Vehicle Routing: Methods and Studies
B.L. Golden and A.A. Assad (Editors)

AN ADDITIVE APPROACH FOR THE OPTIMAL SOLUTION OF THE PRIZE-COLLECTING TRAVELLING SALESMAN PROBLEM*

Matteo Fischetti and Paolo Toth
D.E.I.S., University of Bologna, Italy

Consider the following generalization of the well-known Travelling Salesman Problem. Given a depot, at which a vehicle is stationed, and a set of cities, each having an associated non-negative *prize* p_i, let $c_{i,j}$ be the cost of routing city j just after city i and γ_i be the cost of leaving city i unrouted. The *Prize-Collecting Travelling Salesman Problem* (*PC-TSP*) is to find a minimum-cost route for the vehicle, visiting each city at most once and collecting a total prize not less than a given *goal* g. Such a problem arises in several routing and scheduling applications and belongs to the class of $\mathcal{NP}$-hard problems. In this paper, we introduce several mathematical models of the problem and point out its main substructures. Additive bounding procedures are then designed yielding sequences of increasing lower bounds on the optimum value of the problem. Extensive computational results on randomly generated test problems are reported, comparing the performances of the proposed bounding procedures. A branch and bound algorithm for the optimal solution of *PC-TSP* is finally described and computationally analyzed.

1. Introduction

Consider the following routing problem, which generalizes the well-known *Travelling Salesman Problem* (*TSP*). Given a depot at which a vehicle is stationed, and a set of cities, each having an associated non-negative *prize* p_i, let $c_{i,j}$ be the cost of routing city j just after city i and γ_i be the cost of leaving city i unrouted. The *Prize-Collecting Travelling Salesman Problem* (*PC-TSP*) is to find a minimum-cost route for the vehicle, visiting each city at most once and collecting a total prize not less than a given *goal* g.

Such a problem arises, for instance, when a factory (located at city 1) needs a given amount g of a product, which can be provided by a set of suppliers (located at cities $2, \cdots, n$).

* Work supported by the Ministero della Pubblica Istruzione, Italy

vehicle has to collect the maximum possible prize p^* through a route whose total cost does not exceed a given bound $\bar{c}$. Such a problem could be optimally solved through binary search for the maximum feasible prize p^*, exploiting the property that for each total prize p we have $p^* \geq p$ iff there exists a feasible solution to the instance of *PC-TSP* with goal $g = p$, whose cost does not exceed $\bar{c}$. The existence of such a solution can be checked by optimally solving the corresponding *PC-TSP* instance. Heuristic algorithms for the Orienteering Problem and some generalizations have been proposed by Tsiligirides (1984), Golden, Levy and Vohra (1985, 1987), Golden, Storchi and Levy (1986) and Golden, Wang and Liu (1987), while optimal methods are given in Laporte and Martello (1987).

In Section 2, integer linear programming models are given, pointing out different substructures of the problem. In Section 3 we propose several lower bounds, and combine them to obtain additive procedures yielding a combined bound which is generally superior to the individual ones used to produce it. Section 4 experimentally analyzes the performances of the proposed bounds on randomly generated test problems. A branch and bound algorithm is then described and computationally evaluated in Section 5.

2. Mathematical models

Several integer linear programming models can be given for *PC-TSP*. For each $i, j \in V$, $i \neq j$, let $x_{i,j}$ be a binary variable set to 1 if arc (i,j) is in the optimal solution, set to 0 otherwise. For each vertex $i \in V$, let y_i be a binary variable set to 1 if vertex i is routed, set to 0 otherwise. A first model is

$$(PC-TSP) \quad v(PC-TSP) = \min \sum_{i \in V} \sum_{j \in V \setminus \{i\}} c_{i,j}\, x_{i,j} + \sum_{i \in V} \gamma_i (1 - y_i) \tag{1}$$

subject to

$$y_i = \sum_{h \in V \setminus \{i\}} x_{h,i}\,, \qquad \text{for each } i \in V \tag{2}$$

$$\sum_{h \in V \setminus \{i\}} x_{i,h} = \sum_{h \in V \setminus \{i\}} x_{h,i}\,, \qquad \text{for each } i \in V \tag{3}$$

$$\sum_{i \in V} p_i\, y_i \geq g \tag{4}$$

$$\sum_{i \in S} \sum_{j \in V \setminus S} x_{i,j} \geq y_h\,, \qquad \begin{array}{c}\text{for each } h \in V \setminus \{1\} \text{ and for each} \\ S \subset V : 1 \in S, h \in V \setminus S\end{array} \tag{5}$$

$$\left.\begin{array}{ll} x_{i,j} \in \{0,1\}\,, & \text{for each } i, j \in V, i \neq j \\ y_i \leq 1\,, & \text{for each } i \in V \end{array}\right\}. \tag{6}$$

disjoint subtours (possibly visiting only one vertex) covering all the vertices. Constraints (12) impose an upper bound on the total prize of the unrouted vertices (i.e., of the vertices covered by loops in the *AP* solution), while constraints (13) ensure that the routed vertices can be reached from vertex 1. An example of a feasible solution to model (9) – (14) is given in Figure 1.

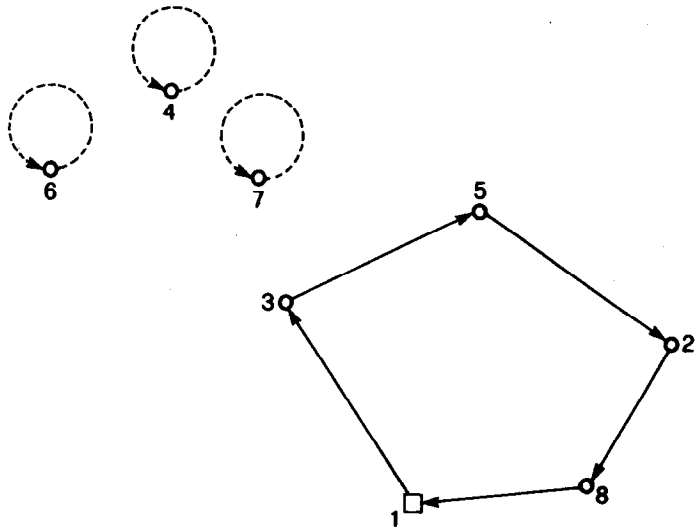

Figure 1. Feasible solution to model (9) – (14), assuming (p_j)=(0,8,5,2,4,5,3,6), $g = 22$.

It is also possible to point out the underlying arborescence problem substructure of *PC-TSP* by rearranging model (1) – (6) as follows. Let us consider an augmented complete graph $G' = (V', A')$, where $V' = V \cup \{0\}$, and for each $i \in V$ define $x_{0,i} = 1$ if vertex i is left unrouted, and $x_{0,i} = 0$ otherwise (i.e., $x_{0,i} = 1 - y_i$). Consequently, for each $i \in V$ we set $c_{0,i} = \gamma_i$ (the cost incurred if vertex i is left unrouted). In addition, we set $c_{i,0} = 0$ for each $i \in V \setminus \{1\}$, and $c_{0,0} = c_{1,0} = \infty$. *PC-TSP* can now be formulated as

$$(PC-TSP) \quad v(PC-TSP) = \min \sum_{i \in V'} \sum_{j \in V'} c_{i,j}\, x_{i,j} \tag{15}$$

subject to

$$\sum_{h \in V'} x_{h,i} = 1, \qquad \text{for each } i \in V' \tag{16}$$

$$\sum_{i \in S} \sum_{j \in V' \setminus S} x_{i,j} \geq 1, \qquad \text{for each } S \subset V' : 1 \in S \tag{17}$$

$$\sum_{h \in V} x_{i,h} = \sum_{h \in V} x_{h,i}, \qquad \text{for each } i \in V \tag{18}$$

$$\sum_{i \in V} p_i\, x_{0,i} \leq \sum_{j \in V} p_j - g \tag{19}$$

$$x_{i,j} \in \{0,1\}, \qquad \text{for each } i,j \in V'. \tag{20}$$

3.1. An additive lower bounding procedure

Let us suppose r bounding procedures $\mathcal{L}^{(1)}, \mathcal{L}^{(2)}, \cdots, \mathcal{L}^{(r)}$ are available for *PC-TSP*, each based on a different relaxation. Let $\tilde{I} = [\tilde{n}, (\tilde{c}_{i,j}), (\tilde{\gamma}_i), (\tilde{p}_i), \tilde{g}]$ be a given *instance* of *PC-TSP*, defined through input data $\tilde{n}$, $(\tilde{c}_{i,j})$, $(\tilde{\gamma}_i)$, $(\tilde{p}_i)$, and $\tilde{g}$, and let $v(\tilde{I})$ be the corresponding optimal solution value. Let us also suppose that, for $h = 1, 2, \cdots, r$ and for any instance $\tilde{I}$, procedure $\mathcal{L}^{(h)}(\tilde{I})$, when applied to $\tilde{I}$, returns a lower bound $\delta^{(h)}$ as well as a *residual instance* $I^{(h)} = [n^{(h)}, (c_{i,j}^{(h)}), (\gamma_i^{(h)}), (p_i^{(h)}), g^{(h)}]$ such that:

i) $v(I^h) \geq 0$;

ii) $\delta^{(h)} + v(I^h) \leq v(\tilde{I})$.

(For example, a residual instance $I^{(h)}$ associated with bound $\delta^{(h)}$ could be obtained by taking $n^{(h)} = \tilde{n}$, $(p_i^{(h)}) = (\tilde{p}_i)$, $g^{(h)} = \tilde{g}$, and by conveniently "reducing" the input costs $(\tilde{c}_{i,j})$ and $(\tilde{\gamma}_i)$ so as to produce non-negative *residual costs* $(c_{i,j}^{(h)})$ and $(\gamma_i^{(h)})$ satisfying condition ii)).

The additive approach generates a sequence of instances of problem *PC-TSP*, each obtained from the previous one by applying a different bounding procedure. A Pascal-like outline of the approach follows.

ALGORITHM ADD-PCTSP:

```
1. input instance I = [n, (c_{i,j}), (γ_i), (p_i), g];
2. output lower bound δ;
   begin
3.    initialize I^(0) := I, δ := 0;
4.    for h := 1 to r do
         begin
5.          apply L^(h)(I^(h-1)), thus obtaining value δ^(h) and the residual instance I^(h);
6.          δ := δ + δ^(h)
         end
   end.
```

In order to show that each value δ computed at step 6 is a valid lower bound for *PC-TSP*, we prove by induction on h that

$$\sum_{l=1}^{h} \delta^{(l)} + v(I^{(h)}) \leq v(I) \tag{21}$$

holds for $h = 1, 2, \cdots, r$. The basis, $h = 1$, is trivially true. Suppose now that inequality (21) holds for $h = \bar{h} - 1$. From conditions ii) on the residual instance, we have

$$\sum_{l=1}^{\bar{h}-1} \delta^{(l)} + \delta^{(\bar{h})} + v(I^{(\bar{h})}) \leq \sum_{l=1}^{\bar{h}-1} \delta^{(l)} + v(I^{(\bar{h}-1)})$$

$$\Delta = \sum_{j \in V} p_j - g\,,$$
$$\tilde{c}_{i,j}(\lambda) = c_{i,j}\,, \qquad \text{for } i,j \in V, i \neq j\,,$$
$$\tilde{c}_{i,i}(\lambda) = c_{i,i} + \lambda\, p_i\,, \qquad \text{for } i \in V\,.$$

$L(\lambda)$ can be efficiently computed by solving the instance of the assignment problem $AP(\lambda)$ defined by cost matrix $(\tilde{c}_{i,j}(\lambda))$. As for the solution of *LD-AP*, the maximization of $L(\lambda)$ over $\lambda \geq 0$ is obtained in an effective way by applying an iterative subgradient optimization technique specialized for the scalar multiplier case, yielding a sequence $\lambda_1, \lambda_2, \cdots$ of multipliers. A "good" value for λ_1 can be heuristically computed through the solution of the following continuous *Knapsack Problem*:

$$\min \sum_{i=1}^{n} a_i\, y_i$$

subject to

$$\sum_{i=1}^{n} p_i\, y_i \geq g$$
$$0 \leq y_i \leq 1\,, \qquad \text{for } i = 1, \cdots, n\,,$$

where $a_i = \min\{c_{h,i} : h = 1, \cdots, n, h \neq i\} - \gamma_i$ is a lower bound on the extra cost incurred for visiting vertex i instead of leaving it unvisited $(i = 1, \cdots, n)$. Value λ_1 is set equal to the optimal dual multiplier associated with constraint $\sum_{i=1}^{n} p_i\, y_i \geq g$.

Computing $L(\lambda_1)$ requires $O(n^3)$ time, e.g., through the Hungarian algorithm (see, e.g., Lawler (1976)). Computation of $L(\lambda_2), L(\lambda_3), \cdots$ can be speeded-up through parametric techniques which take advantage of the fact that, at each subgradient iteration, only coefficients $\tilde{c}_{i,i}(\lambda)$ over the main diagonal are affected by the change of λ with respect to the previous iteration.

Let $\lambda^* \geq 0$ be an optimal (or near optimal) solution of *LD-AP*. Consider the assignment problem $AP(\lambda^*)$ defined by cost matrix $(\tilde{c}_{i,j}(\lambda^*))$, and let $(u_i^*) - (v_j^*)$ be the optimal solution of the associated linear programming dual problem. It is well known that $\sum_{i \in V}(u_i^* + v_i^*)$ gives the optimal solution value of $AP(\lambda^*)$. Hence $L(\lambda^*) = -\lambda^* \Delta + \sum_{i \in V}(u_i^* + v_i^*)$ is a valid lower bound for *PC-AP* and for *PC-TSP* as well. As for the corresponding residual instance $\overline{I}$, we define $\overline{I} = [\overline{n}, (\overline{c}_{i,j}), (\overline{\gamma}_i), (\overline{p}_i), \overline{g}]$, where $\overline{n} = n$, $(\overline{\gamma}_i) = (\overline{c}_{i,i})$, $(\overline{p}_i) = (p_i)$, $\overline{g} = g$, and

$$\overline{c}_{i,j} = \tilde{c}_{i,j}(\lambda^*) - u_i^* - v_j^* \geq 0\,, \qquad \text{for each } i,j \in V\,, \tag{22}$$

i.e., the *residual-cost matrix* $(\overline{c}_{i,j})$ is given by the non-negative *reduced-cost matrix* associated with the optimal solution to $AP(\lambda^*)$. In fact, consider an optimal solution to instance I and let $(x_{i,j})$ be the corresponding incidence matrix in formulation (9) – (14). Because of (10) and (11), we have:

$$L(\lambda^*) + \sum_{i \in V} \sum_{j \in V} \overline{c}_{i,j}\, x_{i,j} =$$

A second approach is to relax in a Lagrangean fashion the formulation of *PC-TSP* given by (15) – (20) by imbedding constraints (18), with $i \in V \setminus \{1\}$, and (19) in the objective function. The corresponding Lagrangean dual problem is

$$(LD-SSAP) \quad v(LD-SSAP) = \max\{L(\lambda,u) : \lambda \geq 0, u \in \Re^n, u_1 = 0\}$$

where

$$L(\lambda,u) = -\lambda\,\Delta + \min \sum_{i\in V'} \sum_{j\in V'} \tilde{c}_{i,j}(\lambda,u)x_{i,j} \tag{24}$$

subject to (16), (17), (20) and

$$\sum_{h\in V'} x_{1,h} = 1 \tag{25}$$

with

$$\Delta = \sum_{j\in V} p_j - g\,,$$
$$\tilde{c}_{i,j}(\lambda,u) = c_{i,j} + u_i - u_j\,, \qquad \text{for } i,j \in V\,,$$
$$\tilde{c}_{0,j}(\lambda,u) = c_{0,j} + \lambda\, p_j\,, \qquad \text{for } j \in V\,,$$
$$\tilde{c}_{i,0}(\lambda,u) = c_{i,0}\,, \qquad \text{for } i \in V'\,.$$

It is worth noting that u_1 is set to 0, since the corresponding constraint (18) (with $i = 1$) has not been removed, but transformed to (25) (recall $c_{1,0} = c_{0,1} = \infty$). $L(\lambda,u)$ can be efficiently computed in $O(n^2)$ time by solving an equivalent instance of the shortest-spanning 1-arborescence on G' defined by cost matrix $(c'_{i,j}(\lambda,u))$, where

$$c'_{i,j}(\lambda,u) = \tilde{c}_{i,j}(\lambda,u)\,, \qquad \text{for } i,j \in V', i \neq 1\,,$$
$$c'_{1,j}(\lambda,u) = \tilde{c}_{1,j}(\lambda,u) + M\,, \qquad \text{for } j \in V'\,,$$

with M a sufficiently large positive number (so as to ensure out-degree one for vertex 1).

The maximization of $L(\lambda,u)$ can be obtained by applying standard subgradient optimization techniques. Let $\lambda^* \geq 0$ and $u^* \in \Re^n$ be the optimal (or near optimal) solution of *LD-SSAP*. Hence, $L(\lambda^*,u^*)$ is a valid lower bound for *PC-TSP*.

The corresponding residual instance $\overline{I}$ can be defined as follows. It is well known that the problem given by (24), (16), (17), (25), and (20) is equivalent to its linear programming relaxation (see Edmonds (1967) and Fulkerson (1974)). Let (v_i^*), $(\mu^*(S))$ and α^* $(= M)$ be the optimal linear programming dual variables associated, respectively, with constraints (16), (17), and (25), when $\lambda = \lambda^*$ and $u = u^*$. Hence $L(\lambda^*,u^*) = -\lambda^*\,\Delta + (\sum_{i\in V'} v_i^* + \sum_{S\subset V':1\in S} \mu^*(S) - \alpha^*)$. We can now define $\overline{I} = [\overline{n},(\overline{c}_{i,j}),(\overline{\gamma}_i),(\overline{p}_i),\overline{g}]$, where $\overline{n} = n$, $(\overline{\gamma}_i) = (\overline{c}_{0,i})$, $(\overline{p}_i) = (p_i)$, $\overline{g} = g$, and

$$\overline{c}_{i,j} = c'_{i,j}(\lambda^*,u^*) - v_j^* - \sum_{\substack{S\in V':1\in S,\\ i\in S, j\in V'\setminus S}} \mu^*(S) \geq 0\,, \qquad \text{for } i \in V', j \in V\,.$$

is obtained as follows. Define $\overline{n} = n$, $(\overline{p}_i) = (p_i)$, and $\overline{g} = g$. Now let $\overline{I}^{(1)} = [\overline{n}, (\overline{c}_{i,j}^{(1)}), (\overline{\gamma}_i^{(1)}), (\overline{p}_i), \overline{g}]$ be the residual instance associated with $\delta_1(S_{h^*})$ computed over instance $I^{(1)}(S_{h^*})$ as described in Subsection 3.2 (with $\tilde{c}_{i,j}^{(1)} = \infty$ and $\tilde{\gamma}_i^{(1)} = \infty$ for $i,j \in V$ when $\sum_{i \in S_{h^*}} p_i < g$ and hence $\delta_1(S_{h^*}) = \infty$). So, $\delta_1(S_{h^*}) + v(\overline{I}^{(1)}) \leq v(I^{(1)}(S_{h^*}))$. Define $\overline{I}^{(2)} = [\overline{n}, (\overline{c}_{i,j}^{(2)}), (\overline{\gamma}_i^{(2)}), (\overline{p}_i), \overline{g}]$, where $\overline{\gamma}_i^{(2)} = \gamma_i$ for $i \in V$, and $\overline{c}_{i,j}^{(2)} = \min\{c_{i,j} + f_i - f_j, c_{i,j} - b_i + b_j\}$ for each $i,j \in V$. Values $\overline{c}_{i,j}^{(2)}$ are non-negative because of the definition of f_h and b_h as shortest path costs. Residual costs $(\overline{c}_{i,j})$ and $(\overline{\gamma}_i)$ of instance $\overline{I}$ can now be computed as: $\overline{c}_{i,j} = \min\{\overline{c}_{i,j}^{(1)}, \overline{c}_{i,j}^{(2)}\}$, $\overline{\gamma}_i = \min\{\overline{\gamma}_i^{(1)}, \overline{\gamma}_i^{(2)}\}$ for $i,j \in V$. In fact, $v(\overline{I}) \geq 0$ clearly holds. To prove that $\delta(S_{h^*}) + v(\overline{I}) \leq v(I)$, consider an optimal solution σ of instance I defined through arc set $A(\sigma)$ and the set of unrouted vertices $U(\sigma)$ (hence $v(I) = \sum_{(i,j) \in A(\sigma)} c_{i,j} + \sum_{i \in U(\sigma)} \gamma_i$). If solution σ routes only vertices in S_{h^*}, then $\delta(S_{h^*}) + v(\overline{I}) \leq \delta_1(S_{h^*}) + v(\overline{I}^{(1)}) \leq v(I^{(1)}(S_{h^*})) = v(I)$. If at least one vertex $k \in V \setminus S_{h^*}$ is routed in solution σ (i.e. $k \notin U(\sigma)$), a more complex analysis is needed. Partition arc set $A(\sigma)$ into arc sets A_1 and A_2, A_1 defining a path from vertex 1 to vertex k, and A_2 defining a path from vertex k to vertex 1. We have $\sum_{(i,j) \in A_1} (c_{i,j} + f_i - f_j) = f_1 - f_k + \sum_{(i,j) \in A_1} c_{i,j}$, and $\sum_{(i,j) \in A_2} (c_{i,j} - b_i + b_j) = b_1 - b_k + \sum_{(i,j) \in A_2} c_{i,j}$, where $f_1 = b_1 = 0$, and $f_k + b_k \geq \delta_2(S_{h^*})$ (since $k \in V \setminus S_{h^*}$). Hence $\delta(S_{h^*}) + v(\overline{I}) \leq \delta_2(S_{h^*}) + \sum_{(i,j) \in A(\sigma)} \tilde{c}_{i,j}^{(2)} + \sum_{i \in U(\sigma)} \overline{\gamma}_i^{(2)} \leq f_k + b_k + \sum_{(i,j) \in A_1} (c_{i,j} + f_i - f_j) + \sum_{(i,j) \in A_2} (c_{i,j} - b_i + b_j) + \sum_{i \in U(\sigma)} \gamma_i = \sum_{(i,j) \in A_1} c_{i,j} + \sum_{(i,j) \in A_2} c_{i,j} + \sum_{i \in U(\sigma)} \gamma_i = v(I)$.

3.5. Instance transformation

In this subsection we introduce a technique to transform an instance $I = [n, (c_{i,j}), (\gamma_i), (p_i), g]$ of *PC-TSP* to a residual one, obtaining a lower bound of value 0, but allowing the following procedures of the additive scheme to produce better bounds. Let $S_1, S_2, \cdots, S_m$ be a partition of V with $1 \in S_1$ and $\sum_{j \in S_1} p_j < g$. A "residual" instance $\overline{I} = [\overline{n}, (\overline{c}_{i,j}), (\overline{\gamma}_i), (\overline{p}_i), \overline{g}]$ of *PC-TSP*, defined on a reduced complete graph $\overline{G} = (\overline{V}, \overline{A})$ with $|\overline{V}| = \overline{n} = m$, can be built up as follows. Vertices $1, 2, \cdots, \overline{n}$ of $\overline{V}$ correspond, in graph G, to subsets $S_1, S_2, \cdots, S_m$, respectively. We define $\overline{g} = g$, $\overline{p}_i = \sum_{j \in S_i} p_j$ and $\overline{\gamma}_i = \sum_{j \in S_i} \gamma_j$ for $i = 1, 2, \cdots, \overline{n}$. The residual costs $\overline{c}_{i,j}$ $(i,j \in \overline{V})$ are obtained in two steps.

Step 1 (*Shrinking*): define $c'_{i,j} = \min\{c_{h,k} : h \in S_i, k \in S_j\}$ for $i,j \in \overline{V}$, $i \neq j$;

Step 2 (*Compression*): let $L = \{i \in \overline{V} : |S_i| \geq 2\}$, and define $\overline{c}_{i,j}$ = cost of the shortest path from vertex i to vertex j, computed with respect to costs $c'_{i,j}$ and allowing as intermediate vertices only those in L, for $i,j \in \overline{V}$, $i \neq j$.

Step 1 requires $O(n^2)$ time, while step 2 requires $O(\overline{n}^2 |L|)$ time through a straightforward

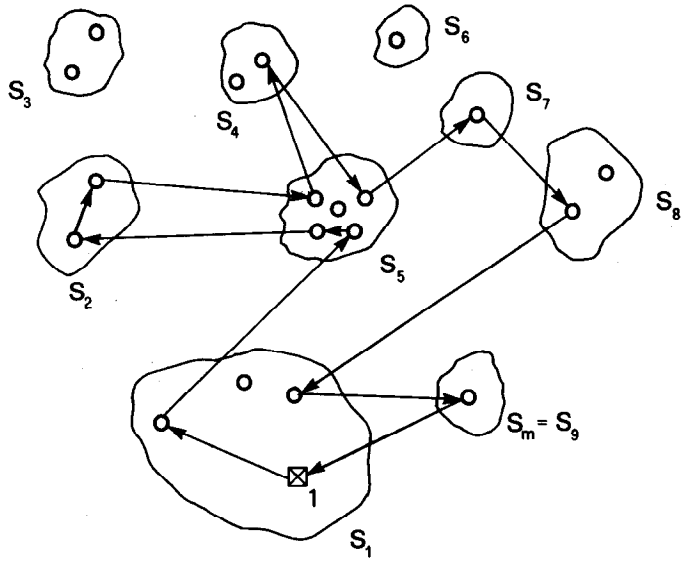

Figure 3a. A feasible solution σ of instance I.

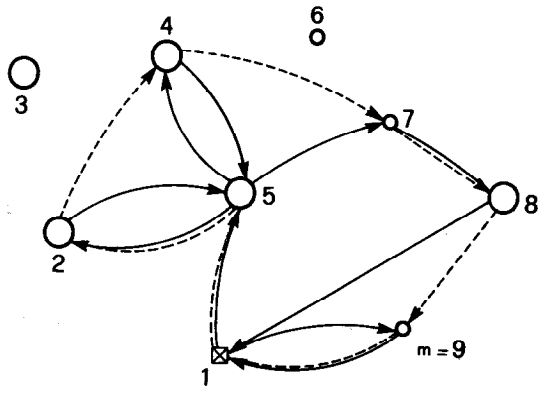

Figure 3b. Solutions $\tilde{\sigma}$ (in continuous line) and $\bar{\sigma}$ (in dotted line) of instance $\bar{I}$.

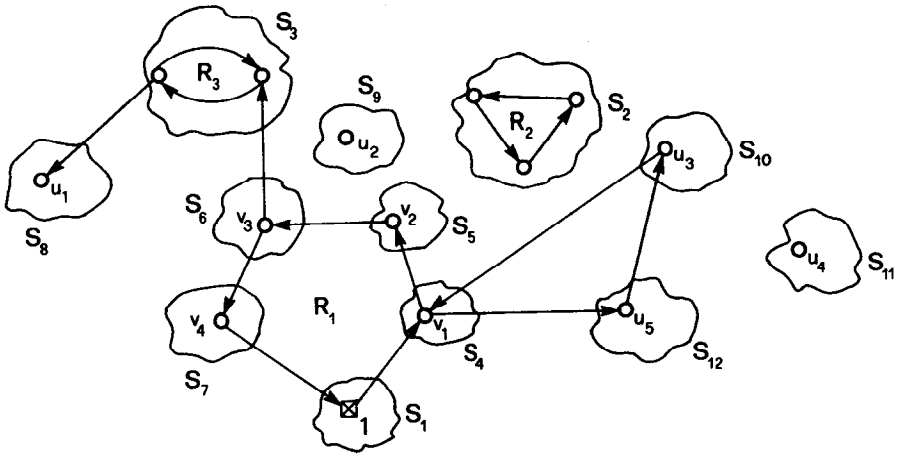

Figure 4a. Current instance $\bar{I}$ and the corresponding partition $S_1, \cdots, S_m$ (with $m = q + t + r = 3 + 4 + 5 = 12$); only arcs having zero residual-cost are drawn.

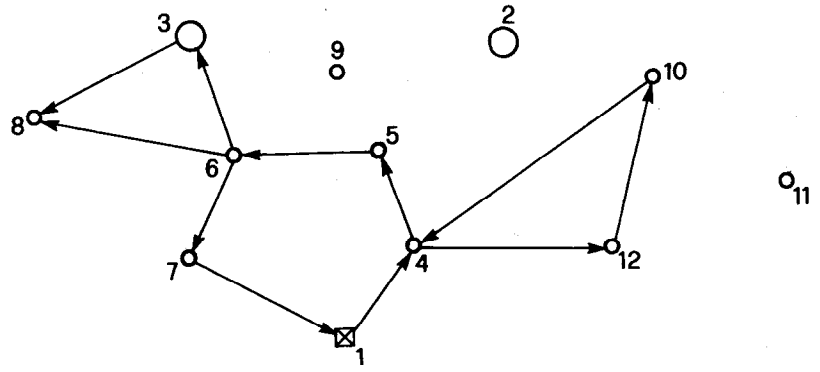

Figure 4b. Transformed instance (only arcs having zero residual-cost are drawn).

(n =20,40,60,80,100) have been considered. Note that when $\alpha = 1.0$, *PC-TSP* reduces to *TSP*.

Tables 1 and 2 compare bounding procedures LB1, LB2, LB3, LB4 and LB5 on test problems of classes A and C, respectively (results corresponding to test problems of class B are not given since they are equivalent to those of class A). Each table gives, for each bounding procedure, the average values (computed over five instances) of the ratios (lower bound)/(lower bound computed by LB1), and of the computing times (in VAX 11/780 seconds).

Table 1

Class A (pure asymmetric cost matrices)

Lower bounds comparison (time is given in VAX 11/780 seconds)

α	n	LB1		LB2		LB3		LB4		LB5	
		ratio	time	ratio	time	ratio	time	ratio	time	ratio	time
0.2	20	1.000	0.05	0.949	0.49	1.345	0.03	1.137	0.09	1.142	0.08
	40	1.000	0.20	0.985	1.32	1.160	0.15	1.125	0.38	1.109	0.27
	60	1.000	0.37	0.899	2.58	1.033	0.52	1.007	0.55	1.020	0.51
	80	1.000	1.00	0.853	4.43	1.002	2.59	1.021	1.63	1.018	1.25
	100	1.000	1.65	0.850	6.89	0.982	3.44	1.015	3.36	1.009	2.07
0.5	20	1.000	0.06	1.022	0.49	1.110	0.07	1.097	0.11	1.076	0.08
	40	1.000	0.25	0.983	1.32	0.854	0.32	1.053	0.34	1.034	0.35
	60	1.000	0.67	0.954	2.88	0.619	0.68	1.006	0.79	1.001	0.89
	80	1.000	1.32	0.928	4.70	0.518	1.31	1.005	1.91	1.004	1.62
	100	1.000	2.56	0.942	6.90	0.430	2.20	1.006	3.37	1.001	3.49
0.8	20	1.000	0.06	1.027	0.54	0.736	0.05	1.049	0.12	1.049	0.09
	40	1.000	0.31	0.987	1.34	0.491	0.18	1.013	0.41	1.011	0.42
	60	1.000	0.88	0.977	3.00	0.294	0.60	1.004	1.11	1.001	1.20
	80	1.000	1.89	0.982	5.01	0.258	1.43	1.005	2.67	1.001	2.47
	100	1.000	3.02	0.977	7.05	0.204	1.98	1.003	3.90	1.001	3.82
1.0	20	1.000	0.02	0.938	0.67	0.382	0.02	1.002	0.04	1.002	0.03
	40	1.000	0.06	0.941	1.68	0.255	0.04	1.000	0.11	1.000	0.10
	60	1.000	0.19	0.942	3.36	0.153	0.08	1.000	0.26	1.000	0.26
	80	1.000	0.45	0.972	5.46	0.141	0.14	1.001	0.63	1.001	0.59
	100	1.000	0.74	0.924	7.82	0.102	0.21	1.000	0.93	1.000	0.93

The computing times of procedure LB1, and hence those of LB3, LB4 and LB5, are considerably shorter when $\alpha = 1.0$ since, in this case, no unidimensional subgradient iteration is required (the optimal value of λ being ∞).

improvement given by LB4 and LB5 decreases when α tends to 1. This is mainly due to the decrease of the gap between the optimal solution value and the bound computed by LB1 (see Table 3 of Section 5). In addition, procedures LB4 and LB5 have been designed for *PC-TSP* , rather than for the pure TSP (more effective additive bounding procedures for both the Asymmetric and the Symmetric TSP have been proposed by Fischetti and Toth (1987a) and Carpaneto, Fischetti and Toth (1987), respectively).

According to the computational results above, a good overall bounding procedure to be imbedded within a branch and bound algorithm can be obtained by applying LB5 for small values of both n and the ratio $g/\sum_{i=1}^{n} p_i$, and LB4 otherwise.

5. A branch and bound algorithm

We now describe a simple branch and bound algorithm for the optimal solution of *PC-TSP*, based on bounding procedures LB4 and LB5 (see Section 4). As for branching, we do not propose an "ad hoc" scheme, but use a straightforward adaptation of the subtour elimination scheme for the Asymmetric *TSP*, as implemented by Carpaneto and Toth (1980).

At each node ν of the branch decision tree, arcs in $I^{(\nu)}$ and $F^{(\nu)}$ are, respectively, imposed and forbidden in the current solution. For each arc $(i,j) \in F^{(\nu)}$, cost $c_{i,j}$ is set to ∞; for each arc $(i,j) \in I^{(\nu)}$, costs $c_{i,h}$ for $h \in V \setminus \{j\}$, $c_{h,j}$ for $h \in V \setminus \{i\}$, γ_i and γ_j are set to ∞. We apply bounding procedure LB5 if $g \leq 0.3 \sum_{i=1}^{n} p_i$ and $n \leq 50$, bounding procedure LB4 otherwise. Let σ^* be the solution of the assignment problem $AP(\lambda^*)$ (see Subsection 3.2), found by LB1 if procedure LB5 is applied, or by the first execution of Step 1 of *ADD*1 if procedure LB4 is applied. If the subtour of σ^* passing through vertex 1 collects a total prize not less than g, we evaluate (using the original costs $(c_{i,j})$ and (γ_i)) the cost of the corresponding feasible solution to *PC-TSP* and possibly update the incumbent optimal solution. In any case, we compare the current lower bound δ_ν with the cost of the incumbent optimal solution z^*. If $\delta_\nu \geq z^*$, node ν is fathomed and a backtracking step is performed. Otherwise, we choose in σ^* a subtour $(v_1, v_2, \cdots, v_h)$, with $h \geq 2$, having the minimum number of non-imposed arcs, and generate h descendent nodes $\nu_1, \nu_2, \cdots, \nu_h$ defined by:

$$\left.\begin{array}{ll} I^{(\nu_i)} = & I^{(\nu)} \cup \{(v_1, v_2), \cdots, (v_{i-1}, v_i)\} \\ F^{(\nu_i)} = & F^{(\nu)} \cup \{(v_i, v_{i+1})\} \end{array}\right\} \quad i = 1, \cdots, h,$$

where $v_{h+1} = v_1$. Nodes ν_i for which $I^{(\nu_i)} \cap F^{(\nu_i)} \neq \emptyset$ are clearly not generated.

The branch and bound algorithm has been implemented in FORTRAN ANSI, and run on a Digital VAX 11/780 by considering the test problems of Section 4. Table 3 gives the average values (computed over five instances) of the global computing time, of the number

based branch and bound algorithms.

Additional computational experience has been performed, for large-size instances of class A, on the Digital VAX 8650 computer of the Carnegie Mellon University of Pittsburgh (this computer turned out to be about 6 times faster than Digital VAX 11/780). For each goal ($\alpha = 0.2, 0.5, 0.8$), 3 different values of n ($n = 120, 160, 200$) have been considered. Table 4 gives the average results computed over 5 problems.

Table 4

Branch and bound algorithm (time is given in VAX 8650 seconds)

Large-size instances of class A

n	$\alpha = 0.2$			$\alpha = 0.5$			$\alpha = 0.8$		
	time	nodes	ratio	time	nodes	ratio	time	nodes	ratio
120	27.8	125	1.076	93.9	739	1.023	80.6	567	1.023
160	132.6	365	1.077	616.1	1994	1.020	309.9	1147	1.011
200	408.9	489	1.061	794.1	1531	1.023	664.5	1347	1.006

The performance of the branch and bound algorithm could be improved upon considerably, mainly for instances of classes B and C, by designing an "ad hoc" branching scheme and introducing dominance criteria to deal with almost-equivalent solutions, and reduction procedures to impose or exclude subsets of vertices from the optimal solution.

Acknowledgment

We thank the Graduate School of Industrial Administration of the Carnegie Mellon University, Pittsburgh, for the use of Digital VAX 8650.

References

E. Balas (1987), "The Prize-Collecting Traveling Salesman Problem", Research Report No. MSRR-539, GSIA, Carnegie Mellon University, Pittsburgh.

E. Balas and G. Martin (1985), "ROLL-A-ROUND: Software Package for Scheduling the Rounds of a Rolling Mill", Copyright Balas and Martin Associates, 104 Maple Heights Road, Pittsburgh.

Problem", forthcoming in *Naval Research Logistics.*

G. Laporte and S. Martello (1987), "The Selective Travelling Salesman Problem", Working Paper, EHEC, Universitè de Montreal, Quebec, Canada (presented at the ORSA/TIMS Meeting, New Orleans, May 1987).

E.L. Lawler (1976), *Combinatorial Optimization: Networks and Matroids*, Holt, Rinehart and Winston, New York.

T. Tsiligirides (1984), "Heuristic Methods Applied to Orienteering", *Journal of the Operational Research Society* 35, 797-809.

S. Warshall (1962), "A Theorem on Boolean Matrices", *Journal of ACM* 9, 11-12.

G. Carpaneto, M. Dell'Amico, M. Fischetti and P. Toth (1987), "A Branch and Bound Algorithm for the Multiple Depot Vehicle Scheduling Problem", Working Paper OR/87/2, DEIS, University of Bologna, Italy.

G. Carpaneto, M. Fischetti and P. Toth (1987), "New Lower Bounds for the Symmetric Travelling Salesman Problem", Working Paper OR/87/7, DEIS, University of Bologna, Italy.

G. Carpaneto, S. Martello and P. Toth (1987), "Algorithms and Codes for the Assignment Problem", forthcoming in G. Gallo, F. Maffioli, S. Pallottino, B. Simeone and P. Toth, eds., *Fortran Codes for Network Optimization*, Annals of Operational Research, J.C. Baltzer AG, Basel.

G. Carpaneto and P. Toth (1980), "Some New Branching and Bounding Criteria for the Asymmetric Travelling Salesman Problem", *Management Science* 26, 736-743.

N. Christofides (1972), "Bounds for the Traveling Salesman Problem", *Operations Research*, 20, 1044-1055.

J. Edmonds (1967), "Optimum Branchings", *Journal of Research of the National Bureau of Standards-B.Mathematics and Mathematical Physics* 71B, 233-240.

M. Fischetti and P. Toth (1986), "An Additive Bounding Procedure for Combinatorial Optimization Problems", Working Paper OR/86/4, DEIS, University of Bologna, Italy, forthcoming in *Operations Research*.

M. Fischetti and P. Toth (1987a), "An Additive Bounding Procedure for the Asymmetric Travelling Salesman Problem", Working Paper OR/87/5, DEIS, University of Bologna, Italy.

M. Fischetti and P. Toth (1987b), "An Efficient Algorithm for the Min-sum Arborescence Problem", Working Paper OR/87/6, DEIS, University of Bologna, Italy.

R.W. Floyd (1962), "Algorithm 97: Shortest Path", *Communications of ACM*, 5, 345.

D.R. Fulkerson (1974), "Packing Rooted Cuts in a Weighted Directed Graph", *Mathematical Programming* 6, 1-13.

B. Golden, L. Levy and R. Vohra (1985), "Some Heuristics for the Generalized Traveling Salesman Problem", *Proceedings of 1985 Southeast TIMS Conference* (J. Hammesfahr, ed.), Myrtle Beach, 168-170.

B. Golden, L. Levy and R. Vohra (1987), "The Orienteering Problem", *Naval Research Logistics* 34 (3), 307-318.

B. Golden, G. Storchi and L. Levy (1986), "A Time-Relaxed Version of the Orienteering Problem", *Proceedings of 1986 Southeast TIMS Conference* (J.A. Pope and A. Ardalar, eds.), Myrtle Beach, 35-37.

B. Golden, Q. Wang and L. Liu (1987), "A Multi-Faceted Heuristic for the Orienteering

of nodes generated in the branch decision tree, and of the ratio (optimum value)/(lower bound computed by LB1 at the root node). Instances of class C with $n \geq 60$ have not been tried because of excessive computing time.

Table 3

Branch and bound algorithm (time is given in VAX 11/780 seconds)

α	n	Class A			Class B			Class C		
		time	nodes	ratio	time	nodes	ratio	time	nodes	ratio
0.2	20	0.4	7	1.360	0.9	17	1.362	5.2	75	2.969
	40	3.6	25	1.300	21.8	144	1.300	166.0	645	3.589
	60	8.5	39	1.140	69.7	238	1.140	—	—	—
	80	44.9	77	1.102	267.5	386	1.098	—	—	—
	100	94.6	75	1.073	256.8	217	1.073	—	—	—
0.5	20	1.7	38	1.303	9.4	214	1.237	88.0	773	2.245
	40	17.7	154	1.122	116.6	1116	1.120	1377.1	2237	2.291
	60	67.4	365	1.059	485.1	2148	1.051	—	—	—
	80	174.3	412	1.032	677.7	1318	1.028	—	—	—
	100	283.7	461	1.040	753.0	897	1.034	—	—	—
0.8	20	3.6	115	1.153	22.6	859	1.109	216.1	2429	1.697
	40	40.4	457	1.057	102.6	1533	1.037	1193.6	2971	1.597
	60	164.6	1060	1.036	795.7	5045	1.024	—	—	—
	80	307.2	855	1.028	1079.8	5583	1.023	—	—	—
	100	253.8	468	1.015	932.3	2149	1.014	—	—	—
1.0	20	0.4	16	1.057	0.3	9	1.028	31.6	803	1.277
	40	3.7	65	1.017	2.2	30	1.015	463.8	2420	1.278
	60	7.0	76	1.012	1.2	10	1.002	—	—	—
	80	7.7	20	1.009	1.7	4	1.004	—	—	—
	100	41.3	138	1.006	5.7	12	1.001	—	—	—

Problems of class A can be solved within small computing times even for large instances. The computational effort increases with α, except for $\alpha = 1.0$ (the pure TSP case).

Problems of class B are harder than those of class A, although the gap between the optimal solution value and the lower bound at the root node is almost the same. This behaviour can be explained by considering that many solutions exist for these instances, with approximately the same cost, and that the branching rule used is not effective in dealing with such situations.

Problems of class C can be solved only for small values of n, due to the well-known difficulty of solving symmetric and Euclidean travelling salesman problems through AP-

Table 2

Class C (Euclidean symmetric cost matrices)

Lower bounds comparison (time is given in VAX 11/780 seconds)

α	n	LB1		LB2		LB3		LB4		LB5	
		ratio	time	ratio	time	ratio	time	ratio	time	ratio	time
0.2	20	1.000	0.04	1.101	0.35	2.368	0.03	2.133	0.18	2.250	0.05
	40	1.000	0.08	1.181	1.50	2.421	0.07	2.859	0.94	2.523	0.14
	60	1.000	0.16	1.250	3.05	3.517	0.15	4.267	2.13	3.665	0.32
	80	1.000	0.29	1.136	5.53	1.966	0.29	2.669	4.18	2.186	0.55
	100	1.000	0.43	1.253	9.18	1.669	0.42	2.498	7.44	1.893	0.82
0.5	20	1.000	0.04	1.148	0.55	1.603	0.05	1.772	0.18	1.707	0.07
	40	1.000	0.11	1.285	1.63	1.388	0.16	2.010	0.59	1.576	0.21
	60	1.000	0.23	1.343	3.42	1.424	0.50	2.119	1.32	1.743	0.50
	80	1.000	0.41	1.263	5.83	1.146	0.96	1.948	2.63	1.462	0.84
	100	1.000	0.63	1.184	10.73	1.046	1.56	1.788	4.29	1.301	1.25
0.8	20	1.000	0.05	1.150	0.70	1.198	0.07	1.433	0.14	1.322	0.08
	40	1.000	0.13	1.209	1.90	0.840	0.14	1.446	0.37	1.288	0.23
	60	1.000	0.29	1.280	4.13	0.880	0.41	1.510	0.90	1.274	0.60
	80	1.000	0.52	1.239	6.92	0.611	0.54	1.470	1.70	1.191	1.02
	100	1.000	0.84	1.255	9.78	0.568	0.86	1.480	3.05	1.148	1.71
1.0	20	1.000	0.02	1.039	0.93	0.637	0.02	1.188	0.06	1.116	0.03
	40	1.000	0.04	1.204	2.93	0.462	0.04	1.190	0.18	1.107	0.09
	60	1.000	0.11	1.199	6.57	0.433	0.08	1.167	0.40	1.084	0.21
	80	1.000	0.17	1.211	10.33	0.326	0.14	1.184	0.79	1.082	0.35
	100	1.000	0.29	1.238	15.89	0.296	0.21	1.204	1.45	1.065	0.56

Procedure LB2 provides lower bound values worse than those of LB4, with higher computing times.

Procedure LB3 shows good performance for problems with small values of α and n, but gives poor results in the other cases. This behavior can be explained by considering that shortest paths give a reliable approximation of the optimal tour only when both α and n are small.

Procedures LB4 and LB5 exhibit the best performances. In particular, LB4 requires longer computing times but provides tighter bounds, mainly for problems of class C.

Note that the improvements on the value of the bound obtained through procedures LB4 and LB5, with respect to LB1, are remarkable for problems of class C, mainly for small values of α. On the other hand, for problems of classes A and B, the extra computational effort with respect to LB1 is generally small (about 2 to 1 in the worst case). The bound

compression steps to produce an instance leading to an increase of the bound. A main difference is the relaxation used, at each iteration, to compute the extra bound (*LD-AP* instead of *AP*). In addition, the proof of correctness given by Christofides cannot be straightforwardly extended to prove the correctness of procedure *ADD*1 which, instead, immediately follows from the general additive approach.

4. An experimental analysis of lower bounds

The lower bounds proposed in the previous section have been computationally evaluated on different classes of randomly generated test problems.

We report an analysis of the following bounding procedures:

- LB1, computing the lower bound based on the assignment problem substructure (see Subsection 3.2);
- LB2, computing the lower bound based on the arborescence problem substructure (see Subsection 3.3), and imposing at most 30 subgradient iterations;
- LB3, computing the lower bound based on disjunction (see Subsection 3.4);
- LB4, computing the lower bound based on instance transformation (see procedure *ADD*1 of Subsection 3.6);
- LB5, applying in sequence procedures LB1 and LB3 according to the additive approach.

Analysis of other possible sequences of bounding procedures is not reported, since the higher computing times are generally not rewarded by significant improvements in the bound quality. All procedures have been implemented in FORTRAN ANSI, and run on a Digital VAX 11/780. Solution of the assignment problems has been obtained through procedure *APC* given in Carpaneto, Martello and Toth (1987), while solution of the arborescence problems was accomplished via the procedure given in Fischetti and Toth (1987b).

Test problems have been randomly generated according to the following distributions:

p_i = uniformly random integer in range (1,100) for $i = 2, \cdots, n$;

$g = \lfloor \alpha \sum_{i=2}^{n} p_i \rfloor$ (with α =0.2, 0.5, 0.8, 1.0).

We have considered $\gamma_i = 0$ for $i = 2, \cdots, n$, as in several practical applications.

Three different generations have been considered for costs $c_{i,j}$ $(i, j = 1, \cdots, n; i \neq j)$:

Class A (pure asymmetric): $c_{i,j}$ = uniformly random integer in range (1,1000);

Class B (triangularized asymmetric): $c_{i,j}$ = shortest path cost from i to j with respect to a pure asymmetric cost matrix;

Class C (Euclidean symmetric): $c_{i,j} = \lfloor \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \rfloor$, with (x_i), (y_i) uniformly random integers in range (1,100).

For each class (A,B,C) and each goal (α =0.2, 0.5, 0.8, 1.0), 5 different values of n

Procedure ADD1:

1. **input** instance $I = [n, (c_{i,j}), (\gamma_i), (p_i), g]$;
2. **output** lower bound $\delta 1$, residual instance $\overline{I} = [\overline{n}, (\overline{c}_{i,j}), (\overline{\gamma}_i), (\overline{p}_i), \overline{g}]$;

begin

3. initialize $\overline{I} := I$, $\delta 1 := 0$;

repeat

4. (**comment** step 1: *LD-AP* solution)
5. solve *LD-AP* over instance $\overline{I}$, thus obtaining lower bound $L(\lambda^*)$, the new residual instance $\overline{I}$ and the optimal solution σ^* of $AP(\lambda^*)$ (see Subsection 3.2);
6. $\delta 1 := \delta 1 + L(\lambda^*)$;
7. let $R_1, R_2, \cdots, R_q$ (with $1 \in R_1$ and $|R_h| \geq 2$ for $h = 1, 2, \cdots, q$) be the subtours in solution σ^* found at step 5, and let $1, v_1, \cdots, v_t$ be the vertices visited by subtour R_1, and $\pi = p_1 + \sum_{k=1}^{t} p_{v_k}$ be the total prize collected by R_1;
8. **if** $(q > 1)$ **or** $(\pi < g)$ **then**
9. **begin** (**comment** step 2: instance transformation)
10. let $u_1, \cdots, u_r$ be the vertices not visited by any subtour R_h $(h = 1, \cdots, q)$;
11. **if** $q > 1$ **then**
    define $S_1 := \{1\}$; $S_k := \{\text{vertices in } R_k\}$ for $k = 2, \cdots, q$; $S_{q+k} := \{v_k\}$ for $k = 1, \cdots, t$; $S_{q+t+k} := \{u_k\}$ for $k = 1, \cdots, r$; $m := q + t + r$
12. **else** define $S_1 := \{\text{vertices in } R_1\}$; $S_{1+k} := \{u_k\}$ for $k = 1, \cdots, r$; $m := r + 1$;
13. apply the instance transformation on $\overline{I}$ with respect to partition $S_1, \cdots, S_m$, obtaining the new residual instance $\overline{I}$

end

14. **until** $(q = 1)$ **and** $(\pi \geq g)$

end.

Lines 9 to 13 define an appropriate partition $S_1, S_2, \cdots, S_m$ of the vertex set of the current instance $\overline{I}$ (see Figure 4a) and the corresponding transformed instance (see Figure 4b). Each subtour R_k is collapsed into a single vertex, for which both the extra costs to route it and to leave it unrouted are generally greater than zero. However, this transformation leads to a "loss of information" both because of the shrinking step (degree one constraint is neglected for the vertices in R_k) and the compression step (costs are decreased). Hence we choose not to collapse subtour R_1 (if possible) so as to avoid a loss of information in the "neighbourhood" of vertex 1 (which probably contains the optimal solution).

Execution of step 5 can be accelerated through parametric techniques exploiting the closeness between two consecutive residual instances.

The approach can be viewed as a generalization of that proposed by Christofides (1972) for the Asymmetric Travelling Salesman Problem, both approaches using the shrinking and

modification of the Floyd-Warshall shortest-path algorithm (see Floyd (1962) and Warshall (1962)).

Residual costs $\bar{c}_{i,j}$ satisfy the triangle inequality $\bar{c}_{i,k} + \bar{c}_{k,j} \geq \bar{c}_{i,j}$ for each $i,j \in \overline{V}$, $i \neq j$, and for each $k \in L$.

We now prove that $\overline{I}$ is a valid residual instance associated with lower bound $\delta = 0$. Condition $v(\overline{I}) \geq 0$ trivially follows from hypotheses $c_{i,j} \geq 0$ and $\gamma_i \geq 0$ for each $i,j \in V$, which imply $\bar{c}_{i,j} \geq 0$ and $\bar{\gamma}_i \geq 0$ for each $i,j \in \overline{V}$. As for condition $v(\overline{I}) \leq v(I)$, it is enough to show that any feasible solution σ of instance I corresponds to a feasible solution $\bar{\sigma}$ of instance $\overline{I}$ such that the cost of $\bar{\sigma}$ does not exceed the cost of σ. Feasible solution $\bar{\sigma}$ can be obtained in two steps as follows. First, an intermediate solution $\tilde{\sigma}$ of instance $\overline{I}$ is built up, containing an arc $(i,j) \in \overline{A}$ for each arc (h,k) in σ with $h \in S_i$, $k \in S_j$ and $i \neq j$ (see Figure 3). Because of the definition of $(\overline{p}_i)$, the sum of the prizes of the vertices in $\overline{V}$ visited (one or more times) by solution $\tilde{\sigma}$ is not less than $\overline{g} = g$. Note however that solution $\tilde{\sigma}$ is generally infeasible, since some vertices could be visited more than once. The cost of solution $\tilde{\sigma}$ is less or equal to the cost of solution σ, because of the definition of $(\overline{\gamma}_i)$ and $(\bar{c}_{i,j})$ (recall that $\bar{c}_{i,j} \leq c'_{i,j}$ for each $i,j \in \overline{V}$; see the compression step). A feasible solution $\bar{\sigma}$ of instance $\overline{I}$ is then obtained from solution $\tilde{\sigma}$ by using *shortcuts*, i.e., by replacing the pair of arcs (i,k) and (k,j) of $\tilde{\sigma}$ with arc (i,j) so as to avoid visiting vertex k more than once (see Figure 3b). Because of the compression step, the cost of $\bar{\sigma}$ does not exceed the cost of $\tilde{\sigma}$, and hence that of σ.

3.6. An additive bounding procedure based on instance transformation

We now combine the bound based on the assignment problem substructure (Subsection 3.2) with the instance transformation of the previous subsection to obtain a first additive bounding procedure.

LD-AP is first solved, obtaining a residual instance $\overline{I}$ in which a family of zero-cost subtours exists. A further solution of *LD-AP* on $\overline{I}$ is clearly useless, since no increase of the bound can occur. However, applying the instance transformation (with an appropriate partition of the vertex set) on $\overline{I}$ generally yields a residual instance $\overline{\overline{I}}$ in which no family of zero-cost subtours collecting the goal exists. So, the solution of *LD-AP* on instance $\overline{\overline{I}}$ generally leads to an increase of the lower bound. The two steps (*LD-AP* solution – instance transformation) can be iterated until the assignment problem solution found on the current instance contains only one subtour collecting the goal, thus solving the current *PC-TSP*. The corresponding bounding procedure is given below.

So the residual-cost matrix $(\bar{c}_{i,j})$ is given by the non-negative reduced-cost matrix associated with the optimal solution of the instance of the shortest-spanning 1-arborescence defined by cost matrix $(c'_{i,j}(\lambda^*, u^*))$. This reduced cost matrix can be computed in $O(n^2)$ time (see Fischetti and Toth (1987b)). The proof of correctness, based on standard LP manipulations of the objective function, is similar to that of the previous subsection and is hence omitted.

3.4. A bound based on disjunction

Let us consider instance $I = [n, (c_{i,j}), (\gamma_i), (p_i), g]$ of *PC-TSP*, and let σ be any feasible solution to I. For each vertex subset $S \subset V$ with $1 \in S$, two cases can occur:

1) solution σ routes only vertices in S;

2) solution σ routes at least one vertex in $V \setminus S$.

A valid lower bound $\delta(S)$ for instance I is then given by the minimum between the lower bounds associated with the two problems, say P_1 and P_2, obtained by imposing condition 1) or 2).

A lower bound $\delta_1(S)$ for problem P_1 can be obtained by defining instance $I^{(1)}(S)$ derived from I by setting $c_{i,j} = \infty$ if $i \in V \setminus S$ or $j \in V \setminus S$, with $i \neq j$ (thus imposing as unrouted all the vertices in $V \setminus S$), and by solving the corresponding *LD-AP*, as described in Section 3.2 (obviously, $\delta_1(S) = \infty$ if $\sum_{i \in S} p_i < g$).

As for problem P_2, let f_h (resp. b_h) be the cost of the shortest path from vertex 1 to vertex h (resp. from vertex h to vertex 1), for each $h \in V$. A lower bound $\delta_2(S)$ for P_2 is then given by:

$$\delta_2(S) = \min\{f_h + b_h : h \in V \setminus S\}.$$

In order to obtain the best lower bound δ, one has to choose subset S so as to maximize $\delta(S) = \min\{\delta_1(S), \delta_2(S)\}$. To this end, we note that for any vertex subset S' with $1 \in S'$ and $S' \subseteq S$, conditions $\delta_1(S') \geq \delta_1(S)$ and $\delta_2(S') \leq \delta_2(S)$ hold. Now, for any given subset S define $S' = \{j \in V : f_j + b_j < \delta_2(S)\}$. Clearly, $\delta_2(S') = \delta_2(S)$ (from the definition of S'), while $S' \subseteq S$ (since no vertex $j \in V \setminus S$ can have $f_j + b_j < \delta_2(S)$), and then $\delta_1(S') \geq \delta_1(S)$. It follows that $\delta(S') \geq \delta(S)$, and therefore the maximization of $\delta(S)$ requires that we explicitly consider only subsets $S_2, S_3, \cdots, S_n$, with $S_h = \{j \in V : f_j + b_j < f_h + b_h\}$ for $h = 2, 3, \cdots, n$ (since $\delta_2(S)$ can attain only values $f_2 + b_2$, $f_3 + b_3$, $\cdots$, $f_n + b_n$). Let S_{h^*} $(2 \leq h^* \leq n)$ be the subset maximizing $\delta(S_h)$. Since $S_2 \subseteq S_3 \subseteq \cdots \subseteq S_n$, we have $\delta_1(S_2) \geq \delta_1(S_3) \geq \cdots \geq \delta_1(S_n)$ and $\delta_2(S_2) \leq \delta_2(S_3) \leq \cdots \leq \delta_2(S_n)$. Index h^* can then be determined through binary search by exploiting the property that, for each h $(2 \leq h \leq n)$, $h^* \geq h$ if $\delta_1(S_h) \geq \delta_2(S_h)$, while $h^* \leq h$ if $\delta_1(S_h) \leq \delta_2(S_h)$.

A residual instance $\bar{I} = [\bar{n}, (\bar{c}_{i,j}), (\bar{\gamma}_i), (\bar{p}_i), \bar{g}]$ corresponding to lower bound $\delta(S_{h^*})$

$$= -\lambda^* \Delta + \sum_{i \in V} (u_i^* + v_i^*) + \sum_{i \in V} \sum_{j \in V \setminus \{i\}} (c_{i,j} - u_i^* - v_j^*) x_{i,j} +$$

$$\sum_{i \in V} (c_{i,i} + \lambda^* p_i - u_i^* - v_i^*) x_{i,i} =$$

$$= \lambda^* (\sum_{i \in V} p_i x_{i,i} - \Delta) + \sum_{i \in V} u_i^* (1 - \sum_{j \in V} x_{i,j}) + \sum_{j \in V} v_j^* (1 - \sum_{i \in V} x_{i,j}) +$$

$$\sum_{i \in V} \sum_{j \in V} c_{i,j} x_{i,j} = \lambda^* (\sum_{i \in V} p_i x_{i,i} - \Delta) + \sum_{i \in V} \sum_{j \in V} c_{i,j} x_{i,j} .$$

Hence, because of (12) and since $\lambda^* \geq 0$,

$$L(\lambda^*) + \sum_{i \in V} \sum_{j \in V} \bar{c}_{i,j} x_{i,j} \leq \sum_{i \in V} \sum_{j \in V} c_{i,j} x_{i,j} \tag{23}$$

holds. Since $(x_{i,j})$ is clearly feasible for reduced instance $\bar{I}$, we have

$$L(\lambda^*) + v(\bar{I}) \leq L(\lambda^*) + \sum_{i \in V} \sum_{j \in V} \bar{c}_{i,j} x_{i,j} \leq \sum_{i \in V} \sum_{j \in V} c_{i,j} x_{i,j} = v(I) .$$

In addition, $v(\bar{I}) \geq 0$ follows from (22), so both conditions i) and ii) on the residual instance $\bar{I}$ are satisfied.

It is worth noting that, from (23), residual costs $\bar{c}_{i,j}$ and $\bar{\gamma}_i = \bar{c}_{i,i}$ give a lower bound on the extra cost incurred if, respectively, arc (i,j) is included in the solution and vertex i is left unrouted ($\bar{\gamma}_i$ can be greater than 0 even when $\gamma_i = 0$ in the original instance).

Also note that the solution of $AP(\lambda^*)$ is generally infeasible for *PC-TSP* both because it can contain more than one subtour of cardinality greater than one, and because the total prize "collected" can be less than g.

3.3. A bound based on the arborescence problem substructure

Let us consider instance $I = [n, (c_{i,j}), (\gamma_i), (p_i), g]$ and the formulation of *PC-TSP* given by (15) – (20). Removing equilibrium constraints (18) produces a relaxed problem in which one calls for a shortest spanning 1-arborescence in the augmented graph G' such that the sum of the prizes of vertices j for which arc $(0,j)$ is not in the solution, is greater or equal to goal g. Such a problem is $\mathcal{NP}$-hard, since the 0-1 Knapsack Problem easily transforms to it. Note that the relaxed problem cannot be viewed as a *Prize-Collecting 1-Arborescence Problem*, since removing artificial vertex 0 could produce a solution given by a family of disconnected branchings. Due to the lack of reachability from vertex 1 of the vertices whose prize has been "collected", this problem generally leads to a poor lower bound.

so inequality (21) holds for $h = \overline{h}$ as well.

Each value δ computed at step 6 is then a valid lower bound for *PC-TSP* since, because of condition i), $v(I^{(h)}) \geq 0$. The sequence of values δ is non-decreasing, since increments $\delta^{(h)}$ are clearly non-negative for $h = 2, \cdots, r$.

More details on the additive approach are given in Fischetti and Toth (1986). Applications to the Asymmetric Travelling Salesman Problem, the Multiple Depot Vehicle Scheduling Problem and the Symmetric Travelling Salesman Problem are reported, respectively, in Fischetti and Toth (1987a), Carpaneto, Dell'Amico, Fischetti and Toth (1987), and Carpaneto, Fischetti and Toth (1987).

Algorithm *ADD-PCTSP* requires the computation of valid residual instances corresponding to all the available lower bounds, except the last (since residual instance $I^{(r)}$ is not used to increase the final value of δ). In the following, we propose different relaxations of *PC-TSP*, and describe how to obtain the corresponding lower bounds and residual instances.

3.2. A bound based on the assignment problem substructure

Let us consider instance $I = [n, (c_{i,j}), (\gamma_i), (p_i), g]$, and the formulation of *PC-TSP* given by (9) – (14). Removing connectivity constraints (13) produces a relaxed problem in which one calls for a minimum cost collection of subtours covering all the vertices and such that the sum of the prizes of the vertices not covered by loops is not less than goal g. Such a *Prize-Collecting Assignment Problem* (*PC-AP*) turns out to be $\mathcal{NP}$-hard, since it contains the *0-1 Knapsack Problem* (formulated as a minimization problem) as a particular case, arising when $c_{i,j} = b_j$ (for $i, j \in V, i \neq j$), $c_{j,j} = 0$ (for $j \in V \setminus \{1\}$), $c_{1,1} = \infty$, where $b_j > 0$ (for $j \in V \setminus \{1\}$), and $b_1 = 0$.

A lower bound for *PC-AP*, and hence for *PC-TSP*, could be obtained by solving its continuous relaxation, $\overline{PC\text{-}AP}$, given by (9) – (12) and

$$x_{i,j} \geq 0, \qquad \text{for each } i, j \in V.$$

A different approach (leading to the same value of the lower bound as $\overline{PC\text{-}AP}$) is to relax *PC-AP* in a Lagrangean fashion by imbedding constraint (12) in the objective function (9), and to solve the corresponding Lagrangean dual problem:

$$(LD - AP) \qquad v(LD - AP) = \max_{\lambda \geq 0} L(\lambda)$$

where

$$L(\lambda) = -\lambda \Delta + \min \sum_{i \in V} \sum_{j \in V} \tilde{c}_{i,j}(\lambda) x_{i,j}$$
$$\text{subject to (10), (11) and (14),}$$

with

Constraints (16), (17), and (20) – with objective function (15) – give the well-known *Shortest Spanning 1-Arborescence Problem* (*1-SSAP*). Any solution to *1-SSAP* is a collection of $|V'|-1$ arcs defining a directed spanning tree rooted at vertex 1, plus an arc entering vertex 1. Constraints (18) ensure that each vertex $i \in V$ is either unrouted (when $x_{0,i}=1$ and hence, from (16), $\sum_{h \in V} x_{h,i}=0$) or routed once. Constraint (19) imposes an upper bound on the total prize of the unrouted vertices. An example of a feasible solution to model (15) – (20) is given in Figure 2.

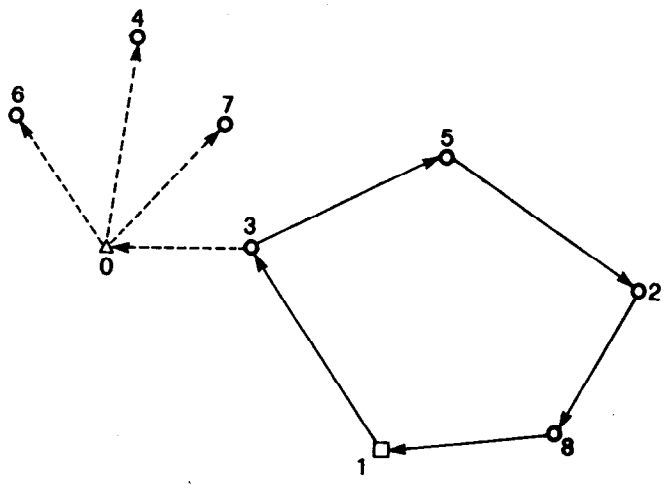

Figure 2. Feasible solution to model (15) – (20), assuming (p_j)= (0,8,5,2,4,5,3,6), $g=22$.

3. Lower bounds

Different lower bounds for *PC-TSP* can be obtained by exploiting different relaxations of the problem as, for example, those based on the assignment problem or on the arborescence problem substructures pointed out in the previous section. Since none of these bounds dominates the others, a possible way to obtain a strengthened bound is to compute the maximum among them. In this way, however, only one substructure is taken into account, while all the others are completely lost. To partially overcome this drawback and to exploit the complementarity of the available bounds, we use the *additive approach* proposed by Fischetti and Toth (1986), leading to a sequence of increasing lower bounds.

Constraints (2), (3) and (6) ensure that each vertex i is either unrouted (in which case no arc enters or leaves vertex i) or routed once. Constraints (4) ensure that the sum of the prizes of the routed vertices is not less than goal g. Any feasible solution satisfying constraints (2), (3),(4) and (6) can be viewed as a family of disjoint subtours (each of cardinality at least 2) visiting a subset of vertices whose total prize is at least g. Constraints (5) ensure the "connectivity" of the solution, in the sense that each routed vertex h can be reached from vertex 1.

Constraints (5) can be replaced by the "subtour elimination" constraints

$$\sum_{i \in S} \sum_{j \in S \setminus \{i\}} x_{i,j} \leq |S| - 1, \qquad \text{for each } S \subset V : 1 \in V \setminus S \tag{7}$$

forbidding subtours not visiting vertex 1.

A tighter formulation of the problem could be obtained by replacing constraint (4) with constraints

$$\sum_{i \in S} \sum_{j \in S \setminus \{i\}} x_{i,j} \leq |S| - 1, \qquad \text{for each } S \subset V : 1 \in S, \sum_{j \in S} p_j < g \tag{8}$$

which eliminate subtours visiting vertex 1 but not collecting a sufficient prize. On the assumption that $\gamma_i = 0$ for $i \in V \setminus \{1\}$, constraints (5) (or (7)) become redundant, and then (1), (2), (3), (6) and (8) give a valid formulation of *PC-TSP*.

Model (1) – (6) can be rearranged so as to point out the assignment problem substructure of *PC-TSP*. To this end, for each $i \in V$ we define $x_{i,i} = 1$ if vertex i is left unrouted, and $x_{i,i} = 0$ otherwise (i.e., $x_{i,i} = 1 - y_i$). Consequently, for each $i \in V$ we set $c_{i,i} = \gamma_i$ (the cost incurred if vertex i is left unrouted). With these definitions, a valid formulation of the problem is

$$(PC-TSP) \qquad v(PC-TSP) = \min \sum_{i \in V} \sum_{j \in V} c_{i,j}\, x_{i,j} \tag{9}$$

subject to

$$\sum_{i \in V} x_{i,j} = 1, \qquad \text{for each } j \in V \tag{10}$$

$$\sum_{j \in V} x_{i,j} = 1, \qquad \text{for each } i \in V \tag{11}$$

$$\sum_{i \in V} p_i\, x_{i,i} \leq \sum_{j \in V} p_j - g \tag{12}$$

$$\sum_{i \in V} \sum_{j \in V \setminus S} x_{i,j} \geq 1 - x_{h,h}, \qquad \begin{array}{c}\text{for each } h \in V \setminus \{1\} \text{ and for each} \\ S \subset V : 1 \in S, h \in V \setminus S\end{array} \tag{13}$$

$$x_{i,j} \in \{0,1\}, \qquad \text{for each } i,j \in V. \tag{14}$$

Constraints (10), (11), and (14) – with objective function (9) – define the well-known *Linear Min-sum Assignment Problem* (*AP*). Any feasible solution to *AP* gives a family of

Let p_i be the (indivisible) amount supplied at city i, β_i the corresponding cost ($i = 2, \cdots, n$), and $c_{i,j}$ the transportation cost from city i to city j ($i, j = 1, \cdots, n$). Assuming that only one trip is required, such a problem can be formulated as an instance of *PC-TSP* in which $\gamma_i = -\beta_i$ is the saving from city i not supplying the product (the total cost being $\sum_{i=2}^{n} \beta_i +$ optimal value of the *PC-TSP* instance).

The problem also arises in several scheduling problems. Balas and Martin (1985) introduced *PC-TSP* as a model for scheduling the daily operation of a steel rolling mill. A rolling mill produces steel sheets from slabs by hot or cold rolling. Let $c_{i,j}$ be the "cost" of processing order j just after order i, and p_i the weight of the slab assigned to order i. Scheduling the daily operation consists of selecting a subset of orders that satisfies a given lower bound g on the total weight, and of sequencing them so as to minimize the global cost.

PC-TSP can be formulated through a graph theory model as follows. Let $G = (V, A)$ be a directed complete graph, where $V = \{1, 2, \cdots, n\}$ is the vertex set (vertex 1 corresponding to the depot) and A the arc set. For each $(i, j) \in A$ let $c_{i,j}$ be the cost of arc (i, j) (with $c_{i,i} = \infty$ for each $i \in V$) and, for each $i \in V$, let γ_i and p_i be respectively the cost and the prize associated with vertex i (with $\gamma_1 = \infty$ and $p_1 = 0$). A vertex subset $S \subseteq V$ is *feasible* iff $\sum_{i \in S} p_i \geq g$ and $1 \in S$. The *PC-TSP* is to find a Hamiltonian circuit in the subgraph induced by a feasible vertex subset S, so as to minimize the sum of the costs of the arcs in the circuit plus the sum of the costs associated with vertices in $V \setminus S$.

PC-TSP contains as a particular case the *TSP* obtained when $\sum_{i \in V} p_i = g$ (or when $\gamma_i = \infty$ for each $i \in V$), and hence it belongs to the class of $\mathcal{NP}$-hard problems (in the strong sense).

PC-TSP can also be viewed as a generalization of the minimization form of the *0-1 Single Knapsack Problem*, arising when $\gamma_i = 0$ for each $i \in V \setminus \{1\}$, and when all the arcs entering the same vertex have the same cost.

Without loss of generality we assume:

$$c_{i,j} \geq 0 \quad \text{for each } (i, j) \in A, \text{ and } \gamma_i \geq 0 \quad \text{for each } i \in V .$$

In fact, for each $i \in V$ the addition of any constant α_i to γ_i and $c_{i,j}$ ($j \in V$) does not alter the relative ranking among the feasible solutions to *PC-TSP*. We also assume the feasibility condition $\sum_{i \in V} p_i \geq g$.

We do not assume that the triangle inequality ($c_{i,k} + c_{k,j} \geq c_{i,j}$ for $i, j, k \in V$ and $i \neq j$) holds. If, however, the triangle inequality applies and $\gamma_i = 0$ for $i \in V \setminus \{1\}$, as in several practical applications, we can also assume that $p_i > 0$ for $i \in V \setminus \{1\}$, since vertices with zero prize are not worth routing; for the same reason, only minimal (with respect to deletion of one element) feasible vertex subsets S can lead to optimal solutions.

To our knowledge, no optimal algorithm has been proposed for *PC-TSP*. Heuristic methods and structural properties have been discussed in Balas and Martin (1985) and in Balas (1987), respectively. A related problem is the *Orienteering Problem*, in which the

Problem", forthcoming in *Naval Research Logistics.*

G. Laporte and S. Martello (1987), "The Selective Travelling Salesman Problem", Working Paper, EHEC, Universitè de Montreal, Quebec, Canada (presented at the ORSA/TIMS Meeting, New Orleans, May 1987).

E.L. Lawler (1976), *Combinatorial Optimization: Networks and Matroids*, Holt, Rinehart and Winston, New York.

T. Tsiligirides (1984), "Heuristic Methods Applied to Orienteering", *Journal of the Operational Research Society* 35, 797-809.

S. Warshall (1962), "A Theorem on Boolean Matrices", *Journal of ACM* 9, 11-12.