On the Solution of Traveling Salesman Problems

David Applegate, Robert Bixby,
Vašek Chvátal and William Cook[1]

Abstract. Following the theoretical studies of J.B. Robinson and H.W. Kuhn in the late 1940s and the early 1950s, G.B. Dantzig, R. Fulkerson, and S.M. Johnson demonstrated in 1954 that large instances of the TSP could be solved by linear programming. Their approach remains the only known tool for solving TSP instances with more than several hundred cities; over the years, it has evolved further through the work of M. Grötschel, S. Hong, M. Jünger, P. Miliotis, D. Naddef, M. Padberg, W.R. Pulleyblank, G. Reinelt, G. Rinaldi, and others. We enumerate some of its refinements that led to the solution of a 13,509-city instance.

1991 Mathematics Subject Classification: 90C10 90C27
Keywords and Phrases: traveling salesman; cutting planes

The *traveling salesman problem*, or TSP for short, is easy to state: given a finite number of "cities" along with the cost of travel between each pair of them, find the cheapest way of visiting all of the cities and returning to your starting point. The travel costs are symmetric in the sense that traveling from city X to city Y costs just as much as traveling from Y to X; the "way of visiting all the cities" is simply the order in which the cities are visited. The simplicity of this problem, coupled with its apparent intractability, makes it an ideal platform for exploring new algorithmic ideas. Surveys of work on the TSP can be found in Bellmore and Nemhauser [1968], Lawler, Lenstra, Rinnooy Kan, and Shmoys [1985], Reinelt [1994], and Jünger, Reinelt, and Rinaldi [1995].

The origins of the TSP are obscure. In the 1920's, the mathematician and economist Karl Menger publicized it among his colleagues in Vienna. In the 1930's, the problem reappeared in the mathematical circles of Princeton. In the 1940's, it was studied by statisticians (Mahalanobis [1940], Jessen [1942]) in connection with an agricultural application and the mathematician Merrill Flood popularized it among his colleagues at the RAND Corporation. Eventually, the TSP gained notoriety as the prototype of a hard problem in combinatorial optimization.

A breakthrough came when Dantzig, Fulkerson, and Johnson [1954] published a description of a method for solving the TSP and illustrated the power of this method by solving an instance with 49 cities, an impressive size at that time. Riding the wave of excitement over the numerous applications of the simplex

[1]Supported by ONR Grant N00014-98-1-0014.

method (designed by Dantzig in 1947) and following the studies of Robinson [1949] and Kuhn [1955], Dantzig, Fulkerson, and Johnson attacked the salesman with linear programming as follows.

Each TSP instance with n cities can be specified by a vector c with $n(n-1)/2$ components, whose values are the travel costs; each tour through the n cities can be represented as its incidence vector with $n(n-1)/2$ components; if $\mathcal{S}$ denotes the set of the incidence vectors of all the tours, then the problem is to

$$\text{minimize } c^T x \text{ subject to } x \in \mathcal{S}. \tag{1}$$

Like the man searching for his lost wallet not in the dark alley where he actually dropped it, but under a street lamp where he can see, Dantzig, Fulkerson and Johnson begin not with the problem they *want to* solve, but with a related problem they *can* solve,

$$\text{minimize } c^T x \text{ subject to } Ax \leq b \tag{2}$$

with some suitably chosen system $Ax \leq b$ of linear inequalities satisfied by all x in $\mathcal{S}$: solving linear programming problems such as (2) is precisely what the simplex method is for. Since (2) is a *relaxation* of (1) in the sense that every feasible solution of (1) is a feasible solution of (2), the optimal value of (2) provides a lower bound on the optimal value of (1).

The ground-breaking idea of Dantzig, Fulkerson, and Johnson was that solving (2) can help with solving (1) in a far more substantial way than just by providing a lower bound: having determined that the wallet is not under the street lamp, one can pick the street lamp up and bring it a little closer to the place where the wallet was lost. If (2) has an optimal solution and if the polyhedron $\{x : Ax \leq b\}$ has an extreme point, then the simplex method finds an optimal solution x^* of (2) such that x^* is an extreme point of $\{x : Ax \leq b\}$; in particular, if x^* is not a member of $\mathcal{S}$, then some linear inequality is satisfied by all the points in $\mathcal{S}$ and violated by x^*. Such an inequality is called a *cutting plane* or simply a *cut*. Having found cuts, one can add them to the system $Ax \leq b$, solve the resulting tighter relaxation by the simplex method, and iterate this process until one arrives at a linear programming relaxation of (1) and its optimal solution x^* such that $x^* \in \mathcal{S}$.

The influence of this work reached far beyond the narrow confines of the TSP: the *cutting-plane method* can be used to attack any problem (1) such that $\mathcal{S}$ is a finite subset of $\mathbf{R}^m$ and an efficient algorithm to recognize points of $\mathcal{S}$ is available. Many problems in *combinatorial optimization* have this form: in the maximum clique problem, $\mathcal{S}$ consists of the incidence vectors of all cliques in the input graph; in the maximum cut problem, $\mathcal{S}$ consists of the incidence vectors of all edge-cuts in the input graph; and so on. Applications of the cutting-plane method to these problems stimulated the development of the flourishing field of polyhedral combinatorics. Another important class of problems (1) are the *integer linear programming* problems, where $\mathcal{S}$ is specified as the set of all integer solutions of a prescribed system of linear inequalities. For this class, Gomory [1958] designed efficient procedures to generate cutting planes in a way that guarantees the cutting-plane method's termination.

The efficiency of the cutting-plane method is a different matter. Where the TSP is concerned, there are reasons to believe that the method may require prohibitively large amounts of time even on certain reasonably small instances: R.M. Karp, E.L. Lawler, and R.E. Tarjan (see Karp [1972]) proved that the decision version of the TSP is an $\mathcal{NP}$-complete problem.

When $\mathcal{S}$ consists of all tours through a set of cities V, Dantzig, Fulkerson, and Johnson let the initial polyhedron consist of all vectors x, with components subscripted by edges of the complete graph on V, that satisfy

$$0 \le x_e \le 1 \qquad \text{for all edges } e, \tag{3}$$

$$\sum (x_e : v \in e) = 2 \qquad \text{for all cities } v. \tag{4}$$

In solving the 49-city problem, Dantzig, Fulkerson, and Johnson tightened this initial LP relaxation first by a number of *subtour inequalities*,

$$\sum (x_e : e \cap S \neq \emptyset, e - S \neq \emptyset) \ge 2 \quad \text{with } S \subset V,\ S \neq \emptyset,\ S \neq V, \tag{5}$$

and then by two additional cuts, after which x^* became the incidence vector of a tour; to show that these two inequalities are satisfied by incidence vectors of all tours, Dantzig, Fulkerson, and Johnson used ad hoc combinatorial arguments.

When an LP relaxation of a TSP instance includes all constraints (3), (4), a nonempty set of cuts can be found easily whenever $x^* \notin \mathcal{S}$: on the one hand, if x^* is not an integer vector, then Gomory's procedures find a nonempty set of cuts; on the other hand, if x^* is an integer vector, then it is the incidence vector of the edge-set of a disconnected graph and each connected component of this graph yields a subtour cut. This scheme is used, with embellishments, in the computer code of Martin [1966], which seems to be the first computer code for solving the TSP. Eventually, subtour inequalities became a staple of TSP cuts but, when new ways of finding TSP cuts emerged, Gomory cuts fell into disuse as TSP cuts.

1 Finding cuts

Hypergraph cuts

Given a subset S of V and given an x satisfying (3), (4), we write

$$\eta(S, x) = \sum (x_e : e \cap S \neq \emptyset, e - S \neq \emptyset) - 2.$$

A *hypergraph* is an ordered pair $(V, \mathcal{F})$ such that V is a finite set and $\mathcal{F}$ is a family of (not necessarily distinct) subsets of V; elements of V are called the *vertices* of the hypergraph and the elements of $\mathcal{F}$ are called the *edges* of the hypergraph. Given a hypergraph $(V, \mathcal{F})$ denoted $\mathcal{H}$, we write $\mathcal{H} \circ x = \sum (\eta(S, x) \ : \ S \in \mathcal{F})$ and we let $\mu(\mathcal{H})$ stand for the minimum of $\mathcal{H} \circ x$ taken over the incidence vectors of tours through V. Every linear inequality satisfied by all the incidence vectors of tours through V is the sum of a linear combination of equations (4) and a *hypergraph inequality*,

$$\mathcal{H} \circ x \ge t$$

with $t \le \mu(\mathcal{H})$. Subtour inequalities are the simplest instances of hypergraph inequalities; one class of more complex instances is as follows.

The *intersection graph* of a hypergraph $(V, \mathcal{F})$ is the graph with vertex-set $\mathcal{F}$ and with two vertices adjacent if and only if these two members of $\mathcal{F}$ intersect. A *clique tree* is any hypergraph $\mathcal{H}$ such that

- the intersection graph of $\mathcal{H}$ is a tree

and such that the edge-set of $\mathcal{H}$ can be partitioned into a set of "handles" and a set of "teeth" with the following properties:

- there is at least one handle,
- the handles are pairwise disjoint,
- the teeth are pairwise disjoint,
- the number of teeth that each handle intersects is odd and at least three,
- each tooth includes a point that belongs to no handle.

Grötschel and Pulleyblank [1986] introduced this notion and proved that, for every clique-tree $\mathcal{H}$ with s teeth, the incidence vector x of any tour through V satisfies

$$\mathcal{H} \circ x \geq s - 1. \tag{6}$$

Let us give a short proof of this theorem here. Consider a clique tree with handles $H_1, \ldots, H_r$ and teeth $T_1, \ldots, T_s$; let t_j denote the number of handles that intersect tooth T_j and let h_i denote the number of teeth that intersect handle H_i; write

$$c_{ij} = \begin{cases} 1 & \text{if the tour includes an edge from } H_i \cap T_j \text{ to } T_j - H_i, \\ 0 & \text{otherwise.} \end{cases}$$

Since the teeth are pairwise disjoint, we have $\eta(H_i, x) \geq \sum_j c_{ij} - 2$; by definition, we have $\sum_j c_{ij} \leq h_i$; since $\eta(H_i, x)$ is even and h_i is odd, we conclude that

$$\eta(H_i, x) \geq 2 \sum_{j=1}^{s} c_{ij} - h_i - 1. \tag{7}$$

The restriction of the tour on a tooth T_j consists of $1 + \eta(T_j, x)/2$ segments; one of these segments passes through the point of T_j that belongs to no handle; since the handles are pairwise disjoint, each i such that $H_i \cap T_j \neq \emptyset$ and $c_{ij} = 0$ adds a new segment; we conclude that

$$\eta(T_j, x) \geq 2(t_j - \sum_{i=1}^{r} c_{ij}). \tag{8}$$

From (7) and (8), we obtain $\mathcal{H} \circ x \geq 2 \sum_j t_j - \sum_i h_i - r = \sum_j t_j - r$; since the intersection graph of $\mathcal{H}$ is a tree, we have $\sum_j t_j = r + s - 1$ and (6) follows.

Clique-trees with precisely one handle are called *combs* and the corresponding inequalities (6) are called *comb inequalities*.

FACET-INDUCING CUTS AND THE TEMPLATE PARADIGM

Some cuts are better than others. The ultimate measure of quality of a cut is its contribution to reducing the total running time of the cutting-plane method.

It is well known (Grötschel and Padberg [1975], Maurras [1975]) that the affine hull of the set $\mathcal{S}$ of all tours through V consists of all solutions x of (4); it follows that every cut is the sum of

- a linear combination of equations (4) and
- a nonnegative combination of linear inequalities that induce facets of the convex hull of $\mathcal{S}$.

Appealing to this fact, one may argue for preferring facet-inducing cuts to all others. This point of view suggests a two-phase paradigm for finding TSP cuts:

(i) describe linear inequalities that induce facets of the convex hull of $\mathcal{S}$,

(ii) for each template obtained in phase (i), design an efficient algorithm that, given an x^*, finds a cut matching that template, if such a cut exists.

Algorithms designed in phase (ii) are called *exact separation algorithms*; algorithms that attempt to find a cut matching the template, and may fail even if such a cut exists, are called *heuristic separation algorithms*.

The template paradigm was championed by Grötschel and Padberg [1979a, 1979b] and by Padberg and Hong [1980]. As for its phase (i), Grötschel and Padberg [1979a, 1979b] proved that both subtour inequalities and comb inequalities induce facets of the convex hull of $\mathcal{S}$; Grötschel and Pulleyblank [1986] proved that clique tree inequalities induce facets of the convex hull of $\mathcal{S}$; Naddef and Rinaldi [1998] proved that *path inequalities* (another generalization of comb inequalities, introduced by Cornuéjols, Fonlupt, and Naddef [1985]) induce facets of the convex hull of $\mathcal{S}$.

A polynomial-time exact separation algorithm for subtour inequalities was pointed out by Hong [1972]. It uses the observation that the problem of minimizing $\eta(S, x^*)$ subject to $S \subset V$, $S \neq \emptyset$, $S \neq V$ reduces to $|V|-1$ instances of the problem

$$\text{minimize } \eta(S, x^*) \text{ subject to } S \subset V,\ s \in S,\ t \notin S \tag{9}$$

with s fixed and t ranging through the remaining cities; it relies on the fact that (9) can be solved in polynomial time by variations on the max-flow min-cut theme of Ford and Fulkerson [1962]. The appeal of this scheme for actual computations is much enhanced when the input size is first reduced by "shrinking procedures" designed by Crowder and Padberg [1980] and by Padberg and Rinaldi [1990]; these procedures alone, without the subsequent max-flow min-cut computations, constitute fast heuristic separation algorithms for subtour inequalities.

A comb with each tooth having exactly two vertices is called a *blossom*. Padberg and Rao [1982] designed a polynomial-time exact separation algorithm for blossom inequalities. Their algorithm is an important tool in the computer codes of Grötschel and Holland [1991] and Padberg and Rinaldi [1991]: besides delivering blossom cuts, it is also used in heuristic separation algorithms for the more general comb inequalities. (The idea is to select sets S such that $\eta(S, x^*) = 0$ and to shrink each of these sets into a single vertex: blossom inequalities over the shrunken image of V yield comb inequalities over the original V.)

Other heuristic separation algorithms for comb inequalities, and for 2-handled clique tree inequalities, were designed by Padberg and Rinaldi [1991]; guided by the structure of the graph with vertex-set V and edge-set $\{e : 0 < x^*_e < 1\}$, they attempt to build the desired hypergraph in a greedy fashion. Heuristic separation algorithms for path inequalities and other templates of TSP cuts were designed by Clochard and Naddef [1993] and by Christof and Reinelt [1995].

We have written a computer code for the TSP that follows in part the template paradigm. Our separation algorithms are

- an exact separation algorithm for subtour cuts that consists of Padberg-Rinaldi shrinking followed by repeated calls of the push-relabel method, as implemented by Cherkassky and Goldberg [1997], to solve max-flow min-cut problems,
- the Padberg-Rao exact separation algorithm for blossom cuts,
- the Grötschel-Holland and Padberg-Rinaldi heuristics for comb cuts,
- a greedy heuristic of the Clochard-Naddef kind for certain path cuts,

and heuristic separation algorithms that we have designed. Three of them that turned out to be important in solving the more difficult instances are as follows.

- Like most TSP codes, ours maintains the best tour $\overline{x}$ that we know of. One may suspect that, as both x^* and $\overline{x}$ approximate an optimal tour, sets S with $\eta(S, x^*) < 0$ are likely to satisfy $\eta(S, \overline{x}) < 2$, and so constitute single segments of the tour $\overline{x}$; our computational experience confirms this suspicion. We have designed an algorithm that, given x^* and $\overline{x}$, returns a family of segments $S_v (v \in V)$ of $\overline{x}$ such that each S_v minimizes $\eta(S, x^*)$ over all segments S that begin at v; its running time is in $O(m \log |V|)$, where m is the number of positive components of x^*. (We have used this algorithm not only in solving TSP instances, but also in computing lower bounds for TSP instances with up to 500,000 cities.)
- Having collected a family $\mathcal{F}$ of sets S such that $\eta(S, x^*) < 2$, we search for combs with handle H and teeth T_1, T_2, T_3 such that $H, T_1, T_2, T_3 \in \mathcal{F}$ and such that x^* violates the corresponding comb inequality. The search is guided by the observation that the desired $\{H, T_1, T_2, T_3\}$, as well as $\{H, T_1, T_2, V - T_3\}$, is a minimal family without the *consecutive ones property*; as an oracle for testing the consecutive ones property, we use *PQ-trees*, an efficient data structure designed by Booth and Lueker [1976].
- One way of showing that comb inequalities are satisfied by all tours is related to the framework for describing Gomory cuts propounded by Chvátal [1973]. We decided to turn the argument into an algorithm and search for comb cuts by solving certain systems of linear congruences mod 2. Our implementation of this plan uses PQ-trees once again, this time as a compact device for storing families of sets S such that $\eta(S, x^*) = 0$: variables in our system of linear congruences are in a one-to-one correspondence with Q-nodes of our PQ-tree. (Later on, Adam Letchford pointed out to us how our algorithm could be adjusted to search for the more general path cuts.)

Beyond the template paradigm

There are routine and well known algorithms that, given a finite subset $\mathcal{S}$ of some $\mathbf{R}^m$ and given a point x^* in $\mathbf{R}^m$, either express x^* as a convex combination of points in $\mathcal{S}$ or find a linear inequality that is satisfied by all points of $\mathcal{S}$ and violated by x^*. Using these algorithms directly to find cuts would be insane, since their running time is prohibitively long when m is large; using them in conjunction with the trick of first projecting $\mathcal{S}$ and x^* into a lower-dimensional space was a crucial ingredient in our solution of a 13,509-city TSP instance.

Given x^*, we choose many different linear mappings $\phi : \mathbf{R}^m \to \mathbf{R}^d$; for each of our choices of ϕ, we express x^* as a convex combination of points in $\phi(\mathcal{S})$ or find a linear inequality $a^T\xi \geq b$ that is satisfied by all points ξ of $\phi(\mathcal{S})$ and $a^T\phi(x^*) < b$; in the latter case, inequality $a^T\phi(x) \geq b$ is a cut. This scheme is feasible as long as d is reasonably small: large size of $\phi(\mathcal{S})$ presents no difficulty provided that $\phi(\mathcal{S})$ can be accessed by an efficient oracle which, given any vector c in $\mathbf{R}^d$, returns an element ξ of $\phi(\mathcal{S})$ that maximizes $c^T\xi$.

True, it may happen that $\phi(x^*)$ belongs to the convex hull of $\phi(\mathcal{S})$ even though x^* lies outside the convex hull of $\mathcal{S}$; however, this is not always the case, and $\phi(\mathcal{S})$ is easier to handle than $\mathcal{S}$. Going a step further in this spirit adds flexibility to the method: for $\phi(\mathcal{S})$, we may substitute any $\mathcal{T}$ such that $\phi(\mathcal{S}) \subseteq \mathcal{T}$. True, it may happen that $\phi(x^*)$ belongs to the convex hull of $\mathcal{T}$ even though it lies outside the convex hull of $\phi(\mathcal{S})$; however, this is not always the case, and $\mathcal{T}$ may be easier to handle than $\phi(\mathcal{S})$.

Success of this method depends on making choices of ϕ and $\mathcal{T}$ in such a way that $\phi(x^*)$ has a reasonable chance of lying outside the convex hull of $\mathcal{T}$ and yet $\mathcal{T}$ is reasonably easy to handle. In the special case where $\mathcal{S}$ consists of all tours through a set V, our computer code makes each choice of ϕ by choosing a partition of V into nonempty sets $V_0, V_1, \ldots, V_k$. The corresponding ϕ is defined by shrinking each of these sets into a single point: the component of $\phi(x)$ that is indexed by i and j ($0 \leq i < j \leq k$) has value $\sum(x_e : e \cap V_i \neq \emptyset, e \cap V_j \neq \emptyset)$. Our $\mathcal{T}$ consists of all nonnegative integer vectors ξ with components indexed by edges of the complete graph with vertex-set $\{0, 1, \ldots, k\}$ such that

- the graph with vertex-set $\{0, 1, \ldots, k\}$ and edge-set $\{e : \xi_e > 0\}$ is connected,
- $\sum(\xi_e : v \in e)$ is even whenever $v \in \{0, 1, \ldots, k\}$.

(Cornuéjols, Fonlupt, and Naddef [1985] call the problem of minimizing a prescribed linear function over this $\mathcal{T}$ the *graphical traveling salesman problem.*) We let k range between 8 and 30; our choices of $V_0, V_1, \ldots, V_k$ are guided by the structure of x^*; in particular, $\eta(V_j, x^*) = 0$ for all $j = 1, 2, \ldots, k$.

We do not know how useful this approach might prove in finding cuts for other problems (1); possibly its success in our experience with the TSP comes at least in part from the peculiar nature of the TSP; let us elaborate. The algorithm that we use to deal with $\mathcal{T}$ and $\phi(x^*)$ either expresses x^* as a convex combination of points in $\mathcal{T}$ or finds an inequality $a^T\xi \geq b$ that induces a facet of the convex hull of $\mathcal{T}$ and is violated by $\phi(x^*)$. In the latter case, we transform $a^T\xi \geq b$ into a hypergraph inequality $\mathcal{H} \circ \xi \geq t$ before substituting $\phi(x)$ for ξ; in our experience, these hypergraph inequalities are often (but not always) *tight triangular*; a conjecture implicit in the work of Naddef and Rinaldi [1992] suggests that, under this condition, inequality $\mathcal{H} \circ \phi(x) \geq t$ induces a facet of the convex hull of $\mathcal{S}$.

Another algorithm for finding TSP cuts that strays off the beaten path of the template paradigm, but starts from the Naddef-Rinaldi notion of tight triangular inequalities, has been designed by Carr [1998].

Alterations while you wait

Watching our computer code run, we have observed that optimal solutions x^* of the successive LP relaxations often react to each new cut we add by shifting the defect prohibited by this cut to an area just beyond the cut's control. The remedy is obvious: we respond to each slight adjustment of $x^\star$ with a slight adjustment of our hypergraph cuts.

Given a hypergraph $\mathcal{H}$ with edges $E_1, \ldots, E_m$, we set

$$\alpha(I, \mathcal{H}) = \bigcap_{i \in I} E_i - \bigcup_{i \notin I} E_i$$

for each subset I of $\{1, \ldots, m\}$; we refer to each $\alpha(I, \mathcal{H})$ as an *atom* of $\mathcal{H}$; we write $\mathcal{H} \sqsubseteq \mathcal{H}'$ to signify that $\mathcal{H}$ and $\mathcal{H}'$ are hypergraphs with the same set of vertices and the same number of edges, and such that $\alpha(I, \mathcal{H}') \neq \emptyset$ whenever $\alpha(I, \mathcal{H}) \neq \emptyset$. It can be shown that $\mathcal{H} \sqsubseteq \mathcal{H}'$ implies $\mu(\mathcal{H}') \geq \mu(\mathcal{H})$. By *tightening* a hypergraph $\mathcal{H}$ with respect to a vector x^*, we mean a swift attempt to modify $\mathcal{H}$ in such a way that the resulting hypergraph, $\mathcal{H}'$, satisfies

- $\mathcal{H} \sqsubseteq \mathcal{H}'$ and $\mathcal{H}' \circ x^* < \mathcal{H} \circ x^*$.

We tighten $\mathcal{H}$ by a greedy algorithm that moves single vertices from one atom to another if such a move decreases $\mathcal{H} \circ x^*$ (or, with some restrictions, if the move at least does not increase $\mathcal{H} \circ x^*$). Some of these permissible moves are more appealing than others; all of them are kept in a priority queue, which is updated after each move is made. We make extensive use of tightening in our computer code. Every cut that we find is tightened before it is added to the LP relaxation. We also periodically run through all constraints of the LP relaxation and tighten each of them.

We use one additional technique for adjusting comb inequalities. Let us refer to a comb with some of its teeth removed as a *generalized comb*; let us say that a tooth of a generalized comb is *big* if its size is at least three; for every generalized comb $\mathcal{H}_0$, let $\Delta(\mathcal{H}_0, x^*)$ denote the minimum of $\mathcal{H} \circ x^* - \mu(\mathcal{H})$ over all combs $\mathcal{H}$ such that $\mathcal{H}$ and $\mathcal{H}_0$ have the same handle and all big teeth of $\mathcal{H}$ are teeth of $\mathcal{H}_0$. We have designed a dynamic programming algorithm that, given a generalized comb $\mathcal{H}_0$, finds either

- a comb $\mathcal{H}$ such that all big teeth of $\mathcal{H}$ are teeth of $\mathcal{H}_0$ and, if $\Delta(\mathcal{H}_0) \leq 0$, then $\mathcal{H} \circ x^* - \mu(\mathcal{H}) \leq \Delta(\mathcal{H}_0)$

or else a subtour inequality violated by x^*. We refer to this algorithm as *teething*, and we apply it to comb constraints in the LP relaxation.

2 The branch-and-cut method

Progress of the cutting-plane method towards solving a particular problem instance is often estimated by the increase in the optimal value of its LP relaxation; as more and more cuts are added, these increases tend to get smaller and smaller. When they become too small, the sensible thing is to *branch*: having partitioned the set $\mathcal{S}$ of tours into sets $\mathcal{S}_1, \mathcal{S}_2$, apply the cutting-plane method first to one of the subproblems

$$\text{minimize } c^T x \text{ subject to } x \in \mathcal{S}_i$$

and then to the other. At some later time, one or both of these subproblems can be split into sub-subproblems, and so on. In the resulting binary tree of subproblems, each leaf has been either solved by the cutting-plane method without recourse to branching or else found irrelevant when the optimal value of its LP relaxation turned out to be at least as large as the cost of a previously known tour. The standard way of splitting a problem into subproblems is

$$\mathcal{S}_1 = \{x \in \mathcal{S} : x_e = 0\}, \quad \mathcal{S}_2 = \{x \in \mathcal{S} : x_e = 1\} \tag{10}$$

for a suitably chosen edge e; Clochard and Naddef [1993] advocated

$$\mathcal{S}_1 = \{x \in \mathcal{S} : \eta(S,x) = 0\}, \quad \mathcal{S}_2 = \{x \in \mathcal{S} : \eta(S,x) \geq 2\} \tag{11}$$

for a suitably chosen subset S of V. Our computer code chooses the most appealing of all options (10), (11); in our experience with the larger TSP instances, this policy reduces the size of the tree of subproblems.

Every subproblem in the tree has the form

$$\text{minimize } c^T x \text{ subject to } x \in \mathcal{S},\ Cx \leq d$$

for some system $Cx \leq d$ of linear inequalities. When this subproblem is attacked by the cutting-plane method, the initial LP relaxation is

$$\text{minimize } c^T x \text{ subject to } Ax \leq b,\ Cx \leq d$$

with $\mathcal{S} \subseteq \{x : Ax \leq b\}$ and each cut added to $Ax \leq b$, $Cx \leq d$ is satisfied by all x in $\mathcal{S} \cap \{x : Cx \leq d\}$; this is a variant of the *branch-and-bound* method. In the *branch-and-cut* method, used by Hong [1972], Miliotis [1976], Padberg and Rinaldi [1987, 1991], and others, cuts are restricted to those satisfied by all x in $\mathcal{S}$ and added to $Ax \leq b$; this system, acquiring more and more inequalities as more and more subproblems are being processed, may be used to initialize the cutting-plane method on any as yet unprocessed subproblem. Our computer code uses the branch-and-cut method.

3 Experimental results

Computer codes for the TSP have become increasingly more sophisticated over the years. A conspicuous sign of these improvements is the increasing size of the nontrivial instances that have been solved: a 120-city problem by Grötschel [1980], a 318-city problem by Crowder and Padberg [1980], a 532-city problem by Padberg and Rinaldi [1987], a 666-city problem by Grötschel and Holland [1991], a 1,002-city problem and a 2,392-city problem by Padberg and Rinaldi [1991].

In the table below, we report the results of running our computer code on these instances, as well as on five others. With the exception of the 13,509-city instance, our code was run on a single processor of a Digital AlphaServer 4100 (400 MHz). The 13,509-city TSP was run on a network of 48 workstations, including Digital Alphas, Intel Pentium IIs and Pentium Pros, and Sun UltraSparcs. The

Name	Cities	Tree of subproblems	Running time
gr120	120	1 node	3.3 seconds
lin318	318	1 node	24.6 seconds
pr1002	1,002	1 node	94.7 seconds
gr666	666	1 node	260.0 seconds
att532	532	3 nodes	294.3 seconds
pr2392	2,392	1 node	342.2 seconds
ts225	225	1 node	438.9 seconds
pcb3038	3,038	193 nodes	1.5 days
fnl4461	4,461	159 nodes	1.7 days
pla7397	7,397	129 nodes	49.5 days
usa13509	13,509	9,539 nodes	∼10 years

reported time for this instance is an estimate of the cumulative CPU time spent on the individual machines.

The problems reported in the table come from the set TSPLIB of test instances collected by Reinelt [1991]. We sorted them by their solution time, rather than by their size, to emphasize that the difficulty of an instance depends on factors other than just its number of cities. In particular, ts225 is a contrived nasty instance that was first solved only in 1994—three years after it first appeared in TSPLIB. We will present results for the full set of 110 TSPLIB problems in a comprehensive report of our TSP work that we are preparing.

Our computer code (written in the C programming language) is available for research purposes. It can be obtained over the internet at the page:

`http://www.caam.rice.edu/~keck/concorde.html`

REFERENCES

[1968] M. Bellmore and G.L. Nemhauser, "The traveling salesman problem: a survey", *Operations Research* 16 (1968) 538–558.

[1976] K.S. Booth and G.S. Lueker, "Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithms", *Journal of Computer Systems and Science* 13 (1976) 335–379.

[1998] R. Carr, "Separation and lifting of TSP inequalities", manuscript, 1998.

[1997] B.V. Cherkassky and A.V. Goldberg, "On implementing the push-relabel method for the maximum flow problem", *Algorithmica* 19 (1997) 390–410.

[1995] T. Christof and G. Reinelt, "Parallel cutting plane generation for the TSP", in: *Parallel Programming and Applications* (P. Fritzson, L. Finmo, eds.), IOS Press, 1995, pp. 163–169.

[1995] V.Chvátal, "Edmonds polytopes and a hierarchy of combinatorial problems", *Discrete Mathematics* 4 (1973) 305–337.

[1993] J.-M. Clochard and D. Naddef, "Using path inequalities in a branch and cut code for the symmetric traveling salesman problem", in: *Third IPCO Conference* (G. Rinaldi and L. Wolsey, eds), 1993, pp. 291–311.

[1985] G. Cornuéjols, J. Fonlupt, and D. Naddef, "The traveling salesman problem on a graph and some related integer polyhedra", *Mathematical Programming* 33 (1985) 1–27.

[1980] H. Crowder and M.W. Padberg, "Solving large-scale symmetric travelling salesman problems to optimality", *Management Science* 26 (1980) 495–509.

[1954] G.B. Dantzig, R. Fulkerson, and S.M. Johnson, "Solution of a large-scale traveling salesman problem", *Operations Research* 2 (1954) 393–410.

[1962] L.R. Ford, Jr. and D.R. Fulkerson, *Flows in Networks*, Princeton University Press, 1962.

[1958] R.E. Gomory, "Outline of an algorithm for integer solutions to linear programs", *Bulletin of the American Mathematical Society* 64 (1958) 275–278.

[1980] M. Grötschel, "On the symmetric travelling salesman problem: solution of a 120-city problem", *Mathematical Programming Study* 12 (1980) 61–77.

[1991] M. Grötschel and O. Holland, "Solution of large-scale symmetric travelling salesman problems", *Mathematical Programming* 51 (1991) 141–202.

[1975] M. Grötschel and M.W. Padberg, *On the Symmetric Travelling Salesman Problem*, Report No.7536-OR, Institut für Ökonometrie und Operations Research, Universität Bonn, 1975.

[1979a] M. Grötschel and M.W. Padberg, "On the symmetric travelling salesman problem I: Inequalities", *Mathematical Programming* 16 (1979) 265–280.

[1979b] M. Grötschel and M.W. Padberg, "On the symmetric travelling salesman problem II: lifting theorems and facets", *Mathematical Programming* 16 (1979) 281–302.

[1986] M. Grötschel and W.R. Pulleyblank, "Clique tree inequalities and the symmetric travelling salesman problem", *Mathematics of Operations Research* 11 (1986) 1–33.

[1972] S. Hong, *A Linear Programming Approach for the Traveling Salesman Problem*, Ph.D. Thesis, The Johns Hopkins University, 1972.

[1942] R.J. Jessen, "Statistical investigation of a sample survey for obtaining farm facts", *Research Bulletin* #304, Iowa State College of Agriculture, 1942.

[1995] M. Jünger, G. Reinelt, and G. Rinaldi, "The traveling salesman problem", in: *Handbook on Operations Research and Management Sciences: Networks* (M. Ball, T. Magnanti, C.L. Monma, and G. Nemhauser, eds.), North-Holland, 1995, pp. 225–330.

[1972] R.M. Karp, "Reducibility among combinatorial problems", in: *Complexity of Computer Computations* (R.E. Miller and J.W. Thatcher, eds.), Plenum Press, 1972, pp. 85–103.

[1955] H.W. Kuhn, "On certain convex polyhedra", Abstract 799t, *Bulletin of the American Mathematical Society* 61 (1955) 557–558.

[1985] E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, and D.B. Shmoys, eds., *The Traveling Salesman Problem*, Wiley, Chichester, 1985.

[1940] P.C. Mahalanobis, "A sample survey of the acreage under jute in Bengal", *Sankhyu* 4 (1940) 511–530.

[1966] G.T. Martin, "Solving the traveling salesman problem by integer linear programming", *Operations Research* 14 (Supplement 1), Abstract WA7.10.

[1975] J.F. Maurras, "Some results on the convex hull of Hamiltonian cycles of symmetric complete graphs", in: *Combinatorial Programming: Methods and Applications* (B. Roy, ed.), Reidel, Dordrecht, 1975, pp. 179–190.

[1976] P. Miliotis, "Integer programming approaches to the travelling salesman problem", *Mathematical Programming* 10 (1976) 367–378.

[1992] D. Naddef and G. Rinaldi, "The graphical relaxation: A new framework for the symmetric traveling salesman polytope", *Mathematical Programming* 58 (1992) 53–88.

[1998] D. Naddef and G. Rinaldi, "The symmetric traveling salesman polytope: New facets from the graphical relaxation", in preparation.

[1980] M.W. Padberg and S. Hong, "On the symmetric travelling salesman problem: a computational study", *Mathematical Programming Study* 12 (1980) 78–107.

[1982] M.W. Padberg and M.R. Rao, "Odd minimum cut-sets and *b*-matchings", *Mathematics of Operations Research* 7 (1982) 67–80.

[1987] M. Padberg and G. Rinaldi, "Optimization of a 532-city symmetric traveling salesman problem by branch and cut", *Operations Research Letters* 6 (1987) 1–7.

[1990] M. Padberg and G. Rinaldi, "An efficient algorithm for the minimum capacity cut problem", *Mathematical Programming* 47 (1990) 19–36.

[1991] M. Padberg and G. Rinaldi, "A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems", *SIAM Review* 33 (1991) 60–100.

[1991] G. Reinelt, "TSPLIB - A traveling salesman library", *ORSA Journal on Computing* 3 (1991) 376–384.

[1994] G. Reinelt, *The Traveling Salesman: Computational Solutions for TSP Applications*, Springer-Verlag, Berlin, 1994.

[1949] J.B. Robinson, "On the Hamiltonian game (a traveling-salesman problem)", *RAND Research Memorandum* RM-303, 1949.

David Applegate
Rice University
Computational and Applied Math
Houston, TX 77005-1892
david@caam.rice.edu

Robert Bixby
Rice University
Computational and Applied Math
Houston, TX 77005-1892
bixby@caam.rice.edu

Vašek Chvátal
Rutgers University
Department of Computer Science
New Brunswick, NJ 08903
chvatal@cs.rutgers.edu

William Cook
Rice University
Computational and Applied Math
Houston, TX 77005-1892
bico@caam.rice.edu