A Branch-And-Cut Algorithm for the Symmetric Generalized Traveling Salesman Problem

Matteo Fischetti; Juan Jose Salazar Gonzalez; Paolo Toth

Operations Research, Vol. 45, No. 3 (May - Jun., 1997), 378-394.

Stable URL:
http://links.jstor.org/sici?sici=0030-364X%28199705%2F06%2945%3A3%3C378%3AABAFTS%3E2.0.CO%3B2-B

Operations Research is currently published by INFORMS.

A BRANCH-AND-CUT ALGORITHM FOR THE SYMMETRIC GENERALIZED TRAVELING SALESMAN PROBLEM

MATTEO FISCHETTI

University of Padova, Italy

JUAN JOSÉ SALAZAR GONZÁLEZ

University of La Laguna, Spain

PAOLO TOTH

University of Bologna, Italy

(Received January 1994; revisions received May 1995; accepted August 1995)

We consider a variant of the classical symmetric Traveling Salesman Problem in which the nodes are partitioned into clusters and the salesman has to visit at least one node for each cluster. This *NP*-hard problem is known in the literature as the symmetric Generalized Traveling Salesman Problem (GTSP), and finds practical applications in routing, scheduling and location-routing. In a companion paper (Fischetti et al. 1995) we modeled GTSP as an integer linear program, and studied the facial structure of two polytopes associated with the problem. Here we propose exact and heuristic separation procedures for some classes of facet-defining inequalities, which are used within a branch-and-cut algorithm for the exact solution of GTSP. Heuristic procedures are also described. Extensive computational results for instances taken from the literature and involving up to 442 nodes are reported.

Routing and Scheduling problems often require the determination of optimal sequences subject to a given set of constraints. The best known problem of this type is the classical *Traveling Salesman Problem* (TSP), calling for a minimum cost Hamiltonian cycle on a given graph. This problem has been extensively studied in the last decades; see the book edited by Lawler et al. (1985) for a comprehensive cover of the literature up to 1985.

In several applications a feasible cycle is allowed to visit only a subset of the nodes of the graph, chosen according to a specified criterion. For example, in the *Prize Collecting TSP* each node has an associated prize, and a salesman calls for a minimum cost cycle covering a node subset whose total prize is not less than a given value. This problem has been studied, among others, by Balas (1989, 1993) and Fischetti and Toth (1988).

In this paper we consider a different "nonspanning" version of the classical TSP, known in the literature as the *Generalized Traveling Salesman Problem* (GTSP), in which the nodes are partitioned into *clusters*, and the problem calls for a minimum cost cycle visiting at least one node for each cluster.

A different version of the problem, called E-GTSP in the sequel (where E stands for Equality), arises when imposing the additional constraint that exactly one node of each cluster must be visited.

GTSP and E-GTSP are useful models for problems involving simultaneous selection and sequencing decisions, e.g., as in location-routing problems. They find practical applications in warehouse order picking with multiple stock locations, sequencing computer files, routing of welfare clients through governmental agencies, airport selection and routing for courier planes, flexible manufacturing scheduling, postal routing, and the design of ring networks; see Noon (1988) and Noon and Bean (1991).

Both GTSP and E-GTSP are clearly *NP*-hard, as they reduce to TSP when each cluster contains only one node. They have been studied, among others, by Laporte and Nobert (1983), Salazar (1992), and Sepehri (1991); their asymmetric counterparts have been investigated in Laporte et al. (1987), and Noon and Bean (1991).

In this paper we propose an exact algorithm for GTSP, based on a cutting plane/branch-and-bound approach. At each node of the branch-decision tree, a lower bound on the optimal solution value is obtained by solving an LP relaxation of GTSP. The relaxation is iteratively tightened by adding valid inequalities that are violated by the current LP optimal solution; these inequalities are identified through exact or heuristic separation procedures.

A basic model of GTSP is given in Section 1. In Section 2 we review some polyhedral results on GTSP described in fuller detail in Fischetti et al. (1995). Exact and heuristic separation procedures for finding violated inequalities are proposed in Section 3. Several heuristic algorithms for finding approximate GTSP solutions are presented in Section 4, where we also show that E-GTSP is polynomially solvable when the sequence of the m clusters is known. Section 5 gives the description of the overall enumerative algorithm for the exact GTSP solution. Extensive computational results on several classes of test problems are finally reported in Section 6.

Subject classifications: Networks/graphs: Generalized Traveling Salesman. Programming, integer, cutting plane: Branch-and-cut algorithm.
Area of review: OPTIMIZATION.

0030-364X/97/4503-0378 $05.00

1. BASIC MODEL

We are given a complete (loop-free) undirected graph $G = (N, E)$ with node set $N := \{1, \ldots, n\}$ and edge set $E := \{[i, j] : i, j \in N, i \neq j\}$. In addition, a proper partition $C_1, \ldots, C_m$ of N is given in which the node subsets C_h are called *clusters*. Let c_e be the *cost* associated with each edge $e \in E$. A cycle is called *feasible* if it is simple and goes through each cluster at least once. GTSP then consists of finding a feasible cycle $T \subset E$ whose global cost $\Sigma_{e \in T} c_e$ is a minimum. The problem involves two related decisions:

(i) choosing a node subset $S \subseteq N$, such that $|S \cap C_h| \geq 1$ for all $h = 1, \ldots, m$;

(ii) finding a minimum cost Hamiltonian cycle in the subgraph of G induced by S.

We now introduce the main notation used in the sequel. For each $S \subseteq N$, let

$$E(S) := \{[i, j] \in E : i \in S, j \in S\},$$

$$\delta(S) := \{[i, j] \in E : i \in S, j \notin S\},$$

$$\mu(S) := |\{h : C_h \subseteq S\}|,$$

$$\eta(S) := |\{h : C_h \cap S \neq \emptyset\}|.$$

For $v \in N$ we write $\delta(v)$ instead of $\delta(\{v\})$, and denote by $C_{h(v)}$ the cluster containing v. We also define

$$W := \{v \in N : |C_{h(v)}| = 1\}.$$

An integer linear programming model for GTSP is as follows. Let $x_e = 1$ if edge $e \in E$ is chosen in the optimal solution, $x_e = 0$ otherwise. In addition, let $y_v = 1$ if node $v \in N$ is visited, $y_v = 0$ otherwise. GTSP then calls for

$$v(\text{GTSP}) := \min \sum_{e \in E} c_e x_e, \tag{1.1}$$

subject to

$$\sum_{e \in \delta(v)} x_e = 2y_v \quad \text{for } v \in N, \tag{1.2}$$

$$\sum_{v \in C_h} y_v \geq 1 \quad \text{for } h = 1, \ldots, m, \tag{1.3}$$

$$\sum_{e \in \delta(S)} x_e \geq 2(y_i + y_j - 1) \quad \text{for } S \subset N,\ 2 \leq |S| \leq n - 2,\ i \in S,\ j \in N \backslash S \tag{.14}$$

$$x_e \in \{0, 1\} \quad \text{for } e \in E, \tag{1.5}$$

$$y_v \in \{0, 1\} \quad \text{for } v \in N. \tag{1.6}$$

Constraints (1.2) require the number of edges incident with a node to be either 2 (if v is visited) or 0 (otherwise). Constraints (1.3) impose that at least one node in each cluster is visited. Inequalities (1.4) are connectivity constraints saying that each cut separating two visited nodes (i and j) must be crossed at least twice.

Constraints (1.4) are valid whenever one looks for a single (not necessarily Hamiltonian) cycle; their asymmetric counterpart has been studied by Balas (1989, 1993) for the Prize Collecting TSP. Stronger inequalities can in some cases be derived for GTSP and E-GTSP, which exploit the particular structure of these problems (see Section 2).

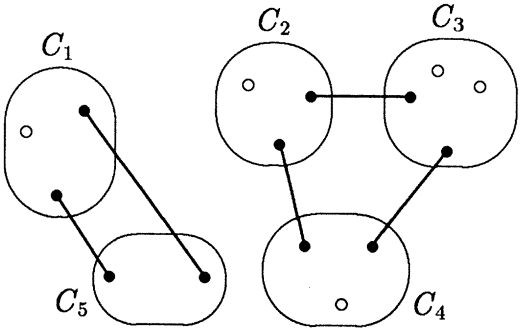

Figure 1. Infeasible GTSP solution satisfying (1.2)–(1.5). The drawn edges have $x_e = 1$, the blank nodes $y_v = 0$, and the black nodes $y_v = 1/2$.

As to E-GTSP, a mathematical model is obtained from (1.1)–(1.6) by replacing (1.3) with

$$\sum_{v \in C_h} y_v = 1 \quad \text{for } h = 1, \ldots, m. \tag{1.7}$$

Notice that GTSP and E-GTSP are equivalent when the costs satisfy the triangle inequality, i.e., $c_{ij} \leq c_{ik} + c_{kj}$ for all node triples (i, j, k).

Model (1.1)–(1.6) heavily relies on the integrality of the y variables. If this requirement is relaxed, solutions like those of Figure 1 become feasible. Therefore the LP relaxation of this model can be very poor. In Section 2 we will describe additional valid inequalities whose introduction in the model leads to considerable strengthening of its LP relaxation.

2. POLYHEDRAL RESULTS

Let P and $P^=$ denote the GTSP and E-GTSP *polytopes*, respectively, i.e.,

$$P := \text{conv}\{(x, y) \in \mathbb{R}^{E \cup N} : (x, y) \text{ satisfies (1.2)–(1.6)}\},$$

and

$$P^= := \text{conv}\{(x, y) \in \mathbb{R}^{E \cup N} : (x, y) \text{ satisfies (1.2), (1.7) and (1.4)–(1.6)}\}.$$

We briefly summarize some polyhedral results for these polytopes given in Fischetti et al. (1995). As a technical assumption we require $m \geq 5$.

2.1. The GTSP Polytope

We first address the GTSP polytope, P.

Theorem 2.1. $\dim(P) = |E| - |W|$.

Theorem 2.2. *Let $S \subset N$ be such that $2 \leq |S| \leq n - 2$. Then the* Generalized Subtour Elimination Constraint (*GSEC, for short*):

$$\sum_{e \in \delta(S)} x_e \geq 2 \quad \text{if } \mu(S) \neq 0,\ \mu(N \backslash S) \neq 0, \tag{2.1}$$

$$\sum_{e \in \delta(S)} x_e \geq 2y_i \quad \text{if } \mu(S) = 0,\ \mu(N \backslash S) \neq 0,\ i \in S, \tag{2.2}$$

$$\sum_{e\in\delta(S)} x_e \geq 2(y_i + y_j - 1)$$

$$\text{if } \mu(S) = \mu(N\backslash S) = 0,\ i \in S,\ j \in N\backslash S, \tag{2.3}$$

defines a facet of P.

Notice that (2.3) coincides with (1.4), whereas (2.1) and (2.2) are strengthened versions of (1.4) obtained when S and/or $N\backslash S$ contain a whole cluster.

By possibly interchanging the role of S and $N\backslash S$, one can always assume that inequalities (2.1) and (2.3) are written for $S \subset N$ such that $|S| \leq \lfloor n/2 \rfloor$. The same holds for inequalities (2.2) by choosing $i \in N\backslash S$ when $\mu(S) \neq 0$ and $\mu(N\backslash S) = 0$.

Notice that (2.2) are also valid (but not facet-inducing) when $\mu(S) \neq 0$. Analogously, inequalities (2.3) hold for any $S \subset N$ and coincide with (1.4).

By exploiting Equations (1.2), the inequalities above can be rewritten as:

$$\sum_{e\in E(S)} x_e \leq \sum_{v\in S} y_v - 1 \quad \text{if } \mu(S) \neq 0,\ \mu(N\backslash S) \neq 0, \tag{2.4}$$

$$\sum_{e\in E(S)} x_e \leq \sum_{v\in S\backslash\{i\}} y_v \quad \text{if } \mu(S) = 0,\ \mu(N\backslash S) \neq 0,\ i \in S, \tag{2.5}$$

$$\sum_{e\in E(S)} x_e \leq \sum_{v\in S\backslash\{i\}} y_v - y_j + 1$$

$$\text{if } \mu(S) = \mu(N\backslash S) = 0,\ i \in S,\ j \in N\backslash S. \tag{2.6}$$

This form of the constraints has the advantage of having fewer nonzero coefficients (assuming $|S| \leq \lfloor n/2 \rfloor$), hence it is more suitable for a cutting plane approach.

Notice that, because of (1.2), inequality (1.3) is equivalent to

$$\frac{1}{2}\sum_{v\in C_h}\sum_{e\in\delta(v)} x_e = \sum_{e\in E(C_h)} x_e + \frac{1}{2}\sum_{e\in\delta(C_h)} x_e \geq 1,$$

hence it is dominated by inequality (2.1) written for $S = C_h$.

Particular cases of GSECs arise when $|S| = 2$, leading to

$$x_e \leq y_v \quad \text{for } v \in N,\ e \in \delta(v). \tag{2.7}$$

These constraints prevent infeasibilities like those of Figure 1.

We now address *comb* inequalities. A comb is a family $C = (H, T_1, \ldots, T_s)$ of $s + 1$ node subsets, where $s \geq 3$ is an odd integer; see Figure 2 for an illustration. H is called the *handle* of C, whereas $T_1, \ldots, T_s$ are called *teeth*. Moreover, the following conditions must be satisfied:

(i) $T_1, \ldots, T_s$ are pairwise disjoint;

(ii) $T_j \cap H \neq \emptyset$ and $T_j \setminus H \neq \emptyset$ for $j = 1, \ldots, s$.

The *size* of C is defined as $\sigma(C) := |H| + \sum_{j=1}^{s} (|T_j| - 1) - (s + 1)/2$.

The *comb inequality* associated with C reads

$$\sum_{e\in E(H)} x_e + \sum_{j=1}^{s}\sum_{e\in E(T_j)} x_e \leq \sigma(C), \tag{2.8}$$

and is known to be valid and facet-defining for TSP.

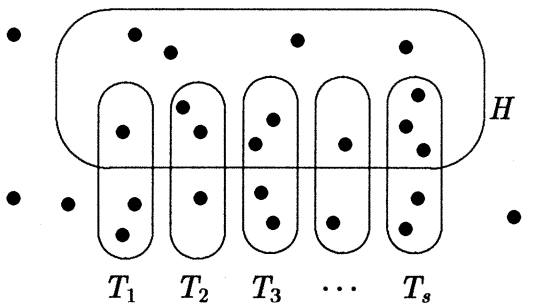

Figure 2. A comb.

Starting with (2.8), one can obtain related facet-defining inequalities for GTSP.

Theorem 2.3. *Let $C = (H, T_1, \ldots, T_s)$ be a comb. For $j = 1, \ldots, s$, let a_j be any node in $T_j \cap H$ if $\mu(T_j \cap H) = 0$, $a_j = 0$ (a dummy value) otherwise; and let b_j be any node in $T_j\backslash H$ if $\mu(T_j\backslash H) = 0$, $b_j = 0$ otherwise. Then the following* generalized comb inequality *defines a facet of P:*

$$\sum_{e\in E(H)} x_e + \sum_{j=1}^{s}\sum_{e\in E(T_j)} x_e + \sum_{v\in N} \beta_v(1 - y_v) \leq \sigma(C), \tag{2.9}$$

where $\beta_v = 0$ for all $v \in N\backslash(H \cup T_1 \cup \ldots \cup T_s)$, $\beta_v = 1$ for all $v \in H\backslash(T_1 \cup \ldots \cup T_s)$, and for $j = 1, \ldots, s$:

$$\beta_v = 2 \quad \text{for } v \in T_j \cap H,\ v \neq a_j;$$

$$\beta_{a_j} = 1 \quad \text{if } a_j \neq 0;$$

$$\beta_v = 1 \quad \text{for } v \in T_j\backslash H,\ v \neq b_j;$$

$$\beta_{b_j} = 0 \quad \text{if } b_j \neq 0.$$

In analogy with TSP, when $|T_j| = 2$ for all j we call the generalized comb inequality a *generalized 2-matching inequality.*

2.2. The E-GTSP Polytope

We now address the polyhedral structure of the E-GTSP polytope, $P^{=}$. This polytope is clearly a face of P, hence all facet-defining inequalities for P studied in Section 2.1 are also valid (but not necessarily facet-defining) for $P^{=}$.

Since in E-GTSP exactly one node of each cluster must be visited, we can drop intra-cluster edges, and re-define the edge-set as

$$E := \{[i, j] : i \in N,\ j \in N\backslash C_{h(i)}\}.$$

In view of this reduction, constraints (1.7) are equivalent to

$$\sum_{e\in\delta(C_h)} x_e = 2 \quad \text{for } h = 1, \ldots, m. \tag{2.10}$$

An alternative model for E-GTSP can be obtained from (1.1)–(1.6) by replacing constraints (1.4) and (1.6) by the two families of constraints:

$$\sum_{e \in E(S)} x_e \leqslant r - 1$$

$$\text{for } S = \cup_{i=1}^{r} C_{l_i} \text{ and } 2 \leqslant r \leqslant m - 2, \tag{2.11}$$

$$\sum_{e \in \delta(C_l) \cap \delta(w)} x_e \leqslant y_w \quad \text{for } l = 1, \ldots, m \text{ and } w \in N \backslash C_l, \tag{2.12}$$

respectively. Inequalities (2.11) will be called *Basic Generalized Subtour Elimination Constraints* (Basic GSECs), and (2.12) will be called *fan inequalities*. Both are particular cases of the GSECs (2.4) because of (1.7). Indeed, (2.11) are equivalent to the GSECs (2.4) written for $S = \cup_{i=1}^{r} C_{l_i}$, where $\Sigma_{v \in S} y_v = r$ since $\Sigma_{v \in C_{l_i}} y_v = 1$ for $i = 1, \ldots, r$. Analogously, (2.12) arise from (2.4) when $S = C_l \cup \{w\}$, since $E(S) = \delta(C_l) \cap \delta(w)$ and $\Sigma_{v \in S} y_v = \Sigma_{v \in C_l} y_v + y_w = 1 + y_w$.

Theorem 2.4. $\dim(P^{=}) = |E| - m$.

Theorem 2.5. *GSECs* (2.5) *and* (2.6) *do not define facets of* $P^{=}$.

Theorem 2.6. *The GSEC* (2.4) *defines a facet of* $P^{=}$ *if and only if one of the following conditions holds*:

(i) $S \subseteq W$ *and* $|S| = 2$,

(ii) $S = C_l \cup \{w\}$ *for some* $w \in N \backslash (C_l \cup W)$,

(iii) $\eta(S) \geqslant 3$ *and* $\eta(N \backslash S) \geqslant 3$.

Corollary 2.1. *The Basic GSEC* (2.11) *defines a facet of* $P^{=}$ *if and only if* $3 \leqslant r \leqslant m - 3$, *or* $r = 2$ *and* $|S| = 2$, *or* $r = m - 2$ *and* $|N \backslash S| = 2$.

Corollary 2.2. *The fan inequality* (2.12) *defines a facet of* $P^{=}$ *if and only if* $w \notin W$, *or* $w \in W$ *and* $|C_l| = 1$.

Theorem 2.7. *The generalized comb inequality* (2.9) *defines a facet of* $P^{=}$ *in case* $\mu(T_j \cap H) \neq 0$ *and* $\mu(T_j \backslash H) \neq 0$ *for all* $j = 1, \ldots, s$.

3. SEPARATION ALGORITHMS

In this section we address the following *separation* (or *identification*) *problem*: Given a (fractional) point $(x^*, y^*) \in [0, 1]^{E \cup N}$, find a member $\alpha x + \beta y \geqslant \gamma$ of a given family $\mathcal{F}$ of valid inequalities for GTSP, such that $\alpha x^* + \beta y^* < \gamma$. An effective exact/heuristic solution of this problem is of fundamental importance in order to use the inequalities of $\mathcal{F}$ within a cutting plane algorithm.

3.1. An Exact Separation Algorithm for Generalized Subtour Elimination Constraints

We consider the family $\mathcal{F}$ of the generalized subtour elimination constraints, in their cut form (2.1)–(2.3).

We start with constraints (2.3):

$$\sum_{e \in \delta(S)} x_e \geqslant 2(y_i + y_j - 1)$$

$$\text{if } \mu(S) = \mu(N \backslash S) = 0, \; i \in S, \; j \in N \backslash S.$$

Suppose nodes i and j have been fixed. Then, finding a most violated inequality (2.3) calls for the minimum-capacity cut $(S, N \backslash S)$ with $i \in S$ and $j \in N \backslash S$ in the capacitated undirected graph G^* obtained from G by imposing a capacity x_e^* for each $e \in E$. This can be done in $O(n^3)$ time, as it amounts to finding the maximum flow from i to j (see, e.g., Ahuja et al. 1989). If the maximum flow value is not less than $2(y_i^* + y_j^* - 1)$, then all the inequalities (2.3) for the given pair (i, j) are satisfied; otherwise the capacity of the minimum cut separating i and j is strictly less than $2(y_i^* + y_j^* - 1)$ and a most violated inequality (2.3) has been detected among those for the given pair (i, j). Trying all possible pairs (i, j) then produces an $O(n^5)$ overall separation algorithm. Actually, a better algorithm having overall $O(n^4)$ time complexity can be obtained, in analogy with the TSP case (see Padberg and Grötschel 1985) by using the Gomory-Hu (1961) scheme for the multiterminal flow problem. A simpler algorithm with the same time complexity is based on the simple observation that, for any S, the most violated inequality (2.3) arises when the chosen i and j are such that $y_i^* = \max\{y_v^* : v \in S\}$ and $y_j^* = \max\{y_v^* : v \in N \backslash S\}$. Therefore, any node s with $y_s^* = \max\{y_v^* : v \in N\}$ can always be fixed to play the role of, say, node i. In this way, one as to solve (at most) $n - 1$ max-flow problems in the attempt to send $2(y_s^* + y_j^* - 1)$ units of flow from s to any $j \in N \backslash \{s\}$. Clearly, nodes j with $y_s^* + y_j^* - 1 \leqslant 0$ need not be considered.

We now address inequalities (2.2):

$$\sum_{e \in \delta(S)} x_e \geqslant 2y_i \quad \text{if } \mu(S) = 0, \; \mu(N \backslash S) \neq 0, \; i \in S.$$

As before, we assume that cluster C_h and node $i \notin C_h$ are fixed. In this case a most violated constraint (2.2) corresponds to a minimum-capacity cut $(S, N \backslash S)$ with $i \in S$ and $C_h \subseteq N \backslash S$ in the capacitated graph G^*. Hence it can be detected by finding the maximum flow from i to t, where t is an additional node connected with each $j \in C_h$ through an edge having very large capacity (this corresponds to shrinking cluster C_h into a single node). Trying all (i, C_h) pairs leads to an $O(mn^4)$ time algorithm. Clearly, nodes i with $y_i^* = 0$ need not be considered.

As to constraints (2.1)

$$\sum_{e \in \delta(S)} x_e \geqslant 2 \quad \text{if } \mu(S) \neq 0, \; \mu(N \backslash S) \neq 0,$$

for all pairs (C_h, C_k) of distinct clusters a most violated inequality (2.1) is found by finding the maximum flow from s to t, where s (resp. t) is an additional node connected with each $j \in C_h$ (resp. $j \in C_k$) by means of an edge having very large capacity. The overall time complexity of this phase is $O(m^2 n^3)$.

Notice that a violated inequality (2.2) or (2.3) found by the above described separation algorithm, is not necessarily facet-defining. For (2.3) this occurs when there exists a cluster C_h contained in S or $N \backslash S$; for (2.2), this happens when there exists a cluster contained in the shore of the cut including node i. In these cases one should obviously reject the inequality in favour of its facet-inducing strengthening (2.1) or (2.2).

According to the above scheme the separation algorithm for the overall family $\mathcal{F}$ containing inequalities (2.1)–(2.3) requires $O(mn^4)$ time in the worst case. In practice, the computing time required is typically much smaller as the capacitated graph G^* is very sparse, and has many isolated nodes. Moreover, as previously explained, several max-flow computations can be avoided because some entries of y^* have a small value. In addition, parametric considerations on the structure of the cuts can further reduce the number of max-flow computations.

We now consider the important case in which $y_s^* := \max\{y_v^* : v \in N\} = 1$, that arises very frequently during the cutting-plane algorithm. In this case one can find a most violated generalized subtour elimination constraint by computing no more than $n + m - 2$ max-flows, with overall $O(n^4)$ time complexity. Indeed, the degree of violation of any inequality (2.1) with, say, $C_h \subseteq S$ and $C_k \subseteq N\backslash S$ is the same as that associated with inequality (2.2) written for the same S and for $i = s$. Hence inequalities (2.1) need not be considered. Now consider any inequality (2.2) with $i \neq s$. To fix the ideas, let $i \in S$ and $C_h \subseteq N\backslash S$. If $s \in S$, then the degree of violation of the inequality does not decrease by replacing i with s. Otherwise, the degree of violation is the same as that of inequality (2.3) written for $j = s$. It follows that inequalities (2.2) with $i \neq s$ need not be considered. As a result, one has to consider explicitly only the inequalities (2.2) with $i = s$, and the inequalities (2.3) (for which $i = s$ can again be assumed).

A final comment is in order. Let us consider a flow from s to a given node j, and let f_v denote the amount of flow going through any node $v \in N$. It is then easy to see that for all $v \in N$ there exists a feasible flow from s to v of value $\lambda_v = f_v + \min\{f_v, f_j - f_v\}$. Indeed the flow from s to j is carried through a family of (not necessarily arc disjoint) directed paths. Some of these paths go through v, thus reverting the flow direction of the portion of the paths from v to j produces a feasible flow from s to v of value λ_v.

A Pascal-like description of the separation algorithm follows. The algorithm maintains a label α_v representing a lower bound on the value of the maximum flow from s to any $v \in N\backslash\{s\}$. Moreover, we have a label β_h giving a lower bound on the value of the maximum flow from s to any cluster $C_h \neq C_{h(s)}$.

procedure GSEC_SEP;
input: the point $(x^*, y^*) \in [0, 1]^{E \cup N}$;
output: a sequence $S_1, \ldots, S_r$ of (not necessarily distinct) node subsets corresponding to violated GSECs;
begin

1. rename the nodes so that $y_1^* \geq y_2^* \geq \ldots \geq y_n^*$;
2. **for all** $v \in N$ **do** $\alpha_v := 0$;
3. $r := 0$; $s := 1$;
4. **for** $j := 2$ **to** n **such that** $\alpha_j < 2(y_s^* + y_j^* - 1)$ **do**
5. **begin** (**comment:** inequalities (2.3))
6. compute the maximum flow from s to j, and let f_v be the flow entering $v \in N$, and $(S, N\backslash S)$ be a minimum capacity cut from s to j;
7. **for all** $v \in N$ **do** $\alpha_v := \max\{\alpha_v, \lambda_v\}$, where $\lambda_v := f_v + \min\{f_v, f_j - f_v\}$;
8. **if** $f_j < 2(y_s^* + y_j^* - 1)$ **then** $r := r + 1$, $S_r := S$

end;

9. **for all** $h \in \{1, \ldots, m\}$ do $\beta_h := \max\{\alpha_v : v \in C_h\}$;
10. **for all** $h \in \{1, \ldots, m\}\backslash\{h(s)\}$ **such that** $\beta_h < 2y_s^*$ and $y_j^* < 1$ for all $j \in C_h$ **do**
11. **begin** (**comment**: inequalities (2.2))
12. compute the maximum flow from s to C_h, and let φ be its value, and $(S, N\backslash S)$ be the corresponding minimum capacity cut;
13. **if** $\varphi < 2y_s^*$ **then** $r := r + 1$, $S_r := S$

end

end.

Notice that, at Step 10, clusters C_h containing a node j with $y_j^* = 1$ are not considered, since the degree of violation of the corresponding inequality (2.2) cannot exceed that of the inequality (2.3) written for the same set S and for $i = s$ (compare Steps 8 and 13, and observe that $\varphi \geq f_j$).

The above separation algorithm is exact when $\max\{y_v^* : v \in N\} = 1$; in this case, one of the returned subsets $S_1, \ldots, S_r$ corresponds to a most violated inequality. When $\max\{y_v^* : v \in N\} < 1$, instead, the algorithm is heuristic in nature. In any case, each returned $S \in \{S_1, \ldots, S_r\}$ can lead to a violated facet-inducing inequality, determined as follows.

procedure GSEC_BUILD;
input: a node subset S, and the point (x^*, y^*);
output: a most violated facet-defining GSEC associated with S;
begin

1. $S := S\backslash\{i : y_i^* = 0\}$;
   for each i such that $y_i^* = 0$ **do**
   let $j \in C_{h(i)}$ be the node with $y_j^* \neq 0$ closest to i (with respect to the original costs c_e), and set $S := S \cup \{i\}$ in case $j \in S$;
2. **if** there exist $C_h \subseteq S$ and $C_k \subseteq N\backslash S$ **then return** the inequality (2.1);
3. **if** there exists $C_h \subseteq S$ **then**
   return the inequality (2.2) with $i := \arg\max\{y_v^* : v \in N\backslash S\}$;
4. **if** there exists $C_h \subseteq N\backslash S$ **then**
   return the inequality (2.2) with $i := \arg\max\{y_v^* : v \in S\}$;
5. **return** the inequality (2.3) with $i := \arg\max\{y_v^* : v \in S\}$, and $j := \arg\max\{y_v^* : v \in N\backslash S\}$

end.

At Step 1, set S is modified in an attempt to have complete clusters inside S and outside S.

For the E-GTSP instances, in order to obtain stronger inequalities, Steps 2 to 5 can be replaced by:

2a. **if** no $C_h \subseteq S$ exists **then** $S := S \cup C_{h(i)}$, where $i := \arg\max\{y_v^* : v \in S\}$;

3a. **if** no $C_h \subseteq N\backslash S$ exists **then** $S := S\backslash C_{h(j)}$, where $j := \arg\max\{y_v^* : v \in N\backslash S\}$;
4a. **return** the inequality (2.1)

The above procedure runs in $O(n)$ time.

3.2. A Heuristic Separation Algorithm for Generalized Subtour Elimination Constraints

The separation algorithm GSEC_SEP of the previous subsection can be excessively time consuming. We now describe two faster heuristic procedures.

The first procedure, GSEC_H1, considers the subset of the inequalities (2.1):

$$\sum_{e\in\delta(S)} x_e \geq 2 \quad \text{if } \mu(S) \neq 0,\ \mu(N\backslash S) \neq 0,$$

with S containing one of the smallest cluster C_l. For each $h \in \{1, \ldots, m\}\backslash\{l\}$, the procedure computes a most violated inequality (2.1) with $C_l \subseteq S$ and $C_h \subseteq N\backslash S$ by finding the maximum flow from C_l to C_h. This procedure has $O(mn^3)$ time complexity, and typically runs much faster than GSEC_SEP.

Both GSEC_SEP and GSEC_H1 produce a list of violated inequalities chosen on the basis of their individual degree of violation, rather than on their combined effect. In order to speed up the convergence of the cutting plane phase, instead, for each round of separation it is advisable to produce a family of violated inequalities "spanning" the overall graph. To support this point, consider the simplest problem involving subtour elimination constraints, namely the *Shortest Spanning Tree* (SST) problem. It is known from matroid theory that the node subsets whose associated subtour elimination constraints are active at the optimum, define a nested family covering all the nodes of the graph. Therefore, a cutting plane SST algorithm that adds violated cuts chosen only on the basis of their individual degree of violation, is likely to require a high number of iterations before producing the optimal family of cuts. In this view, the *shrinking* technique used in Padberg and Rinaldi (1987, 1990) for TSP, besides reducing the computational effort spent in each separation, has the advantage of quickly producing a nested family of constraints spanning the graph.

We next describe a heuristic separation algorithm for GSECs, based on the previous considerations. In order to illustrate the basic idea underlying the algorithm, let us restrict our attention to the standard TSP. Given the fractional point x^*, we look for a family of violated subtour elimination constraints. To this end, we consider the polytope

$$Q^{\text{SEC}} := \{x \geq 0 : \sum_{e\in E(S)} x_e \leq |S| - 1, \quad \text{for } S \subseteq N,\ |S| \geq 2\},$$

whose vertices are the incidence vectors of the forests spanning the graph. We find a vertex of Q^{SEC} "close" to x^*, say $\tilde{x}$, and check the violation of (some of) the subtour elimination constraints defining facets of Q^{SEC} passing through $\tilde{x}$. To be more specific, we find $\tilde{x}$ by solving the problem $\max\{x^*x : x \in Q^{\text{SEC}}\}$, i.e., we find a maximum weight spanning tree of G with edge weights $x_e^* \geq 0$, $e \in E$. We use the classical greedy algorithm of Kruskal (1956) and check the violation of the $n - 1$ SECs associated with the subsets $S_i \subset N$, $i = 1, \ldots, n - 1$, corresponding to the connected components iteratively found. From matroid theory (see, e.g., Nemhauser and Wolsey 1988, p. 669), the SECs associated with these subsets S_i are the only ones needed to prove the optimality of $\tilde{x}$ (since all other SECs can be relaxed without affecting the optimality of $\tilde{x}$), hence they are likely to be violated by x^*. Notice that (some of) the S_i sets found by the above sketched procedure could equivalently be found by detecting the connected components of the subgraphs induced by $E(\vartheta) := \{e \in E : x_e^* \geq \vartheta\}$ for all possible threshold values $\vartheta \in \{x_e^* > 0 : e \in E\}$. In this view, our heuristic is an improved version of the one used in Grötschel and Holland (1991) that checks the connected components of the subgraph $G' = (N, E(\vartheta))$ for $\vartheta = \min\{x_e^* > 0 : e \in E\}$.

The above scheme can easily be adapted to deal with generalized SECs, as outlined in the following PASCAL-like procedure.

procedure GSEC_H2;
input: the point $(x^*, y^*) \in [0, 1]^{E\cup N}$;
output: a sequence $S_1, \ldots, S_r$ of nested node subsets corresponding to violated GSECs;
begin

1. sort the edges so that $x_{e_1}^* \geq x_{e_2}^* \geq \ldots \geq x_{e_t}^* > 0$, where $t := |\{e \in E : x_e^* > 0\}|$;
2. $r := 0$; $\tilde{T} := \emptyset$;
3. **for each** $i \in N$ **do** $V_i := \{i\}$; $\mu := n$;
4. **comment:** $\tilde{T}$ is the set of the edges in the current forest, whereas $V_1, \ldots, V_\mu$ are the connected components of $\tilde{G} := (N, \tilde{T})$;
5. **for** $j := 1$ **to** t **do**
6. **if** e_j connects two distinct components V_a and V_b (say) **then**
   begin
7. $\tilde{T} := \tilde{T} \cup \{e_j\}$;
8. **if** the most violated GSEC associated with $S := V_a \cup V_b$ (as computed by procedure GSEC_BUILD of Section 3.1) is violated by (x^*, y^*) **then** $r := r + 1$, $S_r := S$;
9. replace V_a and V_b with $V_a \cup V_b$ and decrease μ
   end

end.

The time complexity of GSEC_H2 is $O(t \log t + n^2)$, i.e., $O(n^2 \log n)$ in the worst case. The final forest $\tilde{G} := (N, \tilde{T})$ computed by the procedure is heuristically used to produce additional violated GSECs, associated with the connected components of the subgraphs induced by $\tilde{T}\backslash\{e\}$ for $e \in \tilde{T}$.

A different heuristic approach to detect violated GSECs could be obtained by changing the rule used in GSEC_H2 to select the two components V_a and V_b to be merged. Namely, V_a and V_b could be chosen, in a greedy way, as the two components whose union $S := V_a \cup V_b$ produces the greatest amount of violation for the corresponding GSEC. By using parametric techniques, the modified algorithm can be implemented to run in $O(n^3)$ time. According to our computational experience, however, this approach is outperformed by GSEC_H2.

3.3. Heuristic Separation Algorithms for Generalized Comb Inequalities

Two simple heuristic separation procedures for generalized comb inequalities are next described.

We first consider the generalized 2-matching constraints. Using a construction similar to that proposed by Padberg and Rao (1982) for the b-matching problem, one can transform the separation problem for generalized 2-matching inequalities into a minimum capacity odd cut problem; hence this separation problem is exactly solvable in polynomial time. This task is however rather time consuming, hence for our branch-and-cut code we used the following simple heuristic, derived from similar procedures proposed for TSP (see, e.g., Padberg and Grötschel 1985). Given the fractional point (x^*, y^*), we define the subgraph $\tilde{G} = (\tilde{N}, \tilde{E})$ induced by $\tilde{E} := \{e \in E : 0 < x_e^* < 1\}$. We then consider, in turn, each connected component H of $\tilde{G}$ as the handle of a possibly violated generalized 2-matching inequality, whose two-node teeth correspond to the edges $e \in \delta(H)$ with $x_e^* = 1$ (if the number of these edges is even, the inequality is clearly rejected). The procedure takes $O(n + |\tilde{E}|)$ time, if properly implemented.

Our second separation procedure consists of applying the above described heuristic for generalized 2-matching inequalities after having shrunk each cluster into a single supernode, in a vein similar to that described in Padberg and Rinaldi (1990).

4. HEURISTIC ALGORITHMS

A number of known tour construction and tour improvement heuristic algorithms for TSP (see, e.g., Golden and Stewart 1985) can be adapted to both GTSP and E-GTSP. We next concentrate on heuristics producing feasible E-GTSP (and hence GTSP) solutions.

As to tour construction procedures, we describe a possible adaptation of the well-known *farthest insertion* TSP procedure; *nearest insertion* and *cheapest insertion* procedures can be adapted in a similar way. For each pair of clusters C_h and C_k, let the corresponding *distance* d_{hk} be defined as $d_{hk} := \min\{c_{ij} : i \in C_h, j \in C_k\}$. We start by choosing the two clusters, say C_a and C_b, that are farthest from each other, and define a partial tour T between the two closest nodes $i \in C_a$ and $j \in C_b$. At each iteration, T is enlarged by first determining the uncovered cluster C_h farthest from the clusters currently visited by T, and then

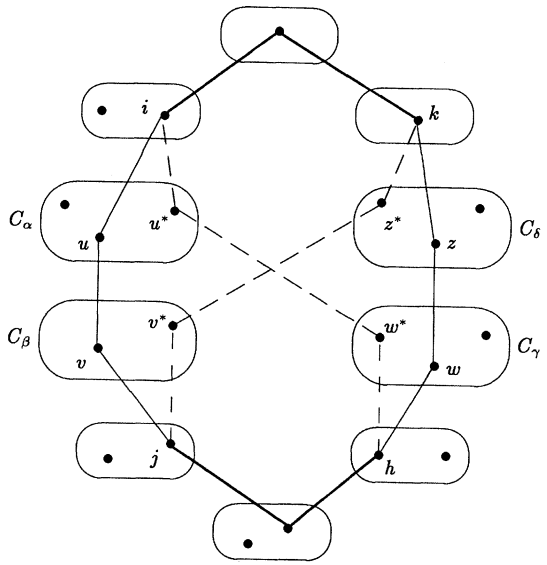

Figure 3. The generalized 2-opt exchange.

by inserting a node v of C_h between two consecutive nodes i and j of T so as to minimize $c_{iv} + c_{vj} - c_{ij}$. The procedure stops when T covers all the clusters. As in the TSP case, the procedure is likely to produce better solutions when the costs satisfy the triangle inequality.

We next describe two tour improvement procedures.

The first procedure, RP1, is based on two-opt and three-opt exchanges. Let T be the current E-GTSP solution, visiting exactly one node for each cluster, and let $S \subseteq N$ be the set of the visited nodes. Clearly any near-optimal TSP solution on the subgraph induced by S, can give an improved GTSP solution. Such a TSP solution can be found heuristically through classical two-opt or three-opt exchanges over T. This approach never changes the set S of the visited nodes. In order to remove this restriction, we propose the following generalized two-opt scheme. Let $(\ldots, C_\alpha, C_\beta, \ldots, C_\gamma, C_\delta, \ldots)$ be the cluster sequence corresponding to the current tour T. Refer to Figure 3, where T is in continuous line. All the edges of T not incident with the nodes in $C_\alpha \cup C_\beta \cup C_\delta \cup C_\gamma$ (drawn in heavy line in the figure) are fixed. We try to exchange the current cluster sequence into $(\ldots, C_\alpha, C_\gamma, \ldots, C_\beta, C_\delta, \ldots)$. To this end we determine the two node pairs (u^*, w^*) and (v^*, z^*) such that

$$c_{iu^*} + c_{u^*w^*} + c_{w^*h} = \min\{c_{ia} + c_{ab} + c_{bh} : a \in C_\alpha, b \in C_\gamma\},$$

$$c_{jv^*} + c_{v^*z^*} + c_{z^*k} = \min\{c_{ja} + c_{ab} + c_{bk} : a \in C_\beta, b \in C_\delta\},$$

where nodes i, j, h and k (see Figure 3) are the nodes visited by T belonging to the clusters preceding C_α, following C_β, preceding C_γ, and following C_δ, respectively.

This computation requires $|C_\alpha|\,|C_\gamma| + |C_\beta|\,|C_\delta|$ comparisons. On the whole, trying all the possible pairs (C_α, C_β) and (C_γ, C_δ) leads to an $O(n^2)$ time complexity, since each edge of G needs to be considered only twice.

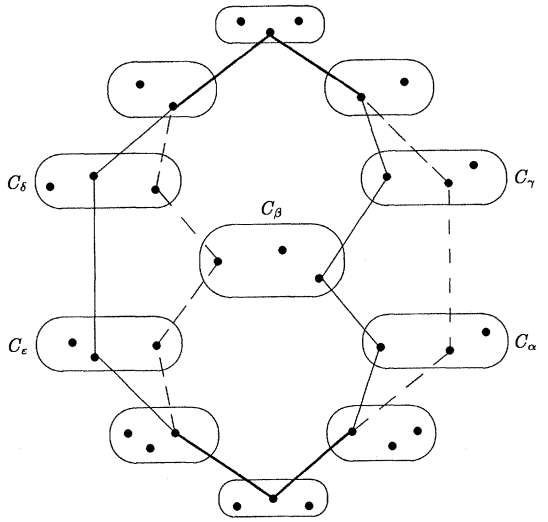

Figure 4. Changing the position of cluster C_β.

Moreover, RP1 considers the three-opt exchange of Figure 4, in which we try modifying the cluster sequence $(\ldots, C_\alpha, C_\beta, C_\gamma, \ldots, C_\delta, C_\epsilon, \ldots)$ into $(\ldots, C_\alpha, C_\gamma, \ldots, C_\delta, C_\beta, C_\epsilon, \ldots)$.

We next propose a second refinement procedure, called RP2 in the sequel, that proved to be rather effective in our computational study. Let T be the current E-GTSP solution and $(C_{h_1}, \ldots, C_{h_m})$ be the sequence in which T goes through the clusters. Our refinement consists of finding the best feasible tour, T^*, visiting the clusters according to the given sequence. This can be done, in polynomial time, by solving $|C_{h_1}|$ shortest path problems, as described below.

We construct a layered network, LN, having $m + 1$ layers corresponding to clusters $C_{h_1}, \ldots, C_{h_m}, C_{h_1}$; see Figure 5. LN contains all the nodes of G, plus an extra node j' for each $j \in C_{h_1}$. There is an arc (i, j) for each $i \in C_{h_t}$ and $j \in C_{h_{t+1}}$ $(t = 1, \ldots, m - 1)$, having cost c_{ij}. Moreover, there is an arc (i, j') for each $i \in C_{h_m}$ and $j \in C_{h_1}$, having cost c_{ij} (these arcs connect the last two layers of the network). For a given $w \in C_{h_1}$, any path in LN from w to w' visits exactly one node for each layer (cluster), hence it gives a feasible E-GTSP tour. Conversely, every E-GTSP tour visiting the clusters according to sequence $(C_{h_1}, \ldots, C_{h_m})$ corresponds to a path in LN from a certain $w \in C_{h_1}$ to w'. It then follows that the best E-GTSP tour T^* visiting

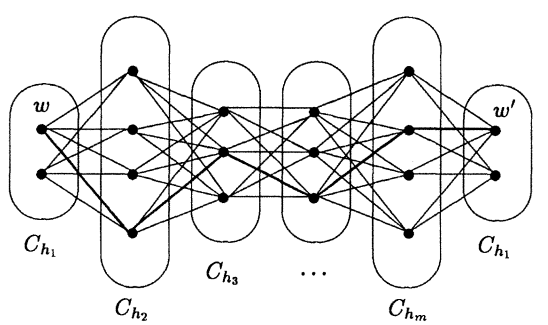

Figure 5. The layered network LN.

the clusters in the given sequence, can be found by determining the shortest path from each $w \in C_{h_1}$ to the corresponding w'. The overall time complexity is then $|C_{h_1}|O(n^2)$, i.e., $O(n^3)$ in the worst case. In practice, the time typically spent is significantly reduced by choosing C_{h_1} as the cluster with minimum cardinality, and using a shortest-path algorithm specialized for acyclic graphs.

Notice that the above refinement procedure leads to an $O((m - 1)!n^3)$-time exact algorithm for E-GTSP, obtained by trying all the $(m - 1)!$ possible cluster sequences. Therefore E-GTSP is polynomially solvable for fixed m (independently of n).

5. THE ENUMERATIVE ALGORITHM

In this section we propose an enumerative algorithm for the exact solution of the problem. Since all the instances we consider in our computational study have triangular costs, we give a rather detailed description of the implementation of the algorithm for E-GTSP. The algorithm can easily be adapted to GTSP. We assume that all costs c_e are integer.

The algorithm follows a branch-and-bound scheme, in which lower bounds are computed by solving an LP relaxation of the problem. The relaxation is iteratively tightened by adding valid inequalities to the current LP, according to the so-called *cutting plane* approach. The overall method is commonly known as a *branch-and-cut* algorithm; we refer Padberg and Rinaldi (1991) and Jünger et al. (1995) for a thorough description of the technique. We next describe some important implementation issues, including the best parameter setting resulting from our computational experience.

5.1. Lower Bound Computation

At each node of the decision tree, the lower bound is computed by solving the LP problem defined by (1.1), (1.2), (1.7), the bound constraints on the variables, the constraints derived from branching, plus a subset of GSECs and generalized comb inequalities. This subset initially coincides with that of the parent node (for the root node an "ad hoc" initialization procedure, based on Lagrangian optimization, will be described later). Notice that the y variables are not projected away through Equations (1.2), as this would result in a much denser LP coefficient matrix. We start by retrieving the optimal LP basis of the parent node. (In order to save memory, this information is written in a scratch file.) We then iteratively solve the LP, and add to it some inequalities that are violated by the current LP optimal solution. All the violated constraints found (except fan inequalities) are permanently stored in compact form in a global data structure called the *pool*. Whenever an inequality introduced in the current branch-node is slack for five (say) consecutive LP solutions, we remove it from the LP (but not from the pool). Moreover, whenever the LP-solver takes more than five (say) minutes

to solve the current LP we remove all the slack inequalities introduced in the previous nodes.

Identification of violated inequalities. The violated inequalities to be added to the current LP are identified by means of the following separation procedure.

procedure SEPARATION;
input: the LP optimal solution (x^*, y^*);
output: a set $\mathcal{T}$ of violated inequalities to be added to the LP;
begin

1. evaluate the degree of violation of the inequalities in the pool (skip those present in the LP);
2. evaluate the degree of violation of all fan inequalities;
3. let $\mathcal{T}$ be the set of inequalities with degree of violation greater than 0.1;
   if $|\mathcal{T}| \geq 1$ **then return**;
4. apply heuristic GSEC_H2 of Section 3.2, and evaluate the degree of violation of the GSECs found;
5. let $\mathcal{T}$ be the set of inequalities with degree of violation greater than 0.1;
   if $|\mathcal{T}| \geq 1$ **then return**;
6. apply heuristic GSEC_H1 of Section 3.2, and evaluate the degree of violation of the GSECs found;
7. **for** $\epsilon \in \{0.1, 0.01, 0.001\}$, by decreasing values, **do begin**
8. let $\mathcal{T}$ be the set of the (distinct) inequalities found at steps 1 and 6, and having degree of violation greater than ϵ;
   if $|\mathcal{T}| \geq 1$ **then return**
   end;
9. apply the separation algorithm GSEC_SEP of Section 3.1, and the heuristic separation algorithms for the generalized comb inequalities described in Section 3.3;
10. let $\mathcal{T}$ contain the GSECs and the generalized comb inequalities found at step 9, and having a degree of violation greater than 0.001

end.

The time spent at Step 1 depends on the way the inequalities are stored, in compact form, in the pool. In our implementation, for each such inequality we store its *type*, its right-hand side, a pointer to its row in the current LP (= 0 if not present), plus additional information related to the inequality type. For *type* = "fan inequality" (see (2.12)), we store l and w. For *type* = "GSEC" (see (2.4)–(2.6)), we store a 0-1 n-dimensional vector to represent the node subset S, plus i and j when needed. In order to have a unique representation of a GSEC, we impose $|S| \leq |N\backslash S|$, and $1 \in S$ in case $|S| = |N\backslash S|$. For *type* = "generalized comb inequality" (see (2.9)), we store a 0-1 n-dimensional vector to represent the handle, plus an integer n-dimensional vector giving for each node the index of the tooth containing it (= 0 if no such tooth exists). Moreover, we store for each tooth j the two nodes a_j and b_j. Using this data structure, one can efficiently evaluate the degree of violation of an inequality in the pool by scanning each nonzero entry of (x^*, y^*) and retrieving (in constant time) the associated coefficient.

At Step 2, the degree of violation of the fan inequalities is evaluated as follows. For $h = 1, \ldots, m$ and $j \in N\backslash C_h$, we initialize $\text{degree}(h, j) := -y^*_j$. Then for each $[i, j] \in E^* := \{e \in E : x^*_e > 0\}$, we update

$$\text{degree}(h(i), j) := \text{degree}(h(i), j) + x^*_{[i,j]},$$

$$\text{degree}(h(j), i) := \text{degree}(h(j), i) + x^*_{[i,j]}.$$

In this way Step 2 requires $O(mn + |E^*|)$ time.

As to Step 6, its execution is avoided when the heuristic procedure GSEC_H2 succeeds in finding significantly violated GSECs. However, we obtained improved performances by forcing the execution of Step 6 at each tenth call of SEPARATION within the same decision node. This allows early detection of violated GSECs that are seldom found by GSEC_H2.

Procedure SEPARATION requires the check for inequality distinctness. This is done by comparing their associated representations in compact form, and takes $O(n)$ time for each inequality pair. A considerable speed-up is achieved by associating with each inequality a four-byte "hash" value, i.e., a value obtained through a many-to-one transformation from all possible compact forms to four-byte integers. This ensures that different hash values correspond to different inequalities.

Lagrangian relaxation. At the beginning of the root node computation, we apply Lagrangian optimization with the aim of determining a good subset of constraints for the initial LP, as well as a near-optimal heuristic solution. We consider the following (simplified) model for E-GTSP, in which the y variables have been projected away through (1.2).

$$\min \sum_{e \in E} c_e x_e, \tag{5.1}$$

subject to

$$\sum_{e \in E} x_e = m, \tag{5.2}$$

$$\sum_{e \in \delta(C_h)} x_e = 2 \quad \text{for } h = 1, \ldots, m, \tag{5.3}$$

$$\sum_{e \in \delta(C_h) \cap \delta(v)} x_e \leq \sum_{e \in \delta(v) \backslash \delta(C_h)} x_e$$

$$\text{for } h = 1, \ldots, m;\ v \in N\backslash C_h, \tag{5.4}$$

$$\sum_{e \in E(S)} x_e \leq r - 1$$

$$\text{for } S = \cup_{i=1}^{r} C_{l_i},\ C_1 \subset N\backslash S,\ 2 \leq r \leq m - 2, \tag{5.5}$$

$$x_e \in \{0, 1\} \quad \text{for } e \in E. \tag{5.6}$$

Equation (5.2) is redundant in this formulation. Inequalities (5.4) and (5.5) are fan and Basic GSECs, respectively (notice however that not all GSECs are included in the model).

We dualize, in a Lagrangian fashion, the fan inequalities (5.4), plus the degree constraints (5.3) for $h \neq 1$. The Lagrangian relaxed problem calls for $m - 2$ edges (each connecting two different clusters) in $E\backslash\delta(C_1)$ inducing no intracluster cycles, plus two edges incident with C_1. Therefore, it can be efficiently solved as follows:

(i) *shrink* G with respect to the m clusters, i.e., replace each cluster C_h with a single *super-node* h, and define for each super-node pair h, k a *super-edge* $[h, k]$ with cost

$$\bar{c}_{hk} := \min\{c'_{ij} : i \in C_h, j \in C_k\}, \tag{5.7}$$

where c'_{ij} is the Lagrangian cost of edge $[i, j] \in E$;

(ii) compute the min-cost *1-tree* (Held and Karp 1971) on the shrunken graph;

(iii) obtain an optimal solution to the Lagrangian relaxed problem by replacing each super-edge $[h, k]$ in the 1-tree found at Step ii, with its corresponding edge $[i, j] \in E$ (the one producing the minimum in (5.7)).

The computation of near-optimal Lagrangian multipliers is done through classical subgradient optimization. In our implementation, we iteratively update the multipliers through two nested loops. In the external loop we update the multipliers for the fan inequalities (5.4). With these multipliers fixed, we enter the internal loop in which the multipliers for the degree constraints (5.3) are adjusted so as to hopefully produce a tour in the shrunken graph. This is in the spirit of the successful Help-Karp approach to the standard TSP. At the end of the internal loop, if the final 1-tree on the shrunken graph is a tour we determine a heuristic E-GTSP solution through the refinement procedure RP2 of Section 4, where the cluster sequence $C_{h_1}, \ldots, C_{h_m}$ is the one induced by the tour in the shrunken graph. This approach computationally proved to be quite effective in determining near optimal solutions at the very beginning of the root node computation. In our implementation we allow for, at most, 1000 and 50 subgradient iterations in the external and internal loops, respectively.

Root node initialization. Let λ_h^* and μ_{hv}^* be the best Lagrangian multipliers for constraints (5.3) and (5.4), respectively. The initial LP at the root node contains constraints (1.2), (2.10), the bound restrictions on the variables, plus the subset of the fan inequalities (2.12) with $\mu_{hv}^* > 0$. Moreover, the LP contains the Basic GSECs (5.5) that were active in the computation of the 1-tree on the shrunken graph with respect to (λ^*, μ^*). To be more specific, we include in the LP all the constraints (5.5) whose subset S corresponds to a connected component detected by the Kruskal (1956) algorithm for determining the best 1-tree on the shrunken graph. With this initialization, the optimal value of the first LP relaxation is guaranteed to be at least as good as the one provided by the Lagrangian relaxation.

5.2. Variable Pricing

The LP problems to be solved would, in principle, involve a huge number of x-variables. In practice, only a small subset of these variables, say $\{x_e : e \in J\}$, needs to be considered explicitly. J is heuristically initialized, at the root node, to contain the edges having zero reduced cost at the end of the Lagrangian optimization. These reduced costs are obtained, in $O(n^2)$ time, from the reduced costs of the 1-tree computed for the best Lagrangian multipliers (λ^*, μ^*), as described in Carpaneto et al. (1989). Moreover, we select the five smallest-cost edges incident with each node.

After the solution of each LP problem, we *price-out* the variables x_e for $e \in E\backslash J$, i.e., we compute the LP reduced costs $\bar{c}_e := c_e - \sum_{i=1}^{q} \alpha_{ie} u_i^*$, where q is the number of constraints in the current LP, u_i^* is the optimal dual variable for the i-th constraint, and α_{ie} is the coefficient of x_e in the i-th constraint. (Reduced costs are also used for variable fixing.) Notice that for any given pair (i, e), the coefficient α_{ie} can be computed in constant time through the information stored in the pool.

After pricing, set J is updated by adding the edges $e \in E^- := \{f \in E : \bar{c}_f < 0\}$ (if $|E^-| > 100$, only the 100 variables with smallest reduced costs are added). Whenever $E^- \neq \emptyset$, we again solve the LP problem with the same set of constraints and with the enlarged set of variables. In this way, the separation phase is only entered when all variables price-out correctly.

In order to reduce the number of LP variables, when $E^- = \emptyset$ we remove from J all the edges e with $\lceil LB + 4\bar{c}_e \rceil \geq UB$, where LB is the value of the current LP relaxation, and UB the value of the best known solution.

The standard approach for variable pricing considers, in turn, each variable x_e with $e \in E\backslash J$ and computes the associated reduced cost $\bar{c}_e$. This "column-by-column" approach can be rather time consuming, since for each x_e one has to consider explicitly the coefficient of x_e in each constraint of the LP (even when this coefficient is zero). Some speed-up can be achieved by heuristically determining a small subset $\tilde{E} \subset E$ containing the edges that will more likely be in the optimal solution (*core edge set*), and pricing only the variables x_e for $e \in \tilde{E}\backslash J$. The remaining variables are priced out only at the end of the branching node, or after every k (say) LP relaxations. For details see, e.g., Jünger et al. (1995).

We have implemented an alternative "row-by-row" pricing scheme that in some cases, as those considered in this paper, is more efficient than the standard one. We initialize $\bar{c}_e := c_e$ for all $e \in E$. Then, for each LP row $i \in \{1, \ldots, q\}$ with $u_i^* \neq 0$ we determine the edge subset $J_i := \{e \in E : \alpha_{ie} \neq 0\}$. By using the information in the pool, J_i is easily defined in $O(n + |J_i|)$ time. We then update the reduced costs $\bar{c}_e$ for $e \in J_i$ by setting $\bar{c}_e := \bar{c}_e - \alpha_{ie} u_i^*$. In this way the whole set of variables is priced out in $O(|E| + nq + \sum_{i=1}^{q} |J_i|)$ time, whereas the standard approach takes $O(|E'|q)$ time to price a given edge set E'. Therefore the new approach performs better when pricing a large set of variables for sparse constraints, as in the E-GTSP case. Notice however that the new pricing scheme requires

$O(n^2)$ bytes of memory to store the real vector $\bar{c}_e$, hence for very large scale instances it may become impractical.

In our cutting plane algorithm, we apply the row-by-row pricing procedure after each LP solution.

5.3. Upper Bound Computation

At the root node we apply the farthest insertion, nearest insertion and cheapest insertion procedures, each followed by the tour improvement procedures, as described in Section 4. Moreover, as explained in Section 5.1, for each tour among clusters found during the Lagrangian relaxation we get a new feasible solution through procedure RP2. All solutions found are refined through the tour improvement procedures of Section 4.

At any branching node we exploit the information associated with the fractional point available after each LP solution, in the attempt of improving the current *UB*. To this end, let (x^*, y^*) be the optimal LP solution. We initialize a heuristic solution by taking all the edges e with $x_e^* = 1$, and then complete it through a nearest insertion scheme. Again, the resulting solution is refined through the tour improvement procedures.

5.4. Branching

We considered two possibilities for branching: branching on variables and branching on cuts. Let (x^*, y^*) be the fractional LP solution at the end of the current node.

Branching on variables (the standard approach for branch-and-cut) consists of selecting a fractional x_e^*, and generating two descendent nodes by fixing the value of x_e to either 0 or 1. In our implementation we choose x_e^* as close as possible to 0.5 (ties are broken by choosing the edge e having maximum cost c_e).

Branching on cuts consists of choosing a subset $S \subset N$ such that $\Sigma_{e\in\delta(S)} x_e^*$ is not an even integer, and imposing the disjunction

$$\left(\sum_{e\in\delta(S)} x_e \leqslant 2k\right) \quad \text{or} \quad \left(\sum_{e\in\delta(S)} x_e \geqslant 2k+2\right),$$

where $k := \lfloor \Sigma_{e\in\delta(S)} x_e^*/2 \rfloor$. In our implementation S is determined as follows.

Let $v_1, \ldots, v_m$ be the vertex sequence corresponding to the current best E-GTSP solution, say $(\bar{x}, \bar{y})$, where the subscripts of v are intended to be taken as modulo m. We restrict ourselves to sets S obtained as the union of consecutive clusters in the sequence (i.e., $S := C_{h(v_a)} \cup C_{h(v_{a+1})} \cup \ldots \cup C_{h(v_b)}$ for some pair (a, b)), and such that $2 + \epsilon \leqslant \Sigma_{e\in\delta(S)} x_e^* \leqslant 4 - \epsilon$ for $\epsilon = 0.2$. Among these sets S, if any, we choose the one maximizing

$$L(S) := \min\{d(v_i, v_j) : i = a, a+1, \ldots, b-1 \text{ and } j = b+1, b+2, \ldots, a-2\},$$

where $d(v_i, v_j) := c_{v_iv_j} + c_{v_{i+1}v_{j+1}} - c_{v_iv_{i+1}} - c_{v_jv_{j+1}}$ is the additional cost corresponding to the new solution obtained from $(\bar{x}, \bar{y})$ by exchanging the edge pair $([v_i, v_{i+1}], [v_j, v_{j+1}])$ with $([v_i, v_j], [v_{i+1}, v_{j+1}])$. $L(S)$ is an estimate on the increase of the cost of the optimal solution (and then of the LP lower bound) when imposing $\Sigma_{e\in\delta(S)} x_e \geqslant 4$. Choosing $L(S)$ as large as possible then hopefully produces a significant increase in the lower bound of one of the two children of the current node.

In our computational study we used the "branching on cuts" strategy (that turned out to be superior), and resort to the "branching on variables" approach when our procedure does not find a suitable cut $\delta(S)$. Since the heuristic solutions computed at the root node are quite good, we have decided to implement a depth-first tree search scheme (although, in general, this is not the best strategy one can choose).

6. COMPUTATIONAL RESULTS

The enumerative algorithm described in Section 5 has been implemented in C, and it ran on a Hewlett Packard 9000 Series 720 Apollo and on a DEC station 5000/240. As to the LP solver we used the package CPLEX 2.1 implementing both primal and dual Simplex algorithms.

We considered two small *geographical* test problems, 15SPAIN47 and 27EUROPE47, corresponding to 47 cities and 15 clusters (regions) of Spain, and to 47 cities and 27 clusters (countries) of Europe, respectively (the corresponding data are available, on request, from the authors). Larger instances were obtained by taking all the TSP test problems from the Reinelt TSPLIB library (1991) having $48 \leqslant n \leqslant 442$. The node clustering has been done so as to simulate geographical regions, according to the following procedure. For a given instance, we fix the number of clusters to $m := \lceil n/5 \rceil$. Then we determine m *centers* by considering m nodes as far as possible one from each other. The clusters are finally obtained by assigning each node to its nearest center. The resulting clustering procedure is as follows.

procedure CLUSTERING;
input: n, m, c_e for $e \in E$ (with $c_{[v,v]} := -\infty$ for all v);
output: $h(v)$ for $v \in N$;
comment let $far(S) := \arg\max\{\min\{c_{[v,w]} : w \in S\} : v \in N\backslash S\}$ be the furthest node from a given $S \subseteq N$;
begin

1. $center_1 := far(\{1\})$;
2. **for** $i := 2$ **to** m **do** $center_i := far(\{center_1, \ldots, center_{i-1}\})$;
3. **for** $v := 1$ **to** n **do** $h(v) := \arg\min\{c_{[v,center_i]} : i = 1, \ldots, m\}$

end.

(In arg min{·} and arg max{·}, ties are broken by choosing the smallest argument.)

In addition, for the geographical problems GR96 (Africa), GR137 (America), GR202 (Europe), GR229 (Australia-Asia) and GR431 (Australia-Asia-Europe), described in Grötschel and Holland (1991), we have also considered the natural clustering, in which clusters correspond to

Table I
Root Node Statistics

Problem Name	Lagr-LB	Lagr-UB	Lagr-t	Basic-LB	Root-LB	Root-UB	Root-t
15SPAIN47	91.50	100.00	0.4	91.76	100.00	100.00	1.5
27EUROPE47	93.70	100.01	2.6	93.71	100.00	100.00	3.8
50GR96	94.94	100.37	9.4	95.01	99.81	100.37	16.2
35GR137	84.82	100.00	8.0	86.81	99.44	100.00	31.9
31GR202	85.19	100.21	13.8	85.35	99.85	100.05	464.8
61GR229	85.26	102.39	24.8	85.42	99.81	100.87	156.8
92GR431	86.36	103.55	83.8	86.49	99.84	100.05	2256.5
10ATT48	88.00	100.00	0.9	88.60	100.00	100.00	2.1
10GR48	89.75	100.00	0.5	90.53	100.00	100.00	1.9
10HK48	84.40	100.00	1.1	84.58	100.00	100.00	3.8
11EIL51	87.35	100.00	0.4	87.59	100.00	100.00	2.9
12BRAZIL58	93.11	100.00	1.4	93.13	100.00	100.00	3.0
14ST70	80.38	100.00	1.2	80.60	100.00	100.00	7.3
16EIL76	79.90	100.00	1.4	79.92	100.00	100.00	9.4
16PR76	84.26	100.00	0.6	87.05	100.00	100.00	12.9
20GR96	90.56	100.00	4.3	90.85	99.95	100.00	18.4
20RAT99	78.67	100.00	3.1	78.60	100.00	100.00	51.4
20KROA100	85.90	100.00	2.4	86.24	100.00	100.00	18.3
20KROB100	90.52	100.00	3.1	91.09	100.00	100.00	22.1
20KROC100	86.90	100.00	2.2	87.61	100.00	100.00	14.3
20KROD100	84.09	100.00	2.5	84.28	100.00	100.00	14.2
20KROE100	83.81	100.00	0.9	87.35	100.00	100.00	12.9
20RD100	87.29	100.08	2.6	87.56	100.00	100.00	16.5
21EIL101	81.52	100.00	1.7	81.54	100.00	100.00	25.5
21LIN105	90.38	100.00	2.0	90.62	100.00	100.00	16.2
22PR107	99.11	100.00	2.1	99.25	100.00	100.00	7.3
24GR120	84.00	101.99	4.9	84.45	100.00	100.00	41.8
25PR124	91.29	100.00	3.7	91.40	100.00	100.00	25.7
26BIER127	97.09	100.00	11.2	97.29	100.00	100.00	23.3
28PR136	93.04	100.82	7.2	93.28	100.00	100.00	42.8
28GR137	86.53	101.02	4.6	86.64	100.00	100.00	96.7
29PR144	99.64	100.00	2.3	99.82	100.00	100.00	8.0
30KROA150	84.13	100.00	7.6	84.24	100.00	100.00	100.0
30KROB150	88.15	100.00	9.9	88.35	100.00	100.00	60.3
31PR152	94.91	100.00	9.6	95.14	98.45	100.00	51.4
32U159	86.84	100.00	10.9	86.97	99.96	100.00	139.6
39RAT195	81.50	101.87	8.2	81.71	100.00	100.00	245.5
40D198	93.85	100.48	12.0	93.90	100.00	100.00	762.5
40KROA200	82.64	100.00	15.3	82.88	99.99	100.00	183.3
40KROB200	83.29	100.05	19.1	83.47	100.00	100.00	268.0
41GR202	92.30	100.05	20.9	92.44	100.00	100.00	1021.3
45TS225	83.54	100.09	19.4	83.62	99.11	100.09	1298.4
46PR226	96.70	100.00	14.6	97.21	100.00	100.00	106.2
46GR229	89.83	100.37	49.6	89.92	99.58	100.00	995.2
53GIL262	85.29	103.75	15.8	85.44	99.80	100.89	1443.5
53PR264	91.90	100.33	24.3	91.98	100.00	100.00	336.0
60PR299	84.48	100.00	33.2	84.67	100.00	100.00	811.4
64LIN318	92.48	100.36	52.5	92.64	99.79	100.36	847.8
80RD400	85.28	103.16	59.8	85.52	99.94	102.97	5031.5
84FL417	93.61	100.13	77.2	93.67	100.00	100.00	16714.4
87GR431	93.46	101.18	408.3	93.49	99.94	100.54	26774.0
88PR439	92.26	101.42	146.6	92.39	100.00	100.00	5418.9
89PCB442	82.74	104.22	78.8	83.03	99.49	100.29	5353.9

countries. The resulting instances are 50GR96, 35GR137, 31GR202, 61GR229, and 92GR431.

Tables I, II, and III give computational results for the above test problems. Times are given in HP 9000/720 CPU seconds. For each problem Table I gives the following information for the root node:

Problem name : in the form *mXXXXn*, where *m* is the number of clusters, and *XXXXn* is the name of the problem in TSPLIB (*n* gives the number of nodes);

Lagr-LB : percentage ratio LB/(optimal solution value), where LB is the lower bound value computed through the Lagrangian relaxation of Section 5.1;

Table II
Branch-and-Cut Statistics

Problem Name	Optval	Total-t	LP-t	SEP-t	Nodes	Separ.	Cuts	Row	Col
15SPAIN47	3271	1.5	0.7	0.1	0	10	39	135	186
27EUROPE47	18246	3.8	0.7	0.1	0	7	42	162	214
50GR96	36292	18.1	4.2	1.0	2	11	84	338	412
35GR137	28709	41.5	13.0	6.2	6	59	162	293	432
31GR202	14416	504.7	366.7	46.6	2	134	865	547	780
61GR229	65508	215.6	109.3	28.5	4	94	487	627	989
92GR431	75616	3365.2	2342.8	278.2	4	231	1659	1210	1942
10ATT48	5394	2.1	0.7	0.1	0	9	19	139	259
10GR48	1834	1.9	1.0	0.1	0	10	34	142	220
10HK48	6386	3.8	1.9	0.1	0	14	69	164	265
11EIL51	174	2.9	1.8	0.1	0	13	56	175	180
12BRAZIL58	15332	3.0	1.0	0.1	0	10	23	156	297
14ST70	316	7.3	4.6	0.3	0	12	100	242	317
16EIL76	209	9.4	5.9	0.3	0	20	117	253	321
16PR76	64925	12.9	8.9	0.5	0	33	144	236	396
20GR96	29072	19.4	10.2	1.1	2	27	153	321	391
20RAT99	497	51.5	36.8	3.4	0	50	344	323	404
20KROA100	9711	18.4	11.1	0.8	0	31	123	290	388
20KROB100	10328	22.2	13.5	0.6	0	36	123	309	515
20KROC100	9554	14.4	8.7	0.4	0	19	143	310	411
20KROD100	9450	14.3	7.9	0.5	0	21	107	314	512
20KROE100	9523	13.0	8.3	0.5	0	22	128	283	397
20RD100	3650	16.6	9.1	1.1	0	26	118	315	465
21EIL101	249	25.6	16.7	1.6	0	46	197	321	411
21LIN105	8213	16.4	8.8	0.7	0	24	113	310	404
22PR107	27898	7.4	2.9	0.2	0	4	14	340	701
24GR120	2769	41.9	24.9	3.2	0	55	237	364	468
25PR124	36605	25.9	14.4	0.9	0	20	117	371	578
26BIER127	72418	23.6	7.3	0.5	0	18	109	356	462
28PR136	42570	43.0	21.1	3.8	0	50	222	416	543
28GR137	35957	96.9	65.0	6.6	0	75	354	437	617
29PR144	45886	8.2	2.5	0.1	0	2	7	382	776
30KROA150	11018	100.3	63.3	8.0	0	73	368	476	701
30KROB150	12196	60.6	33.0	5.2	0	37	284	453	641
31PR152	51576	94.8	54.0	6.9	2	84	350	424	705
32U159	22664	146.4	98.0	11.2	2	65	354	486	775
39RAT195	854	245.9	167.7	26.3	0	101	776	634	861
40D198	10557	763.1	571.8	56.9	0	146	901	582	940
40KROA200	13406	187.4	108.2	27.4	2	70	420	598	943
40KROB200	13111	268.5	182.7	23.5	0	90	623	685	962
41GR202	23239	1022.2	764.1	119.3	0	176	1331	574	852
45TS225	68340	37875.9	34071.0	2481.6	190	1418	8252	1010	1273
46PR226	64007	106.9	45.4	5.0	0	48	215	570	1141
46GR229	71641	1187.5	870.3	132.2	2	172	1129	658	1042
53GIL262	1013	6624.1	5342.7	943.4	16	322	2250	911	1374
53PR264	29549	337.0	204.6	34.4	0	73	608	779	1170
60PR299	22615	812.8	583.9	43.3	0	122	929	869	1400
64LIN318	20765	1671.9	1038.8	292.6	10	214	1194	952	1259
80RD400	6361	7021.4	4721.7	1658.3	2	317	2425	1260	2165
84FL417	9651	16719.4	12232.3	2964.2	0	627	4506	1139	3395
87GR431	101523	31544.6	24873.2	4540.0	2	454	3833	1174	2739
88PR439	60099	5422.8	3636.5	896.9	0	296	2330	1340	2386
89PCB442	21657	58770.5	43753.5	12712.1	46	894	8735	1520	2326

Lagr-UB : percentage ratio UB/(optimal solution value), where UB is the upper bound value at the end of the Lagrangian relaxation (see Section 5.3);

Lagr-t : CPU time, in seconds, for the Lagrangian relaxation;

Basic-LB : percentage ratio LB/(optimal solution value), where LB is the optimal value of the LP relaxation of the simplified model (5.1)–(5.6);

Root-LB : percentage ratio LB/(optimal solution value), where LB is the final lower bound at the root node;

Table III
Statistics on Separation Procedures

Problem Name	Cuts	Fan	GSEC_H2	GSEC_H1	Gcomb	Pool
15SPAIN47	96	52 (44)	31	0	0	0
27EUROPE47	109	49 (42)	35	0	0	0
50GR96	259	133 (127)	77	0	0	1
35GR137	296	150 (101)	63	22	8	20
31GR202	1112	325 (218)	580	129	0	49
61GR229	793	355 (247)	303	30	2	44
92GR431	2219	713 (460)	1026	146	2	242
10ATT48	88	66 (61)	14	0	0	0
10GR48	98	62 (56)	28	0	0	0
10HK48	140	78 (62)	53	0	0	1
11EIL51	131	82 (65)	39	1	0	0
12BRAZIL58	95	70 (62)	15	0	0	0
14ST70	208	115 (96)	80	1	0	0
16EIL76	220	112 (89)	94	0	0	0
16PR76	241	121 (83)	83	11	0	12
20GR96	302	157 (131)	124	1	0	2
20RAT99	492	167 (130)	277	19	0	11
20KROA100	269	162 (128)	77	5	0	7
20KROB100	284	176 (143)	74	8	0	8
20KROC100	275	150 (114)	99	0	0	8
20KROD100	251	149 (126)	76	2	0	6
20KROE100	260	139 (114)	103	0	0	0
20RD100	266	159 (130)	83	4	0	2
21EIL101	337	175 (121)	115	20	0	8
21LIN105	253	157 (121)	76	0	0	1
22PR107	211	178 (177)	13	0	0	0
24GR120	410	208 (151)	168	0	0	12
25PR124	302	188 (162)	83	6	0	2
26BIER127	272	167 (139)	76	0	0	5
28PR136	431	220 (183)	155	18	0	12
28GR137	549	237 (169)	231	26	0	29
29PR144	209	175 (175)	7	0	0	0
30KROA150	594	254 (198)	264	6	0	42
30KROB150	511	243 (199)	193	41	0	6
31PR152	574	246 (195)	236	14	5	44
32U159	599	260 (215)	232	56	0	21
39RAT195	1104	395 (291)	536	98	0	38
40D198	1189	334 (250)	544	206	0	67
40KROA200	710	339 (252)	262	46	4	21
40KROB200	947	358 (286)	440	79	0	32
41GR202	1597	335 (227)	715	439	0	69
45TS225	8590	499 (295)	789	2300	165	4785
46PR226	513	314 (254)	129	10	0	16
46GR229	1428	393 (255)	573	300	14	104
53GIL262	2676	516 (375)	567	860	27	655
53PR264	1016	479 (357)	340	107	0	39
60PR299	1358	536 (371)	579	106	0	79
64LIN318	1680	585 (424)	562	305	1	165
80RD400	3092	762 (589)	1093	759	0	400
84FL417	5102	962 (514)	2182	1624	0	378
87GR431	4354	672 (436)	1466	1471	5	655
88PR439	2979	778 (563)	1641	277	0	197
89PCB442	9427	949 (605)	1824	2767	38	3762

Root-UB : percentage ratio UB/(optimal solution value), where UB is the final upper bound at the root node;

Root-t : CPU time, in seconds, for the root node (including *Lagr-t*).

According to the table, the upper bound computed through Lagrangian relaxation is quite tight. Its coincides with the optimal value for 30 instances, i.e., in 56.6% of cases, and on average is about 0.5% over the optimum. On the other hand, the quality of the Lagrangian lower bound is rather poor, with an average gap of 11.7%. This is due both to the heuristic solution of the Lagrangian dual problem (through subgradient optimization), and to the fact that it is derived from the simplified model (5.1)–(5.6). Notice that the best theoretical lower bound for Lagrangian

Table IV
Some Computational Results with Different Clusterizations

Problem Name	$\mu = 3$			$\mu = 5$			$\mu = 7$			$\mu = 10$		
	Time	Nodes	m	Time	Nodes	m	Time	Nodes	m	Time	Nodes	m
ATT48	3.6	0	18	3.3	0	13	1.8	0	7	1.8	0	7
EIL51	1.9	0	25	1.4	0	16	1.6	0	9	1.6	0	9
ST70	5.5	0	24	3.9	0	16	3.9	0	16	5.2	0	9
EIL76	42.0	10	34	5.5	0	16	5.4	0	16	5.8	0	9
PR76	17.2	2	31	7.6	0	22	7.4	0	15	10.5	0	9
GR96	21.7	4	34	26.3	0	22	28.4	0	15	28.4	0	15
RAT99	25.6	0	36	26.4	0	25	56.5	0	16	56.5	0	16
KROA100	14.9	0	43	19.5	0	23	14.4	0	16	14.4	0	16
KROB100	39.2	4	44	11.5	0	25	20.6	0	16	20.6	0	16
KROC100	51.0	18	42	38.3	0	25	14.3	0	16	14.3	0	16
KROD100	24.1	2	42	19.8	0	24	25.0	0	16	25.0	0	16
KROE100	15.5	0	42	11.8	0	25	8.5	0	16	8.5	0	16
RD100	84.5	16	36	20.1	4	24	11.0	0	16	11.0	0	16
EIL101	139.2	16	36	18.2	0	25	20.3	0	16	20.3	0	16
LIN105	14.3	0	45	10.0	0	30	15.2	0	16	15.2	0	16
PR107	5.0	0	42	4.2	0	22	9.2	0	16	16.0	0	12
GR120	24.8	0	46	52.4	0	28	56.4	0	21	46.9	0	15
PR124	30.0	6	45	15.2	0	25	13.9	0	19	26.7	0	14
BIER127	234.6	16	50	210.7	6	26	69.4	0	19	25.7	0	14
PR136	181.4	28	60	15.1	0	34	22.3	0	20	13.6	0	16
GR137	81.7	0	46	94.7	0	28	284.5	0	23	972.0	0	14
PR144	28.8	0	48	25.3	0	30	9.9	0	21	21.8	0	16
KROA150	56.3	2	57	72.5	0	36	82.3	0	25	111.2	0	16
KROB150	40.5	0	56	100.0	2	36	117.4	0	25	79.2	0	16
PR152	68.7	4	54	455.2	42	33	67.9	2	24	35.5	0	16
U159	36.5	0	58	154.3	0	38	107.4	0	23	107.4	0	23
RAT195	84.4	2	81	1409.5	18	49	902.1	1	36	423.5	0	25
D198	539.5	0	67	591.2	0	40	1986.9	0	32	2849.9	0	25
KROA200	207.2	2	72	1696.3	30	47	512.2	0	35	339.5	0	25
KROB200	2031.9	54	76	119.1	0	48	132.6	0	36	422.2	0	25
GR202	2644.3	34	73	727.4	2	43	731.2	0	31	450.1	0	21
TS225	453.4	14	75	—	—	45	5427.0	12	35	10601.5	0	25
PR226	130.7	0	78	65.6	0	50	61.6	0	33	105.7	0	24
GR229	312.0	0	80	1895.1	8	46	2524.0	0	34	9391.9	2	23
GIL262	5674.6	104	96	10763.1	42	63	8160.7	49	49	1141.0	0	36
PR264	747.1	16	101	109.6	0	55	352.8	2	42	376.8	0	27
PR299	761.3	2	102	4629.9	12	69	5296.9	6	47	2730.6	0	35
LIN318	37117.4	220	108	16784.8	40	64	1355.9	0	49	71010.7	26	36
RD400	21764.6	118	135	87308.1	198	81	8947.7	6	64	21156.8	2	49
FL417	7687.9	0	142	10373.7	0	93	5364.4	0	61	919.2	0	43
PR439	1905.7	6	163	18876.5	14	96	5189.9	0	74	35652.6	8	48
PCB442	23226.1	86	155	39155.3	24	96	21268.5	0	64	15266.6	0	48

Problem TS225 with $\mu = 5$ required more than 100,000 CPU seconds.

relaxation equals the optimal value of the LP relaxation of model (5.1)–(5.6). The latter value was computed through a simplified version of our cutting plane algorithm, and is reported in the table (column *Basic-LB*). It can be seen that the improvement with respect to the Lagrangian bound is negligible, hence the large gap observed derives from the slackness of the simplified model.

Table II shows the performance of the overall enumerative algorithm. For each problem the table gives:

Problem name;

Optval : value of the optimal solution;

Total-t : CPU time, in seconds, for the overall execution;

LP-t : overall CPU time, in seconds, spent by the LP solver;

SEP-t : overall CPU time, in seconds, spent for separation;

Nodes : number of nodes of the branch-decision tree (= 0 if no branching is required);

Separ. : total number of calls to procedure SEPARATION;

Cuts : total number of distinct cuts produced by SEPARATION;

Row : maximum number of constraints in the LP;

Col : maximum number of nonslack variables in the LP.

The table shows that all the considered instances can be solved to optimality within an acceptable computing time. Moreover, a significant part of the total computing time is spent within the LP solver. In about 70% of the cases, no branching is needed. The results also show that natural clustering produces easier instances than those obtained through our procedure CLUSTERING.

Additional statistics on the type of the generated cuts are reported in Table III, whose columns give:

Problem name;

Cuts : total number of cuts generated, including those found by the Lagrangian initialization (Section 5.1) and those recovered from the pool;

Fan : total number of fan inequalities generated (in parentheses those found by the Lagrangian initialization);

GSEC_H2 : total number of GSECs found by the heuristic procedure GSEC_H2 of Section 3.2;

GSEC_H1 : total number of GSECs found by the heuristic procedure GSEC_H1 of Section 3.2;

Gcomb : total number of generalized comb inequalities generated;

Pool : total number of violated cuts recovered from the pool.

As to procedure GSEC_SEP, it never found violated cuts, with the only exception of instance 45TS225 for which nine cuts were detected. This proves the effectiveness of our heuristic separations for GSECs. The inequalities which are most frequently recovered from the pool are the GSECs (2.1).

In order to evaluate the effect of different clusterizations of the nodes, we have also considered a second clustering procedure to simulate geographical regions. Given a TSP instance, let (x_i, y_i) be the geographical coordinates of the ith node ($i = 1, \ldots, n$). This information is provided in TSPLIB for all the instances considered in Table IV. Let $x$min, $x$max, $y$min, and $y$max be the minimum and maximum x- and y-coordinates, respectively. We considered the rectangle whose vertices have coordinates ($x$min, $y$min), ($x$min, $y$max), ($x$max, $y$max), and ($x$max, $y$min), and subdivided it so as to obtain an $NG \times NG$ grid in which each cell has edges of length ($x$max $-$ $x$min)/NG and ($y$max $-$ $y$min)/NG. Each cell of the grid containing at least one node corresponds to a cluster. As to NG, it is determined so as to have a prefixed average number μ (an input parameter) of nodes in each cluster. To this end, let CLUSTER(H) be the number of nonempty clusters corresponding to the $H \times H$ grid, and define NG as the minimum integer such that CLUSTER(NG) $\geq n/\mu$.

Table IV gives, for each test problem and value of μ = 3, 5, 7, 10, the overall CPU time (in HP 9000/720 CPU seconds), the number of nodes of the branch-decision tree, and the number m of clusters.

Comparing Table IV (for $\mu = 5$) and Table II shows that the grid clusterization produces harder instances. No correlation exists, instead, between the difficulty of the problem and the average number of nodes in each cluster.

On the whole, the computational performances of the branch-and-cut algorithm are quite satisfactory. All the test problems in our test bed were solved to optimality within acceptable computing time, with the only exception of problem TS225 with grid clusterization (case $\mu = 5$ of Table IV). Moreover, the heuristic algorithms proposed allow one to compute very good solutions within short computing time. As shown in Table I, after the Lagrangian phase the average percentage error with respect to the optimum is 0.5% (see column Lagr-UB), and 0.1% at the end of the root node (see column root-UB).

Table V
Least-Square Regression

	α		β	
Time	Value	p-value	Value	p-value
total-t	2.4E-7	6.0E-19	3.99	5.7E-24
LP-t	3.9E-8	1.3E-18	4.23	1.3E-22
SEP-t	4.5E-10	2.2E-21	4.70	4.7E-23

To summarize the empirical performance of the branch-and-cut algorithm, we ran a least-square estimate on the various running times reported in Table II with respect to n, according to the model $Time = \alpha n^{\beta}$ and with a confidence level of 95%. The estimates are given in Table V.

ACKNOWLEDGMENT

Work supported by M.U.R.S.T. and by C.N.R., Progetto Finalizzato Trasporti 2 (Italy), by the EEC-Human Capital & Mobility Program (contract Nr ERBCIPDCT940623), and by the Royal Spanish College in Bologna. We thank CIOC-CNR and the PROMETHEUS research project for the use of the DECstation 5000/240, and "Departamento de Estadística, Investigación Operativa y Computación" (University of La Laguna) for the use of the HP 9000/720 computer.

REFERENCES

AHUJA, R. K., T. L. MAGNANTI, AND J. B. ORLIN. 1989. Network Flows. In G. L. Nemhauser, A. H. G. Rinnooy Kan, M. J. Todd (eds.), *Optimization*, Vol. I. North-Holland, 211–370.

BALAS, E. 1989. The Prize Collecting Traveling Salesman Problem. *Networks,* **19,** 621–636.

BALAS, E. 1993. The Prize Collecting Traveling Salesman Problem: II. Polyhedral Results. Working Paper, Carnegie Mellon University.

CARPANETO, G., M. FISCHETTI, AND P. TOTH. 1989. New Lower Bounds for the Symmetric Traveling Salesman Problem. *Math. Programming,* **45,** 233–254.

FISCHETTI, M., J. J. SALAZAR, AND P. TOTH. 1995. The Symmetric Generalized Traveling Salesman Polytope. *Networks,* **26,** 113–123.

FISCHETTI, M., AND P. TOTH. 1988. An Additive Approach for the Optimal Solution of the Prize-Collecting Traveling Salesman Problem. In *Vehicle Routing: Methods and Studies*, B. L. Golden and A. A. Assad (eds.), North-Holland, 319–343.

GOLDEN, B. L., AND W. R. STEWART. 1985. Empirical Analysis of Heuristics. In *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*, E. L. Lawler, J. K. Lenstra, A. H. G. Rinnoy Kan and D. B. Shmoys (Eds.), Wiley, New York, 207–245.

GOMORY, R. E., AND T. C. HU. 1961. Multi-Terminal Network Flows. *SIAM J.* **9,** 551–570.

GRÖTSCHEL, M., AND O. HOLLAND. 1991. Solution of Large-Scale Symmetric Traveling Salesman Problems. *Math. Programming,* **51,** 141–202.

HELD, M., AND R. M. KARP. 1971. The Traveling Salesman Problem and Minimum Spanning Trees: Part II. *Math. Programming,* **1,** 6–25.

JÜNGER, M., G. REINELT, AND G. RINALDI. 1995. The Traveling Salesman Problem. In *Handbook in Operations Research and Management Science: Network Models*. M. O. Ball, T. L. Magnanti, C. L. Monma, G. L. Nemhauser (eds.), Elsevier, Amsterdam.

KRUSKAL, J. B. 1956. On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem. *Proc. AMS,* **7,** 48–50.

LAPORTE, G., H. MERCURE, AND Y. NOBERT. 1987. Generalized Traveling Salesman Problem Through *n* Sets of Nodes: The Asymmetrical Cases. *Discrete Appl. Math.* **18,** 185–197.

LAPORTE, G., AND Y. NOBERT. 1983. Generalized Traveling Salesman Through *n* Sets of Nodes: An Integer Programming Approach. *INFOR,* **21,** 61–75.

LAWLER, E. L., J. K. LENSTRA, A. H. G. RINNOOY KAN, AND D. B. SHMOYS (EDS). 1985. *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*. Wiley, NY.

NEMHAUSER, G. L., AND L. A. WOLSEY. 1988. *Integer and Combinatorial Optimization*. Wiley, NY.

NOON, C. E. 1988. The Generalized Traveling Salesman Problem. Unpublished Dissertation, Department of Industrial and Operations Research, University of Tennessee.

NOON, C. E., AND J. C. BEAN. 1991. A Lagrangian Based Approach for the Asymmetric Generalized Traveling Salesman Problem. *Opns. Res.* **39,** 623–632.

PADBERG, M. W., AND M. GRÖTSCHEL. 1985. Polyhedral Computations. In *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*, E. L. Lawler, J. K. Lenstra, A. H. G. Rinnoy Kan and D. B. Shmoys (Eds.), Wiley, New York, 307–360.

PADBERG, M. W., AND M. R. RAO. 1982. Odd Minimum Cut-Sets and b-Matching, *Math. O.R.* **7,** 67–80.

PADBERG, M. W., AND G. RINALDI. 1987. Optimization of a 532-city Symmetric Traveling Salesman Problem. *O. R. Lett.* **6,** 1–7.

PADBERG, M. W., AND G. RINALDI. 1990. Facet Identification for the Symmetric Traveling Salesman Polytope. *Math. Programming,* **47,** 219–257.

PADBERG, M. W., AND G. RINALDI. 1991. A Branch-and-cut Algorithm for the Resolution of Large-scale Symmetric Traveling Salesman Problems. *SIAM Review,* **33,** 60–100.

REINELT, G. 1991. TSPLIB—A Traveling Salesman Problem Library. *ORSA J. Comput.* **3,** 376–384.

SALAZAR, J. J. 1992. Algoritmi per il Problema del Commesso Viaggiatore Generalizzato. Ph.D. Dissertation, Royal Spanish College in Bologna, Italy.

SEPEHRI, M. M. 1991. The Symmetric Generalized Traveling Salesman Problem. Ph.D. Dissertation, University of Tennessee, Knoxville.