Mathematical Programming 45 (1989) 233-254
North-Holland

NEW LOWER BOUNDS FOR THE SYMMETRIC TRAVELLING SALESMAN PROBLEM

G. CARPANETO
Dipartimento di Economia Politica, University of Modena, Italy

M. FISCHETTI and P. TOTH
DEIS, University of Bologna, Italy

Received 17 September 1987
Revised manuscript received 1 June 1988

In this paper new lower bounds for the Symmetric Travelling Salesman Problem are proposed and combined in additive bounding procedures. Efficient implementations of the algorithms are given; in particular, fast procedures for computing the linear programming reduced costs of the Shortest Spanning Tree (SST) Problem and for finding all the r-SST of a given graph, are presented. Computational results on randomly generated test problems are reported.

Key words: Symmetric Travelling Salesman Problem, additive bounding procedures, reduced costs computation, algorithms implementation.

1. Introduction

Let $G=(V, E)$ be a given undirected complete graph, where $V=\{1,\dots,n\}$ is the vertex set and $E=\{l_1,\dots,l_m\}$ the edge set, and let $c_l \geqslant 0$ be the cost associated with edge $l \in E$. Each edge $l \in E$ will also be denoted through the unordered pair $[i, j]$, where i and j are the extreme vertices of l (without loss of generality we will assume $c_{[i,i]}=\infty$ for each $i \in V$). For each vertex $i \in V$, let Γ_i denote the subset of the edges incident at i. A *Hamiltonian cycle* (*tour*) of G is a partial graph $\tilde{G}=(V, \tilde{E})$ of G such that:

(i) $\tilde{G}$ is connected,

(ii) each vertex $i \in V$ has degree equal to two in $\tilde{G}$ (i.e., $|\Gamma_i \cap \tilde{E}|=2$ for each $i \in V$).

The *Symmetric Travelling Salesman Problem* (STSP) is to find a tour $G^*=(V, E^*)$ of G whose *cost* $\sum_{l\in E^*} c_l$ is minimum. Such a problem is known to be $\mathcal{NP}$-hard. For a comprehensive survey on the Travelling Salesman Problem, see Lawler, Lenstra, Rinnooy Kan and Shmoys (1985).

The exact solution of STSP has been successfully approached through branch and bound algorithms (Held and Karp, 1970, 1971; Christofides, 1970; Helbig Hansen and Krarup, 1974; Smith and Thompson, 1977; Volgenant and Jonker, 1982; Gavish and Srikanth, 1986) and polyhedral-based techniques (see the survey of Padberg and Grötschel, 1985). Although the exact solution of the largest instances of STSP have been obtained by Padberg and Rinaldi (1987) through the polyhedral "branch and cut" technique, branch and bound algorithms cannot be dismissed out of hand. In fact, branch and bound is much simpler to implement, can easily take into account additional constraints, and can give good approximate feasible solutions if its execution is interrupted before the end.

It is well known that the effectiveness of branch and bound algorithms greatly depends on the availability of fast procedures computing tight lower bounds. In this paper new bounding procedures for STSP are presented which produce sequences of increasing lower bounds according to the general framework of the *additive approach* proposed by Fischetti and Toth (1989). Such bounding procedures can also be useful to improve the lower bound for the *Asymmetric Travelling Salesman Problem* (ATSP), mainly for "almost symmetric" instances. This can be achieved by decomposing the cost $c_{i,j}$ of each arc (i, j) into $\sigma_{i,j}+\alpha_{i,j}$, with $\sigma_{i,j}=\sigma_{j,i}=\min\{c_{i,j}, c_{j,i}\}$, and hence by computing a lower bound δ for ATSP as $\delta=\delta_\sigma+\delta_\alpha$, where δ_σ is a lower bound for the instance of STSP defined by the symmetric costs $\sigma_{i,j}$, and δ_α is a lower bound for the instance of ATSP defined by the asymmetric costs $\alpha_{i,j}$ (for more details, see Fischetti and Toth, 1990, Section 4.2).

In Section 2, the additive approach is briefly described, and its properties relevant for STSP are pointed out. In Section 3 new lower bounds are proposed, and combined in Section 4 to obtain three bounding procedures. Section 5 gives data structures and algorithms for efficient procedure implementation. Computational results on randomly generated test problems are reported in Section 6.

2. The additive approach

We now briefly outline the additive approach to design bounding procedures; fuller details are given in Fischetti and Toth (1989).

Problem STSP can be formulated as

$$\text{(STSP)} \quad v(\text{STSP})=\min\left\{\sum_{l\in E} c_l x_l \colon (x_l)\in F(\text{STSP})\right\}$$

where $x_l=1$ if edge l is in the optimal tour, $x_l=0$ otherwise ($l\in E$), and F(STSP) is the set of the 0–1 incidence vectors of tours in G.

Let us suppose that q bounding procedures $\mathscr{L}^{(1)}, \mathscr{L}^{(2)}, \ldots, \mathscr{L}^{(q)}$ are available, each exploiting a different substructure of STSP to produce a different lower bound. If none of the q bounds dominates the others, an overall lower bound could be obtained by taking the maximum among them. In this way, however, only one substructure is taken into account, while all the others are lost. In order to exploit all the substructures, an overall additive bounding procedure can be designed as follows.

Suppose that, for $h=1,\dots,q$ and for any cost vector $\bar{c}=(\bar{c}_l)$, procedure $\mathscr{L}^{(h)}(\bar{c})$, when applied to the instance of STSP having costs $\bar{c}_l$, returns its lower bound $\delta^{(h)}$ as well as a *residual cost* vector $c^{(h)}=(c_l^{(h)})$ such that:

(i) $c_l^{(h)} \geqslant 0$ for each $l \in E$;

(ii) $\delta^{(h)}+\sum_{l\in E} c_l^{(h)}x_l \leqslant \sum_{l\in E} \bar{c}_l x_l$ for each $(x_l)\in F(\mathrm{STSP})$.

According to the additive approach, procedures $\mathscr{L}^{(1)}, \mathscr{L}^{(2)}, \dots, \mathscr{L}^{(q)}$ are considered in sequence: procedure $\mathscr{L}^{(h)}$ is applied to the residual cost vector $c^{(h-1)}$ returned by procedure $\mathscr{L}^{(h-1)}$, and computes a lower bound increment $\delta^{(h)}$ ($h=1,2,\dots,q$; $c^{(0)}$ being the original cost vector c). It is easy to show that $\delta=\delta^{(1)}+\delta^{(2)}+\cdots+\delta^{(q)}$ is a valid lower bound for STSP. Final residual cost vector $c^{(q)}$ can be used for reduction purposes.

The additive approach, which is related to the restricted Lagrangian relaxation proposed by Balas and Christofides (1981), has been applied to the Asymmetric Travelling Salesman Problem (Fischetti and Toth, 1990), to the Multiple Depot Vehicle Scheduling Problem (Carpaneto, Dell'Amico, Fischetti and Toth, 1989), and to the Prize Collecting Travelling Salesman Problem (Fischetti and Toth, 1988).

A key step of the algorithm above is computation of the residual costs. To this end different techniques can be used, corresponding to different ways of computing the lower bound.

2.1. *Bounds from linear programming relaxation*

Let A and b be a $(k\times m)$ matrix and a k vector, respectively, such that

$$F(\mathrm{STSP}) \subseteq \{x\in\mathbb{R}^m \colon x\geqslant 0,\ Ax\geqslant b\}.$$

A well-known lower bound, δ', for the instance of STSP defined by cost vector $\bar{c}$, can be computed by solving the linear programming problem

$$(\mathrm{LP}) \qquad \delta' = \min\left\{\sum_{l\in E} \bar{c}_l x_l \colon x\geqslant 0,\ Ax\geqslant b\right\},$$

while the corresponding residual cost vector $c'=(c'_l)$ can be obtained as

$$c'_l = \bar{c}_l - \sum_{i=1}^{k} u_i^* A_{i,l} \quad \text{for each } l\in E,$$

where u^* is the optimal solution vector of the dual of LP (i.e., c' is the linear programming *reduced cost* vector associated with the optimal solution to LP).

2.2. *Bounds from variables decomposition*

Let us suppose that it is possible to define two sets $Y^{(1)}$ and $Y^{(2)}$ with $Y^{(t)}\subseteq \{y\in\mathbb{R}^m \colon y\geqslant 0\}$ $(t=1,2)$, with the property that each vector $x\in F(\mathrm{STSP})$ can be decomposed into two vectors $y^{(1)}\in Y^{(1)}$ and $y^{(2)}\in Y^{(2)}$ such that $x=y^{(1)}+y^{(2)}$. For

$t=1,2$, let $\vartheta^{(t)}$ and $\gamma^{(t)}$ be, respectively, a lower bound value and the corresponding residual cost vector for the following *partial problem*:

$$(\mathrm{PP}^{(t)}) \quad v(\mathrm{PP}^{(t)}) = \min\left\{ \sum_{l\in E} \bar{c}_l y_l^{(t)} \colon (y_l^{(t)}) \in Y^{(t)} \right\}.$$

Hence

$$\delta' = \vartheta^{(1)} + \vartheta^{(2)}$$

is a valid lower bound for the instance of STSP defined by cost vector $\bar{c}$, and

$$c_l' = \min\{\gamma_l^{(1)}, \gamma_l^{(2)}\} \quad \text{for each } l \in E,$$

are the corresponding residual costs.

2.3. Bounds from projection

A particular case of variables decomposition is *projection*. Let $E^{(1)}$ and $E^{(2)}$ be a partition of edge set E, and $\mathrm{FP}^{(t)}$ $(t=1,2)$ denote the projection of $F(\mathrm{STSP})$ with respect to $E^{(t)}$, i.e. the set of all the $|E^{(t)}|$-dimensional vectors obtainable from vectors $x \in F(\mathrm{STSP})$ by removing the components associated with edges $l \in E \setminus E^{(t)}$. Suppose that for each *partial subproblem*

$$(\mathrm{PSP}^{(t)}) \quad v(\mathrm{PSP}^{(t)}) = \min\left\{ \sum_{l\in E^{(t)}} \bar{c}_l z_l \colon (z_l) \in \mathrm{FP}^{(t)} \right\}$$

a bounding procedure is available, producing a lower bound $\vartheta^{(t)} \leq v(\mathrm{PSP}^{(t)})$ and a corresponding residual cost vector $\varphi^{(t)}$. Then inequality

$$\vartheta^{(t)} + \sum_{l\in E^{(t)}} \varphi_l^{(t)} z^{(t)} \leq \sum_{l\in E^{(t)}} \bar{c}_l z_l$$

holds for each $z \in \mathrm{FP}^{(t)}$ $(t=1,2)$, and we have

$$\vartheta^{(1)} + \vartheta^{(2)} + \sum_{l\in E^{(1)}} \varphi_l^{(1)} z_l + \sum_{l\in E^{(2)}} \varphi_l^{(2)} z_l \leq \sum_{l\in E} \bar{c}_l z_l \quad \text{for each } (z_l) \in F(\mathrm{STSP}).$$

Hence

$$\delta' = \vartheta^{(1)} + \vartheta^{(2)}$$

is a valid lower bound for the instance of STSP defined by cost vector $\bar{c}$ (since $\varphi_l^{(t)}$ is non-negative for each $l \in E^{(t)}$), and the corresponding residual costs are

$$c_l' = \varphi_l^{(1)} \quad \text{for each } l \in E^{(1)},$$

$$c_l' = \varphi_l^{(2)} \quad \text{for each } l \in E^{(2)}.$$

2.4. *Bounds from Lagrangian relaxation*

Let A and b be a $(k \times m)$ matrix and a k vector, respectively, such that

$$F(\text{STSP}) \subseteq \{x \in \mathbb{R}^m : Ax \geqslant b\}.$$

Let $\tilde{u} \in \mathbb{R}^k$ be a vector of non-negative Lagrangian multipliers, and consider costs

$$\tilde{c}_l = \bar{c}_l - \sum_{i=1}^{k} \tilde{u}_i A_{i,l} \quad \text{for each } l \in E.$$

Then inequality

$$\sum_{i=1}^{k} \tilde{u}_i b_i + \sum_{l \in E} \tilde{c}_l x_l \leqslant \sum_{l \in E} \bar{c}_l x_l$$

holds for each $x \in F(\text{STSP})$.

Now apply any bounding procedure to the instance of STSP defined by costs $\tilde{c}_l$, obtaining a lower bound value ϑ and the corresponding residual costs γ_l. So inequalities

$$\sum_{i=1}^{k} \tilde{u}_i b_i + \left(\vartheta + \sum_{l \in E} \gamma_l x_l \right) \leqslant \sum_{i=1}^{k} \tilde{u}_i b_i + \sum_{l \in E} \tilde{c}_l x_l \leqslant \sum_{l \in E} \bar{c}_l x_l$$

hold for each $x \in F(\text{STSP})$. Hence

$$\delta' = \sum_{i=1}^{k} \tilde{u}_i b_i + \vartheta$$

is a valid lower bound for the instance of STSP defined by costs $\bar{c}_l$ (since γ_l is non-negative for each $l \in E$), and

$$c'_l = \gamma_l \quad \text{for each } l \in E$$

are the corresponding residual costs.

Note that if $F(\text{STSP}) \subseteq \{x \in \mathbb{R}^m : Ax = b\}$, Lagrangian multipliers $\tilde{u}_i$ are not restricted to being non-negative.

2.5. *Bounds from the Asymmetric Travelling Salesman Problem*

Any instance of STSP defined by cost vector $(\bar{c}_l)$ is clearly equivalent to the instance of ATSP with symmetric cost matrix $(\bar{c}_{i,j})$ defined by $\bar{c}_{i,j} = \bar{c}_l$ for each $l \equiv [i, j] \in E$. Hence any bounding procedure for ATSP, producing a lower bound δ' and a residual cost matrix $\tilde{c}$, also applies to STSP. Since, however, residual cost matrix $\tilde{c}$ is generally asymmetric even when the input cost matrix $\bar{c}$ is symmetric, such a procedure could be inserted in an additive algorithm for STSP only as the final procedure of the sequence. We now describe how to overcome this drawback by transforming $\tilde{c}$ into an equivalent symmetric residual cost matrix c'.

We first show that, if $\tilde{c}$ is a valid residual cost matrix (associated with lower bound δ') for the instance of ATSP defined by symmetric cost matrix $(\bar{c}_{i,j})$, then so is its transposed matrix $\tilde{c}^{\mathrm{T}} = (\tilde{c}^{\mathrm{T}}_{i,j})$. In fact, let $(x_{i,j})$ be the 0–1 incidence matrix of any feasible solution to ATSP. Since, clearly, its transposed matrix $(x^{\mathrm{T}}_{i,j})$ also defines a feasible solution to ATSP, we have

$$\delta' + \sum_{i\in V}\sum_{j\in V} \tilde{c}_{i,j} x^{\mathrm{T}}_{i,j} \leqslant \sum_{i\in V}\sum_{j\in V} \bar{c}_{i,j} x^{\mathrm{T}}_{i,j},$$

which can be rewritten as

$$\delta' + \sum_{i\in V}\sum_{j\in V} \tilde{c}^{\mathrm{T}}_{i,j} x_{i,j} \leqslant \sum_{i\in V}\sum_{j\in V} \bar{c}^{\mathrm{T}}_{i,j} x_{i,j} = \sum_{i\in V}\sum_{j\in V} \bar{c}_{i,j} x_{i,j}.$$

Because of the arbitrariness of $(x_{i,j})$, the thesis follows.

Clearly, any convex combination of residual cost matrices gives a valid residual cost matrix, so the symmetric matrix c' defined as

$$c'_{i,j} = \tfrac{1}{2}(\tilde{c}_{i,j} + \tilde{c}_{j,i}) \quad \text{for each } i, j \in V,$$

is a valid residual cost matrix for ATSP.

Since any instance of ATSP defined by a symmetric cost matrix can be viewed as an instance of STSP, vector (c'_l) with $c'_l = c'_{i,j}$ for each $l \equiv [i, j] \in E$ is a valid residual cost vector (with respect to lower bound δ') for the instance of STSP defined by cost vector $(\bar{c}_l)$.

A bounding procedure to compute a lower bound δ' and a residual cost vector (c'_l) for STSP, based on transformation to ATSP, can be outlined as follows:

(i) define a symmetric cost matrix $\bar{c}$ with $\bar{c}_{i,j} = \bar{c}_{j,i} = \bar{c}_l$ for each $l \equiv [i, j] \in E$;

(ii) apply a bounding procedure to the instance of ATSP defined by cost matrix $\bar{c}$, thus obtaining a lower bound value δ' and a residual cost matrix $\tilde{c}$;

(iii) for each $l \equiv [i, j] \in E$, define the residual cost c'_l as $\tfrac{1}{2}(\tilde{c}_{i,j} + \tilde{c}_{j,i})$.

2.6. *A simple check for residual costs*

Many bounding procedures for STSP are based on greedy techniques and are, therefore, algorithmically easy to analyze.

Let $\mathscr{L}(\bar{c})$ be one of these procedures, producing lower bound δ'. Given a non-negative vector c', a simple proof that c' is a valid residual cost vector associated with δ' is to show that procedure $\mathscr{L}(\bar{c} - c')$ produces a lower bound $\delta'' \geqslant \delta'$. In this case, in fact, inequalities $\delta' \leqslant \delta'' \leqslant \sum_{l\in E} (\bar{c}_l - c'_l)x_l$ hold for each $x \in F(\mathrm{STSP})$. Hence, residual cost vector $c' \geqslant 0$ for procedure $\mathscr{L}(\bar{c})$ can be obtained as the (maximal) decrease of cost vector $\bar{c}$ which does not yield a decrease in the lower bound value.

3. Lower bounds

In this section we describe several lower bounding procedures and show how to compute the corresponding residual costs. For each bounding procedure, $(\bar{c}_l)$ denotes the input cost vector, and δ' and (c'_l) the lower bound and the corresponding residual cost vector, respectively.

3.1. A bound from ATSP

According to Section 2.5, any bounding procedure for ATSP also applies to STSP. Several bounding procedures for ATSP have been proposed by Fischetti and Toth (1990). In particular, an $O(n^3)$ additive bounding Procedure P1 can be outlined as follows:

Procedure P1

```
   begin
1. foreach $l \equiv [i, j] \in E$ do $\tilde{c}_{i,j} := \tilde{c}_{j,i} := \bar{c}_l$;
2. solve the Assignment Problem (AP) on cost matrix $\tilde{c}$, obtaining lower bound
   $v$(AP) and the reduced cost matrix $\tilde{c}$; $\delta' := v$(AP);
3. compute on cost matrix $\tilde{c}$ the cost $\tilde{L}_h$ of the shortest path from vertex 1 to each
   vertex $h \in V$;
   foreach $i, j \in V$ do $\tilde{c}_{i,j} := \tilde{c}_{i,j} + \tilde{L}_i - \tilde{L}_j$;
4. solve the Shortest Spanning 1-Antiarborescence Problem (1-SAAP) on cost matrix
   $\tilde{c}$, obtaining lower bound $v$(1-SAAP) and the reduced cost matrix $\tilde{c}$;
   $\delta' := \delta' + v$(1-SAAP);
5. foreach $l \equiv [i, j] \in E$ do $c'_l := \frac{1}{2}(\tilde{c}_{i,j} + \tilde{c}_{j,i})$
   end.
```

At Step 4, the Shortest Spanning 1-Antiarborescence Problem 1-SAAP is solved on the current cost matrix $\tilde{c}$. Relaxation 1-SAAP of ATSP is to find a minimum cost partial graph of G such that: (i) exactly one arc leaves each vertex, and (ii) a path from each vertex to vertex 1 exists. Such a problem can be solved by determining the shortest spanning arborescence rooted at vertex 1 with respect to the transposed matrix $\tilde{c}^T$ (e.g., through the $O(n^2)$ algorithm of Tarjan, 1977), and by adding the minimum-cost arc leaving vertex 1. Since 1-SAAP is a linear programming problem, its reduced costs are valid residual costs (see Fischetti and Toth, 1987, for an $O(n^2)$ algorithm for the arborescence reduced costs computation).

It is easy to show that, after Step 4, each vertex pair can be connected (via vertex 1) through a pair of zero-cost directed paths in the digraph associated with the asymmetric residual-cost matrix $\tilde{c}$.

3.2. A bound from projection

Let $W = \{v_1, v_2, \ldots, v_p\}$ (with $p < \frac{1}{2}n$) be a given vertex subset of V, and define $E^{(1)} = \Gamma_{v_1} \cup \Gamma_{v_2} \cup \cdots \cup \Gamma_{v_p}$ as the subset of the edges incident at W, and $E^{(2)} = E \setminus E^{(1)}$. Consider any tour $\tilde{G} = (V, \tilde{E})$ of G, and define $\tilde{E}^{(t)} = \tilde{E} \cap E^{(t)}$ for $t = 1, 2$ (see Figure 1). It is easy to show that the following properties hold:

(1) $\tilde{E}^{(1)}$ induces an acyclic graph and contains at least $m_1 = p + 1$ edges;

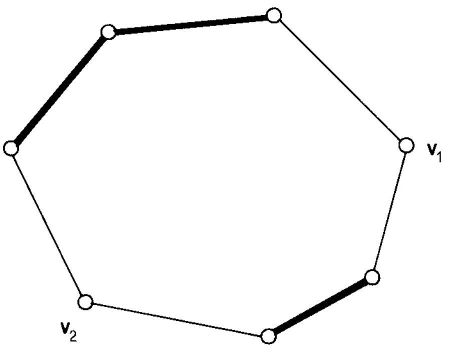

Fig. 1. Tour $\tilde{G}$; $p = 2$: edges in $\tilde{E}^{(2)}$ are in bold line.

(2) $\tilde{E}^{(2)}$ induces an acyclic graph and contains at least $m_2 = n - 2p$ edges.

According to Section 2.3, a valid lower bound $\vartheta^{(t)}$ for each partial subproblem PSP$^{(t)}$ $(t = 1, 2)$ can be obtained by determining the minimum-cost subset $\bar{E}^{(t)}$ of $E^{(t)}$ which satisfies property (t). This can be achieved through the following straightforward adaptation of the *greedy algorithm* of Kruskal (1956) for graphic matroids (assuming $\bar{c}_l \geqslant 0$ for each $l \in E$):

```
initialize $\bar{E}^{(t)} := \phi$ and $\vartheta^{(t)} := 0$;
**repeat**
    let $l$ be the next minimum-cost edge in $E^{(t)}$;
    **if** $\bar{E}^{(t)} \cup \{l\}$ induces an acyclic graph
    **then** set $\bar{E}^{(t)} := \bar{E}^{(t)} \cup \{l\}$ and $\vartheta^{(t)} := \vartheta^{(t)} + \bar{c}_l$
**until** $|\bar{E}^{(t)}| = m_t$.
```

As for the corresponding residual cost $\varphi_l^{(t)}$ associated with each edge $l \in E^{(t)}$, this can be computed as

$$\varphi_l^{(t)} = \bar{c}_l - \max\{\bar{c}_k : k \in C_l\} \geqslant 0,$$

where C_l is the set of the edges in $\bar{E}^{(t)}$ which make a cycle with edge l (with $C_l = \{l\}$ if $l \in \bar{E}^{(t)}$, and $C_l = \bar{E}^{(t)}$ if $\bar{E}^{(t)} \cup \{l\}$ induces an acyclic graph). The correctness of these residual costs follows from matroid theory. An easier proof can however be derived, according to Section 2.6, from the fact that reapplying the greedy algorithm on costs $\bar{c}_l - \varphi_l^{(t)} = \max\{\bar{c}_k : k \in C_l\}$ would produce the same value $\vartheta^{(t)}$ of the lower bound (since the cost of edges $l \in \bar{E}^{(t)}$ would be unchanged, while the change in the relative ranking of the remaining edges could not cause an edge l excluded from $\bar{E}^{(t)}$ to be considered before all the edges in C_l, using an appropriate tie-break rule).

An overall lower bound for STSP is then $\delta' = \vartheta^{(1)} + \vartheta^{(2)}$, and the corresponding residual costs are $c_l' = \varphi_l^{(t)}$ for $l \in E^{(t)}$ $(t = 1, 2)$.

A particular case of the bound based on projection leads to the well-known *r-Shortest Spanning Tree Problem* (r-SST) (see Held and Karp, 1970, 1971), arising when $p=1$ and $W=\{r\}$.

3.3. A bound from disjoint edge pairs

Let S be a given vertex subset such that $2 \leqslant |S| \leqslant n-2$, and denote with $K \equiv (S, V \backslash S)$ the *cut-set* containing all the edges $[i, j] \in E$ such that $i \in S$ and $j \in V \backslash S$. In addition, let $\bar{l} \equiv [v_1, v_2]$ be a given edge in K. Consider any tour $\tilde{G} = (V, \tilde{E})$ of G. One can verify that $\tilde{E}$ can always be partitioned into $\tilde{E}^{(1)}$ and $\tilde{E}^{(2)}$, where $\tilde{E}^{(1)}$ and $\tilde{E}^{(2)}$ satisfy the following properties (see Figure 2):

(1) $\tilde{E}^{(1)} = \{l_1, l_2\}$, with l_1 and l_2 disjoint edges in K;

(2) $|\tilde{E}^{(2)}| = n-2$, $\bar{l} \notin \tilde{E}^{(2)}$, $\tilde{E}^{(2)} \cup \{\bar{l}\}$ induces an acyclic graph.

Note that property (2) is satisfied iff l_1 and l_2 are not in the same path connecting vertices v_1 and v_2 in $\tilde{G}$ (hence, $l_1 = \bar{l}$ if $\bar{l} \in \tilde{E}$), and that l_1 and l_2 can always be chosen disjoint, since $2 \leqslant |S| \leqslant n-2$.

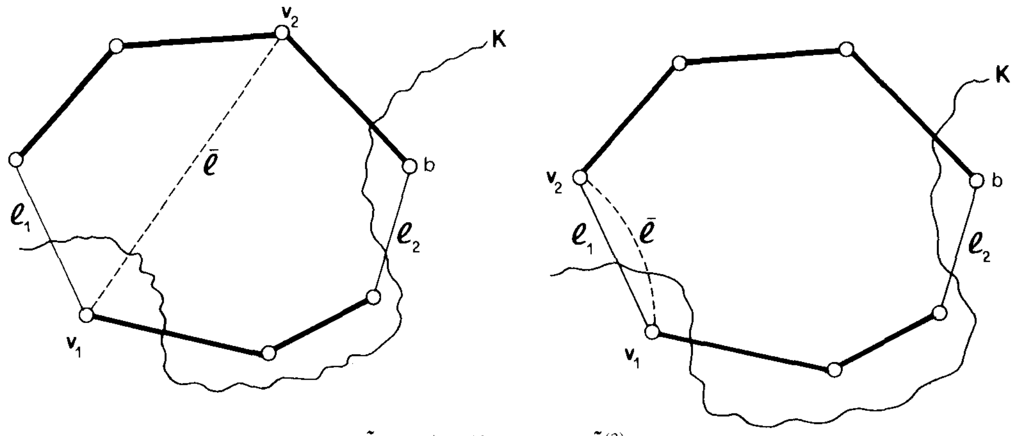

Fig. 2. Tour $\tilde{G}$; $S = \{v_1, b\}$; edges in $\tilde{E}^{(2)}$ are in bold line.

According to the above properties and following the variables decomposition approach of Section 2.2, STSP can be decomposed into partial problems $\mathrm{PP}^{(1)}$ and $\mathrm{PP}^{(2)}$. Partial problem $\mathrm{PP}^{(t)}$ ($t = 1, 2$) is to find the 0-1 incidence vector ($y_l^{(t)}$: $l \in E$) corresponding to a minimum-cost edge set $\bar{E}^{(t)}$ which satisfies property (t). Both problems $\mathrm{PP}^{(1)}$ and $\mathrm{PP}^{(2)}$ can be exactly solved in polynomial time (see the following two subsections). Let $\vartheta^{(t)}$ be the optimal solution value of problem $\mathrm{PP}^{(t)}$, and $\gamma^{(t)}$ the corresponding residual cost vector ($t = 1, 2$). A valid lower bound δ' for STSP can then be computed as $\vartheta^{(1)} + \vartheta^{(2)}$, while the corresponding residual costs c_l' are given by $\min\{\gamma_l^{(1)}, \gamma_l^{(2)}\}$ for $l \in E$.

Solving partial problem $\mathrm{PP}^{(1)}$

Problem $\mathrm{PP}^{(1)}$ can be formulated as an integer linear programming problem as follows:

$$(\mathrm{PP}^{(1)}) \quad \vartheta^{(1)} = \min \sum_{l\in E} \bar{c}_l y_l^{(1)}$$

subject to

$$y_l^{(1)} = 0 \quad \text{for each } l \in E\backslash K,$$

$$\sum_{l\in K} y_l^{(1)} = 2,$$

$$\sum_{l\in \Gamma_i \cap K} y_l^{(1)} \leqslant 1 \quad \text{for each } i \in V,$$

$$y_l^{(1)} \geqslant 0 \text{ and integer} \quad \text{for each } l \in K.$$

Let $\overline{\mathrm{PP}}^{(1)}$ be the linear programming relaxation of $\mathrm{PP}^{(1)}$. We now prove that $\overline{\mathrm{PP}}^{(1)}$ is a tight formulation of $\mathrm{PP}^{(1)}$ by showing that it always has an optimal integer solution.

Let us denote:
- $\lambda \equiv [a, b]$ as the minimum-cost edge in K;
- α as the minimum-cost edge in $K\backslash\{\lambda\}$ incident at a;
- β as the minimum-cost edge in $K\backslash\{\lambda\}$ incident at b;
- ε as the minimum-cost edge in $K\backslash(\Gamma_a \cup \Gamma_b)$, i.e. disjoint from λ.

Clearly, both (λ, ε) and (α, β) are pairs of disjoint edges in K. Let $(l_1, l_2) = (\lambda, \varepsilon)$ if $\bar{c}_\lambda + \bar{c}_\varepsilon \leqslant \bar{c}_\alpha + \bar{c}_\beta$, $(l_1, l_2) = (\alpha, \beta)$ otherwise, and define $\Delta = \bar{c}_{l_1} + \bar{c}_{l_2}$ $(= \min\{\bar{c}_\lambda + \bar{c}_\varepsilon, \bar{c}_\alpha + \bar{c}_\beta\})$. We claim that the 0-1 incidence vector corresponding to the edge pair (l_1, l_2) is an optimal solution to $\overline{\mathrm{PP}}^{(1)}$.

Consider in fact the dual problem of $\overline{\mathrm{PP}}^{(1)}$:

$$(\overline{\mathrm{DPP}}^{(1)}) \quad v(\overline{\mathrm{DPP}}^{(1)}) = \max\left(2w - \sum_{i\in V} u_i\right)$$

subject to

$$\bar{c}_l + z_l \geqslant 0 \quad \text{for each } l \in E\backslash K,$$

$$\bar{c}_l + u_i + u_j - w \geqslant 0 \quad \text{for each } l \equiv [i, j] \in K,$$

$$u_i \geqslant 0 \quad \text{for each } i \in V,$$

and define the following dual solution:

$$w = \Delta - \bar{c}_\lambda; \qquad u_a = \max\{0, w - \bar{c}_\alpha\}; \qquad u_b = w - \bar{c}_\lambda - u_a;$$

$$u_i = 0 \quad \text{for each } i \in V\backslash\{a, b\}; \qquad z_l = \infty \quad \text{for each } l \in E\backslash K.$$

We first show that such a solution is feasible for the dual problem $\overline{\mathrm{DPP}}^{(1)}$.

Clearly $u_a \geqslant 0$ and $u_b = w - \bar{c}_\lambda - \max\{0, w - \bar{c}_\alpha\} = \min\{\Delta - 2\bar{c}_\lambda, \bar{c}_\alpha - \bar{c}_\lambda\} \geqslant 0$, while $\bar{c}_l + z_l \geqslant 0$ for each $l \in E\backslash K$. In addition:
- $\bar{c}_\lambda + u_a + u_b - w = 0$;
- for each edge $l \in K\backslash\{\lambda\}$ incident at a, we have $\bar{c}_l + u_a + 0 - w \geqslant \bar{c}_\alpha + u_a - w \geqslant \bar{c}_\alpha + (w - \bar{c}_\alpha) - w = 0$;

- for each edge $l \in K \backslash \{\lambda\}$ incident at b, we have $\bar{c}_l + 0 + u_b - w \geq \bar{c}_\beta + u_b - w = \bar{c}_\beta - \bar{c}_\lambda - u_a = \min\{\bar{c}_\beta - \bar{c}_\lambda, \bar{c}_\beta + \bar{c}_\alpha - \Delta\} \geq 0$;
- for each edge $l \in K \backslash (\Gamma_a \cup \Gamma_b)$, we have $\bar{c}_l + 0 + 0 - w \geq \bar{c}_\varepsilon - w = \bar{c}_\varepsilon + \bar{c}_\lambda - \Delta \geq 0$.

As for the optimality of both primal and dual solutions, we have it that the value of the dual solution is $2w - (u_a + u_b) = 2\Delta - 2\bar{c}_\lambda - w + \bar{c}_\lambda = \Delta$, which is also the value of the primal solution.

The residual costs $\gamma_l^{(1)}$ associated with $\vartheta^{(1)} = \Delta$ coincide with the reduced costs associated with the optimal solution of $\overline{\mathrm{PP}}^{(1)}$, that is:

- $\gamma_l^{(1)} = \infty$ for each $l \in E \backslash K$;
- $\gamma_\lambda^{(1)} = 0$;
- $\gamma_l^{(1)} = \bar{c}_l + u_a - w$ for each edge $l \in K \backslash \{\lambda\}$ incident at a;
- $\gamma_l^{(1)} = \bar{c}_l + u_b - w$ for each edge $l \in K \backslash \{\lambda\}$ incident at b;
- $\gamma_l^{(1)} = \bar{c}_l - w$ for each edge $l \in K \backslash (\Gamma_a \cup \Gamma_b)$.

To optimally solve problem $\mathrm{PP}^{(1)}$ and compute the corresponding residual cost vector requires $O(|K|)$ time to find edges λ, α, β and ε, and to define costs $\gamma_l^{(1)}$ for edges $l \in K$.

Solving partial problem $\mathrm{PP}^{(2)}$

Problem $\mathrm{PP}^{(2)}$ consists in finding an edge subset $\bar{E}^{(2)}$ containing $n-2$ edges different from $\bar{l}$, having minimum cost $\theta^{(2)} = \sum_{l \in \bar{E}^{(2)}} \bar{c}_l$ and such that the addition of edge $\bar{l}$ produces a spanning tree. Such a problem can be solved by applying any shortest spanning tree algorithm on the modified costs obtained by setting $\bar{c}_{\bar{l}} = -\infty$ (so as to impose edge $\bar{l}$ in the optimal tree), and then by removing edge $\bar{l}$ from the solution. Residual costs $\gamma_l^{(2)}$ are obtained from the reduced costs of the Shortest Spanning Tree Problem simply by setting $\gamma_{\bar{l}}^{(2)} = \infty$ (an easy proof of correctness can be derived from the fact that reapplying the algorithm on costs $\bar{c}_l - \gamma_l^{(2)}$ would produce the same value of $\vartheta^{(2)}$).

Choosing edge $\bar{l}$

Given vertex subset S with $2 \leq |S| \leq n-2$, different lower bounds can be obtained by choosing different edges $\bar{l} \in K \equiv (S, V \backslash S)$. However, it is easy to show that choosing $\bar{l} = \lambda$ (the minimum-cost edge in K) produces the best lower bound. In fact, $\vartheta^{(1)}$ does not depend on $\bar{l}$. As for $\vartheta^{(2)}$, its value is computed as the cost of the shortest spanning tree containing edge $\bar{l}$, minus $\bar{c}_{\bar{l}}$. Now let $G^T = (V, T)$ be the shortest spanning tree of G, which is known to contain edge λ. Since for each edge $\bar{l} \in K$ an edge $l' \in K \cap T$ always exists such that $G' = (V, T \backslash \{l'\} \cup \{\bar{l}\})$ is a spanning tree (with $l' = \bar{l}$ if $\bar{l} \in T$), we have

$$\vartheta^{(2)} \leq \left(\sum_{l \in T} \bar{c}_l - \bar{c}_{l'} + \bar{c}_{\bar{l}} \right) - \bar{c}_{\bar{l}} \leq \sum_{l \in T} \bar{c}_l - \bar{c}_\lambda .$$

Therefore, lower bound $\vartheta^{(2)}$, computed with respect to any edge $\bar{l} \in K$, cannot exceed that computed with respect to λ.

In the following, the lower bound based on disjoint edge pairs will always be computed with respect to the minimum cost edge in the chosen cut.

3.4. *A combined bound*

Let S be a given vertex subset of V. A lower bound associated with S as well as the corresponding residual costs can be obtained by combining the results of Sections 3.2 and 3.3 as follows.

If $|S| = 1$ (resp. $|S| = n - 1$), lower bound and residual costs are computed according to Section 3.2 with respect to $W = S$ (resp. $W = V\backslash S$), i.e. by solving the r-Shortest Spanning Tree Problem with $\{r\} = S$ (resp. $\{r\} = V\backslash S$).

Otherwise ($2 \leq |S| \leq n - 2$) lower bound and residual costs are computed according to Section 3.3 by considering $K = (S, V\backslash S)$ and $\bar{l}$ as the minimum cost edge in K.

4. Additive procedures

In this section we propose three bounding procedures, P2, P3 and P4.

4.1. *Procedure* P2

Bounding procedure P2 is based on the projective lower bound of Section 3.2, which is computed for all vertex subsets W having cardinality 1. For each vertex $r \in V$, Procedure P2 increases the current lower bound in additive way by computing the r-shortest spanning tree on the current residual costs (i.e. by considering $W = \{r\}$).

Procedure P2 takes $O(n^2)$ time if properly implemented (see Section 5.2).

At each iteration of the procedure, let $G_0 = (V, E_0)$ be the partial graph of G defined through the edges having current residual-cost equal to zero. Vertex r is said to be an *articulation point* of G_0 iff its removal produces a disconnected graph. After the iteration of P2 in which vertex r is considered, G_0 contains an r-spanning tree: r is then certainly not an articulation point of G_0, and at least two edges of E_0 are incident at r. Since residual costs are never increased, graph G_0 contains no articulation point at the end of the procedure, while each vertex has at least degree two in G_0.

4.2. *Procedure* P3

Bounding procedure P3 is based on the combined bound of Section 3.4. The approach is the following. First, the shortest spanning tree $G^T = (V, T)$ of G is computed with respect to the input costs. Let $\bar{l}$ be the edge in T having the maximum cost, and S the vertex set of one of the two components of G^T obtained by removing edge $\bar{l}$. The combined bound of Section 3.4 is then computed with respect to S. Note that in this case $n - 2$ of the n edges needed to compute the lower bound are those in $T\backslash\{\bar{l}\}$.

Procedure P3 takes $O(n^2)$ time, by using the algorithm of Prim (1957) for the shortest spanning tree computation. Our choice of vertex subset S is motivated by the fact that cut $K = (S, V\backslash S)$ is that whose minimum edge $(\bar{l})$ has the maximum cost.

The value of the lower bound can be further increased through Lagrangian relaxation (see Section 2.4). Procedure P3 considers the following set of valid linear equalities for STSP:

$$\sum_{l \in \Gamma_i} x_l = 2 \quad \text{for each } i \in V.$$

Lagrangian multipliers $\tilde{u}_i$ $(i \in V)$, which are not limited to being non-negative, are heuristically determined through subgradient-based techniques so as to produce the maximum value of the lower bound.

4.3. *Procedure* P4

Bounding procedure P4 exploits, in an additive way, the lower bound of Section 3.4 to increase the current bound. Procedure P4 is assumed to be applied to an input cost vector $\bar{c}$ such that partial graph $G_0 = (V, E_0)$ of the edges l with $\bar{c}_l = 0$, is connected (e.g., this is the case if P4 is applied after Procedure P3).

A graph search algorithm is first applied, starting from vertex 1, to find a depth-first spanning tree $G^T = (V, T)$ in G_0. Then, for each $l \in T$, the combined bound of Section 3.4 is computed with respect to vertex subset S corresponding to one of the two components of G^T obtained by removing edge l. Note that $n-2$ of the n edges needed for each bound computation are those in $T\backslash\{l\}$.

Procedure P4 takes $O(n^2)$ time if properly implemented (see Section 5.3).

The procedure computes the combined bound on $n-1$ vertex subsets S, whose choice is motivated by the following arguments. Clearly, the combined bound cannot produce a bound increase if $|S| = 1$ (or $|S| = n-1$) and at least two zero residual-cost edges exist in the cut $(S, V\backslash S)$. Hence the only cuts of this type which can produce a bound improvement are those associated with the leaves of tree G^T, which are all considered by Procedure P4. As for the remaining cuts, consider any $(S, V\backslash S)$ with $2 \leq |S| \leq n-2$, and let $l \equiv [i, j]$ be a minimum residual-cost edge in this cut (clearly $l \in T$). One can easily verify that if both vertices i and j are not articulation points of G_0, the bound based on disjoint edge pairs gives no improvement. On the other hand, if vertex i (or j) is an articulation point, a lower bound improvement is guaranteed if S is one of the components $C_1, C_2, \ldots, C_t$ obtained by removing vertex i from G_0. Because of the known properties of depth-first trees (see, e.g., Aho, Hopcroft and Ullman, 1974), we have it that $|(C_h, V\backslash C_h) \cap T| = 1$ for each component C_h non containing vertex 1 (the root of the depth-first search: see Figure 3). Hence our procedure considers as subsets S all components C_h $(h = 1, \ldots, t)$ except perhaps that containing vertex 1. For each such S, the corresponding cost reduction induces a zero-cost pair of disjoint edges in cut $(S, V\backslash S)$, thus introducing in G_0 a new edge connecting S with a different component. It turns out that, at the

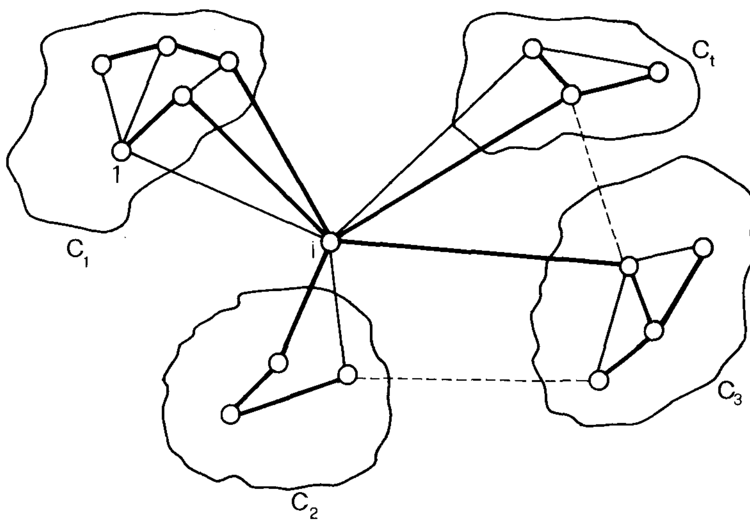

Fig. 3. Graph G_0: bold line, the depth-first tree G^T; dotted line, the new zero-cost edges at the end of Procedure P4.

end of Procedure P4, the number of components obtained by removing vertex i from G_0 is, at most, $\lfloor\frac{1}{2}(t-1)\rfloor+1=\lceil\frac{1}{2}t\rceil$ (the worst case arising when the same edge, connecting components C_h and C_k, is chosen in the disjoint pair when $S=C_h$ and $S=C_k$; see Figure 3).

The iterative application of Procedure P4 until no lower bound improvement is obtained, ensures that no articulation point exists in G_0, while each vertex has at least degree two. Since at each application of P4 the number of components corresponding to each articulation point is at least halved, no more than $\lceil\log_2 n\rceil$ executions of Procedure P4 are needed. In this way the exponential number of all possible vertex subsets S which can lead to a bound improvement, is implicitly considered in $O(n^2\log n)$ time.

5. Algorithm implementation

In this section efficient implementations of Procedures P2 and P4 are presented. In addition, a fast $O(n^2)$ algorithm for computation of the reduced costs of the Shortest Spanning Tree Problem is given.

5.1. SST *reduced costs computation*

Let $G^T=(V,T)$ be a shortest spanning tree of G, and for each edge $l\in E$ let C_l define the set containing the edges in T which make a cycle with l (with $C_l=\{l\}$ if

$l \in T$). The reduced cost c'_l of each edge $l \in E$ is $\bar{c}_l - \alpha_l$, where $\bar{c}_l$ is the input cost of edge l, and $\alpha_l = \max\{\bar{c}_k\colon k \in C_l\}$.

Suppose tree G^T is represented by an oriented tree rooted at vertex v_1, and let p_j be the predecessor of vertex j in the oriented tree. Define a topological ordering $v_1, \ldots, v_n$ of vertices $1, \ldots, n$ such that $p_{v_h} = v_k$ implies $k < h$. Given the tree, ordering $v_1, \ldots, v_n$ can easily be computed in $O(n)$ time (if the SST algorithm of Prim is used, the order in which the vertices are connected to the growing tree gives a suitable topological ordering). The following $O(n^2)$ algorithm considers the vertices according to the topological ordering (so as to ensure that each vertex is considered after its predecessor) and computes values α_l for all $l \in E$ (see also Volgenant and Jonker, 1983).

```
$\alpha_{[v_1,v_1]} := -\infty$;
for $h := 2$ to $n$ do
  begin (comment values $\alpha_{[v_h,v_t]}$ are computed for $t < h$)
    $j := v_h$; $i := p_j$; $\alpha_{[j,j]} := -\infty$;
    for $t := 1$ to $h-1$ do
      begin (comment $C_{[v_t,j]} = C_{[v_t,i]} \cup \{[i,j]\}$)
        $k := v_t$;
        $\alpha_{[k,j]} := \max\{\alpha_{[k,i]}, \bar{c}_{[i,j]}\}$
      end
  end.
```

5.2. *An* $O(n^2)$ *implementation of Procedure* P2

Procedure P2 computes, in an additive way, the r-SST with respect to all vertices $r \in V$. We assume that partial graph G_0 (containing the edges l whose current residual cost $\bar{c}_l$ is zero) is connected at the beginning of the procedure (if this is not the case, the shortest spanning tree is computed, the lower bound is increased and costs are correspondingly reduced). Hence let $G^T = (V, T)$ be any spanning tree of G_0, represented as an oriented tree. Our implementation is based on the property that any connected component S of G_0 containing no articulation point and no vertex with degree one, is "shrinkable" into a single *supervertex* (multiple edges incident at the supervertex being replaced by their minimum-cost edge). This operation does not affect the lower bound computation.

Vertices are considered in "postorder" (see, e.g., Tarjan, 1983), so as to ensure that each vertex is considered after its successors in the oriented tree G^T. Let r be the vertex considered at the current iteration, and $s_1, \ldots, s_k$ be the "son-vertices" of r in G^T (if any). For each $h = 1, \ldots, k$, vertex s_h and all its successors have been considered in the previous iterations (hence defining a shrinkable component of current graph G_0) and have been collapsed into a single supervertex S_h (see Figure 4). The r-SST is now computed on the current shrunk graph. After the corresponding

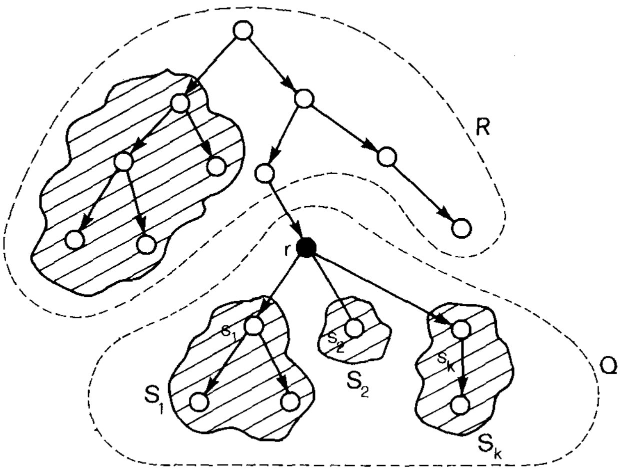

Fig. 4. Tree G^T and the shrunk graph when r-SST is computed.

cost reduction, vertices r and $S_1, \ldots, S_k$ define a new shrinkable component of G_0, and are collapsed into a single supervertex Q.

In our implementation, vertices r which are "leaves" of the undirected tree G^T are not elaborated. In fact, computing the r-SST for such vertices can only increase the bound by the value of the second minimum-cost edge incident at r, and introduce this edge in G_0. However, the same result is achieved as a byproduct when the "father-vertex" of r (or the unique "son-vertex" of r, if r is the root of the oriented tree), is elaborated.

For all non-leaves vertices r, the r-SST is computed as follows. In the current shrunk graph the (super)vertex set can be partitioned into $\{r\}$, $\{S_1, \ldots, S_k\}$ and R (see Figure 4). Determination of the two minimum-cost edges incident at r is clearly useless, since vertex r has at least degree two in G_0. As for computation of the SST of the graph obtained from the shrunk one by removing vertex r, this can be speeded up by temporarily collapsing the zero-cost component R into a single super-vertex S_{k+1}. This operation takes $O(kn)$ time to find, for each S_h $(h = 1, \ldots, k)$, the minimum-cost edge $[S_h, t]$ with $t \in R$. The spanning tree computation now takes $O((k+1)^2)$ time to compute both the value and the corresponding cost reductions $\alpha'_{[S_i, S_j]}$ for $i, j = 1, \ldots, k+1$ (see Section 5.1). Supervertex S_{k+1} is then re-expanded, while (super)vertices $S_1, \ldots, S_k$ and vertex r are permanently collapsed into supervertex Q. During this operation, costs are also reduced by subtracting value $\alpha'_{[S_h, S_{k+1}]}$ from the cost of all edges $[S_h, t]$ with $h = 1, \ldots, k$ and $t \in R$ (the cost of the edges $[r, i]$ with $i \in R$ and $[i, j]$ with $i, j \in R$ being unchanged). Also this phase takes $O(kn)$ time. Hence, the complete computation of the r-SST, the corresponding cost reduction, and the shrinking of r, $S_1, \ldots, S_k$ into supervertex Q, globally takes $O(kn)$ time, and removes k (super)vertices from the current graph. The $O(n^2)$ global time complexity follows.

The overall reduced costs for the original graph can be computed during the execution of Procedure P2 as sketched below. We define a "reduction credit" β_i associated with each vertex $i \in V$. At each iteration, for all edges $[a, b]$ having a and b in the same supervertex, the *final* cost reduction $\alpha_{[a,b]}$ has been computed, while for all the remaining edges $[i, j]$ the *current* cost reduction is $\beta_i + \beta_j$. Initially, $\beta_i = 0$ for each $i \in V$. When vertex r is elaborated, a further cost reduction $\alpha'_{[S_i, S_j]}$ between current (super)vertices S_i and S_j $(i, j = 1, \ldots, k+1)$ is computed. Before collapsing vertex r and (super)vertices $S_1, \ldots, S_k$, we define the final cost reduction of each edge $[a, b]$ having $a \in S_i$ and $b \in S_j$ $(i, j = 1, \ldots, k; i \neq j)$ as $\alpha_{[a,b]} = \beta_a + \beta_b + \alpha'_{[S_i, S_j]}$, while $\alpha_{[r,j]}$ is set to $\beta_r + \beta_j$ for each edge $[r, j]$ having $j \in S_h$ $(h = 1, \ldots, k)$. Values β_i for each vertex $i \in S_h$ $(h = 1, \ldots, k)$ are then updated as $\beta_i = \beta_i + \alpha'_{[S_h, S_{k+1}]}$.

The overhead time complexity of cost reduction is $O(n^2)$, since each value $\alpha_{[a,b]}$ is defined once in constant time, while each β_i is updated (in constant time), at most, n times.

It is worth noting that a straightforward modification of Procedure P2, consisting of removing cost reduction, allows computation of all the r-SST's $(r = 1, \ldots, n)$ in $O(n^2)$ time.

5.3. An $O(n^2)$ *implementation of Procedure* P4

Procedure P4 computes, in an additive way, the combined bound of Section 3.4. For the sake of simplicity, the $O(n^2)$ implementation given below is similar to that of the previous section. Slightly more effective implementations are also possible.

Let $G^T = (V, T)$ be the depth-first spanning tree of G_0, represented as an oriented tree rooted at vertex 1, and let p_j be the predecessor of vertex j in the tree. Determining G^T requires $O(n^2)$ time. Our implementation is based on the property that when the cut associated with edge $[r, p_r]$ is considered, all the vertices different from r and p_r which belong to the same "side" of the cut, are shrinkable into a single supervertex without affecting the bound computations.

Vertices are considered in postorder so as to ensure that each vertex is considered after its successors in G^T. Let r be the vertex considered at the current iteration, and $(S, V \backslash S)$ be the cut associated with edge $[r, p_r]$ (where S contains vertex r and all its successors in the tree). If $|S| = 1$ or $|S| = n - 1$, the lower bound associated with this cut, as well as the cost reductions, are easily computed in $O(n)$ time. Otherwise, let $s_1, \ldots, s_k$ be the "son-vertices" of r in G^T. Since vertex s_h $(h = 1, \ldots, k)$ and all its successors will always belong to the same "side" of all the cuts considered in the following iterations, they define a shrinkable vertex subset and have been collapsed into a single supervertex S_h in the previous iterations. Computing the minimum-cost disjoint edge pair and the corresponding dual variables w, u_{p_r} and u_r takes $O((k+1)n)$ time since, at most, $(k+1)n$ edges of the current shrunk graph cross the cut. Vertices r and $S_1, \ldots, S_k$ now define a new shrinkable vertex subset, and hence are collapsed into a single supervertex (during shrinking, costs are reduced according to dual variables w, u_{p_r} and u_r). Each such iteration globally requires $O((k+1)n)$ time and removes k vertices from the current graph. The overall time complexity of Procedure P4 is then $O(n^2)$.

The overall reduced costs for the original graph can be computed in $O(n^2)$ time by using a technique similar to that used in the previous section, the main difference being that, at each iteration, the final cost reduction $\alpha_{[a,b]}$ is defined for all edges $[a, b]$ having $a \in S_i$ and $b \in S_j$ $(i, j = 1, \ldots, k;\ i \neq j)$, for all edges $[r, j]$ having $j \in S_h$ $(h = 1, \ldots, k)$, as well as for the edges $[p_r, j]$ having $j \in S_h$ $(h = 1, \ldots, k)$ or $j = r$.

It is worth noting that a straightforward modification of Procedure P4—consisting of removing cost reductions and finding, for each cut $(S, V \backslash S)$ with $2 \leq |S| \leq n-2$, the minimum-cost edge not in T which crosses the cut (instead of the minimum-cost disjoint edge pair)—allows an $O(n^2)$-time sensitivity analysis of the $n-1$ edges belonging to a given shortest spanning tree $G^T = (V, T)$.

6. Computational results

The lower bounds proposed in the previous sections have been computationally evaluated and compared with the 1-SST relaxation with subgradient ascent (although tighter bounds can be obtained through linear programming relaxations incorporating several classes of facet-defining inequalities, as in Padberg and Rinaldi (1987), 1-SST with subgradient ascent is the bounding procedure commonly used in branch and bound algorithms). All the bounding procedures have been coded in FORTRAN IV and run on a Digital VAX 11/780.

The Assignment Problem and the Shortest Spanning Arborescence Problem, addressed in Procedure P1, have been solved through the Hungarian algorithm APC described in Carpaneto, Martello and Toth (1988) and through the implementation of Fischetti and Toth (1987) of the Edmonds (1967) algorithm, respectively (for more details on the overall implementation of P1, see Fischetti and Toth, 1990). The shortest paths have been computed through the algorithm of Dijkstra (1959), and the shortest spanning trees through that of Prim (1957).

As for the subgradient optimization procedure, we have used the one proposed by Volgenant and Jonker (1982), with the first step-size computed through a line search along the first available subgradient.

Three classes of randomly generated problems have been considered:

Class A: c_l uniformly random in range (1–1000).

Class B: $c_{[i,j]} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$, where $(x_i) - (y_i)$ are uniformly random in range (1–1000);

Class C: as for Class B, with points (x_i, y_i) uniformly generated inside five "clusters" (each cluster is represented through a square whose area is 20 000 and whose center is a uniformly random point in the 1000×1000 square).

Table 1
Lower bounds growth during ascent

Class	n	Ascent on 1-SST				Procedure P3			
		1 iterat.	10 iterat.	50 iterat.	100 iterat.	1 iterat.	10 iterat.	50 iterat.	100 iterat.
A	50	0.609	0.949	0.993	1.000	0.643	0.954	0.994	1.000
		(0.03)	(0.34)	(1.69)	(3.35)	(0.02)	(0.33)	(1.70)	(3.35)
	100	0.593	0.924	0.988	1.000	0.606	0.930	0.991	1.001
		(0.08)	(1.38)	(6.94)	(14.05)	(0.08)	(1.38)	(6.92)	(14.01)
	150	0.640	0.949	0.993	1.000	0.648	0.952	0.994	1.000
		(0.20)	(3.15)	(15.80)	(30.80)	(0.19)	(3.14)	(15.78)	(30.67)
	200	0.622	0.939	0.987	1.000	0.631	0.943	0.989	1.000
		(0.35)	(5.65)	(28.31)	(57.27)	(0.35)	(5.66)	(28.44)	(57.52)
B	50	0.877	0.964	0.996	1.000	0.897	0.969	0.997	1.000
		(0.03)	(0.35)	(1.71)	(3.46)	(0.02)	(0.36)	(1.80)	(3.63)
	100	0.898	0.969	0.996	1.000	0.910	0.971	0.997	1.001
		(0.08)	(1.41)	(7.02)	(14.19)	(0.09)	(1.42)	(7.20)	(14.63)
	150	0.903	0.963	0.991	1.000	0.912	0.968	0.994	1.000
		(0.20)	(3.19)	(15.99)	(32.33)	(0.20)	(3.21)	(16.09)	(32.53)
	200	0.902	0.975	0.995	1.000	0.911	0.980	0.997	1.001
		(0.36)	(5.75)	(28.76)	(58.16)	(0.36)	(5.77)	(28.88)	(58.38)
C	50	0.869	0.947	0.989	1.000	1.018	1.087	1.127	1.138
		(0.02)	(0.35)	(1.72)	(3.47)	(0.02)	(0.37)	(1.87)	(3.77)
	100	0.905	0.957	0.988	1.000	1.020	1.076	1.100	1.106
		(0.08)	(1.37)	(6.88)	(13.91)	(0.10)	(1.55)	(7.75)	(15.65)
	150	0.898	0.970	0.994	1.000	1.009	1.077	1.104	1.110
		(0.19)	(3.16)	(15.81)	(31.97)	(0.22)	(3.50)	(17.52)	(35.39)
	200	0.908	0.972	0.994	1.000	1.003	1.061	1.084	1.092
		(0.35)	(5.68)	(28.45)	(57.53)	(0.45)	(6.60)	(32.85)	(66.31)

All costs have been truncated so as to obtain integer values.

Problems of Class C represent real-life situations in which the "customers" to be visited are clustered in a few geographical regions.

For each class, four different values of n have been considered ($n = 50$, 100, 150, 200); for each value of n and for each class, five instances were solved.

Table 1 compares the average performance of the 1-SST relaxation having subgradient ascent with that of Procedure P3 (Section 4.2). The maximum number of ascent iterations has been fixed at 100. The table shows the lower bound value and the computing time after 1, 10, 50 and 100 ascent iterations. Each entry gives the average ratio (lower bound)/(final bound of the 1-SST ascent) and, in brackets, the corresponding computing time (in VAX 11/780 seconds).

The results show that for problems of Classes A and B, the initial bound computed by P3 is better than the value of the first 1-SST relaxation, while the difference between the two bounds tends to decrease with the number of ascent iterations (this is not surprising, since 1-SST relaxation with a large number of subgradient iterations is known to produce very tight bounds for these problems). As for problems of Class C, the bound obtained by P3 is initially better than the final 1-SST bound, and significantly grows during the ascent. The better behaviour of P3 is due to the fact that, for clustered problems, Procedure P3 generally adds a costly edge connecting different clusters to those of the shortest spanning tree, while 1-SST uses an edge inside the cluster containing vertex 1. The computing times of the two bounding procedures are equivalent.

Although the ascent consistently increases the lower bound value even in the last iterations, the corresponding times are quite large. Hence at the nodes of the branch decision tree different from the root, one is interested in a faster, although less

Table 2

Bounding procedures comparison

Class	*n*	P1	P1 + P2	P1 + P3 (20 it.)	P1 + P3 (20 it.) + P2	P3 (20 it.)	P3 (20 it.) + P4
A	50	0.929	0.961	0.947	0.966	0.996	0.997
		(0.29)	(0.42)	(1.11)	(1.17)	(0.78)	(0.91)
	100	0.914	0.938	0.927	0.942	0.987	0.990
		(1.29)	(1.80)	(4.57)	(4.83)	(3.22)	(3.69)
	150	0.921	0.953	0.933	0.955	0.992	0.993
		(3.05)	(4.24)	(10.48)	(11.09)	(7.36)	(8.42)
	200	0.897	0.940	0.911	0.943	0.989	0.991
		(5.89)	(7.99)	(19.23)	(20.32)	(13.27)	(15.18)
B	50	0.920	0.962	0.945	0.963	0.992	0.995
		(0.24)	(.036)	(1.04)	(1.11)	(0.83)	(0.95)
	100	0.937	0.965	0.948	0.969	0.995	0.997
		(0.97)	(1.48)	(4.40)	(4.66)	(3.32)	(3.77)
	150	0.935	0.958	0.951	0.962	0.994	0.995
		(2.26)	(3.46)	(9.78)	(10.39)	(7.52)	(8.58)
	200	0.932	0.971	0.948	0.974	0.994	0.995
		(4.02)	(6.17)	(17.49)	(18.58)	(13.48)	(15.39)
C	50	1.119	1.160	1.153	1.165	1.109	1.126
		(0.24)	(0.36)	(1.06)	(1.13)	(0.86)	(0.99)
	100	1.087	1.108	1.112	1.115	1.097	1.100
		(0.90)	(1.41)	(4.19)	(4.45)	(3.63)	(4.10)
	150	1.103	1.134	1.126	1.139	1.096	1.106
		(2.10)	(3.25)	(10.04)	(10.63)	(8.20)	(9.44)
	200	1.067	1.085	1.081	1.089	1.081	1.086
		(3.91)	(5.99)	(17.57)	(18.62)	(15.40)	(17.34)

accurate, bounding procedure. To this end, six different bounding procedures have been computationally evaluated:

- Procedure P1 (Section 3.1)
- Procedures P1 and P2 (Section 4.1) in sequence, according to the additive approach;
- Procedures P1 and P3 in sequence;
- Procedures P1, P3 and P2 in sequence;
- Procedure P3;
- Procedures P3 and P4 in sequence (with P4 reapplied until no bound increase occurs, see Section 4.3).

The maximum number of ascent iterations of Procedure P3 has been fixed at 20.

Procedures P2 and P4 proved to be almost equivalent for both computing time and bound quality, and could be interchanged.

Table 2 compares the average performances of the six bounding procedures above. Table entries give the same information as those of Table 1.

Table 2 shows that Procedures P1 and P1 + P2 are very fast, and produce very good results for problems of Class C. Procedures P1 + P3 and P1 + P3 + P2 require longer computing times, but for Classes A and B yield only small bound improvements with respect to P1 and P1 + P2, respectively (these results could probably be improved by tuning the ascent technique used in P3 so as to take into account the fact that reduced costs are considered instead of the original ones). Procedures P3 and P3 + P4 produce, for problems of Classes A and B, lower bounds comparable with those obtained after 50 ascent iterations of the 1-SST relaxation with computing times approximately halved (see Table 1); as for problems of Class C, they both outperform the final 1-SST lower bound.

Computational results show that the new lower bounds improve on the performance of that based on 1-SST relaxation, mainly for problems of Class C. The practical effect of the new bounds on the behaviour of branch and bound algorithms, as regards both fathomings and initial reductions, is to be investigated.

Acknowledgment

Work supported by Ministero della Pubblica Istruzione, Italy.

References

A.V. Aho, J.E. Hopcroft and J. Ullman, *The Design and Analysis of Computer Algorithms* (Addison-Wesley, Reading, MA, 1974).

E. Balas and N. Christofides, "A restricted Lagrangian approach to the Traveling Salesman Problem," *Mathematical Programming* 21 (1981) 19-46.

G. Carpaneto, M. Dell'Amico, M. Fischetti and P. Toth, "A branch and bound algorithm for the Multiple Depot Vehicle Scheduling Problem," *Networks* 19 (1989).

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 (1988) 193-223.

N. Christofides, "The shortest Hamiltonian chain of a graph," *SIAM Journal on Applied Mathematics* 19 (1970) 689-696.

E.W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik* 1 (1959) 269-271.

J. Edmonds, "Optimum branchings," *Journal of Research of the National Bureau of Standards* 71B (1967) 233-240.

M. Fischetti and P. Toth, "An efficient algorithm for the Min-Sum Arborescence Problem," Technical Report OR/87/7, DEIS, University of Bologna (Bologna, 1987).

M. Fischetti and P. Toth, "An additive approach for the optimal solution of the Prize-Collecting Travelling Salesman Problem," in: B. Golden and A.A. Assad, eds., *Vehicle Routing: Methods and Studies* (North-Holland, Amsterdam, 1988) pp. 319-343.

M. Fischetti and P. Toth, "An additive bounding procedure for combinatorial optimization problems," *Operations Research* 37 (1989) 319-328.

M. Fischetti and P. Toth, "An additive bounding procedure for the Asymmetric Travelling Salesman Problem," submitted to *Mathematical Programming* (1990).

B. Gavish and K. Srikanth, "An optimal method for large-scale multiple traveling salesman problems," *Operations Research* 34 (1986) 698-717.

K. Helbing Hansen and J. Krarup, "Improvements of the Held-Karp Algorithm for the Symmetric Traveling-Salesman Problem," *Mathematical Programming* 7 (1974) 87-96.

M. Held and R.M. Karp, "The Traveling-Salesman Problem and minimum spanning trees," *Operations Research* 18 (1970) 1138-1162.

M. Held and R.M. Karp, "The Traveling-Salesman Problem and minimum spanning trees: Part II," *Mathematical Programming* 1 (1971) 6-25.

J.B. Kruskal, "On the shortest spanning subtree of a graph and the Traveling Salesman Problem," *Proceedings of the American Mathematical Society* 7 (1956) 48-50.

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

E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan and D.B. Shmoys, *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization* (Wiley, Chichester, 1985).

M.W. Padberg and M. Grötschel, "Polyhedral computations," in: E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan and D.B. Shmoys, eds., *The Traveling Salesman Problem: a Guided Tour of Combinatorial Optimization* (Wiley, Chichester, 1985).

M.W. Padberg and G. Rinaldi, "Optimization of a 532-City Symmetric Traveling Salesman Problem by branch and cut," *Operations Research Letters* 6 (1987) 1-8.

R.C. Prim, "Shortest connection networks and some generalizations," *BSTJ* 36 (1957) 1389-1401.

T.H.C. Smith and G.L. Thompson, "A LIFO implicit enumeration search algorithm for the Symmetric Traveling Salesman Problem using Held and Karp's 1-tree relaxation," *Annals of Discrete Mathematics* 1 (1977) 479-493.

R.E. Tarjan, "Finding optimum branchings," *Networks* 7 (1977) 25-35.

T. Volgenant and R. Jonker, "A branch and bound algorithm for the Symmetric Traveling Salesman Problem based on the 1-tree relaxation," *European Journal of Operational Research* 9 (1982) 83-89.

T. Volgenant and R. Jonker, "The Symmetric Traveling Salesman Problem and edge exchanges in minimal 1-trees," *European Journal of Operational Research* 12 (1983) 394-403.