Chapter 13

THE GENERALIZED TRAVELING SALESMAN AND ORIENTEERING PROBLEMS

Matteo Fischetti
D.E.I., University of Padova
Via Gradenigo 6/A, 35100 Padova, Italy
fisch@dei.unipd.it

Juan-José Salazar-González
D.E.I.O.C., University of La Laguna
38271 La Laguna, Tenerife, Spain
jjsalaza@ull.es

Paolo Toth
D.E.I.S., University of Bologna
Viale Risorgimento 2, 40136 Bologna, Italy
ptoth@deis.unibo.it

1. Introduction

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.

In several applications the cycle is allowed to visit only a subset of the nodes of the graph, chosen according to a specified criterion. A basic version of this problem is the following *Simple Cycle Problem* (SCP). We are given a complete undirected graph $K_n = (V, E)$ on $n := |V|$ nodes, a cost c_e associated with each edge $e \in E$, and a prize p_v associated with each node $v \in V$. Recall that a (simple) cycle of K_n is a subset $\tilde{E}$ of E, $|\tilde{E}| \geq 3$, inducing a subgraph $(V(\tilde{E}), \tilde{E})$ which is connected and in

G. Gutin et al. (eds.), *The Traveling Salesman Problem and Its Variations*

which all nodes in $V(\tilde{E})$ have degree two. The *cost* of a cycle $\tilde{E}$ is given by $\sum_{e\in\tilde{E}} c_e - \sum_{v\in V(\tilde{E})} p_v$. The problem is to find a min-cost cycle of K_n.

Without loss of generality one can assume $c_e \geq 0$ for all $e \in E$ and $p_v \geq 0$ for all $v \in V$, since the addition of any constant to all edge costs and to all node prizes does not affect the cycle cost.

SCP is a useful model for problems involving simultaneous selection and sequencing decisions. Indeed, the problem involves two related decisions:

1 choosing a convenient node subset $S \subseteq V$,

2 finding a minimum cost Hamiltonian cycle in the subgraph induced by S.

Many variants of SCP have been studied in the literature, a non-exhaustive list of which is given later in this section. Roughly speaking, we can classify these variants into two main classes, the first including all variants subsuming the TSP (i.e., those for which every Hamiltonian cycle is feasible), and the second including all the variants in which additional constraints may prevent some Hamiltonian cycles from being feasible. This property has important consequences when analyzing the structure of the polytope P associated with a certain SCP variant. Indeed, whenever P contains the TSP polytope, one can apply simple lifting constructions to extend known TSP facets to P, whereas more involved constructions are required for the problems in the second class. We will therefore describe in a rather detailed way a specific variant for each of the two classes, namely the Generalized TSP (Section 2) and the Orienteering Problem (Section 3). These two problems have been chosen as they seem to be the most-widely studied cycle-type problems, along with the Prize-Collecting TSP considered in Chapter 14. In particular, for both problems we will concentrate on exact solution methods based on the branch-and-cut approach, which proved to be the most effective framework for optimally solving cycle-type problems.

1.1. The Simple Cycle Problem

The Simple Cycle Problem, SCP, has the following natural Integer Linear Programming formulation. For any $S \subseteq V$, let $\delta(S)$ represent the set of edges with exactly one endnode in S, and let $E(S)$ be the set of edges with both endnodes in S, i.e.,

$$\begin{aligned} \delta(S) &:= \{(i,j) \in E : i \in S, j \notin S\}, \\ E(S) &:= \{(i,j) \in E : i,j \in S\}. \end{aligned}$$

As it is customary, we write $\delta(v)$ instead of $\delta(\{v\})$ for $v \in V$. Moreover, for any real function $f : Q \to \mathbb{R}$ on a finite domain Q and for any $T \subseteq Q$, we write $f(T)$ instead of $\sum_{q \in T} f_q$.

Our model introduces a binary variable x_e associated with each edge $e \in E$ (where $x_e = 1$ if and only if e belongs to the optimal cycle), and a binary variable y_v associated with each node $v \in V$ (where $y_v = 1$ if and only if v is visited by the optimal cycle). The model reads:

$$v(SCP) \equiv \min \sum_{e \in E} c_e x_e - \sum_{v \in V} p_v y_v \tag{1}$$

subject to:

$$x(\delta(v)) = 2y_v \qquad v \in V \tag{2}$$

$$x(\delta(S)) \geq 2(y_i + y_j - 1) \qquad S \subset V, i \in S, j \in V \setminus S \tag{3}$$

$$y(V) \geq 3 \tag{4}$$

$$x_e \in \{0,1\} \qquad e \in E \tag{5}$$

$$y_v \in \{0,1\} \qquad v \in V. \tag{6}$$

Constraints (2) impose that the number of edges incident to a node v is either 2 (if v is visited) or 0 (otherwise). Inequalities (3) are connectivity constraints saying that each cut separating two visited nodes (i and j) must be crossed at least twice. Constraint (4) forces at least three nodes to be visited by the cycle.

A variant of SCP requires the cycle visits a specified "depot" node, say node 1. This can be easily obtained by adding a large positive value to prize p_1, or by introducing explicitly the additional constraint:

$$y_1 = 1. \tag{7}$$

In this case, the connectivity constraints (3) can be replaced by

$$x(\delta(S)) \geq 2y_v \qquad \text{for } S \subset V, 1 \in S, v \in V \setminus S.$$

SCP is known to be strongly $\mathcal{NP}$-hard, as it contains as a special case the problem of finding a Hamiltonian cycle of a given undirected arbitrary graph $G = (V, \bar{E})$ (just set $c_e = 0$ for all $e \in \bar{E}$, $c_e = 1$ for all $e \in E \setminus \bar{E}$, $p_v = 1$ for all $v \in V$, and check whether the min-cost cycle on K_n has cost equal to $-n$).

1.2. The Weighted Girth Problem

This problem arises from SCP when $p_v = 0$ for all $v \in V$, and negative edge costs are allowed. Notice that, because of (2), one can always

convert node prizes into edge costs, by redefining $c_{(i,j)} := c_{(i,j)} - (p_i + p_j)/2$ for all $(i,j) \in E$, and $p_v = 0$ for all $v \in V$. As a consequence, the Weighted Girth Problem (WGP) is equivalent to SCP, hence it is strongly $\mathcal{NP}$-hard.

A relevant polynomially-solvable special case arises when the graph does not contain negative cycles (negative costs being allowed). Indeed, in this case the problem can be transformed into $|V|$ non-bipartite matching problems on G with loops (see, e.g., Coullard and Pulleyblank [227]), hence it can be solved in $O(|V|^4)$ time. A simpler algorithm can be obtained along the following lines.

We replace each edge $e = (i,j)$ of G by two arcs (i,j) and (j,i) with cost $c_{ij} := c_{ji} := c_e$ (we set $c_{ij} := \infty$ for all missing arcs, including loops). Observe that this construction induces a negative circuit of length 2 for each negative-cost edge, but no negative-cost circuit involving more than 2 arcs. Therefore, WGP can be restated as the problem of finding a minimum-cost (possibly nonsimple) circuit (closed trail) on the new digraph, with the constraint that the circuit contains no 2-length circuit. As such, it can be solved by dynamic programming using the recursions originally proposed by Christofides, Mingozzi and Toth [192] to derive the so-called q-path lower bound for the Vehicle Routing Problem, as outlined next.

Let us consider the case where a certain node r (e.g., $r = 1$) is assumed to be visited by the circuit (the most general case can be solved by trying all possible nodes r), and let $\gamma(j) := \{i \in V : (i,j) \in E\}$. For each node $j \in V$ and for each integer $h = 1, \ldots, n$, let $f(h,j)$ represent the minimum cost of a directed path (not necessarily simple) with h arcs that starts from node r, arrives at node j, and contains no 2-length circuit. Moreover, let $\pi(h,j)$ denote the node immediately preceding node j in the path corresponding to $f(h,j)$, and let $g(h,j)$ be the minimum cost of a path having the same properties as the one corresponding to $f(h,j)$, but with the node immediately preceding j forced to be different from $\pi(h,j)$. Initially, for each $j \in V$ we set $f(1,j) := c_{rj}$, $\pi(1,j) := r$, and $g(1,j) := \infty$. Then, for $h = 2, \ldots, n$ and for each $j \in V$ we compute:

$$f(h,j) := \min_{i \in \gamma(j)} \begin{cases} f(h-1,i) + c_{ij} & if \pi(h-1,i) \neq j \\ g(h-1,i) + c_{ij} & otherwise, \end{cases}$$

(let $\pi(h,j)$ be the node corresponding to the minimum above)

$$g(h,j) := \min_{i \in \gamma(j) \setminus \{\pi(h,j)\}} \begin{cases} f(h-1,i) + c_{ij} & if \pi(h-1,i) \neq j \\ g(h-1,i) + c_{ij} & otherwise. \end{cases}$$

The optimal solution value of WGP can then be computed as:

$$v(WGP) := \min\{f(h,r) : h = 3,\ldots,n\}.$$

The above computation can be clearly performed in $O(|E|n)$ time, hence the method requires $O(n^3)$ time in the worst case.

WGP is a basic relaxation of several problems with important practical applications in routing and location, and has been studied mainly from a theoretical point of view.

The polyhedral structure of the weighted girth problem has been deeply investigated by Bauer [92], who studied several classes of facet-defining inequalities, some of which derived from the TSP polytope. Lifting theorems relating the TSP and the WGP polytopes are given in Salazar [736]. Balas and Oosten [79] investigated the corresponding polytope for a directed graph; see Chapter 14 for details.

1.3. The Prize-Collecting TSP

In the Prize-Collecting TSP each node $v \in V$ has an associated non-negative *weight* w_v, and a cycle $\tilde{E}$ is considered to be feasible only if the total weight $w(V(\tilde{E}))$ of the visited nodes is not less than a given threshold w_0. Therefore, the problem can be formulated as (1)–(6), plus the additional constraint:

$$\sum_{v \in V} w_v y_v \geq w_0. \tag{8}$$

Such an $\mathcal{NP}$-hard problem arises, for instance, when a factory located at node 1 needs a given amount w_0 of a product, which can be provided by a set of suppliers located at nodes $2,\ldots,n$. Let w_v be the indivisible amount supplied at node v, $-p_v$ be the corresponding cost ($v = 2,\ldots,n$), and let $c_{(i,j)}$ be the transportation cost from node i to node j ($i,j \in V, i \neq j$). Assuming that only one trip is required, such a problem can be formulated as an instance of the Prize-Collecting TSP–in the version requiring the visit of node 1.

The directed counterpart of the problem also arises in several scheduling problems. Balas and Martin [77] introduced the Prize-Collecting TSP as a model for scheduling the daily operations of a steel rolling mill. A rolling mill produces steel sheets from slabs by hot or cold rolling. The cost of arc (i,j) is given by the cost of processing order j just after order i, and w_v is the weight of the slab associated with order v. Scheduling the daily operations consists of selecting a subset of orders whose total weight satisfies a given lower bound w_0, and of sequencing them so as to minimize the global cost.

The Prize-Collecting TSP has been mainly investigated in its directed version. Heuristic methods have been proposed by Balas and Martin [77]. Balas [63, 65] analyzed the problem from a polyhedral point of view. Fischetti and Toth [302] proposed a branch-and-bound exact algorithm based on additive bounding procedures. Bienstock, Goemans, Simchi-Levi and Williamson [109] presented an approximation algorithm with constant bound. The reader is referred to Chapter 14 of this book for a comprehensive treatment of the subject.

1.4. The Capacitated Prize-Collecting TSP

The *Capacitated Prize-Collecting TSP* is an extension of the Prize-Collecting TSP where (8) is replaced by

$$\sum_{v \in V} w_v y_v \leq w_0. \tag{9}$$

Here w_v represents the weight of node (customer) v, and w_0 is the capacity of a vehicle originally located at the depot (say node 1). Constraints (9) specify that the total load carried by the vehicle cannot exceed the vehicle capacity. This $\mathcal{NP}$-hard problem was introduced by Bixby, Coullard and Simchi-Levi [110] as a column generation subproblem in a set partitioning formulation of the classical Capacitated Vehicle Routing Problem (CVRP). They also presented a branch-and-cut algorithm and solved to optimality instances ranging in size from 50 to 280 nodes.

1.5. The Orienteering Problem

The *Orienteering Problem* (OP), also called the "Selective TSP", is in a sense the "dual" of the Prize-Collecting TSP. Here, the cycle cost only depends on the node prizes, i.e., $c_e = 0$ for all $e \in E$, and the objective is to maximize the global prize of the visited nodes. On the other hand, each edge $e \in E$ has an associated nonnegative *duration* t_e, and a cycle $\tilde{E}$ is considered to be feasible only if its total duration $t(\tilde{E})$ does not exceed a given threshold t_0. Moreover, the cycle is required to visit node 1. Model (1)–(7) then needs to be amended by the additional constraint:

$$\sum_{e \in E} t_e x_e \leq t_0 \tag{10}$$

OP is strongly $\mathcal{NP}$-hard as it contains as a special case the problem of finding a Hamiltonian cycle of a given undirected arbitrary graph $G = (V, \bar{E})$ (just set $p_v = 1$ for all $v \in V$, $t_e = 0$ for all $e \in \bar{E}$, $t_e = 1$ for

all $e \in E \setminus \bar{E}$, and $t_0 = 0$, and check whether the optimal solution has value n).

The problem derives its name from the Orienteering sport, where each participant has to maximize the total prize to be collected, while returning to the starting point within a given time limit. OP also arises in several routing and scheduling applications, see, e.g., Golden, Levy and Vohra [387].

Heuristic algorithms for OP and some generalizations have been proposed by Tsiligirides [796], Golden, Levy and Vohra [387], Golden, Wang and Liu [389], Chao, Golden and Wasil [178] and Fink, Schneidereit and Voss [290]. Exact branch-and-bound methods have been proposed by Laporte and Martello [536], and by Ramesh, Yoon and Karwan [693]. Leifer and Rosenwein [553] have discussed an LP-based bounding procedure. Recently, Fischetti, Salazar and Toth [301] and Gendreau, Laporte and Semet [355] have proposed branch-and-cut algorithms; see Section 3 for more details.

1.6. The Generalized TSP

In the *Generalized* TSP (GTSP), also known as the "International TSP", we are given a proper partition of V into $m \geq 3$ clusters

$$C_1, C_2, \ldots, C_m,$$

and the cycle is feasible only if it visits each cluster at least once. Typically we also have $p_v = 0$ for all $v \in V$. The corresponding additional constraints for model (1)–(6) are:

$$\sum_{v \in C_h} y_v \geq 1 \qquad \text{for } h = 1, \ldots, m \tag{11}$$

A different version of the problem, called *E-GTSP* (where E stands for Equality), arises when imposing that exactly one node of each cluster must be visited, i.e., (11) is replaced by:

$$\sum_{v \in C_h} y_v = 1 \qquad \text{for } h = 1, \ldots, m \tag{12}$$

The two versions are clearly equivalent when the costs satisfy the triangle inequality, i.e., $c_{(i,j)} \leq c_{(i,k)} + c_{(k,j)}$ for all node triples (i, j, k).

Both GTSP and E-GTSP find practical applications in the design of ring networks, sequencing of computer files, routing of welfare customers through governmental agencies, airport selection and routing for courier planes, flexible manufacturing scheduling, and postal routing; see, e.g., Noon [629], Noon and Bean [630], and Laporte, Asef-Vaziri and Sriskandarajah [535].

The two problems are clearly $\mathcal{NP}$-hard, as they reduce to the TSP when $m = n$, i.e., $|C_h| = 1$ for all h. They have been studied, among others, by Laporte and Nobert [538], Salazar [735], Sepehri [749], and Fischetti, Salazar and Toth [299, 300]. Their asymmetric counterparts have been investigated in Laporte, Mercure and Nobert [537], and Noon and Bean [630]. Transformations from the GTSP to the asymmetric TSP have been proposed by Noon and Bean [631], Lien, Ma and Wah [561], Dimitrijevic and Saric [255], and Laporte and Semet [541]. See also Cvetković, Dimitrijević and Milosavljević [236]. Semet and Renaud [748] presented a tabu-search algorithm for the E-GTSP.

A more detailed analysis of both GTSP and E-GTSP will be given in Section 2.

1.7. The Covering Tour Problem

The *Covering Tour Problem* is a variant of the Generalized TSP arising when the clusters C_k are not necessarily disjoint. This problem was introduced by Gendreau, Laporte and Semet [354] as a model for the location of post boxes and for the planning of routes for medical teams in developing countries, where each cluster C_k corresponds to the subset of nodes located within a given distance of a certain customer k. They analyzed the polyhedral structure of the problem, presented a branch-and-cut algorithm, and tested its performance on random instances with up to 100 nodes. Current and Schilling [233] studied a multi-objective version of the same problem, that they called the *Covering Salesman Problem.*

1.8. The Median Cycle Problem

The *Median Cycle Problem* looks for a min-cost cycle $\tilde{E}$ such that the sum of the distances between each node not in $\tilde{E}$ and its closest node in $\tilde{E}$ does not exceed a given value d_0. In other words, the Median Cycle Problem looks for a min-cost cycle $\tilde{E}$ such that

$$\sum_{i \notin V(\tilde{E})} \min_{j \in V(\tilde{E})} d_{ij} \leq d_0,$$

where d_{ij} represents the distance between nodes i and j.

This problem was introduced by Labbé, Laporte, Rodríguez and Salazar [527] as a location model for circular-shaped transportation infrastructures. These authors provided a polyhedral analysis and a branch-and-cut algorithm tested on random instances with up to 150 nodes. Current and Schilling [234] proposed heuristics for a variant of this problem which consists of finding a cycle $\tilde{E}$ visiting no more than p nodes,

while minimizing the weighted sum of the cycle cost and of the largest distance of the nodes in $\tilde{E}$ from the unrouted nodes (the latter being computed as $\max_{i \notin V(\tilde{E})} \min_{j \in V(\tilde{E})} d_{ij}$). They also studied the problem of minimizing the cycle cost while imposing

$$\max_{i \notin V(\tilde{E})} \min_{j \in V(\tilde{E})} d_{ij} \leq d_0.$$

1.9. The Traveling Purchaser Problem

In the *Traveling Purchaser Problem* node 1 corresponds to the purchaser's domicile, and the other nodes to markets. There are m products to be purchased, the kth product being available in a given cluster of markets C_k. The problem looks for a cycle starting at the domicile and purchasing each product while minimizing the sum of the cycle cost plus the purchasing costs. To be more specific, let f_{ik} be the cost of purchasing product k at node $i \in C_k$. As in the Covering Tour problem, a cycle $\tilde{E}$ is considered feasible if and only if $1 \in V(\tilde{E})$ and $V(\tilde{E}) \cap C_k \neq \emptyset$ for all $k = 1, \ldots, m$. The Traveling Purchaser Problem then calls for a min-cost feasible cycle, the cost of a feasible cycle $\tilde{E}$ being computed as

$$\sum_{e \in \tilde{E}} c_e + \sum_{k=1}^{m} \min\{f_{ik} : i \in C_k \cap V(\tilde{E})\}$$

In the *Capacitated Traveling Purchaser Problem*, for each product k we also have a required amount d_k to be purchased, while the quantity of product k available at each node $i \in C_k$ is q_{ik}. In this version, f_{ik} represents the cost of purchasing one unit of product k at node $i \in C_k$, and the objective is to find a route collecting the required amount d_k for each product k, while minimizing the sum of the routing and the purchasing costs.

The uncapacitated version of the problem was originally introduced by Burstall [152] and Ramesh [694]. Heuristic methods have been proposed by, e.g., Voss [814], Golden, Levy and Dahl [386], Ong [632], Pearn and Chien [662], and Boctor, Laporte and Renaud [119],

Branch-and-bound exact algorithms have been studied by, e.g., Singh and van Oudheusden [762], reporting the solution of 25-node instances. Laporte, Riera and Salazar [540] proposed a branch-and-cut algorithm for the exact solution of the capacitated version, which is capable of solving random instances involving up to 200 nodes.

2. The Generalized Traveling Salesman Problem

As already stated, in the GTSP we are given a proper partition of V into $m \geq 3$ node subsets $C_1, \ldots, C_m$, called *clusters*. A cycle is consid-

ered feasible if it goes through each cluster at least once. The GTSP then consists of finding a feasible cycle $\tilde{E} \subset E$ whose global cost $\sum_{e \in \tilde{E}} c_e$ is a minimum, and can be formulated as

$$v(GTSP) \equiv \min \sum_{e \in E} c_e \, x_e \tag{13}$$

subject to

$$\sum_{e \in \delta(v)} x_e = 2 \, y_v \qquad \text{for } v \in V \tag{14}$$

$$\sum_{v \in C_h} y_v \geq 1 \qquad \text{for } h = 1, \ldots, m \tag{15}$$

$$\sum_{e \in \delta(S)} x_e \geq 2 \, (y_i + y_j - 1) \qquad \text{for } S \subset V, \, 2 \leq |S| \leq n-2, \; i \in S, \, j \in V \setminus S \tag{16}$$

$$x_e \in \{0, 1\} \qquad \text{for } e \in E \tag{17}$$

$$y_v \in \{0, 1\} \qquad \text{for } v \in V. \tag{18}$$

As to the E-GTSP, arising when imposing that each cluster must be visited exactly once, a mathematical model is obtained from (13)–(18) by replacing (15) with

$$\sum_{v \in C_h} y_v = 1 \qquad \text{for } h = 1, \ldots, m. \tag{19}$$

Model (13)–(18) heavily relies on the integrality of the y variables. If this requirement is relaxed, solutions like those of Figure 2 become feasible. Therefore, the LP relaxation of this model can be very poor. Additional valid inequalities will be described in the sequel, whose introduction in the model leads to a considerable strengthening of its LP relaxation.

As shown in [300], the E-GTSP is polynomially solvable when the sequence of the clusters is known, which implies the polynomial solvability for fixed m (see Subsection 2.5 for more details).

The remaining part of the present section is mainly based on the results given by Fischetti, Salazar and Toth in [299] and [300]. We first analyze the facial structure of the GTSP and the E-GTSP polytopes; in particular, in Section 2.2 we introduce a general theorem that allows one to lift any facet of the TSP polytope into a facet of the GTSP polytope. This result is used to derive classes of facet-inducing inequalities related to the *subtour elimination* and *comb* constraints. In Section 2.3 we analyze the E-GTSP polytope and discuss the cases in which the

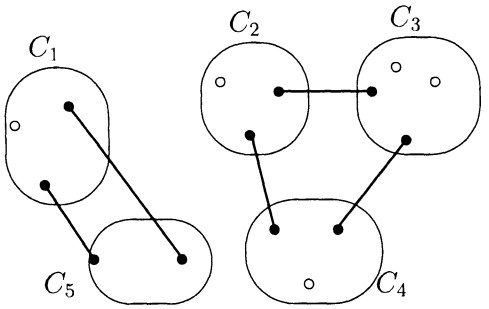

Figure 13.1. Infeasible GTSP solution satisfying (14)–(17). The drawn edges have $x_e = 1$, the blank nodes $y_v = 0$, and the black nodes $y_v = 1/2$.

inequalities of Section 2.2 are facet-inducing. The analysis is based on a general result which allows one to inductively reduce the polyhedral analysis of the E-GTSP polytope to that of the TSP polytope.

These theoretical results are used in Section 2.6 to design a branch-and-cut algorithm for the exact solution of large-scale E-GTSP instances. The algorithm is based on exact/heuristic separation procedures for the main classes of inequalities previously analyzed. We finally report results on test instances involving up to 442 nodes, showing that these inequalities lead to a substantial improvement of the LP relaxation of the original model.

2.1. Basic notations

Let P, $P^=$, and Q denote, respectively, the *GTSP*, *E-GTSP*, and *TSP polytopes*, defined as

$$P := \text{conv}\{(x,y) \in \mathbb{R}^{E\cup V} : \text{(14)-(18) hold}\},$$

$$P^= := P \cap \{(x,y) \in \mathbb{R}^{E\cup V} : \text{(19) holds}\},$$

and

$$Q := P \cap \{(x,y) \in \mathbb{R}^{E\cup V} : y_v = 1 \text{ for all } v \in V\}.$$

Clearly, $P^=$ and Q are faces of P. These faces are disjoint when $m < n$, whereas for $m = n$ the three polytopes P, $P^=$, and Q coincide.

We assume the reader is familiar with the foundations of polyhedral theory. For the sake of simplicity, in the following we will not distinguish between a GTSP (or E-GTSP) solution and its characteristic vector, and assume $m \geq 5$. Moreover, we will make use of the following notation:

$$\mu(S) := |\{h : C_h \subseteq S\}| \qquad \text{for } S \subseteq V,$$

$$\eta(S) := |\{h : C_h \cap S \neq \emptyset\}| \qquad \text{for } S \subseteq V,$$

and denote by $C_{h(v)}$ the cluster containing a given node v. We also define

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

2.2. Facet-defining inequalities for the GTSP polytope

In this section we study the GTSP polytope, P. The facial structure of P is clearly related to that of the TSP polytope, Q, arising when imposing the additional equations $y_v = 1$ for all $v \in V$. In order to link these two polytopes, let us define the intermediate polytopes

$$P(F) := P \cap \{(x, y) \in \mathbb{R}^{E \cup V} : y_v = 1 \text{ for all } v \in F\},$$

where $\emptyset \subseteq F \subseteq V$. By definition, $P(V) = Q$ and $P(\emptyset) = P$.

Our first order of business is to determine the dimension of $P(F)$ for any given F. This amounts to studying the equation system for $P(F)$. This system includes the $|V|$ linearly independent equations (14), plus the variable fixing equations

$$y_v = 1 \quad \text{for all } v \in F \cup W, \tag{20}$$

where W has been defined previously. Actually, no other linearly independent equations satisfied by all the points of $P(F)$ exist, as implied by the following result.

Theorem 1 *For all $F \subseteq V$, $dim(P(F)) = |E| - |F \cup W|$.*

Proof: Clearly $\dim(P(F)) \leq |E| - |F \cup W|$ since $P(F) \subset \mathbb{R}^{E \cup V}$ and the $|V| + |F \cup W|$ valid equations (14) and (20) are linearly independent. We claim the existence of $|E| - |F \cup W| + 1$ affinely independent points in $P(F)$. This will prove $\dim(P(F)) \geq |E| - |F \cup W|$, and hence the theorem. The proof of the claim is by induction on the cardinality of F.

When $|F| = n$ the claim is true, since $P(F)$ corresponds to the TSP polytope (see, e.g., Grötschel and Padberg [405] or Chapter 2).

Assume now the claim holds for $|F| = \sigma$, and consider any node set F' with $|F'| = \sigma - 1$. Let v be any node not in F', and define $F := F' \cup \{v\}$. Because of the induction hypothesis, there exist $|E| - |F \cup W| + 1$ affinely independent points belonging to $P(F)$ hence to $P(F')$. If $v \in W$ then $|F \cup W| = |F' \cup W|$, and we are done. Otherwise, $|F \cup W| = |F' \cup W| + 1$, i.e., we need an additional point. Such a point always exists, and corresponds to any Hamiltonian cycle in the subgraph induced by $V \setminus \{v\}$. ■

Corollary 2 *dim*$(P) = |E| - |W|$.

According to Theorem 1, given any nonempty $F \subseteq V$ and any $v \in F$ one has the following: if $v \in W$ then $\dim(P(F \setminus \{v\})) = \dim(P(F))$, else $\dim(P(F \setminus \{v\})) = \dim(P(F)) + 1$. In other words, the removal of a node from F increases the dimension of $P(F)$ by, at most, one unit. As a consequence, any facet-defining inequality for $P(F)$ can be lifted in a simple way so as to be facet-inducing for $P(F \setminus \{v\})$ as well.

Theorem 3 *Let $F \subseteq V$ and $u \in F$. In addition, let*

$$\sum_{e \in E} \alpha_e \, x_e + \sum_{v \in V} \beta_v \, (1 - y_v) \geq \gamma$$

be any facet-inducing inequality for $P(F)$. Then the lifted inequality

$$\sum_{e \in E} \alpha_e \, x_e + \sum_{v \in V \setminus \{u\}} \beta_v \, (1 - y_v) + \tilde{\beta}_u \, (1 - y_u) \geq \gamma$$

is valid and facet-defining for $P(F \setminus \{u\})$, where $\tilde{\beta}_u$ is an arbitrary value if $u \in W$, whereas

$$\tilde{\beta}_u = \gamma - \min \left\{ \sum_{e \in E} \alpha_e \, x_e + \sum_{v \in V \setminus \{u\}} \beta_v \, (1 - y_v) \; : \; \begin{array}{c} (x, y) \in P(F \setminus \{u\}), \\ \text{and } y_u = 0 \end{array} \right\}$$

holds when $u \notin W$.

Proof: The claim follows from the well-known sequential lifting theorem (Padberg [641]), as described, e.g., in Grötschel and Padberg [405]. ■

Theorem 3 leads to a lifting procedure to be used to derive facet-inducing inequalities for the GTSP polytope from those of the TSP polytope. To this end one has to choose any *lifting sequence* for the nodes, say $\{v_1, \ldots, v_n\}$, and iteratively derive a facet of $P(\{v_{t+1}, \ldots, v_n\})$ from a facet of $P(\{v_t, \ldots, v_n\})$ for $t = 1, \ldots, n$. Different lifting sequences can produce different facets.

By using the above lifting procedure one can easily prove the following results.

Theorem 4 *The following inequalities define facets of P:*

$$x_e \geq 0 \qquad \text{for every } e \in E, \tag{21}$$

$$x_e \leq 1 \qquad \text{whenever } e \in E(W), \tag{22}$$

$$y_v \leq 1 \qquad \text{whenever } v \notin W, \tag{23}$$

$$\sum_{e \in \delta(S)} x_e \geq 2 \qquad \text{for } S \subset V : \ 2 \leq |S| \leq n-2, \ \mu(S) \neq 0, \mu(V \setminus S) \neq 0. \tag{24}$$

$$\sum_{e \in \delta(S)} x_e \geq 2\, y_i \qquad \text{for } S \subset V : \ 2 \leq |S| \leq n-2, \ \mu(S) = 0, \mu(V \setminus S) \neq 0, \ i \in S, \tag{25}$$

$$\sum_{e \in \delta(S)} x_e \geq 2\,(y_i + y_j - 1) \qquad \text{for } S \subset V : \ 2 \leq |S| \leq n-2, \ \mu(S) = \mu(V \setminus S) = 0, \ i \in S, j \in V \setminus S. \tag{26}$$

By possibly interchanging the role of S and $V \setminus S$, one can always assume that inequalities (24) and (26) are written for $S \subset V$ such that $|S| \leq \lfloor n/2 \rfloor$. The same holds for inequalities (25) by choosing $i \in V \setminus S$ when $\mu(S) \neq 0$ and $\mu(V \setminus S) = 0$.

Notice that (25) are also valid (but not facet-inducing) when $\mu(S) \neq 0$. Analogously, inequalities (26) hold for any $S \subset V$ and coincide with (16).

By exploiting equations (14), inequalities (24), (25) and (26) above can be rewritten, for any $S \subset V$ with $2 \leq |S| \leq n-2$, as the following *Generalized Subtour Elimination Constraints* (GSEC's):

$$\sum_{e \in E(S)} x_e \leq \sum_{v \in S} y_v - 1 \qquad \text{for } \mu(S) \neq 0, \mu(V \setminus S) \neq 0, \tag{27}$$

$$\sum_{e \in E(S)} x_e \leq \sum_{v \in S \setminus \{i\}} y_v \qquad \text{for } \mu(S) = 0, \mu(V \setminus S) \neq 0, \ i \in S, \tag{28}$$

$$\sum_{e \in E(S)} x_e \leq \sum_{v \in S \setminus \{i\}} y_v - y_j + 1 \qquad \text{for } \mu(S) = \mu(V \setminus S) = 0, \ i \in S, j \notin S. \tag{29}$$

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.

Particular cases of GSEC's arise when $|S| = 2$, leading to

$$x_e \leq y_v \qquad \text{for } v \in V, \ e \in \delta(v). \tag{30}$$

Note that inequality $\sum_{v \in C_h} y_v \geq 1$ does not define a facet of P for any $h = 1, \ldots, m$. Indeed, because of (14), constraint (15) is equivalent to $\sum_{e \in \delta(C_h)} x_e + 2 \sum_{e \in E(C_h)} x_e \geq 2$, hence it is dominated by the valid

inequality $\sum_{e\in\delta(C_h)} x_e \geq 2$ when $E(C_h) \neq \emptyset$, whereas for $E(C_h) = \emptyset$ (i.e., when $|C_h| = 1$) it defines the improper face of P.

We finally consider the TSP *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. 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 is

$$\sum_{e\in E(H)} x_e + \sum_{j=1}^{s} \sum_{e\in E(T_j)} x_e \leq \sigma(C), \tag{31}$$

and is valid and facet-defining for the TSP (see Grötschel and Padberg [405], or Chapter 2). It is well known that interchanging the role of H and $V \setminus H$ produces an equivalent formulation of (31).

Starting with (31), one can obtain related facet-defining inequalities for the GTSP by using the lifting Theorem 3 (trivially modified so as to deal with "$\leq$" inequalities).

Theorem 5 *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 $T_j \setminus H$ if $\mu(T_j \setminus H) = 0$, $b_j = 0$ otherwise. Then the following* generalized comb inequality *is valid and facet-defining for P:*

$$\sum_{e\in E(H)} x_e + \sum_{j=1}^{s} \sum_{e\in E(T_j)} x_e + \sum_{v\in V} \tilde{\beta}_v(1 - y_v) \leq \sigma(C), \tag{32}$$

where $\tilde{\beta}_v = 0$ for all $v \in V \setminus (H \cup T_1 \cup \ldots \cup T_s)$, $\tilde{\beta}_v = 1$ for all $v \in H \setminus (T_1 \cup \ldots \cup T_s)$, and for $j = 1, \ldots, s$: $\tilde{\beta}_v = 2$ for $v \in T_j \cap H$, $v \neq a_j$; $\tilde{\beta}_{a_j} = 1$ if $a_j \neq 0$; $\tilde{\beta}_v = 1$ for $v \in T_j \setminus H$, $v \neq b_j$; $\tilde{\beta}_{b_j} = 0$ if $b_j \neq 0$.

2.3. Facet-defining inequalities for 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 2.2 are also valid (but not necessarily facet-defining) for $P^=$.

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

$$E := \{(i,j) : i \in V, j \in V \setminus C_{h(i)}\}.$$

In view of this reduction, constraints (19) are equivalent to

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

and a simplified model for the E-GTSP can be obtained by replacing constraints (16) and (18) by the two families of constraints:

$$\sum_{e \in E(S)} x_e \leq r - 1 \qquad \text{for } S = \cup_{i=1}^{r} C_{l_i} \text{ and } 3 \leq r \leq m - 3, \quad (33)$$

$$\sum_{e \in \delta(C_l) \cap \delta(w)} x_e \leq y_w \qquad \text{for } l = 1, \ldots, m \text{ and } w \in V \setminus C_l, \quad (34)$$

respectively.

Inequalities (33) will be called *Basic Generalized Subtour Elimination Constraints* (Basic GSEC's), and (34) will be called *fan inequalities.* Both are particular cases of the GSEC's (27) because of (19). Indeed, (33) are equivalent to the GSEC's (27) written for $S = \cup_{i=1}^{r} C_{l_i}$, where $\sum_{v \in S} y_v = r$ since $\sum_{v \in C_{l_i}} y_v = 1$ for $i = 1, \ldots, r$. Analogously, (34) arise from (27) when $S = C_l \cup \{w\}$, since $E(S) = \delta(C_l) \cap \delta(w)$ and $\sum_{v \in S} y_v = \sum_{v \in C_l} y_v + y_w = 1 + y_w$. Notice that constraints (34) dominate (30).

As in the previous subsection, we aim at relating the facial structure of $P^=$ to that of the TSP polytope, Q, even thought Q is not a relaxation of $P^=$. To this end, let us introduce some basic definitions.

Definition 6 *Given a valid inequality $\alpha x + \beta y \leq \gamma$ for $P^=$, let $\mathcal{H}^=_{\alpha,\beta,\gamma} := P^= \cap \{(x,y) \in \mathbb{R}^{E \cup V} : \alpha x = \beta y + \gamma\}$ denote the face of $P^=$ induced by $\alpha x + \beta y \leq \gamma$. Let $v \in V \setminus W$ be an arbitrary but fixed node, and let $P^=_v$ denote the E-GTSP polytope associated with the subgraph of G induced by $V \setminus \{v\}$. The* v-restriction *of $\alpha x + \beta y \leq \gamma$ is the inequality obtained from $\alpha x + \beta y \leq \gamma$ by dropping the variables y_v and x_e for all $e \in \delta(v)$. The* v-compatibility graph *of $\alpha x + \beta y \leq \gamma$ is the graph $G^*_v = (V \setminus C_{h(v)}, E^*)$ with $(u,w) \in E^*$ if and only if there exists $(x,y) \in \mathcal{H}^=_{\alpha,\beta,\gamma}$ with $x_{(v,u)} = x_{(v,w)} = 1$.*

The rank *of a graph is defined as the rank of its edge-node incidence matrix, i.e., the number of its nodes minus the number of its bipartite connected components. The graph is said to be of* full rank *when its edge-node incidence matrix is of full rank, i.e., when it has an odd cycle for each connected component.*

Lemma 7 *For every valid inequality $\alpha x + \beta y \leq \gamma$ for $P^=$ and every node $v \in V \setminus W$ the dimension of $\mathcal{H}^=_{\alpha,\beta,\gamma}$ is greater or equal to the dimension*

of the face of $P_v^=$ induced by its v-restriction, plus the rank of its v-compatibility graph.

Proof: Let X be the matrix in which every row is an extreme point of $\mathcal{H}^=_{\alpha,\beta,\gamma}$. Since $\mathcal{H}^=_{\alpha,\beta,\gamma}$ is contained in a hyperplane not passing through the origin (e.g., that induced by (19) for $h = 1$), a subset of rows of X is affinely independent if and only if it is linearly independent. Hence the dimension of $\mathcal{H}^=_{\alpha,\beta,\gamma}$ coincides with the rank of X minus 1. Now, X can be partitioned into

$$X = \begin{bmatrix} X_{11} & 0 & 0 \\ X_{21} & X_{22} & 1 \end{bmatrix},$$

where the last column corresponds to variable y_v, and the columns of X_{22} correspond to variables x_e for $e \in \delta(v)$. Then the rank of X is, at least, the sum of the rank of X_{11} plus the rank of $[X_{22}\ 1]$. By construction, the rank of X_{11} is the dimension of the face of $P_v^=$ induced by the v-restriction of $\alpha x + \beta y \leq \gamma$, plus 1. As to X_{22}, we observe that each of its rows contains exactly two 1's and (barring repeated rows) can be viewed as the edge-node incidence matrix of the v-compatibility graph G_v^* associated with $\alpha x + \beta y \leq \gamma$. Moreover, the last column of $[X_{22}\ 1]$ is a linear combination (with coefficients 1/2) of the other columns. This proves the claim. ■

Lemma 7 allows us to extend some known results from the TSP polytope to the E-GTSP case by using induction on $\rho = \sum_{h=1}^{m}(|C_h| - 1) = n - m$.

As shown in Lemma 7, the rank of the v-compatibility graph G_v^* associated with a given inequality $\alpha x + \beta y \leq \gamma$ plays a central role when analyzing the polyhedral structure of $P^=$. Unfortunately, determining whether an edge is present in G_v^* requires the construction of a suitable E-GTSP solution (x, y) with $\alpha x = \beta y + \gamma$, hence it is an $\mathcal{NP}$-hard problem in general. In practice, one is interested in finding sufficient conditions for the existence of an edge in G_v^*. We next describe one such condition, related to the work of Naddef and Rinaldi [618] for the graphical TSP, and of Balas and Fischetti [72, 73] for the asymmetric TSP.

Definition 8 *An inequality $\alpha x + \beta y \leq \gamma$ is said to be* Tight-Triangular *(TT, for short) when for all $v \in V$ one has*

$$\beta_v = \max\{\alpha_{iv} + \alpha_{jv} - \alpha_{ij} \ : \ (i,j) \in E \setminus \delta(C_{h(v)})\}.$$

For $v \in V$, we denote by

$$\Delta(v) := \{(i,j) \in E \setminus \delta(C_{h(v)}) \ : \ \beta_v = \alpha_{iv} + \alpha_{jv} - \alpha_{ij}\}$$

the set of the tight *edges for* v.

Recall that a face $\mathcal{H}$ of $P^=$ is called *trivial* when $\mathcal{H} \subseteq \{(x,y) \in \mathbb{R}^{E \cup V} : x_e = 0\}$ for some $e \in E$; *nontrivial* otherwise.

Lemma 9 *Let* $v \in V \setminus W$, *and let* $\alpha x + \beta y \leq \gamma$ *be a valid TT inequality for* $P^=$ *whose* v*-restriction defines a nontrivial face of* $P_v^=$. *Then* G_v^* *contains all the edges in* $\Delta(v)$.

A more sophisticated lifting procedure for the E-GTSP, which allows one to extend any given facet of the TSP polytope to a facet of $P^=$, is given in [299].

We are now ready to study the facial structure of $P^=$.

Theorem 10 $dim(P^=) = |E| - m$.

Proof: Clearly, $\dim(P^=) \leq |E| - m$ as equations (14) and (19) are linearly independent. Hence, it remains to be proved that the dimension of the (improper and nontrivial) face $\mathcal{H}(0,0,0)$ induced by $0x \leq 0y + 0$ is not less than $|E| - m$. We use induction on $\rho = n - m$.

When $\rho = 0$ we have the standard TSP case, and the claim is true.

Assume now the claim holds for $\rho = \bar{\rho}$, and consider any E-GTSP instance with $\rho = \bar{\rho} + 1$. Then there exists a node $v \in V \setminus W$. Because of Lemma 7, we have $\dim(\mathcal{H}(0,0,0)) \geq d_1 + d_2$, where d_1 is the dimension of the face of $P_v^=$ induced by the v-restriction of $0x \leq 0y + 0$, and d_2 is the rank of the v-compatibility graph G_v^* associated with $0x \leq 0y + 0$. By the induction hypothesis, $d_1 \geq |E \setminus \delta(v)| - m$, thus it remains to be shown that $d_2 = |\delta(v)| = |V \setminus C_{h(v)}|$, i.e., that G_v^* is of full rank. But this follows easily from Lemma 9, since $\Delta(v)$ contains all the edges in $E \setminus \delta(C_{h(v)})$ and, therefore, G_v^* is connected and contains an odd cycle (recall that $m \geq 5$ is assumed). ■

Using similar arguments, one can prove the following results.

Theorem 11 *The following inequalities define facets of* $P^=$*:*

(1) $x_e \geq 0$ *for all* $e \in E$.

(2) $x_e \leq 1$ *for all* $e \in E(W)$.

(3) the GSEC (27) whenever one of the following conditions holds:

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

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

(iii) $\eta(S) \geq 3$ *and* $\eta(V \setminus S) \geq 3$,

where $\eta(\cdot)$ has been defined in Subsection 2.1.

(4) The fan inequality (34) for all $w \notin W$.

(5) The GSEC inequality (27) whenever both S and $V \setminus S$ overlap, at least, 3 clusters each, i.e., when $\eta(S) \geq 3$ and $\eta(V \setminus S) \geq 3$.

Note that, because of (34), $y_v \geq 0$ does not define a facet for any $v \in V$. Analogously, the GSEC's (28) and (29) do not define facets of $P^=$. Indeed, a GSEC (29) can be written as $\sum_{e \in E(S)} x_e - \sum_{v \in S \setminus \{i\}} y_v + y_j - 1 \leq 0$. If $C_{h(i)} = C_{h(j)}$, then $y_i + y_j \leq 1$, hence the inequality (29) is a weakening of $\sum_{e \in E(S)} x_e - \sum_{v \in S} y_v \leq 0$ which is, in turn, strictly dominated by the equation $\frac{1}{2} \sum_{v \in S} (\sum_{e \in \delta(v)} x_e - 2y_v) = 0$. Otherwise (29) is dominated by the GSEC (28) written for $S' := S \setminus C_{h(j)}$, i.e., by $\sum_{e \in E(S')} x_e - \sum_{v \in S' \setminus \{i\}} y_v \leq 0$.

Similarly, one can show that a GSEC (28) is dominated by the GSEC (27) written for $S' := S \cup C_{h(i)}$

The bound constraint $x_{(i,j)} \leq 1$ does not define a facet whenever $i \notin W$ or $j \notin W$, since in this case it is dominated by the fan inequality $\sum_{e \in \delta(C_{h(j)}) \cap \delta(i)} x_e \leq y_i$ (if $i \notin W$), or $\sum_{e \in \delta(C_{h(i)}) \cap \delta(j)} x_e \leq y_j$ (if $j \notin W$). In addition, the bound constraints $y_v \leq 1$ never define a facet of $P^=$ because of equations (19).

Finally, the GSEC's (27) not covered by Theorems 11 do not define facets in that $E(S)$ induces a bipartite graph, hence they can be obtained as the sum of certain fan inequalities, as shown in [299].

2.4. Separation algorithms

In this subsection we address the following *separation* (or *identification*) *problem*: Given a (fractional) point $(x^*, y^*) \in [0,1]^{E \cup V}$, find a member $\alpha x + \beta y \geq \gamma$ of a given family $\mathcal{F}$ of valid inequalities for GTSP (or E-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 for the exact/heuristic solution of the problem. In the following we describe the separation algorithms proposed by Fischetti, Salazar and Toth [300].

2.4.1 An exact separation algorithm for GSEC's. We consider the family $\mathcal{F}$ of the generalized subtour elimination constraints, in their cut form (24)–(26). We will assume that node subset $S \subset V$ satisfies $2 \leq |S| \leq n - 2$.

We start with constraints (26):

$$\sum_{e \in \delta(S)} x_e \geq 2\,(y_i + y_j - 1) \quad \text{if } \mu(S) = \mu(V \setminus S) = 0, i \in S, j \in V \setminus S.$$

Suppose nodes i and j have been fixed. Then, finding a most violated inequality (26) calls for the minimum-capacity cut $(S, V \setminus S)$ with $i \in S$ and $j \in V \setminus S$ 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, Magnanti, Orlin [6]). If the maximum flow value is not less than $2(y_i^* + y_j^* - 1)$, then all the inequalities (26) 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 (26) 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 [642] or Chapter 2) by using the Gomory-Hu [390] 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 (26) arises when the chosen i and j are such that $y_i^* = \max\{y_v^* : v \in S\}$ and $y_j^* = \max\{y_v^* : v \in V \setminus S\}$. Therefore, any node s with $y_s^* = \max\{y_v^* : v \in V\}$ 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 $j \in V \setminus \{s\}$. Clearly, nodes j with $y_s^* + y_j^* - 1 \leq 0$ need not be considered.

We now address inequalities (25):

$$\sum_{e \in \delta(S)} x_e \geq 2\,y_i \quad \text{if } \mu(S) = 0, \mu(V \setminus S) \neq 0, i \in S.$$

As before, we assume that cluster C_h and node $i \notin C_h$ are fixed. In this case a most violated constraint (25) corresponds to a minimum-capacity cut $(S, V \setminus S)$ with $i \in S$ and $C_h \subseteq V \setminus S$ 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 $O(m\,n^4)$ time algorithm. Clearly, nodes i with $y_i^* = 0$ need not to be considered.

We now address constraints (24)

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

For all pairs (C_h, C_k) of distinct clusters, a most violated inequality (24) is detected 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 (25) or (26) found by the above described separation algorithm, is not necessarily facet-defining. For (26) this occurs when there exists a cluster C_h contained in S or $V \setminus S$; for (25), 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 favor of its facet-inducing strengthening (24) or (25).

According to the above scheme, the separation algorithm for the overall family $\mathcal{F}$ containing inequalities (24)–(26) requires $O(m\,n^4)$ 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-flows computations.

We now consider the important case in which $y^*_s := \max\{y^*_v : v \in V\} = 1$, 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 (24) with, say, $C_h \subseteq S$ and $C_k \subseteq V \setminus S$ is the same as that associated with inequality (25) written for the same S and for $i = s$. Hence inequalities (24) need not to be considered. Now consider any inequality (25) with $i \neq s$. To fix the ideas, let $i \in S$ and $C_h \subseteq V \setminus S$. 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 (26) written for $j = s$. It follows that inequalities (25) with $i \neq s$ need not be considered. As a result, one has to consider explicitly only the inequalities (25) with $i = s$, and the inequalities (26) (for which $i = s$ can again be assumed).

The reader is referred to [300] for an efficient (parametric) implementation of the above separation procedures, called GSEC_SEP in the sequel.

A heuristic separation algorithm for GSEC's The exact separation algorithm given in the previous subsection can be excessively time consuming. We now outline two faster heuristic procedures.

The first procedure, GSEC_H1, considers the following subset of the inequalities (24):

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

with S containing a cluster C_l of smallest size. For each $h \in \{1, \ldots, m\} \setminus \{l\}$, the procedure computes a most violated inequality (24) with $C_l \subseteq S$ and $C_h \subseteq V \setminus S$ by finding the maximum flow from C_l to C_h. This procedure has $O(m\, n^3)$ time complexity, and typically runs much faster than GSEC_SEP.

Both the exact separation procedure 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 [645, 647] for the 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 GSEC's, 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, let us consider the polytope

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

whose vertices are the incidence vectors of the forests spanning the graph. A node of Q^{SEC} "close" to x^*, say $\tilde{x}$, is found, and the violation of (some of) the subtour elimination constraints defining facets of Q^{SEC} passing through $\tilde{x}$ is checked. To be more specific, $\tilde{x}$ is determined by solving the problem $\max\{x^* x : x \in Q^{SEC}\}$, i.e., by finding a maximum weight spanning tree of G with edge weights $x^*_e \geq 0$, $e \in E$. The classical

greedy algorithm of Kruskal [523] is used, and a check is performed on the violation of the $n-1$ SEC's associated with the subsets $S_i \subset V$, $i = 1, \ldots, n-1$, corresponding to the connected components iteratively found. From matroid theory (see, e.g., Nemhauser and Wolsey [625, page 669]), the SEC's associated with these subsets S_i are the only ones needed to prove the optimality of $\tilde{x}$ (since all other SEC's 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, the above heuristic is an improved version of the one used in Grötschel and Holland [398], that checks the connected components of the subgraph $G' = (V, E(\vartheta))$ for $\vartheta = \min\{x_e^* > 0 : e \in E\}$.

The above scheme can easily be adapted to deal with generalized SEC's, leading to the heuristic separation procedure called GSEC_H2 in [300].

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 [644] 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 the branch-and-cut code makes use of the following simple heuristic, derived from similar TSP procedures [642]. Given the fractional point (x^*, y^*), the subgraph $\tilde{G} = (\tilde{V}, \tilde{E})$ induced by $\tilde{E} := \{e \in E : 0 < x_e^* < 1\}$ is defined. Then, each connected component H of $\tilde{G}$ is considered, in turn, as the handle of a possibly violated generalized 2-matching inequality, whose 2-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.

The 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 [647].

2.5. Heuristic algorithms

A number of known tour construction and tour improvement heuristic algorithms for the TSP (see, e.g., Golden and Stewart [388] or Chapter 8) can be adapted to both GTSP and E-GTSP. We next concentrate on the heuristics producing feasible E-GTSP (and hence GTSP) solutions proposed by Fischetti, Salazar and Toth [300].

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 $d_{hk} := \min\{c_{ij} : i \in C_h, j \in C_k\}$. The procedure starts by choosing the two clusters, say C_a and C_b, that are farthest from each other (with respect to distances d_{hk}), and defines 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 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 2-opt and 3-opt exchanges. Let T be the current E-GTSP solution, visiting exactly one node for each cluster, and let $S \subseteq V$ be the set of the visited nodes. Clearly, any near-optimal TSP solution on the subgraph induced by S (found heuristically through, e.g., 2- or 3-opt exchanges) can lead to an improved GTSP solution. This approach has however the drawback of leaving the set of visited nodes unchanged. In order to remove this restriction, the following generalized 2-opt scheme has been proposed. Let $(\ldots, C_\alpha, C_\beta, \ldots, C_\gamma, C_\delta, \ldots)$ be the cluster sequence corresponding to the current tour T. All the edges of T not incident with the nodes in $C_\alpha \cup C_\beta \cup C_\gamma \cup C_\delta$ are fixed. The scheme tries to exchange the current cluster sequence into $(\ldots, C_\alpha, C_\gamma, \ldots, C_\beta, C_\delta, \ldots)$. To this end, two node pairs (u^*, w^*) and (v^*, z^*) are determined such that

$$\begin{aligned} c_{iu^*} + c_{u^*w^*} + c_{w^*h} &= \min\{c_{ia} + c_{ab} + c_{bh} : a \in C_\alpha, b \in C_\gamma\}, \\ c_{jv^*} + c_{v^*z^*} + c_{z^*k} &= \min\{c_{ja} + c_{ab} + c_{bk} : a \in C_\beta, b \in C_\delta\}, \end{aligned}$$

where nodes i, j, h and k are the nodes visited by T belonging to the clusters preceding C_α, following C_β, and 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.

Moreover, RP1 considers a 3-opt exchange trying to modify the cluster sequence $(\ldots, C_\alpha, C_\beta, C_\gamma, \ldots, C_\delta, C_\varepsilon, \ldots)$ into $(\ldots, C_\alpha, C_\gamma, \ldots, C_\delta, C_\beta, C_\varepsilon, \ldots)$.

We next describe a second refinement procedure, RP2, that proved to be quite effective in our computational study. Let T be the current E-GTSP solution, and let $(C_{h_1}, \ldots, C_{h_m})$ be the sequence in which T goes through the clusters. The 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 acyclic network, LN, having $m+1$ layers corresponding to clusters $C_{h_1}, \ldots, C_{h_m}, C_{h_1}$; see Figure 2.5, where all edges are directed from left to right. 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 the clusters in the same 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 digraphs (e.g., Bang-Jensen and Gutin [84] and Cormen, Leiserson and Rivest [218]).

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).

2.6. A branch-and-cut algorithm

In this subsection we describe the enumerative algorithm for the exact solution of the problem proposed by Fischetti, Salazar and Toth in [300]. Since all the instances considered in the computational study have trian-

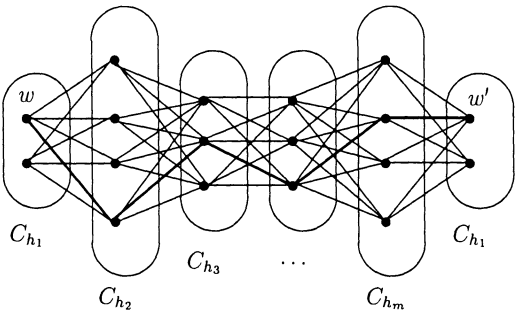

Figure 13.2. The layered network *LN*.

gular costs, we give a rather detailed description of the implementation of the algorithm for the E-GTSP. The algorithm can easily be adapted to the 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 to Padberg and Rinaldi [648] and Jünger, Reinelt and Rinaldi [474] for a thorough description of the technique and to [160] for recent developments. We next describe some important implementation issues, including the best parameter setting resulting from the computational experience.

Lower bound computation At each node of the decision tree, the lower bound is computed by solving the LP problem defined by (13), (14), (19), the bound constraints on the variables, the constraints derived from branching, plus a subset of GSEC's 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 (14), as this would result in a much denser LP coefficient matrix. Then, in an iterative way, the LP is solved, and the computation starts by retrieving the optimal LP basis of the parent node. Some inequalities that are violated by the current LP optimal solution are added. To this end, we applied in sequence the separation procedure for fan inequalities, GSEC_H2, GSEC_H1, and GSEC_SEP. All the violated constraints found (except the fan inequalities) are permanently stored in compact form in a global data structure

called the *constraint pool.* Whenever an inequality introduced in the current branch-node is slack for 5 (say) consecutive LP solutions, it is removed from the LP (but not from the pool). Moreover, whenever the LP-solver takes too long to solve the current LP, all the slack inequalities introduced in the previous nodes are removed.

LAGRANGIAN RELAXATION

At the beginning of the root node computation, Lagrangian optimization is applied with the aim of determining a good subset of constraints for the initial LP, as well as a near-optimal heuristic solution. The following (simplified) model for the E-GTSP, in which the y variables have been projected away through (14), is considered.

$$\min \sum_{e \in E} c_e x_e \tag{35}$$

subject to

$$\sum_{e \in E} x_e = m \tag{36}$$

$$\sum_{e \in \delta(C_h)} x_e = 2 \qquad \text{for } h = 1, \ldots, m \tag{37}$$

$$\sum_{e \in \delta(C_h) \cap \delta(v)} x_e \leq \sum_{e \in \delta(v) \setminus \delta(C_h)} x_e \qquad \text{for } h = 1, \ldots, m;\ v \in V \setminus C_h \tag{38}$$

$$\sum_{e \in E(S)} x_e \leq r - 1 \qquad \text{for } \begin{array}{c} S = \cup_{i=1}^{r} C_{l_i} \\ C_1 \subset V \setminus S \\ 2 \leq r \leq m-2 \end{array} \tag{39}$$

$$x_e \in \{0, 1\} \qquad \text{for } e \in E. \tag{40}$$

Equation (36) is redundant in this formulation. Inequalities (38) and (39) are fan and Basic GSEC's, respectively (notice however that not all GSEC's are included in the model).

The fan inequalities (38), plus the degree constraints (37) for $h \neq 1$, are dualized in a Lagrangian fashion. The Lagrangian relaxed problem calls for $m - 2$ edges (each connecting two different clusters) in $E \setminus \delta(C_1)$ inducing no intra-cluster 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

$$\tilde{c}_{hk} := \min\{c'_{ij} : i \in C_h\,,\ j \in C_k\}, \tag{41}$$

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

ii) compute the min-cost *1-tree* (Held and Karp [445]) 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 (41)).

The computation of near-optimal Lagrangian multipliers is done using classical subgradient optimization. The multipliers are iteratively updated through two nested loops. In the external loop the multipliers for the fan inequalities (38) are updated. With these multipliers fixed, the internal loop adjusts the multipliers for the degree constraints (37) 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, a heuristic E-GTSP solution is determined through the refinement procedure RP2 of Section 2.5, 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. At most 1000 and 50 subgradient iterations in the external and internal loops, respectively, are performed.

ROOT NODE INITIALIZATION

Let λ_h^* and μ_{hj}^* be the best Lagrangian multipliers for constraints (37) and (38), respectively. The initial LP at the root node contains constraints (2), (36), the bound restrictions on the variables, plus the subset of the fan inequalities (34) with $\mu_{hj}^* > 0$. Moreover, the LP contains the Basic GSEC's (39) that were active in the computation of the 1-tree on the shrunken graph with respect to (λ^*, μ^*). To be more specific, the procedure includes in the LP all the constraints (39) whose subset S corresponds to a connected component detected by the Kruskal [523] algorithm used 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.

Upper bound computation At the root node, the farthest insertion, nearest insertion and cheapest insertion procedures are applied, each followed by the tour improvement procedures, as described in Section 2.5. Moreover, as explained above, for each tour among clusters found

during the Lagrangian relaxation a new feasible solution is obtained through procedure RP2. All solutions found are refined through the tour improvement procedures of Section 2.5.

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

Branching Two possibilities for branching are considered: 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. As usual, x_e^* is chosen 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 $S \subset V$ such that $\sum_{e\in\delta(S)} x_e^*$ is not an even integer, and imposing the disjunction

$$\left(\sum_{e\in\delta(S)} x_e \leq 2k\right) \text{ or } \left(\sum_{e\in\delta(S)} x_e \geq 2k+2\right)$$

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

Let $v_1, \ldots, v_m$ be the node sequence corresponding to the current best E-GTSP solution, say $(\tilde{x}, \tilde{y})$, where the subscripts of v are intended to be taken as modulo m. Only a few sets S are considered, namely those obtained as the union of consecutive clusters in the sequence (i.e., of the form $S := C_{h(v_a)} \cup C_{h(v_{a+1})} \cup \ldots \cup C_{h(v_b)}$ for some pair (a, b)), and such that $2 + \varepsilon \leq \sum_{e\in\delta(S)} x_e^* \leq 4 - \varepsilon$ for $\varepsilon = 0.2$. Among these sets S, if any, the one maximizing

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

is chosen, 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 $(\tilde{x}, \tilde{y})$ by exchanging the edge pairs $((v_i, v_{i+1}), (v_j, v_{j+1}))$ and $((v_i, v_j), (v_{i+1}, v_{j+1}))$. $L(S)$ is an estimate on the increase of the cost of the optimal solution (and of the LP lower bound as well) when imposing $\sum_{e\in\delta(S)} x_e \geq 4$. 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 the computational study, the "branching on cuts" strategy (that turned out to be superior) was used, resorting to the "branching on variables" approach when the procedure does not find a suitable set S. Since the heuristic solutions computed at the root node are quite good, a depth-first tree search scheme was implemented (although, in general, this is not the best strategy one can choose).

2.7. Computational results

In this subsection, the computational behaviour of the branch-and-cut algorithm proposed by Fischetti, Salazar and Toth in [300] and described in Section 2.6, is analyzed. The algorithm, implemented in ANSI C, was run on a Hewlett Packard 9000 Series 700 Apollo. As to the LP solver, the package CPLEX 2.1, implementing both primal and dual Simplex algorithms, was used.

The instances of the testbed were obtained by taking all the TSP test problems from the Reinelt TSPLIB library [709] having $137 \leq n \leq 442$. The node clustering has been done so as to simulate geographical regions (using the internal costs as the metric), according to the following procedure. For a given instance, the number of clusters is given by $m := \lceil n/5 \rceil$. Then m *centers* are determined 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.

In addition, for the Grötschel and Holland [398] geographical problems GR137 (America), GR202 (Europe), GR229 (Australia-Asia), and GR431 (Australia-Asia-Europe) the "natural" clustering has been considered, in which clusters correspond to countries. The resulting instances are 35GR137, 31GR202, 61GR229, and 92GR431, respectively.

Tables 13.1 and 13.2 give computational results for the above test problems. Times are given in HP 9000/720 CPU seconds. For each problem, Table 13.1 gives the following information for the root node:

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 2.6;

Lagr-UB : percentage ratio UB/(optimal solution value), where UB is the upper bound value at the end of the Lagrangian relaxation (see Section 2.6);

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

Name	Lagr-LB	Lagr-UB	Lagr-t	basic-LB	r-LB	r-UB	r-time
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
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

Table 13.1. Root node statistics.

basic-LB : percentage ratio LB/(optimal solution value), where LB is the optimal value of the LP relaxation of the simplified model (35)–(40);

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

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

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

According to the table, the upper bound computed using Lagrangian relaxation is quite tight. On the other hand, the quality of the Lagrangian lower bound is rather poor, with an average gap of 11.6%. This is mainly due to the fact that it is derived from the simplified model (35)–(40). Indeed, notice that the best theoretical lower bound

Name	optval	t-time	LP-t	SEP-t	nodes	cuts	fan	GSEC	Gcomb
35gr137	28709	41.5	13.0	6.2	6	296	150	25	8
31gr202	14416	504.7	366.7	46.6	2	1112	325	709	0
61gr229	65508	215.6	109.3	28.5	4	793	355	333	2
92gr431	75616	3365.2	2342.8	278.2	4	2219	713	1172	2
28gr137	35957	96.9	65.0	6.6	0	549	237	257	0
29pr144	45886	8.2	2.5	0.1	0	209	175	7	0
30kroa150	11018	100.3	63.3	8.0	0	594	254	270	0
30krob150	12196	60.6	33.0	5.2	0	511	243	234	0
31pr152	51576	94.8	54.0	6.9	2	574	246	250	5
32u159	22664	146.4	98.0	11.2	2	599	260	288	0
39rat195	854	245.9	167.7	26.3	0	1104	395	634	0
40d198	10557	763.1	571.8	56.9	0	1189	334	750	0
40kroa200	13406	187.4	108.2	27.4	2	710	339	308	4
40krob200	13111	268.5	182.7	23.5	0	947	358	519	0
41gr202	23239	1022.2	764.1	119.3	0	1597	335	1154	0
45ts225	68340	37875.9	34071.0	2481.6	190	8590	499	3089	165
46pr226	64007	106.9	45.4	5.0	0	513	314	139	0
46gr229	71641	1187.5	870.3	132.2	2	1428	393	873	14
53gil262	1013	6624.1	5342.7	943.4	16	2676	516	1427	27
53pr264	29549	337.0	204.6	34.4	0	1016	479	407	0
60pr299	22615	812.8	583.9	43.3	0	1358	536	685	0
64lin318	20765	1671.9	1038.8	292.6	10	1680	585	867	1
80rd400	6361	7021.4	4721.7	1658.3	2	3092	762	1852	0
84fl417	9651	16719.4	12232.3	2964.2	0	5102	962	3806	0
87gr431	101523	31544.6	24873.2	4540.0	2	4354	672	2937	5
88pr439	60099	5422.8	3636.5	896.9	0	2979	778	1918	0
89pcb442	21657	58770.5	43753.5	12712.1	46	9427	949	4591	38

Table 13.2. Branch-and-cut statistics.

for the Lagrangian relaxation equals the optimal value of the LP relaxation of model (35)–(40). The latter value was computed through a simplified version of the cutting plane algorithm, and is reported in the table (column *basic-LB*). It can be seen that the improvement with respect to the Lagrangian lower bound is negligible.

Table 13.2 shows the performance of the overall enumerative algorithm. For each problem the table gives:

Name : the problem name;

optval : value of the optimal solution;

t-time : 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);

cuts : total number of cuts generated, including those found by the Lagrangian initialization (Section 2.6) and those recovered from the pool;

fan : total number of fan inequalities generated;

GSEC : total number of GSEC's found by the heuristic procedures GSEC_H1 and GSEC_H2 of Section 2.4.1.

Gcomb : total number of generalized comb inequalities generated.

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 50% of the cases, no branching is needed. The results also show that natural clustering produces easier instances than those obtained through the clustering procedure.

As to procedure GSEC_SEP, it never found violated cuts, with the only exception of instance 45TS225 for which 9 cuts were detected. This proves the effectiveness of the heuristic separations for GSEC's. The inequalities which are most frequently recovered from the pool are the GSEC's (24).

In order to evaluate the effect of different clusterizations of the nodes, a second clustering procedure has also been considered 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 13.3. Let $xmin$, $xmax$, $ymin$ and $ymax$ be the minimum and maximum x- and y-coordinates, respectively. The procedure considered the rectangle whose vertices have coordinates $(xmin, ymin)$, $(xmin, ymax)$, $(xmax, ymax)$, and $(xmax, ymin)$, and subdivided it so as to obtain an $NG \times NG$ grid in which each cell has edges of length $(xmax - xmin)/NG$ and $(ymax - ymin)/NG$. 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(NG)* be the number of nonempty clusters corresponding to the $NG \times NG$ grid, and define NG as the minimum integer such that *CLUSTER(NG)*$\geq n/\mu$.

Table 13.3 gives, for each test problem and value of $\mu = 3, 5, 10$, 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 13.3 (for $\mu = 5$) and Table 13.2 shows that the grid clusterization produces harder instances. No correlation exists, instead,

Name	$\mu = 3$			$\mu = 5$			$\mu = 10$		
	t-time	nodes	m	t-time	nodes	m	t-time	nodes	m
gr137	81.7	0	46	94.7	0	28	972.0	0	14
pr144	28.8	0	48	25.3	0	30	21.8	0	16
kroa150	56.3	2	57	72.5	0	36	111.2	0	16
krob150	40.5	0	56	100.0	2	36	79.2	0	16
pr152	68.7	4	54	455.2	42	33	35.5	0	16
u159	36.5	0	58	154.3	0	38	107.4	0	23
rat195	84.4	2	81	1409.5	18	49	423.5	0	25
d198	539.5	0	67	591.2	0	40	2849.9	0	25
kroa200	207.2	2	72	1696.3	30	47	339.5	0	25
krob200	2031.9	54	76	119.1	0	48	422.2	0	25
gr202	2644.3	34	73	727.4	2	43	450.1	0	21
ts225	453.4	14	75	–	–	45	10601.5	0	25
pr226	130.7	0	78	65.6	0	50	105.7	0	24
gr229	312.0	0	80	1895.1	8	46	9391.9	2	23
gil262	5674.6	104	96	10763.1	42	63	1141.0	0	36
pr264	747.1	16	101	109.6	0	55	376.8	0	27
pr299	761.3	2	102	4629.9	12	69	2730.6	0	35
lin318	37117.4	220	108	16784.8	40	64	71010.7	26	36
rd400	21764.6	118	135	87308.1	198	81	21156.8	2	49
fl417	7687.9	0	142	10373.7	0	93	919.2	0	43
pr439	1905.7	6	163	18876.5	14	96	35652.6	8	48
pcb442	23226.1	86	155	39155.3	24	96	15266.6	0	48

Problem ts225 with $\mu = 5$ required more than 100,000 CPU seconds.

Table 13.3. Some computational results with different clusterizations.

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 for our families of instances. All the test problems in the 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 13.3). Moreover, the heuristic algorithms proposed allow one to compute very good solutions within short computing time. As shown in Table 13.1, after the Lagrangian phase the average percentage error with respect to the optimum is 0.9% (see column *Lagr-UB*), and 0.2% at the end of the root node (see column *r-UB*).

3. The Orienteering Problem

As stated in the introduction, we are given a set of n nodes, each having an associated nonnegative *prize* p_v, and a distinguished "depot" node, say node 1. Let $t_{(i,j)}$ be the *time* spent for routing nodes i and j in

sequence. The *Orienteering Problem* (OP) is to find a cycle C through node 1 whose total duration $t(C)$ does not exceed a given bound t_0, and visiting a node subset with a maximum total prize. Without loss of generality, we can assume that cycle C contains at least three nodes.

The problem can be formulated as

$$v(\text{OP}) \equiv \max \sum_{v \in V} p_v y_v \tag{42}$$

subject to

$$\sum_{e \in E} t_e x_e \leq t_0, \tag{43}$$

$$x(\delta(v)) = 2\, y_v \qquad \text{for } v \in V, \tag{44}$$

$$x(\delta(S)) \geq 2\, y_v \qquad \text{for } S \subset V, 1 \in S, v \in V \setminus S, \tag{45}$$

$$y_1 = 1, \tag{46}$$

$$x_e \in \{0,1\} \qquad \text{for } e \in E, \tag{47}$$

$$y_v \in \{0,1\} \qquad \text{for } v \in V \setminus \{1\}, \tag{48}$$

Because of the degree constraints (44), inequalities (45) can equivalently be written as

$$x(E(S)) \leq y(S) - y_v \quad \text{for } S \subset V, 1 \in S, v \in V \setminus S \tag{49}$$

and

$$x(E(\bar{S})) \leq y(\bar{S}) - y_v \quad \text{for } \bar{S} \subset V, 1 \in V \setminus \bar{S}, v \in \bar{S}. \tag{50}$$

Notice that the inequalities (16), although valid, are dominated by (45) as $y_i - 1 \leq 0$ for all $i \in V$.

This section is mainly based on the results given by Fischetti, Salazar and Toth in [301]. Section 3.1 discusses a number of additional constraints, which improve the quality of the LP relaxation of the basic model. We also analyze a family of *conditional cuts*, i.e., cuts which cut off the current optimal solution. Separation procedures are described in Section 3.2, whereas Section 3.3 presents heuristic algorithms for finding approximate OP solutions. An overall branch-and-cut algorithm is described in Section 3.4. In that section, an effective way of integrating conditional cuts within the overall framework is also presented. Extensive computational results on several classes of test problems involving up to 500 nodes are presented in Section 3.5.

3.1. Additional inequalities

In this section we describe five classes of additional inequalities for OP. These inequalities are capable of strengthening the LP-relaxation

of model (42)–(48). The first two classes do not rely on the total time restriction (43), and are derived from the *cycle relaxation* of OP [79, 92]. The remaining classes, instead, do exploit the total time restriction.

A polyhedral analysis of the OP appears very difficult, and to our knowledge it has not been addressed in the literature. From the practical point of view, however, these cuts proved to be of fundamental importance for the solution of most instances.

Logical constraints Clearly, $x_e = 1$ for some $e \in \delta(j)$ implies $y_j = 1$. Hence the *logical constraints*

$$x_e \leq y_j \text{ for all } e \in \delta(j), j \in V \setminus \{1\} \tag{51}$$

are valid for OP. Whenever $e = (v, j) \notin \delta(1)$, inequality (51) is a particular case of (50) arising for $\bar{S} = \{v, j\}$. On the other hand, for $e \in \delta(1)$ these inequalities do improve the LP relaxation of model (42)–(48). To see this, consider the fractional point (x^*, y^*) with $x^*_{12} = x^*_{13} = 1$, $y^*_1 = 1$, $y^*_2 = y^*_3 = 1/2$ (all other components being 0). Assuming $t_{12} + t_{13} \leq t_0$, this point satisfies all the constraints of the relaxation, but not the constraints (51) associated with $e = (1, 2)$ and $j = 2$, and with $e = (1, 3)$ and $j = 3$.

We observe that the addition of (51) to model (42)–(48) makes the integrality requirement on the y-variables redundant. Indeed, let (x^*, y^*) be any point satisfying (43)–(47) and $0 \leq y_v \leq 1$ for all $v \in V$, and define $T^* := \{e \in E : x^*_e = 1\}$. Then from (44) we have $y_v = |T^* \cap \delta(v)|/2$ for all $v \in V$, i.e., $y_v \in \{0, 1/2, 1\}$. But $y_v = 1/2$ would imply $T^* \cap \delta(v) = \{e\}$ for some $e \in \delta(v)$, which is impossible since in this case the corresponding logical constraint (51) would be violated.

2-matching inequalities The well-known *2-matching constraints* for the TSP have the following counterpart in the cycle relaxation of OP:

$$x(E(H)) + x(T) \leq y(H) + \frac{|T| - 1}{2}, \tag{52}$$

where $H \subset V$ is called the *handle*, and $T \subset \delta(H)$ is a set with $|T| \geq 3$, $|T|$ odd, pairwise disjoint *teeth*. This inequality is obtained by adding up the degree constraints for all $v \in H$ and the bound constraints $x_e \leq 1$ for all $e \in T$, dividing by 2, and then rounding down all the coefficients to the nearest integer.

Cover inequalities The total time constraint (43), along with the requirements $x_e \in \{0, 1\}$ for $e \in E$, defines an instance of the *0-1 Knapsack Problem* (KP), in which items correspond to edges. Therefore, every

valid KP inequality can be used to hopefully improve the LP relaxation of the OP model. Among the several classes of known KP inequalities, let us consider the *cover inequality* (see, e.g., Nemhauser and Wolsey [625]):

$$x(T) \leq |T| - 1, \tag{53}$$

where $T \subseteq E$ is an inclusion-minimal edge subset with $\sum_{e \in T} t_e > t_0$. This constraint stipulates that not all the edges of T can be selected in a feasible OP solution.

A cover inequality can in some cases be strengthened. In particular, one can easily obtain the valid *extended* inequality

$$x(T \cup Q) \leq |T| - 1, \tag{54}$$

where $Q := \{e \in E \setminus T : t_e \geq \max_{f \in T} t_f\}$.

A different improvement is next proposed, which exploits the fact that the selected edges have to define a cycle. The improvement can only be applied in case T defines an infeasible cycle passing through node 1, and leads to the *cycle cover inequality*:

$$x(T) \leq y(V(T)) - 1. \tag{55}$$

Validity of (55) follows from the easy observation that $x(T) \geq y(V(T))$ would imply $x_e = 1$ for all $e \in T$. Figure 3.1 shows a fractional point violating a cycle cover inequality but not other previous inequalities. More generally, (55) is a valid inequality whenever T does not contain any feasible cycle. This generalization will be studied in the forthcoming subsection on conditional cuts.

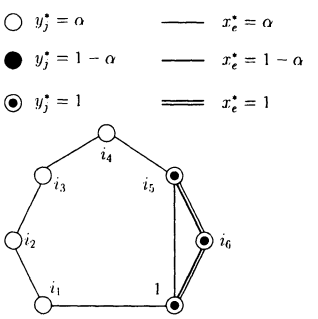

Figure 13.3. Fractional point violating a cycle cover inequality for the OP instance with t_0=6 and t_e=1 for all $e \in E$ ($0 < \alpha \leq 1/2$). Here $T = \{(1, i_1), (i_1, i_2), \ldots, (i_6, 1)\}$, $x(T) = 2 + 5\alpha$, and $y(V(T)) = 3 + 4\alpha$.

Path inequalities The previous classes of additional inequalities (except the cycle cover inequalities) are based either on the cycle or on the knapsack relaxation of the problem. We next introduce a new family of constraints that exploit both relaxations.

Let $P = \{(i_1, i_2), (i_2, i_3), \ldots, (i_{k-1}, i_k)\}$ be any simple path through $V(P) = \{i_1, \ldots, i_k\} \subseteq V \setminus \{1\}$, and define the nodeset:

$$W(P) := \{v \in V \setminus V(P) : P \cup \{(i_k, v)\} \text{ can be part of a feasible OP sol.}\}.$$

We allow P to be infeasible, in which case $W(P) = \emptyset$. Then the following *path inequality*

$$\sum_{j=1}^{k-1} x_{i_j i_{j+1}} - \sum_{j=2}^{k-1} y_{i_j} - \sum_{v \in W(P)} x_{i_k v} \leq 0 \tag{56}$$

is valid for OP. Indeed, suppose there exists a feasible OP solution (x^*, y^*) violating (56). Then

$$x^*_{i_1 i_2} + (x^*_{i_2 i_3} - y^*_{i_2}) + \ldots + (x^*_{i_{k-1} i_k} - y^*_{i_{k-1}}) - \sum_{v \in W(P)} x^*_{i_k v} \geq 1,$$

where $x^*_{i_j i_{j+1}} - y^*_{i_j} \leq 0$ for all $j = 2, \ldots, k-1$. It then follows that $x^*_{i_1 i_2} = 1$ (hence $y^*_{i_2} = 1$), $x^*_{i_2 i_3} - y^*_{i_2} = 0$ (hence $x^*_{i_2 i_3} = 1$ and $y^*_{i_3} = 1$), $\ldots$, $x^*_{i_{k-1} i_k} - y^*_{i_{k-1}} = 0$ (hence $x^*_{i_{k-1} i_k} = 1$), and $x^*_{i_k v} = 0$ for all $v \in W(P)$. But then solution (x^*, y^*) cannot be feasible, since it contains all the edges of P, plus an edge (i_k, w) with $w \notin W(P)$.

Figure 13.3 shows a typical fractional point that is cut off by a path inequality. This point can be viewed as the convex combination of two cycles, one of which is infeasible because of the total time requirement.

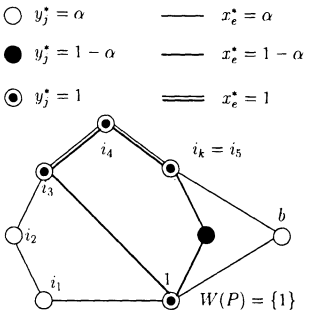

Figure 13.4. A fractional point violating a path inequality for the OP instance with t_0=6 and t_e=1 for all $e \in E$ ($0 < \alpha \leq 1/2$).

Recall that, for any given $F \subseteq E$, $t(F)$ stands for $\sum_{e \in F} t_e$. The definition of $W(P)$ amounts to checking for each $v \in V \setminus V(P)$ whether there exists a cycle of the form $C = P_1 \cup (P \cup \{i_k, v\}) \cup P_2$, where P_1 and P_2 are node-disjoint paths from 1 to i_1 and v, respectively, such that $t(P_1) + t(P) + t_{i_k v} + t(P_2) \leq t_0$. A simpler condition (producing a possibly larger set $W(P)$, and hence a weakened inequality (56)) is obtained by removing the requirement that P_1 and P_2 share no node (except node 1). This leads to the alternative definition of $W(P)$ as

$$W(P) := \{v \in V \setminus V(P) : d(1, i_1) + t(P) + t_{i_k v} + d(1, v) \leq t_0\}, \quad (57)$$

where for each $j \in V \setminus \{1\}$, $d(1, j)$ gives the total time associated with the shortest path from node 1 to node j.

Conditional cuts We next address inequalities that are not guaranteed to be valid for our problem, but can nevertheless be used in a cutting plane context.

Suppose that a heuristic OP solution of value (say) LB is available. In the process of finding an optimal OP solution we are clearly interested in finding, if any, a feasible solution of value strictly better than LB. Therefore, any inequality can be exploited as a cutting plane, provided that it is satisfied by every feasible OP solution of value greater than LB. These inequalities are called *conditional cuts*.

Let us consider a general family of inequalities of the type

$$x(T) \leq y(V(T)) - 1, \quad (58)$$

where $T \subset E$ is chosen in an appropriate way. It can be seen easily that $x(T) \leq y(V(T))$ holds for every feasible solution, no matter how T is chosen. Moreover, $x(T) = y(V(T))$ implies that the OP solution consists of a cycle entirely contained in T. It then follows that (58) can be used as a conditional cut, provided that no feasible OP solution of value strictly greater than LB is contained in T. This occurs, in particular, when

$$T = E(S) \text{ for some } S \subset V \text{ such that } 1 \in S \text{ and } \sum_{v \in S} p_v \leq \text{LB}. \quad (59)$$

A different approach for defining conditional cuts, based on enumeration, will be described in the following section.

3.2. Separation algorithms

In this section we outline exact and/or heuristic algorithms, proposed by Fischetti, Salazar and Toth [301], for the following *separation problem*: Let $\mathcal{F}$ be one of the families of OP inequalities described in Section

3.1; given a point $(x^*, y^*) \in [0,1]^{E \times V}$ which satisfies (43)–(44), find a member $\alpha x + \beta y \leq \gamma$ of $\mathcal{F}$ which is (mostly) violated by (x^*, y^*), if any.

We denote by $G^* = (V^*, E^*)$ the *support graph* associated with the given (x^*, y^*), where $V^* := \{v \in V : y_v^* > 0\}$ and $E^* := \{e \in E : x_e^* > 0\}$.

Cut inequalities (45) Let x_e^* be viewed as a capacity value associated with each edge $e \in E^*$. For any fixed node $v \in V^* \setminus \{1\}$, a most violated inequality (45) (among those for the given v) is determined by finding a minimum-capacity $(1, v)$-cut, say $(S_v, V^* \setminus S_v)$, on G^*. This requires $O(|V^*|^3)$ time, in the worst case, through a max-flow algorithm. Trying all possible $v \in V^* \setminus \{1\}$ then leads to an overall $O(|V^*|^4)$-time separation algorithm.

The nodes v are considered in decreasing order of the associated y_v^*. Whenever a violated inequality (45) is found for (say) the pair S_v and v, the capacity x_{1v}^* is increased by the quantity $2 - x^*(\delta(S_v))$. This prevents the cut $(S_v, V^* \setminus S_v)$ from being generated again in the subsequent iterations. Moreover, in order to increase the number of violated inequalities detected by a single max-flow computation, two minimum-capacity $(1, v)$-cuts are considered for each v, namely $(S'_v, V^* \setminus S'_v)$ and $(V^* \setminus S_v, S_v)$, where S_v (respectively, S'_v) contains the nodes connected to node v (respectively, node 1) in the incremental graph corresponding to the maximum flow vector. Nodeset S_v gives an hopefully violated inequality (45), whereas S'_v is used, as explained later, for producing a conditional cut.

Logical constraints (51) This family can be dealt with by complete enumeration, with an overall $O(|E^*|)$ time complexity.

2-matching constraints (52) These inequalities can be separated in polynomial time through a simple modification of the Padberg and Rao [644] odd-cut separation scheme. In order to reduce the computational effort spent in the separation, however, the following simple heuristic can be implemented. Values x_e^* are interpreted as weights associated with the edges. The greedy algorithm of Kruskal is applied to find a minimum-weight spanning tree on G^*. At each iteration in which this algorithm selects a new edge e, the connected component which contains e, say H, is determined (in the subgraph of G^* induced by all the edges selected so far). The nodeset H is then considered as the handle of an hopefully violated 2-matching constraint. In this way, the procedure generates efficiently all the connected components H of the subgraph $G_\theta = (V, E_\theta)$ induced by $E_\rho := \{e \in E : 0 < x_e^* < \theta\}$ for *every*

possible threshold θ. These sets H have high probability of being the handle of a violated 2-matching constraint, if one exists. For any H, tooth edges are determined, in an optimal way, through the following greedy procedure. Let $\delta(H) = \{e_1, \ldots, e_p\}$ with $x^*_{e_1} \geq x^*_{e_2} \geq \ldots \geq x^*_{e_p}$. The requirement that the teeth have to be pairwise disjoint is initially relaxed. For any given $|T| \geq 3$ and odd, the best choice for T consists of the edges $e_1, \ldots, e_{|T|}$. Therefore, a most violated inequality corresponds to the choice of the odd integer $|T| \geq 3$ which maximizes $x^*_{e_1} + (x^*_{e_2} + x^*_{e_3} - 1) + \ldots + (x^*_{e_{|T|-1}} + x^*_{e_{|T|}} - 1)$. If no violated cut can be produced in this way, then clearly no violated 2-matching constraint exists for the given handle. Otherwise a violated 2-matching constraint exists in which two tooth edges, say e and f, may overlap in a node, say v. In this case, the inequality is *simplified* by defining a new handle-tooth pair (H', T') with $T' := T \setminus \{e, f\}$, and $H' := H \setminus \{v\}$ (if $v \in H$) or $H' := H \cup \{v\}$ (if $v \notin H$). It is then easy to see that the inequality (52) associated with this new pair (H', T') is at least as violated as that associated with the original pair (H, T). Indeed, replacing (H, T) with (H', T') increases the violation by, at least, $1 + y_v - x(\delta(v)) \geq 2y_v - x(\delta(v)) = 0$ (if $v \in H$), or $1 - y_v \geq 0$ (if $v \notin H$). By iterating this simplification step one can then always detect a violated 2-matching constraint with non overalapping teeth. In some cases this procedure could even lead to a 2-matching constraint with $|T| = 1$; if this occurs, the inequality is rejected in favour of an inequality (45) associated with the handle.

Path inequalities (56) Let us assume that the fractional point (x^*, y^*) satisfies all logical constraints (51), and observe that the path inequality associated with a given path P cannot be violated by (x^*, y^*) if $x^*_{i_h i_{h+1}} = 0$ for some $(i_h, i_{h+1}) \in P$. This follows from the fact that (56) can be rewritten as

$$\sum_{j=1}^{h-1}(x_{i_j i_{j+1}} - y_{i_{j+1}}) + x_{i_h i_{h+1}} + \sum_{j=h+1}^{k-1}(x_{i_j i_{j+1}} - y_{i_j}) - \sum_{v \in W(P)} x_{i_k v} \leq 0$$

where all terms involved in the first two summations are nonpositive by assumption. Hence every violated path inequality must be associated with a path P contained in the support graph G^*. Since this graph is usually very sparse, a simple enumeration scheme can be implemented to detect the path P producing a most violated path inequality. The procedure starts with an empty node sequence P. Then, iteratively, the current P is extended in any possible way, and the associated path inequality is checked for violation. Whenever for the current path $P =$

$\{(i_1, i_2), \ldots, (i_{k-1}, i_k)\}$

$$\sum_{j=1}^{k-1} x_{i_j i_{j+1}} - \sum_{j=2}^{k-1} y_{i_j} \leq 0,$$

holds, a backtracking step is performed, since no extension of P can lead to a violated cut.

Cover inequalities (54)–(55) We first address the separation problem for cover inequalities, in their weakest form (53), which calls for an edgeset T with $\sum_{e \in T} t_e > t_0$ which maximizes $x^*(T) - |T| + 1$. It is well known that this problem can be formulated as

$$\sigma^* := \min \sum_{e \in E} (1 - x_e^*) z_e \tag{60}$$

subject to

$$\sum_{e \in E} t_e z_e \geq t_0 + 1, \tag{61}$$

$$z_e \in \{0, 1\} \qquad \text{for } e \in E. \tag{62}$$

Although $\mathcal{NP}$-hard, the knapsack problem (60)–(62) can typically be solved within short computing time by means of specialized codes (see Martello and Toth [586]). Moreover, all variables z_e with $x_e^* = 0$ can be fixed to 0, because $z_e = 1$ would imply $\sigma^* \geq 1$. Analogously, one can set $z_e = 1$ whenever $x_e^* = 1$, since in this case its weight in (60) vanishes.

If $\sigma^* \geq 1$ then no violated cover inequality (53) exists. Otherwise, $T := \{e \in E : z_e = 1\}$ gives a most violated such cut. In both cases, it is worth checking the extended cover inequalities (54) associated with T for violation. Notice that, because of the fact that some weights in (60) can be zero, the edgeset T which gives the optimum in (60) is not guaranteed to be minimal with respect to property (61). Therefore, in order to have a stronger inequality one can make T minimal (in a greedy way) before checking the extended inequality (54) for violation.

A heuristic separation algorithm for the cycle cover inequalities (55), associated with an infeasible cycle T, is now outlined. The heuristic is intended to produce several candidate cycles T with large value of $x^*(T)$. To this end, the values x_e^* are interpreted as weights associated with the edges, and a maximum-weight spanning tree on G is computed. The edges $e \in E^*$ not in the tree are then consider, in turn: if the addition of e to the tree induces a cycle T passing through node 1 and such that $\sum_{e \in T} t_e > t_0$, then a valid inequality (55) is obtained, that is checked for violation.

Conditional cuts (58) Two heuristic separation procedures for conditional cuts have been implemented. Let LB be the value of the current best solution available.

The first procedure is based on condition (59), and is embedded within the max-flow separation algorithm for inequalities (45) described earlier. For each set S'_v therein detected which satisfies $\sum_{v \in S'_v} p_v \leq LB$ and $1 \in S'_v$, the procedure sets $T = E(S'_v)$ and checks (58) for violation.

The second procedure is based on the observation that (58) can *always* be used as a conditional cut, provided that the lower bound value LB is updated by taking into account all the feasible OP solutions entirely contained in T. This amounts to computing

$$\text{LB} := \max\{\text{LB}, v(\text{OP}_T)\},$$

where $v(\text{OP}_T)$ is the optimal OP value when $x_e = 0$ is imposed for all $e \in E \setminus T$. Although the computation of $v(\text{OP}_T)$ requires exponential time in the worst case, for a sufficiently sparse edge set T it is likely that even a simple complete enumeration scheme can succeed in determining $v(\text{OP}_T)$ within short computing time. The procedure defines $T := E^*$, hence ensuring that the corresponding conditional cut (58) is violated since $x^*(T) = x^*(E)$ and $y^*(V(T)) = y^*(V)$, where $x^*(E) = y^*(V)$ because of the degree equations (44). A simple algorithm for solving OP, based on complete enumeration, is then applied on the support graph G^*. If the enumeration ends within a fixed time-limit TL then, after the updating of LB, (58) is guaranteed to be a valid conditional cut to be added to the current LP.

3.3. Heuristic algorithms

The performance of the enumerative exact algorithms improves if one is capable of early detecting "good" feasible OP solutions. To this end, Fischetti, Salazar and Toth [301] proposed the following heuristic procedure, working in two stages. In the first stage, a feasible cycle C is detected, which is likely to contain a large number of edges belonging to an optimal solution. In the second stage, refining procedures are applied to derive from C a better feasible circuit. The method is along the same lines as the heuristic proposed by Ramesh and Brown [692], but uses LP information to guide the search. A brief outline follows.

On input of the first stage, the heuristic receives, for each edge $e \in E$, an estimate w_e, $0 \leq w_e \leq 1$, of the probability of having edge e in an optimal solution. The computation of values w_e is described in Section 3.4. The edges are sorted in decreasing order of w_e, with ties broken so as to rank edges with smaller time t_e first. Then an edge subset T

containing a family of node-disjoint paths with large probability of being part of an optimal solution, is heuristically detected in a greedy way. To be specific, the procedure initializes $T := \emptyset$ and then considers, in turn, each edge e according to the given order: If $T \cup \{e\}$ contains a node with degree larger than 2, the edge is rejected; otherwise T is updated as $T := T \cup \{e\}$ if $T \cup \{e\}$ is cycle-free, else the algorithm is stopped.

Starting with T, the required feasible cycle C is obtained by means of the following steps. First, all the nodes in $V \setminus \{1\}$ that are not covered by T are removed from the graph. Then, the paths of T are linked into a cycle, C, passing through node 1. To this end, a simple nearest-neighbor scheme is applied which starts from node 1, and iteratively moves to the nearest uncovered extreme node of a path in T. At the end of this phase, a check on $\sum_{e \in C} t_e \leq t_0$ is performed. If the condition is not satisfied, the following procedure is applied to make C feasible. For any given node v covered by C, let i_v and j_v denote the two neighbors of v in C. The procedure iteratively removes a node v from C, i.e., replaces (i_v, v) and (j_v, v) with the short-cut (i_v, j_v). At each iteration v is chosen (if possible) as a minimum-prize node whose removal makes C feasible, or else as a node that minimizes the score $p_v/(t_{i_v v} + t_{j_v v} - t_{i_v j_v})$.

In the second stage of the heuristic, the procedure receives as input the feasible cycle C computed in the first stage, and iteratively tries to improve it. At each iteration, 2-optimality edge exchanges inside C are first performed, so as to hopefully reduce its total time. Then an attempt is performed to add to C a maximum-prize node belonging to the set $Q(C)$ containing the nodes v not covered by C, and such that $\min_{(i,j) \in C}\{t_{iv} + t_{jv} - t_{ij}\} \leq t_0 - \sum_{e \in C} t_e$. If $Q(C) \neq \emptyset$, the node insertion is performed and the step is repeated. Otherwise, the whole procedure is re-applied on the cycle obtained from C by removing, in turn, one of its nodes.

3.4. A branch-and-cut algorithm

We next outline the main ingredients of the branch-and-cut algorithm proposed by Fischetti, Salazar and Toth [301] for the optimal solution of OP.

The initialization phase At the root node of the branch-decision tree, a lower bound on the optimal OP value is computed through the heuristic algorithm of Section 3.3, with edge weights $w_e = 0$ for all $e \in E$. In addition, the first Linear Program (LP) to be solved is set-up by taking:

1 all variables y_v, $v \in V$;

2 the variables x_e associated with edges belonging to the initial heuristic solution;

3 for all $v \in V$, the variables x_e associated with the 5 smallest-time edges $e \in \delta(v)$;

4 the total time constraint (43);

5 the n degree equations (44);

6 the lower and upper bounds on the variables.

Finally, the constraint pool (i.e., the data structure used to save the OP constraints that are not included in the current LP) is initialized as empty.

The cutting plane phase At each node of the branch-decision tree, the procedure determines the optimal primal and dual solutions of the current LP, say (x^*, y^*) and u^*, respectively —in case the current LP reveals infeasible, it introduces artificial variables with very large negative prize. Notice that the value of the primal solution, namely $\sum_{v \in V} p_v y_v^*$, is not guaranteed to give an upper bound on the optimal OP value, as the current LP contains only a subset of the x-variables. Then the so-called *pricing phase* is entered, in which the dual solution u^* is used to compute the reduced cost $\bar{c}_e$ of the variables x_e that are not part of the current LP, which are by default set to 0. The variables x_e which price-out with the wrong sign (i.e., $\bar{c}_e > 0$), are added to the LP, which is then re-optimized with the primal simplex algorithm. In order to keep the size of the LP as small as possible, the procedure never adds more than 100 variables at each round of pricing (chosen among those with largest reduced costs). The pricing loop is iterated until all variables price-out correctly, i.e., until the current LP value, say UB, is guaranteed to be an upper bound on the optimal OP value. In this case, if the current node is not fathomed the following purging phase is entered. Let LB denote the value of the best OP solution known so far. The variables x_e with $\lfloor \mathrm{UB} + 4\bar{c}_e \rfloor \leq \mathrm{LB}$, along with the constraints that have been slack in the last 5 iterations, or whose slack exceeds 0.01, are removed from the current LP. Moreover, at the root branch-decision node, all the variables x_e with $\lfloor \mathrm{UB} + \bar{c}_e \rfloor \leq \mathrm{LB}$ are fixed to 0, and all the variables with $\lfloor \mathrm{UB} - \bar{c}_e \rfloor \leq \mathrm{LB}$ are fixed to 1 (this latter condition may only apply to LP variables at their upper bound).

The *separation phase* is next entered, in which constraints violated by (x^*, y^*) are identified and added to the current LP. The separation algorithms described in Section 3.2 are applied. The constraint pool is first searched. Then the procedure checks, in sequence, the logical constraints (51), the inequalities (45), the 2-matching constraints (52), the

cover inequalities (54)–(55), the path inequalities (56), and the conditional cuts (58). The time limit TL used for the enumeration required in the conditional cut separation, is set to 5·TS, where TS is the computing time so far spent in the last round of separation. Whenever a separation procedure succeeds in finding violated cuts, the separation sequence is stopped, and all the cuts found are added to the LP.

In order to reduce tailing off phenomena, a branching is performed whenever the upper bound did not improve by at least 0.001 in the last 10 cutting-plane iterations of the current branching node.

At every fifth application of the separation algorithm, an attempt is performed to improve the current best OP solution through the heuristic algorithm described in Section 3.3. The values w_e required by the algorithm are set to x_e^* for all $e \in E$. This choice computationally proved very effective and typically produces very tight approximate solutions. An additional heuristic is embedded within the second separation procedure for conditional cuts (58), as described in Section 3.2. Indeed, the enumeration of the OP solutions contained in the support graph of x^*, therein required, can in some cases improve the current LB.

The branching step Whenever a branch-decision node cannot be fathomed, a branching step is performed, in a traditional way, by fixing $x_f = 0$ or $x_f = 1$ for a variable x_f chosen as follows. The 15 fractional variables x_e with x_e^* closest to 0.5 are selected. For each such candidate branching variable x_e, two values, say UB_e^0 and UB_e^1, are computed by solving the current LP amended by the additional constraint $x_e = 0$ and $x_e = 1$, respectively. Then, the actual branching variable x_f is chosen as the one that maximizes the score $0.75 \cdot \mathrm{UB}_e^0 + 0.25 \cdot \mathrm{UB}_e^1$.

The overall algorithm At the root node of the branch-decision tree, the initialization phase and the cutting-plane phase are executed. When all separation algorithms fail and the current node is not fathomed, a branching step is performed. However, for the root node only, the following alternative scheme is executed.

According to computational experience, the conditional cut associated with the support graph $G^* = (V(E^*), E^*)$ of the current LP solution (x^*, y^*), namely

$$x(E^*) \leq y(V(E^*)) - 1, \tag{63}$$

is quite effective in closing the integrality gap. Unfortunately, for rather dense G^* the simple enumeration scheme described in Section 3.2 is unlikely to complete the enumeration of all possible OP solutions contained in G^*, within the short time limit allowed. Nevertheless, cut (63) is added to the LP even when this enumeration fails (in this case the

cut is called a *branch cover cut*). This choice may however cut off the optimal OP solution as well, if this solution is contained in G^*. This possibility can be taken into account by storing the graph G^*, with the aim of dealing with it at a later time. With the branch cover cut added to the LP, the cutting plane phase is then re-entered until again all separations fail. Then, if needed, the whole scheme is iterated: the procedure adds a new branch cover cut, stores the current support graph G^*, and re-enters the cutting plane phase.

In this way, a sequence of support graphs, say $G_i^* = (V(E_i^*), E_i^*)$ for $i = 1, \ldots, k$, are produced and stored until the root node is fathomed. At this point, the computation is not over, as it is necessary to consider the best OP solution within each graph $G_1^*, \ldots, G_k^*$ or, alternatively, within the "union" of these graphs, defined as $\tilde{G} = (V(\tilde{E}), \tilde{E} := \cup_{i=1}^{k} E_i^*)$. To this end, all the branch cover cuts are removed from the constraint pool, and the branch-and-cut algorithm is re-applied on the OP instance associated with $\tilde{G}$. In order to guarantee the convergence of the overall algorithm, the generation of branch cover cuts is inhibited in this second branch-and-cut round.

As explained, the branch-and-cut scheme works in two stages. In the first stage branching is avoided by adding branch cover cuts. In the second stage, a sparse graph $\tilde{G}$ (resulting from the branch cover cuts produced in the first stage) is considered, and a classical branching strategy is used to close the integrality gap. The computational experience shows that the overall scheme typically performs better than (although does not dominate) the classical one. Indeed, the second stage takes advantage from a large number of relevant cuts (produced in the first stage and stored in the constraint pool), as well as from a very tight approximate OP solution. On the other hand, for some instances the first stage exhibits a slow convergence in the last iterations, due to tailing-off phenomena. To contrast this behavior, branching is allowed even in the first stage. Namely, at each node a branching step is performed after the addition of 5 branch cover cuts.

3.5. Computational results

The branch-and-cut algorithm proposed by Fischetti, Salazar and Toth [301], and described in the previous section (called FST in the sequel) was implemented in ANSI C language, and run on an Hewlett Packard Apollo 9000/720 computer. CPLEX 3.0 was used as LP solver. Four different classes of test problems are considered. The reader is referred to [301] for more computational results.

The first problem class (Class I) includes 15 instances from the OP and Vehicle Routing Problem (VRP) literature. Problems OP21, OP32, and OP33 are OP instances introduced by Tsiligirides [796], with travel times multiplied by 100 and then rounded to the nearest integer. Problems ATT48, EIL30, EIL31, EIL33, EIL51, EIL76, EIL101, and GIL262 are VRP instances taken from library TSPLIB 2.1 of Reinelt [709]. Problems CMT101, CMT121, CMT151, and CMT200 are VRP instances from Christofides, Mingozzi and Toth [191]. For all the VRP instances, the customer demands are interpreted as node prizes.

The second problem class (Class II) includes all the TSP instances contained in TSPLIB 2.1 involving from 137 to 400 nodes (problems GR137 to RD400). For these instances, the node prizes p_j, for $j \in V \setminus \{1\}$, have been generated in three different ways:

- Generation 1: $p_j := 1$;
- Generation 2: $p_j := 1 + (7141 \cdot (j-1) + 73) \bmod (99)$;
- Generation 3: $p_j := 1 + \lfloor 99 \cdot t_{1j}/\theta \rfloor$, where $\theta := \max_{i \in V \setminus \{1\}} t_{1i}$.

(The above is an *errata corrige* of the prize definition for Generation 2 given in [301] which was pointed out to be incorrect by Fink, Schneidereit and Voss [290].)

Generation 1 produces OP instances in which the goal is to cover as many nodes as possible, as occurs in some applications. Generation 2 is intended to produce pseudo-random prizes in range [1,100], whereas Generation 3 leads to more difficult instances, in which large prizes are assigned to the nodes far away from the depot.

For the third problem class (Class III), random instances have been obtained by using the original Laporte and Martello [536] code. In this class, both prizes and travel times are generated as uniformly random integers in range [1,100], with travel times triangularized through shortest path computation.

For all problem classes, the maximum total travel time t_0 is defined as $\lceil \alpha \cdot v(\text{TSP}) \rceil$, where $v(\text{TSP})$ is the length of the corresponding shortest Hamiltonian tour, and α is a given parameter. For all instances taken from TSPLIB, the value $v(\text{TSP})$ is provided within the library. For problems OP21, OP32, OP33, CMT101, CMT121, CMT151, and CMT200, respectively, the following values for $v(\text{TSP})$ have been used: 4598, 8254, 9755, 505, 545, 699, and 764. As to the random problems of Class III, the approximate value computed by the original Laporte-Martello code has been used, namely $v(TSP) := \lfloor 0.95 \cdot UB(TSP) + 0.5 \rfloor$, where $UB(TSP)$ is the length of the tour obtained by the heuristic algorithm proposed by Rosenkrantz, Stearns and Lewis [730].

Name	t_0	r-time	%-LB	%-UB	nodes	cuts	optval	%-vis	t-time
op21	2299	0.7	0.0	0.0	1	58	205	61.9	0.7
op32	4127	1.3	0.0	0.0	1	91	160	56.2	1.3
op33	4878	1.8	0.0	0.0	1	109	500	63.6	1.8
att48	5314	0.8	0.0	0.0	1	55	30	64.6	0.8
eil30	191	5.2	0.0	0.0	1	170	7600	26.7	5.2
eil31	103	0.5	0.0	0.0	1	32	747	58.1	0.5
eil33	221	8.5	0.0	0.0	1	215	16220	48.5	8.5
eil51	213	2.4	0.0	0.0	1	150	508	54.9	2.4
eil76	269	3.1	0.0	0.0	1	118	907	59.2	3.1
eil101	315	5.6	0.0	0.0	1	159	1049	57.4	5.6
cmt101	253	36.9	1.0	0.0	2	514	1030	56.4	55.2
cmt121	273	411.1	0.0	0.8	39	1350	715	52.9	1525.6
cmt151	350	131.8	0.1	0.1	5	451	1537	55.6	167.3
cmt200	382	147.4	0.0	0.0	17	859	2198	62.0	596.3
gil262	1189	365.8	0.2	0.2	35	2370	8456	54.8	3252.7

Table 13.4. Results for problems of Class I (OP and VRP instances) with $\alpha = 0.50$.

Tables 13.4 to 13.9 report on the computational behavior of the branch-and-cut code FST. Each table (except Table 13.5) gives:

Name : the problem name;

t_0 : the maximum total time (only for Classes I and II);

r-time : the total time spent at the root node;

%-LB : the percentage ratio (optimum − LB)/optimum, where LB is the value of the best heuristic solution computed at the root node;

%-UB : the percentage ratio (UB − optimum)/optimum, where UB is the upper bound computed at the root node;

nodes : the total number of nodes generated (1 means that the problem required no branching);

cuts : the total number of cuts generated (including the total time restriction (43));

optval : the optimal solution value (only for classes I and II);

%-vis : the percentage number of nodes visited by the optimal solution;

t-time : the total computing time spent by the branch-and-cut code.

The computing times reported are expressed in seconds, and refer to CPU times on an HP Apollo 9000/720 computer running at 80 MHz (59 SPEC's, 58 MIPS, 18 MFlops). A time limit of 18,000 seconds (5 hours) has been imposed for each run. For the instances exceeding the time limit, we report 't.l.' in the *t-time* column, and compute the

Name	t-LP	t-sep	cuts	log	gsec	2-mat	cover	path	cond	b-cov
op21	0.4	0.1	58	27	14	0	0	0	16	0
op32	0.9	0.2	91	31	46	2	1	0	10	0
op33	1.0	0.5	109	31	47	2	0	0	28	0
att48	0.4	0.2	55	23	17	11	0	0	3	0
eil30	3.6	0.8	170	43	84	0	4	0	38	0
eil31	0.2	0.2	32	15	5	1	1	0	9	0
eil33	5.4	2.0	215	42	86	2	14	20	50	0
eil51	1.6	0.4	150	48	54	5	0	0	42	0
eil76	1.4	1.1	118	50	49	7	0	0	11	0
eil101	2.7	1.5	159	61	59	19	1	0	18	0
cmt101	23.9	16.0	514	114	362	14	8	6	8	1
cmt121	503.5	652.9	1350	189	719	86	82	63	172	38
cmt151	27.8	121.0	451	127	210	61	7	0	39	6
cmt200	84.8	422.6	859	177	449	82	38	0	94	18
gil262	740.0	1873.6	2370	262	1131	154	83	49	644	46

Table 13.5. Additional results for Class I ($\alpha = 0.50$) problems

corresponding results by considering the best available as the optimal solution value. Hence, for the time-limit instances the column *%-UB* gives an upper bound on the percentage approximation error.

Table 13.4 refers to the instances of Class I with $\alpha = 0.5$. We also report, in Table 13.5, additional information on the overall time spent within the LP solver (*t-LP*) and the separation procedures (*t-sep*), and on the number of logical (*log*), inequalities (45) (*gsec*), 2-matching (*2-mat*), cover (*cover*), path (*path*), conditional (*cond*), and branch cover (*b-cov*) constraints generated.

Tables 13.6 to 13.8 refer to the instances of Class II, with prizes computed according to Generation 1, 2, and 3, respectively. The parameter α has been set to 0.50. Cases $\alpha = 0.25$ and $\alpha = 0.75$ present comparable results.

Table 13.9 reports average results over 10 random instances belonging to Class III, with $\alpha = 0.2$, 0.4, 0.6, and 0.8, and $n = 25$, 50, 100, 300, and 500. Larger instances could be solved as well, since for this class the computing time tends to increase very slowly with n for $n \geq 200$. As a comparison, the branch-and-bound algorithm of Laporte and Martello [536] ran into difficulties when solving instances with $n = 25$ and $\alpha \geq 0.6$, and with $n = 50$ and $\alpha \geq 0.4$. For example, running (on the HP Apollo 9000/720 computer) the Laporte and Martello code on the instances with $n = 25$ required on average 0.1 seconds for $\alpha = 0.2$, 77.2 seconds for $\alpha = 0.4$, more than 2 hours for $\alpha = 0.6$; whereas for $\alpha = 0.8$ no instance was solved within the 5 hour time-limit.

Name	t_0	r-time	%-LB	%-UB	nodes	cuts	optval	%-vis	t-time
gr137	34927	178.6	0.0	0.0	1	553	81	59.1	178.6
pr144	29269	240.3	0.0	0.0	1	1170	77	53.5	240.3
kroa150	13262	582.2	0.0	1.2	85	2594	86	57.3	4669.0
krob150	13065	145.6	0.0	0.0	1	729	87	58.0	145.6
pr152	36841	204.6	0.0	0.0	1	917	77	50.7	204.6
u159	21040	497.6	0.0	0.0	1	1912	93	58.5	497.6
rat195	1162	331.9	0.0	0.0	1	1323	102	52.3	331.9
d198	7890	716.3	0.0	0.0	1	1431	123	62.1	716.3
kroa200	14684	395.0	0.0	0.0	1	1254	117	58.5	395.0
krob200	14719	683.6	0.0	0.0	1	1635	119	59.5	683.6
gr202	20080	150.6	0.0	0.0	1	603	147	72.8	150.6
ts225	63322	9.7	0.0	0.8	107	6040	125	55.5	t.l.
pr226	40185	1955.3	0.0	5.2	25	1019	134	59.3	t.l.
gr229	1765	75.0	0.0	0.0	1	431	176	76.9	75.0
gil262	1189	120.6	0.0	0.0	1	789	158	60.3	120.6
pr264	24568	2860.2	0.0	0.0	1	1694	132	50.0	2860.2
pr299	24096	5726.3	1.2	1.2	8	4524	162	54.2	14244.0
lin318	21045	2558.0	0.0	0.5	5	2653	205	64.5	3169.9
rd400	7641	874.0	1.7	0.4	29	1821	239	59.8	4272.5

Table 13.6. Results for Class II (TSPLIB instances) and Generation 1 ($\alpha = 0.50$)

Name	t_0	r-time	%-LB	%-UB	nodes	cuts	optval	%-vis	t-time
gr137	34927	797.9	0.0	0.3	81	1929	4294	57.7	3193.0
pr144	29269	668.0	0.0	0.7	11	1579	4003	51.4	1409.0
kroa150	13262	460.1	0.6	0.9	47	2876	4918	54.0	3950.6
krob150	13065	735.7	0.0	0.1	5	1795	4869	52.7	1018.1
pr152	36841	188.6	0.0	0.0	1	836	4279	48.0	188.6
u159	21040	518.0	0.4	0.2	21	1853	4960	54.1	1772.4
rat195	1162	1750.1	0.0	0.2	13	2691	5791	48.2	2498.6
d198	7890	1337.8	0.1	0.1	27	1999	6670	56.1	2517.1
kroa200	14684	515.2	0.0	0.1	15	1147	6547	54.5	805.1
krob200	14719	1240.7	0.1	0.1	17	2563	6419	50.5	3522.8
gr202	20080	441.7	0.8	0.0	31	1945	7848	65.8	3847.6
ts225	63322	763.3	0.0	0.1	5	1627	6834	54.2	1195.5
pr226	40185	3973.9	0.1	0.3	27	1842	6615	46.6	t.l.
gr229	1765	329.5	0.5	0.1	47	1415	9187	72.1	4261.4
gil262	1189	2783.0	0.1	0.2	23	2080	8321	50.8	5574.6
pr264	24568	4253.3	0.0	0.0	1	2211	6654	50.0	4253.3
pr299	24096	10803.8	0.0	0.0	5	1900	9161	49.8	t.l.
lin318	21045	1370.0	0.0	0.0	41	1392	10900	60.7	t.l.
rd400	7641	837.6	0.1	0.2	76	3721	13648	54.5	t.l.

Table 13.7. Results for Class II (TSPLIB instances) and Generation 2 ($\alpha = 0.50$)

Name	t_0	r-time	%-LB	%-UB	nodes	cuts	optval	%-vis	t-time
gr137	34927	2401.9	45.5	0.1	5	5386	3979	51.8	4958.7
pr144	29269	1573.8	0.0	0.1	65	2632	3809	43.1	t.l.
kroa150	13262	269.7	0.1	0.6	181	1646	5039	52.7	3828.9
krob150	13065	1112.5	0.0	0.1	13	1472	5314	56.7	1363.9
pr152	36841	1099.0	0.7	1.3	391	3159	3905	48.7	13736.7
u159	21040	1308.4	0.0	0.2	5	2171	5272	52.8	1447.2
rat195	1162	3672.2	0.1	0.1	9	1528	6195	47.7	3975.4
d198	7890	1810.0	0.0	0.2	197	2361	6320	61.6	8635.7
kroa200	14684	3116.5	0.0	0.2	41	2161	6123	51.5	6548.9
krob200	14719	642.6	0.0	0.1	9	905	6266	51.0	783.7
gr202	20080	654.2	0.5	0.3	298	1947	8632	71.3	11113.5
ts225	63322	3437.6	0.0	0.4	27	1598	7575	55.1	5821.8
pr226	40185	3379.5	16.0	0.1	51	5310	6993	52.2	7923.2
gr229	1765	1667.1	0.0	0.0	11	1613	6347	67.7	1891.5
gil262	1189	6177.4	0.2	0.0	27	2386	9246	56.5	9574.0
pr264	24568	4011.3	0.0	0.0	1	2625	8137	39.8	4011.3
pr299	24096	14699.4	0.0	0.0	2	1787	10358	49.8	t.l.
lin318	21045	8597.5	0.0	0.0	12	1308	10382	60.7	t.l.
rd400	7641	14257.1	0.0	0.0	3	1418	13229	55.8	t.l.

Table 13.8. Results for Class II (TSPLIB instances) and Generation 3 ($\alpha = 0.50$)

On the whole, the performance of the branch-and-cut code is quite satisfactory for our families of instances. In most cases, the upper and lower bounds computed at the root node are very tight, and a few branchings are needed. The code was able to solve to proven optimality almost all the random instances of Class III (except 1 instance for $n = 500$), and most of the "real-world" instances of Classes I and II. For the instances exceeding the time limit, the computed solution is very close to the optimal one (see column %-UB).

According to Table 13.5, most of the generated constraints are inequalities (45), 2-matching, logical and conditional cuts. For some "difficult" instances, a relevant number of cover and path inequalities is generated.

Additional computational experience has been performed on the class of random instances considered in the work by Gendreau, Laporte and Semet [355], called Class IV in the sequel. These instances were generated by using the original Gendreau-Laporte-Semet code. The instances are similar to those of Class II and Generation 2, but the nodes are generated as random points in the $[0, 100]^2$ square according to a uniform distribution. The corresponding values of $v(TSP)$ were computed by means of the algorithm of Padberg and Rinaldi [648]. Table 13.10 reports average results over 5 random instances belonging to Class IV, with α=0.1, 0.3, 0.5, 0.7, 0.9, and n=101, 121, 161, 261, and 301.

n	α	r-time	%-LB	%-UB	nodes	cuts	%-vis	t-time
25	0.2	1.1	0.0	0.0	1.1	63.4	28.8	1.2
25	0.4	38.8	0.0	0.9	3.8	138.2	56.8	42.7
25	0.6	46.6	0.0	0.3	2.6	101.0	74.0	63.4
25	0.8	42.1	0.0	0.4	4.0	91.5	86.0	51.5
50	0.2	41.5	0.0	0.4	1.6	169.6	32.8	53.8
50	0.4	32.1	0.0	0.5	10.0	220.2	59.8	99.6
50	0.6	30.8	0.3	0.3	21.8	263.1	76.4	147.0
50	0.8	10.5	0.2	0.1	10.4	128.1	90.2	58.9
100	0.2	61.3	0.0	0.3	8.2	359.6	35.3	149.6
100	0.4	35.9	0.2	0.2	28.4	403.5	60.9	340.3
100	0.6	33.7	0.4	0.1	34.6	421.4	80.8	445.2
100	0.8	41.9	0.3	0.0	35.4	273.9	93.2	403.8
300	0.2	102.7	0.1	0.1	15.0	484.8	30.1	363.5
300	0.4	139.1	0.2	0.0	43.6	652.4	57.5	1816.5
300	0.6	203.9	0.2	0.0	26.2	355.3	80.4	1949.5
300	0.8	237.2	1.5	0.0	4.3	186.6	99.8	2038.5
500	0.2	189.6	0.0	0.0	5.4	300.5	27.0	325.4
500	0.4	317.4	0.3	0.0	9.9	322.9	53.6	1418.0
500	0.6	408.4	1.1	0.0	9.7	284.2	78.8	5327.9
500	0.8	650.2	1.4	0.0	2.4	116.2	100.0	2454.0

Table 13.9. Average results over 10 random instances of Class III

n	α	N_1	N_2	N_3	r-time	%-LB	%-UB	nodes	cuts	%-vis	t-time
101	0.1	5	5	5	60.5	0.0	0.8	3.4	559.8	11.7	193.3
101	0.3	5	5	5	186.7	0.0	0.3	3.0	849.6	33.5	228.7
101	0.5	5	5	5	175.3	0.0	0.3	11.8	806.4	55.6	350.2
101	0.7	5	5	5	70.9	0.3	0.2	37.8	747.6	75.6	584.5
101	0.9	5	5	1	55.2	1.1	0.3	222.6	872.6	90.7	2467.5
161	0.1	5	5	5	233.8	0.0	0.2	1.4	900.6	12.2	275.9
161	0.3	5	5	2	539.6	0.4	0.5	11.0	1638.2	34.9	1254.6
161	0.5	5	5	0	249.4	0.2	0.2	20.2	956.2	55.0	737.2
161	0.7	4	3	0	185.2	0.7	0.2	101.5	1419.5	72.8	3563.2
161	0.9	5	4	-	136.0	3.8	0.1	90.0	697.2	90.3	2981.2
201	0.1	5	5	5	397.9	0.0	0.6	5.8	1345.4	11.4	673.2
201	0.3	5	4	-	555.7	0.2	0.6	37.4	2188.2	34.3	2793.0
201	0.5	5	5	-	412.1	0.0	0.1	9.8	1062.6	54.5	724.1
201	0.7	4	1	-	952.7	1.4	0.1	88.5	2136.2	71.8	6118.0
201	0.9	4	1	-	305.9	2.4	0.1	121.8	1224.5	90.4	9917.1
301	0.1	5	4	1	1358.1	0.3	1.1	17.0	2982.8	11.7	3729.6
301	0.3	2	0	-	914.5	1.5	0.6	98.0	3779.0	35.9	12145.3
301	0.5	3	1	-	530.2	0.9	0.2	59.0	3151.0	54.7	8787.9
301	0.7	1	1	-	616.6	0.3	0.0	14.0	1303.0	72.1	5064.7
301	0.9	0	0	-	—	—	—	—	—	—	t.l.

Table 13.10. Average results over 5 random instances of Class IV

Column N_3 gives the number of instances successfully solved by the Gendreau-Laporte-Semet code within a time limit of 10,000 SUN Sparc station 1000 CPU seconds. According to Dongarra [258], the HP Apollo 9000/720 computer is about 1.8 times faster than that used by Gendreau, Laporte and Semet, hence their time limit corresponds to about 5,555 HP 9000/720 CPU seconds. Columns N_1 and N_2 give the number of instances successfully solved by code FST within a time limit of 18,000 and 5,555 HP 9000/720 CPU seconds, respectively. The remaining columns are as in previous tables, and refer to the execution of code FST with the 18,000 second time limit. As in [355], averages are computed with respect to the instances solved to proven optimality.

A comparison of columns N_2 and N_3 shows that code FST is capable of solving a number of instances substantially larger than the Gendreau-Laporte-Semet code, within approximately the same time limit. Moreover, the values of both the lower and upper bounds computed by FST are tighter than those reported in [355]. For the cases in which the Gendreau-Laporte-Semet code successfully solved all the five instances, the average LB and UB ratios of CFT (0.02% and 0.28%, respectively) compare very favorably with the corresponding values reported in [355] (3.59% and 1.74%, respectively). On the whole, the instances of Class IV appear more difficult than those in the previous classes.

Acknowledgement

The work of the first and third authors has been supported by C.N.R. and by M.I.U.R., Italy. The work of the second author has been supported by the research projects TIC-2000-1750-CO6-02 and PI2000/116, Spain.