euclidean problems. However, a larger difference was observed when Euclidean problems were generated as in Table 1.

An examination of the two columns Sets and Subtours of Tables 1 and 2 reveals that most infeasibilities occurred because the solution of the relaxed subproblems contained unvisited clusters. Note that the relaxation used in [7] ensured that all clusters were visited; but then, flow conservation at the nodes was not always satisfied. Overall, the algorithm developed in this paper is considerably more successful. To the authors' knowledge, it represents by far the most efficient approach ever developed for the GTSP.

References

[1] J. Bovet, The selective travelling salesman problem, Paper presented at the EURO VI Conference, Vienna, 1983.

[2] G. Carpaneto and P. Toth, Some new branching and bounding criteria for the asymmetric travelling salesman problem, Management Sci. 26 (1980) 736–743.

[3] N. Christofides, Bounds for the travelling salesman problem, Operations Research 20 (1972) 1044–1056.

[4] A.L., Henry-Labordere, The record balancing problem: A dynamic programming solution of a generalized travelling salesman problem, RIRO B-2 (1969) 43–49.

[5] G. Laporte and Y. Nobert, Generalized travelling salesman problem through n sets of nodes: an integer programming approach, Infor. 21 (1983) 61–75.

[6] G. Laporte, H. Mercure and Y. Nobert, Optimal tour planning with specified nodes, RAIRO Rech. Opér. 18 (1984) 203–210.

[7] G. Laporte. H. Mercure and Y. Nobert, Finding the shortest Hamiltonian circuit through *n* clusters: a Lagrangean relaxation approach, Cong. Mumer. 48 (1985) 277–290.

[8] J.K. Lenstra and A.H.G. Rinnooy Kan, Complexity of vehicle routing and scheduling problems, Networks, 11 (1981) 221–227.

[9] S. Lin and B.W. Kerninghan, Computer solutions of the travelling salesman problem, Operations Research 21 (1973) 498–516.

[10] J.-M. Rousseau, Private Communication, Centre de recherche sur les transports, University of Montreal, January 1985.

[11] J.P. Saksena, Mathematical model of scheduling clients through welfare agencies, CORS J. 8 (1970) 185–200.

[12] S.S. Srivastava, S. Kumar, R.C. Garg and P. Sen, Generalized travelling salesman problem through *n* sets of nodes, CORS J. 7 (1969) 97–101.