View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by Elsevier - Publisher Connector

Discrete Applied Mathematics 26 (1990) 193-207
North-Holland

THE SELECTIVE TRAVELLING SALESMAN PROBLEM

Gilbert LAPORTE
École des Hautes Études Commerciales de Montréal, Canada

Silvano MARTELLO
DEIS, University of Bologna, Italy

Received 11 June 1987
Revised 6 June 1988

Given a weighted graph with profits associated with the vertices, the selective travelling salesman problem (or orienteering problem) consists of selecting a simple circuit of maximal total profit, whose length does not exceed a prespecified bound. This paper provides integer linear programming formulations for the problem. Upper and lower bounds are then derived and embedded in exact enumerative algorithms. Computational results are reported.

1. Introduction

Let $G=(V,A)$ be a complete graph with vertex set $V=\{v_1,\dots,v_n\}$ and arc set $A=\{(v_i,v_j)\colon v_i,v_j\in V,\ v_i\neq v_j\}$, having a *profit* p_i associated with each vertex $v_i\in V$ and a *distance* (or *cost*) c_{ij} associated with each arc $(v_i,v_j)\in A$ ($c_{ii}=0$ for all i). Throughout this paper, we assume that the cost matrix (c_{ij}) satisfies the *triangle inequality* ($c_{ij}\le c_{ik}+c_{kj}$ for all i, j, k). The *selective travelling salesman problem* (*STSP*) consists of determining a simple circuit of maximal total profit, which includes v_1 and whose length does not exceed a preset value c. More formally, we want to find an ordered vertex set $C^*=\{v_{i_1},v_{i_2},\dots,v_{i_h}\}$ such that

$$v_{i_r}\neq v_{i_s}\quad \text{for } r,s\in\{1,\dots,h\},\ r\neq s,$$

$$v_1\in C^*,$$

$$\sum_{r=1}^{h-1} c_{i_r,i_{r+1}}+c_{i_h,i_1}\le c,$$

$$\sum_{r=1}^{h} p_{i_r}\ \text{is maximized.}$$

Note that the triangularity condition is not restrictive since any distance matrix can be replaced by the corresponding shortest path matrix (in such a case, the STSP solution relative to the original graph may be a nonsimple circuit). Also, observe that unprofitable vertices will never be selected in the solution, so we can assume without loss of generality that $p_i>0$ for all $v_i\in V$.

0166-218X/90/$3.50 © 1990, Elsevier Science Publishers B.V. (North-Holland)

The problem arises in a number of practical contexts. An application to the delivery of home heating oil is described in Golden, Levy and Vohra [4]. Hayes and Norman [6] and Tsiligirides [11] describe it as the *orienteering problem* because of its connection with "orienteering", a treasure-hunt game in which competitors collect scores by reaching "control points" within a prefixed time limit.

The STSP is NP-hard, as can easily be seen by transformation of its recognition version from the *Hamiltonian circuit problem* (*HC*), which is known to be NP-complete (see Garey and Johnson [2]). Given any instance of the HC, relative to a graph $G^h=(V^h,A^h)$ with vertex set $V^h=\{v_1,\dots,v_n\}$ and arc set $A^h=\{(v_i,v_j)\}$, define an instance of STSP having: $V=V^h$; $p_i=1$ for all vertices; $c_{ij}=1$ if $(v_i,v_j)\in A^h$, $c_{ij}=2$ otherwise; $c=|V|$. Then G^h possesses a Hamiltonian circuit if and only if the solution value of this instance equals $|V|$.

Apart from a straightforward dynamic programming formulation by Hayes and Norman [6], only approximate algorithms seem to have been provided in the literature for the STSP (Tsiligirides [11]; Golden, Levy and Vorha [3,4]; Golden, Wang and Liu [5]). In this paper we develop new theoretical results and exact algorithms for the problem.

In the following section we provide integer linear programming formulations and derive a solution approach. In Section 3 we introduce upper and lower bounds for the problem. Enumerative algorithms are presented in Section 4 and computational results in Section 5.

2. An ILP-based approach

The problem described in the previous section, relative to a directed graph, can be formally stated as follows. Let C^* be the ordered set of vertices in the optimal circuit and define a binary variable x_{ij} taking the value 1 if and only if v_i and v_j are two consecutive vertices of C^* (the first vertex is assumed to be consecutive to the last one). The ILP formulation of the STSP is then:

maximize

$$p_1+\sum_{i=2}^{n} p_i \sum_{j=1}^{n} x_{ij},$$

subject to

$$\sum_{i=2}^{n} x_{i1}=\sum_{j=2}^{n} x_{1j}=1, \tag{1}$$

$$\sum_{i=1}^{n} x_{ik}=\sum_{j=1}^{n} x_{kj}\le 1, \quad k=2,\dots,n, \tag{2}$$

$$\sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij}\le c, \tag{3}$$

$$\sum_{v_k \in S} \left(\sum_{i=1}^{n} x_{ik} + \sum_{j=1}^{n} x_{kj} \right) \le |S| \left(\sum_{\substack{v_i \in S \\ v_j \notin S}} x_{ij} + \sum_{\substack{v_i \notin S \\ v_j \in S}} x_{ij} \right), \quad S \subset V \setminus \{v_1\},\ |S| \ge 2, \tag{4}$$

$$x_{ij} \in \{0,1\}, \quad i = 1, \ldots, n;\ j = 1, \ldots, n;\ i \ne j.$$

Constraints (1) specify that vertex v_1 belongs to C^*, while (2) are the flow conservation conditions for the remaining vertices. Constraint (3) defines the upper bound on the optimal circuit length. In connectivity constraints (4), for any proper subset S of $V \setminus \{v_1\}$, the left-hand side is equal to $2|S \cap C^*|$; the right-hand side takes the value 0 if in the optimal solution S is disconnected from its complement, and takes a value not less than $2|S|$ otherwise. Since $|S \cap C^*| \le |S|$, no feasible solution is eliminated, while any solution containing a subtour in S is prevented.

In the particular case where $c_{ij} = c_{ji}$ for all i, j, i.e., when the graph is undirected, the number of constraints and variables can be considerably reduced as follows. Variables x_{ij} need only be defined for $i<j$ and give the number of times *edge* (v_i, v_j) is used in the solution. Since $C^* = \{v_1, v_j\}$ is feasible, these variables can take three values:

$$x_{ij} = \begin{cases} 2, & \text{if } C^* = \{v_1 \equiv v_i, v_j\}, \\ 1, & \text{if } |C^*| > 2 \text{ and } v_i,\ v_j\ (i<j) \text{ are consecutive vertices of } C^*, \\ 0, & \text{otherwise.} \end{cases}$$

By introducing $n-1$ additional binary variables y_i $(i = 2, \ldots, n)$, taking the value 1 if and only if $v_i \in C^*$, we obtain the following model:

maximize

$$p_1 + \sum_{i=2}^{n} p_i y_i,$$

subject to

$$\sum_{j=2}^{n} x_{1j} = 2, \tag{5}$$

$$\sum_{i=1}^{k-1} x_{ik} + \sum_{j=k+1}^{n} x_{kj} = 2y_k, \quad k = 2, \ldots, n, \tag{6}$$

$$\sum_{i=1}^{n} \sum_{j=i+1}^{n} c_{ij} x_{ij} \le c, \tag{7}$$

$$2 \sum_{v_k \in S} y_k \le |S| \left(\sum_{\substack{v_i \in S \\ v_j \notin S}} x_{ij} + \sum_{\substack{v_i \notin S \\ v_j \in S}} x_{ij} \right), \quad S \subset V \setminus \{v_1\},\ |S| \ge 3, \tag{8}$$

$$x_{1j} \in \{0,1,2\}, \quad j = 2, \ldots, n,$$

$$x_{ij} \in \{0,1\}, \quad i = 2, \ldots, n;\ j = i+1, \ldots, n,$$

$$y_i \in \{0,1\}, \quad i = 2, \ldots, n.$$

Constraints (5)–(8) are the counterparts of (1)–(4), respectively. Note that $|S|\geq 3$ in (8), since no subtour of two edges exists in $V\setminus\{v_1\}$. A graphical interpretation of constraints (8) can be found in Laporte [7].

We used a standard constraint relaxation algorithm to solve the second model (ILP-based approaches are known to be generally inefficient for asymmetrical travelling salesman problems). The algorithm starts by relaxing the connectivity constraints and the integrality conditions on the variables. The resulting problem is then solved through linear programming and the violated conditions are gradually introduced through a branch and bound process. Two variants of the algorithm are obtained according to the order in which connectivity and integrality conditions are considered. In the first one, violated connectivity constraints are only generated at an integer solution. In the second variant, they are introduced as soon as a connected (possibly fractional) component disjoint from $\{v_1\}$ is identified, while branching on fractional variables occurs only when no such component exists.

3. Bounds and approximate algorithms

In this section we introduce upper and lower bounds for the STSP. These will be used in the enumerative algorithms described in Section 4.

3.1. Upper bounds

The following theorem provides an upper bound on the optimal STSP solution.

Theorem 1. *Given any instance of the STSP and a real value* α $(0\leq\alpha\leq 1)$, *define* vertex weights

$$w_j=\alpha \min_{i\neq j}\{c_{ij}\}+(1-\alpha)\min_{k\neq j}\{c_{jk}\}, \quad j=1,\dots,n, \tag{9}$$

and let z^* *be the optimal solution value to the following* 0–1 knapsack problem (KP):

maximize

$$z=p_1+\sum_{j=2}^{n}p_jy_j,$$

subject to

$$\sum_{j=2}^{n}w_jy_j\leq c-w_1,$$

$$y_j\in\{0,1\},\quad j=2,\dots,n.$$

Then z^* *is no less than the optimal solution of the STSP instance.*

Proof. Let $I^*=(v_{t_1}\equiv v_1, v_{t_2},\dots,v_{t_h}\equiv v_1)$ be the sequence of vertices in the optimal solution of the STSP instance, thus giving $\sum_{j=1}^{h-1} p_{t_j}$ as optimal value. We prove that the KP has a feasible solution having the same value. From (3) it must be that $\sum_{j=1}^{h-1} c_{t_j,t_{j+1}} \le c$, so

$$(1-\alpha)c_{1,t_2} + \sum_{j=2}^{h-1} (\alpha c_{t_{j-1},t_j} + (1-\alpha)c_{t_j,t_{j+1}}) + \alpha c_{t_{h-1},1} \le c.$$

From (9), $\alpha c_{t_{j-1},t_j} + (1-\alpha)c_{t_j,t_{j+1}} \ge w_{t_j}$ for $j=1,\dots,h-1$. Hence the conclusion follows. □

Since it is known that v_1 necessarily belongs to the solution, a tighter bound can be derived as follows. For any pair of vertices v_r, v_s $(r,s \ne 1)$, let STSP(r,s) be the restricted instance of STSP obtained by imposing that arcs (v_r, v_1) and (v_1, v_s) belong to the solution. Let us assume for the moment that $r \ne s$. If $c_{1s}+c_{sr}+c_{r1} > c$, then STSP(r,s) is infeasible and we can define $z(r,s)=p_1$ as an upper bound on its solution value. Otherwise, let $H(r,s)=\{j \ne 1,r,s\colon c_{1s}+c_{sj}+c_{jr}+c_{r1} \le c\}$ and note that only vertices v_j such that $j \in H(r,s)$ can belong to the solution of STSP(r,s). The minimum cost of including such vertices in the solution becomes

$$w_j(r,s) = \alpha \min_{i \in H(r,s) \cup \{s\} \setminus \{j\}} \{c_{ij}\} + (1-\alpha) \min_{k \in H(r,s) \cup \{r\} \setminus \{j\}} \{c_{jk}\},$$
$$j \in H(r,s). \qquad (10)$$

Hence, an upper bound for STSP(r,s) can immediately be derived from Theorem 1 as

$$z(r,s) = (p_1+p_r+p_s) + \max \sum_{j \in H(r,s)} p_j y_j,$$

subject to

$$\sum_{j \in H(r,s)} w_j(r,s) y_j \le c - \left(c_{1s}+c_{r1}+\alpha \min_{i \in H(r,s)} \{c_{ir}\} + (1-\alpha) \min_{k \in H(r,s)} \{c_{sk}\}\right)$$
$$y_j \in \{0,1\}, \quad j \in H(r,s).$$

In the particular case where $r=s$, we trivially have $z(r,s)=p_1+p_r$ if $c_{1r}+c_{r1} \le c$ and $z(r,s)=p_1$ otherwise. Therefore a valid upper bound for the STSP is $\max_{2 \le r,s \le n} \{z(r,s)\}$.

The above bounds require the optimal solution of a KP, which is known to be NP-hard. However, it is always valid to replace the KP solution value by an upper bound. A number of such bounds, all requiring O(n) time, can be found in Martello and Toth [9]. In our implementation, we used the Martello and Toth bound [8,9]. As a result, computing the upper bound for STSP given by Theorem 1 requires O(n^2) time, since this is the complexity of computing the w_j. It follows that the tighter bound requires O(n^4) time. It is however possible to compute, in O(n^3) time, a relaxation of it by replacing (10) with

$$w_j(r,s) = \alpha \min_{i \ne j,1,r} \{c_{ij}\} + (1-\alpha) \min_{k \ne j,1,s} \{c_{jk}\}, \quad j \in H(r,s).$$

With this, in fact, we can first determine in $O(n^2)$ time the quadruplets of indices $(i'(j), i''(j), k'(j), k''(j); j=2,\dots,n)$ such that

$$c_{i'(j),j} = \min_{i \neq 1,j} \{c_{ij}\} \quad \text{and} \quad c_{i''(j),j} = \min_{i \neq 1,j,i'(j)} \{c_{ij}\},$$

$$c_{j,k'(j)} = \min_{k \neq 1,j} \{c_{jk}\} \quad \text{and} \quad c_{j,k''(j)} = \min_{k \neq 1,j,k'(j)} \{c_{jk}\}.$$

Using this information, each $w_j(r,s)$ can be immediately computed in $O(1)$ time, so the overall complexity for the bound is $O(n^3)$. Computational experiments proved that the relaxation introduced has very marginal effects on the tightness of the resulting bound.

3.2. Approximate algorithms

Approximate algorithms for the STSP can be easily obtained by simple modifications to known *travelling salesman problem* (TSP) heuristics. In this paper, which is mostly concerned with exact algorithms, heuristic methods will only be used to provide an initial feasible solution in the branch and bound process of Section 4. We describe two simple schemes which have proved to be computationally efficient for the STSP.

The first scheme corresponds to the nearest neighbour TSP algorithm (Rosenkrantz, Stearns and Lewis [10]). It gradually extends a path $I = \{v_r, \dots, v_1, \dots, v_s\}$ by adding, at each iteration, an arc (v_i, v_r) or (v_s, v_j) according to a greedy criterion. Let $l(I)$ denote the length of the current path. The general scheme can be outlined as follows.

```
Algorithm H1
begin
  $I := \{v_1\}$, $r := s := 1$;
  while $F = \{v_k \notin I: l(I) + c_{sk} + c_{kr} \leq c\} \neq \emptyset$ do
    begin
      find $v_i$ and $v_j$ such that
      $p_i/c_{ir} = \max\{p_k/c_{kr}: v_k \in F\}$ and $p_j/c_{sj} = \max\{p_k/c_{sk}: v_k \in F\}$;
      if $p_i/c_{ir} > p_j/c_{sj}$ then add $v_i$ to $I$ before $r$ and set $r := i$
      else add $v_j$ to $I$ after $s$ and set $s := j$
    end;
  close the current path by adding arc $(v_s, v_r)$
end
```

The second scheme, derived from the TSP cheapest insertion algorithm [10], extends a tour $T = \{v_1, \dots, v_1\}$ instead of a path. Let $l(T)$ denote the length of the current tour.

```
Algorithm H2
begin
  $T := \{v_1\}$;
  find the maximal ratio $p_j/(c_{1j}+c_{j1})$ such that $j \neq 1$ and $c_{1j}+c_{j1} \leq c$ (if
  no such $j$ exists then stop);
  form the tour $T := \{v_1, v_j, v_1\}$;
  improve := true;
  repeat
    find $v_j \notin T$ and $(v_r, v_s)$ consecutive vertices in $T$ such that $l(T)+$
    $c_{rj}+c_{js}-c_{rs} \leq c$ and $p_j/(c_{rj}+c_{js}-c_{rs})$ is maximal;
    if no such $v_j$ and $(v_r, v_s)$ exist then improve := false
    else insert $v_j$ in $T$ between $v_r$ and $v_s$
  until improve = false
end
```

The time complexities of Algorithms H1 and H2 are the same as those of their TSP counterparts, i.e., $O(n^2)$ and $O(n^2 \log n)$, respectively.

4. Enumerative algorithms

We now describe some exact algorithms for the STSP. These consists of gradually extending a simple path emanating from vertex v_1 through a breath-first branch and bound process.

At the first node of the search tree, we compute the second upper bound described in Section 3.1 and take as initial feasible solution the better of the two solutions provided by Algorithms H1 and H2 of Section 3.2. If the value of this solution equals that of the upper bound, the algorithm terminates. Otherwise, the branching process is initiated.

Let $I(h) = \{v_{t_1} \equiv v_1, v_{t_2}, \ldots, v_{t_h}\}$ denote the sequence of vertices included in the current path at a general node h of the search tree. Descendant nodes are generated by branching on a vertex v_i not already in $I(h)$. Upper bounds on the value of the optimum and lower bounds on the length of the tour attainable from the current path are used for fathoming nodes of the search tree.

4.1. Partitioning schemes

We implemented two partitioning schemes. At node h of the decision tree, let $l(h) = \sum_{i=1}^{h-1} c_{t_i, t_{i+1}}$ denote the *length* of the current path.

4.1.1. First partitioning scheme

In the first scheme, P1, node h generates one descendant node for each vertex v_i which can be included in $I(h)$ after v_{t_h} and which does not yield a dominated solution. At node h, vertex v_i is said to dominate vertex v_j ($j \neq i$) if $c_{t_h,i}+c_{ij}=c_{t_h,j}$, since the sequence (v_{t_h}, v_i, v_j) gives a larger profit than (v_{t_h}, v_j) at the same cost. Define $F(h)=\{v_j \notin I(h): l(h)+c_{t_h,j}+c_{j1} \leq c\}$ and $D(h)=\{v_j \in F(h): v_j$ is dominated by $v_i \in F(h)\}$. Then branches are created for each $v_i \in F(h) \setminus D(h)$.

4.1.2. Second partitioning scheme

The second scheme, P2, requires a more elaborate description. At node h we select a vertex v_i and generate either three or two descendant nodes. In the former case the generation corresponds to the following decisions:

(h_1): v_i is included in the current path immediately after v_{t_h};
(h_2): v_i will be included in the current path, but at a later stage;
(h_3): v_i is permanently excluded from the current path.

At nodes descending from decisions (h_1) or (h_3), v_i will never be reconsidered, while in those descending from decision (h_2) it will reappear in the decision process (unless fathoming occurs). At that point, clearly, only two nodes corresponding to (h_1) and (h_2) will be generated.

In order to formally describe this process, we introduce the following partition of V at every node h of the search tree:
$I(h)$: *included* vertices $(=\{v_{t_1} \equiv v_1, v_{t_2}, \dots, v_{t_h}\})$;
$E(h)$: *excluded* vertices;
$W(h)$: *waiting* vertices (for which decision (h_2) was taken);
$F(h)$: *free* vertices $(= V \setminus (I(h) \cup E(h) \cup W(h)))$.
(If h is the root node, then $I(h)=\{v_1\}$, $E(h)=W(h)=\emptyset$.)

In order to prevent cyclic selection of waiting vertices for branching, a vertex is labelled as soon as decision (h_2) is taken about it. Only free and unlabelled waiting vertices are selected for branching. Taking decision (h_1) for a vertex results in the unlabelling of all the waiting vertices. Hence if

$$P(h)=F(h) \cup \{v_j \in W(h): v_j \text{ is unlabelled}\} \tag{11}$$

is empty, node h can be fathomed. Otherwise define the set of dominated vertices as $D(h)=\{v_j \in P(h): v_j$ is dominated by $v_i \in P(h)\}$ and select v_i from $P(h) \setminus D(h)$ so as to yield the best improvement to the current solution in the following sense. Since $l(h)+c_{t_h,1}$ is the length of the tour implied by $I(h)$, v_i will be such that

$$\frac{p_i}{c_{t_h,i}+c_{i1}-c_{t_h,1}} = \max\left\{\frac{p_j}{c_{t_h,j}+c_{j1}-c_{t_h,1}} : v_j \in P(h) \setminus D(h)\right\}.$$

Vertex v_i belongs either to $F(h)$ or $W(h)$. In the former case, the three nodes h_1, h_2, h_3 generated from h are characterized as follows:

Node h_1: Let $U(h_1)=\{v_j \in F(h) \cup W(h): l(h)+c_{t_h,i}+c_{ij}+c_{j1}>c\}$ be the set of vertices which are now unreachable due to the inclusion of v_i. If $U(h_1) \cap W(h) \neq \emptyset$, then node h_1 can be fathomed; otherwise:

$I(h_1) := I(h) \cup \{v_i\}$;
$E(h_1) := E(h) \cup U(h_1)$;
$W(h_1) := W(h)$ and unlabel all vertices of $W(h_1)$;
$F(h_1) := F(h) \setminus (\{v_i\} \cup U(h_1))$.

Node h_2:
$I(h_2) := I(h)$;
$E(h_2) := E(h)$;
$W(h_2) := W(h) \cup \{v_i\}$ and label v_i;
$F(h_2) := F(h) \setminus \{v_i\}$.

Node h_3:
$I(h_3) := I(h)$;
$E(h_3) := E(h) \cup \{v_i\}$;
$W(h_3) := W(h)$;
$F(h_3) := F(h) \setminus \{v_i\}$.

If $v_i \in W(h)$, then node h_3 is not created, while for nodes h_1 and h_2, the same operations apply, except:
$W(h_1) := W(h) \setminus \{v_i\}$ and unlabel all vertices of $W(h_1)$;
$F(h_1) := F(h) \setminus U(h_1)$;
$W(h_2) := W(h)$ and label v_i;
$F(h_2) := F(h)$.

4.2. *Fathoming criteria*

4.2.1. *Upper bound computation*

For both partitioning schemes, at each decision node h generated, we compute, through Theorem 1, an upper bound $\bar{z}(h)$ on the solution value attainable from the subproblem corresponding to the node. In order to provide a unified description, let us define $W(h) = \emptyset$ and $P(h) = F(h)$ (see equation (11)) for scheme P1. Then the upper bound computation at node h can be described as follows.

First note that the current path $I(h)$ can be considered as a "super-vertex" $\bar{v}$ with profit $\bar{p} = \sum_{v_j \in I(h)} p_j$ and weight $\bar{w} = l(h) + (\alpha \min_{v_i \in F(h) \cup W(h)} \{c_{i1}\} + (1-\alpha) \min_{v_k \in P(h)} \{c_{t_h,k}\})$ for some α $(0 \le \alpha \le 1)$. For all $v_j \in F(h) \cup W(h)$, define $A(j) = F(h) \cup W(h) \setminus \{v_j\}$; the weights of these vertices are then $w_j = \alpha \min_{v_i \in A(j) \cup \{v_{t_h}\}} \{c_{ij}\} + (1-\alpha) \min_{v_k \in A(j) \cup \{v_1\}} \{c_{jk}\}$. If $\bar{w} + \sum_{v_j \in W(h)} w_j > c$, the node can be fathomed. Otherwise, the required $\bar{z}(h)$ is any upper bound on the solution of the following KP:

maximize

$$z(h) = \left(\bar{p} + \sum_{v_j \in W(h)} p_j\right) + \sum_{v_j \in F(h)} p_j y_j,$$

subject to

$$\sum_{v_j \in F(h)} w_j y_j \le c - \left(\bar{w} + \sum_{v_j \in W(h)} w_j\right), \tag{12}$$

$$y_j \in \{0, 1\}, \quad v_j \in F(h).$$

Let p^* denote the value of the best STSP solution so far obtained. Then, clearly, node h can be fathomed if $\bar{z}(h) \le p^*$.

When $\alpha = 0$ or 1, the above bound can be strengthened by exploiting, in the computation of the w_j, the fact that the vertices v_j belonging to C^* must be part of a tour. Consider for example the case where $\alpha = 0$. Since the two arcs $(v_{k'}, v_j)$ and $(v_{k''}, v_j)$ cannot both enter the optimal tour if $k' \neq k''$, it is then valid to impose that the indices k yielding $\min_{v_k \in A(j) \cup \{v_1\}} \{c_{jk}\}$ in the computation of the w_j must all be different. Formally, we must determine the best combination of k's by solving a *transportation problem* (*TP*) where $J(h) = W(h) \cup \{v_{t_h}\}$ and $K(h) = W(h) \cup F(h) \cup \{v_1\}$:

minimize

$$\sum_{v_j \in J(h)} \sum_{v_k \in K(h) \setminus \{v_j\}} c_{jk} x_{jk},$$

subject to

$$\sum_{v_k \in K(h) \setminus \{v_j\}} x_{jk} = 1, \quad v_j \in J(h), \tag{13}$$

$$\sum_{v_j \in J(h) \setminus \{v_k\}} x_{jk} \le 1, \quad v_k \in K(h), \tag{14}$$

$$x_{jk} \in \{0, 1\}, \quad j \neq k, \ v_j \in J(h), \ v_k \in K(h).$$

Let $t^*(\alpha)$ denote the value of the optimal TP solution. Node h can then be fathomed if $l(h) + t^*(\alpha) > c$ and, in the determination of $\bar{z}(h)$, the right-hand side of (12) can be replaced by $c - (l(h) + t^*(\alpha))$. Consider in fact any feasible solution to the STSP including the current path, all the vertices of $W(h)$ and possibly some vertices of $F(h)$: the total length of the arcs emanating from vertices of $I(h) \cup W(h)$ cannot be less than $l(h) + t^*(\alpha)$, while the cost of each arc emanating from a vertex $v_j \in F(h)$ is no less than the corresponding w_j. Hence this solution is also feasible for the KP since (12) is automatically satisfied.

A similar reasoning applies to the case where $\alpha = 1$. The corresponding TP is then obtained by redefining $J(h) = W(h) \cup F(h) \cup \{v_{t_h}\}$, $K(h) = W(h) \cup \{v_1\}$ and by interchanging the "$=$" and "$\le$" signs in (13) and (14).

4.2.2. *Additional criteria for scheme P2*

For the second partitioning scheme, node h can also be fathomed if:

(a) $P(h) = \emptyset$ (see Section 4.1); or

(b) $W(h)$ contains an unreachable node, i.e., if $l(h) + c_{t_h, j} + c_{j1} > c$ for at least one $v_j \in W(h)$; or

(c) $W(h) \neq \emptyset$ and a lower bound $b(h)$ on the length of a tour including the current path as well as all the vertices of $W(h)$ exceeds c. In order to compute such a bound, consider the super-vertex $\bar{v}$ previously introduced and define the distance matrix $C(h)$ induced by $\{\bar{v}\} \cup W(h)$ as follows. Given the cost matrix (c_{ij}), replace the entries of row 1 by the corresponding entries of row t_h, remove all rows and columns j such that $v_j \notin W(h) \cup \{v_1\}$ and set all diagonal entries equal to infinity. In the resulting submatrix, the super-vertex is associated with row and column 1, so the entries c_{1j} for which v_j is labelled can also be set to infinity. The value of $b(h)$ can then be taken as $l(h)+\tau(h)$, where $\tau(h)$ is the optimal solution of the TSP defined by $C(h)$. Since, however, TSP is NP-hard, $\tau(h)$ can be replaced by any lower bound $\underline{\tau}(h)$. We have determined $\underline{\tau}(h)$ as the optimal solution to the *min-sum assignment problem* defined by $C(h)$ and which can be solved in $O(n^3)$ time. (For a recent survey on assignment problem codes, see Carpaneto, Martello and Toth [1]. In our implementation, we used the APC Fortran code [1].) Therefore node h can be fathomed if $l(h)+\underline{\tau}(h)>c$.

5. Computational results

The algorithms described in this paper were tested on a number of problems derived from complete directed and undirected graphs. Distances and profits were generated according to a discrete uniform distribution on [1, 100] and the shortest path lengths c_{ij} were then computed. The value of c was set equal to $\beta\tau^*$ where τ^* is the optimal value of the TSP defined by (c_{ij}) and β, a parameter in (0, 1] controlling the problem tightness: a low value means that few vertices will be included in C^* whereas a value close to 1 indicates that almost all vertices will appear in the optimal solution.

All problems were solved on a VAX 11/780 computer. The maximum CPU time allowed for a single problem was 100 seconds.

Limited tests were first conducted on undirected graphs, using the two ILP based algorithms. Problems involving up to 20 vertices were solved to optimality but for larger sizes, computing times grew quickly and in almost all cases, the time limit was reached. Problems with a larger value of β were, as a rule, easier to solve: the relaxed problem tended to contain relatively more variables having an integer value, thus reducing the expansion of the branch and bound tree. Of the two versions of the algorithm, the first one fared better: it is generally more efficient to check for violated subtour elimination constraints at integer solutions only.

The two enumerative algorithms (corresponding to branching schemes P1 and P2) were then applied to a number of problems associated with directed and undirected graphs. Ten random problems were generated and attempted for several combinations of β between .1 and .9 and of $n \geq 10$. In all cases, the KP bounds were computed with $\alpha = \frac{1}{2}$ after having observed, on a series of tests, that bounds generated with $\alpha = 0$ or 1 were on the average weaker. The first branching scheme was clearly faster than the second and is the only one for which results are reported. Tables 1 and 2 give average results over the successful problems when at least 5 problems out of 10 could be solved to optimality within the set time limit. Otherwise, no results are reported.

Table 1. Computational results for directed graphs.

		Heuristic		Branch and Bound	
β	n	Time	Heuristic/Optimum	Time	Nodes
.1	10	.006	1.000	.002	1
	15	.004	1.000	.005	1
	20	.006	1.000	.005	1
	25	.008	.988	.005	1
	30	.007	.988	.009	1
	40	.011	.910	.029	4
	50	.013	.947	.092	8
	60	.019	.935	.355	21
	70	.028	.788	1.394	68
	80	.041	.822	4.179	119
	90	.064	.799	18.919	473
.2	10	.005	.996	.006	1
	15	.006	1.000	.005	1
	20	.010	.897	.030	5
	25	.013	.902	.142	18
	30	.014	.843	.245	27
	40	.032	.883	3.339	185
	50	.045	.797	33.861	1285
.3	10	.005	.952	.011	2
	15	.008	.940	.031	6
	20	.013	.909	.218	27
	25	.021	.843	1.574	144
	30	.027	.810	4.808	387
	40	.050	.871	63.666 (5)	2588
.4	10	.008	.991	.020	4
	15	.011	.899	.124	17
	20	.017	.882	1.492	158
	25	.025	.861	14.054	892
	30	.035	.794	31.367 (6)	1699
.5	10	.010	.911	.054	13
	15	.013	.953	.465	44
	20	.021	.933	7.579	621
	25	.032	.867	47.658 (7)	2785
.6	10	.009	.953	.090	17
	15	.014	.899	1.177	103
	20	.023	.940	32.493 (9)	2344
.7	10	.010	.928	.180	35
	15	.015	.929	3.091	256
.8	10	.010	.974	.288	49
	15	.016	.950	6.155	415
.9	10	.011	.961	.444	69
	15	.019	.956	11.877	951

Table 2. Computational results for undirected graphs.

		Heuristic		Branch and Bound		
β	n	Time	Heuristic/Optimum	Time		Nodes
.1	10	.003	1.000	.007		1
	15	.005	.951	.007		1
	20	.007	1.000	.033		4
	25	.007	.953	.013		2
	30	.011	.899	.054		8
	40	.019	.914	.380		39
	50	.016	.930	.682		52
	60	.032	.789	13.144	(9)	652
	70	.040	.851	30.518	(5)	1172
.2	10	.007	.993	.008		2
	15	.007	.999	.023		3
	20	.011	.931	.557		71
	25	.013	.888	.345		43
	30	.018	.868	1.992	(9)	180
	40	.030	.858	35.490	(6)	2001
.3	10	.004	.987	.022		4
	15	.011	.937	.121		18
	20	.014	.967	10.441		1147
	25	.020	.862	13.626		1075
	30	.026	.839	31.064	(5)	2252
.4	10	.006	.945	.044		9
	15	.014	.975	.634		84
	20	.016	.936	10.048	(8)	978
	25	.022	.853	31.310	(6)	2365
.5	10	.009	.961	.082		15
	15	.014	.970	2.802		332
	20	.028	.945	49.062	(6)	3304
.6	10	.010	1.000	.180		32
	15	.015	.981	10.821		1150
.7	10	.008	.993	.286		41
	15	.017	.968	22.400	(9)	2112
.8	10	.011	.975	.542		80
	15	.019	.980	50.102		3861
.9	10	.010	1.000	.994		156

The various column headings can be interpreted as follows.

– Heuristic time: average CPU time required to run the two heuristic algorithms described in Section 3;

– Heuristic/Optimum: Best heuristic solution value divided by the value of the optimum;

- Time ( ): average CPU time required to run the branch and bound algorithm followed, in brackets, by the number of successful problems when this is less than 10;
- Nodes: Number of nodes generated during the branch and bound process.

Computational results indicate that the combination of the two heuristics is quite efficient. It often provided the optimum or a solution within a few percentage points of the optimum in insignificant CPU time. This is particularly true in problems where n is small. As expected, problem difficulty increases with β as in "loose" problems, upper bounds become less sharp and fathoming criteria less effective. Problems generated from directed graphs were easier to solve than symmetrical problems. This observation is not particular to the STSP, but also applies to several related routing problems (the TSP, for example).

Overall, our approach appears to have been quite successful. To our knowledge, this is the first time exact algorithms are published for this difficult problem. Depending on problem characteristics, instances involving between 10 and 90 vertices were solved to optimality.

Acknowledgement

This work was supported by the Canadian Natural Science and Engineering Research Council (grant A4747) and by the Ministero della Pubblica Istruzione, Italy. We thank Dr. Matteo Fischetti and Dr. Antonio Mantani for helpful discussions. Thanks are also due to Hélène Mercure for her assistance with programming. The title of this paper was suggested by Prof. John Stafford from the University of Southampton.

References

[1] G. Carpaneto, S. Martello and P. Toth, Algorithms and codes for the assignment problem, in: B. Simeone, P. Toth, G. Gallo, F. Maffioli and S. Pallottino, eds., Fortran Codes for Network Optimization, Annals of Operations Research 13 (Baltzer, Basel, 1988).

[2] M.R. Garey and D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Complete Problems (Freeman, New York, 1979).

[3] B.L. Golden, L. Levy and R. Vohra, Some heuristics for the generalized traveling salesman problem, Working Paper MS/S 85-036, College of Business and Management, University of Maryland (1985).

[4] B.L. Golden, L. Levy and R. Vohra, The orienteering problem, Naval Res. Logist. 34 (1987) 307–318.

[5] B.L. Golden, Q. Wang and L. Liu, A multi-faceted heuristic for the orienteering problem, Naval Res. Logist. 35 (1988) 359–366.

[6] M. Hayes and J.M. Norman, Dynamic programming in orienteering: route choice and siting of controls, J. Oper. Res. Soc. 35 (1984) 791–796.

[7] G. Laporte, Generalized subtour elimination constraints and connectivity constraints, J. Oper. Res. Soc. 37 (1986) 509–514.

[8] S. Martello and P. Toth, An upper bound for the zero-one knapsack problem and a branch and bound algorithm, European J. Oper. Res. 1 (1977) 169–175.

[9] S. Martello and P. Toth, Algorithms for knapsack problems, in: S. Martello, G. Laporte, M. Minoux and C. Ribeiro, eds., Surveys in Combinatorial Optimization, Annals of Discrete Mathematics 31 (North-Holland, Amsterdam, 1987) 213–257.

[10] D.J. Rosenkrantz, R.E. Stearns and P.M. Lewis II, An analysis of several heuristics for the traveling salesman problem, SIAM J. Comput. 6 (1977) 563–581.

[11] T. Tsiligirides, Heuristic methods applied to orienteering, J. Oper. Res. Soc. 35 (1984) 797–809.