Mathematical Programming 51 (1991) 141-202
North-Holland

Solution of large-scale symmetric travelling salesman problems

Martin Grötschel*
Institut für Mathematik, Universität Augsburg, W-8900 Augsburg, Germany

Olaf Holland**
Forschungsinstitut für Diskrete Mathematik, Institut für Operations Research, Universität Bonn, W-5300 Bonn, Germany

Received 18 July 1988
Revised manuscript received 31 July 1989

In this paper we report on a cutting plane procedure with which we solved symmetric travelling salesman problems of up to 1000 cities to optimality. Our implementation is based on a fast LP-solver (IBM's MPSX) and makes effective use of polyhedral results on the symmetric travelling salesman polytope. We describe the important ingredients of our code and give an extensive documentation of its computational performance.

AMS Subject Classifications: 05C04, 05C45, 90C10.

Key words: Travelling salesman problem, cutting plane algorithms, polyhedral combinatorics.

Introduction

Developing theory for the travelling salesman problem (TSP) and solving TSP's has always been one of the central subjects of mathematical programming. The TSP has not only fascinated mathematical programmers, operations researchers, and economists. Now also physicists, engineers, biologists, and chemists get excited about this problem. Reasons for this are certainly the facts that the TSP is easy to state, it has very nice applications, but it is hard to solve.

This paper contributes to the solvability aspects of the TSP. We describe an algorithm and its implementation with which large-scale travelling salesman problems can be solved to optimality. We see our work in a long line of attempts to use linear programming techniques and exploit information about the facet structure of the travelling salesman polytope. The history of this approach — outlined in

* Supported by DFG-Schwerpunkt "Anwendungsbezogene Optimierung und Steuerung", Universität Augsburg, Germany.
** Supported by SFB 303 (DFG), Forschungsinstitut für Diskrete Mathematik, Institut für Operations Research, Universität Bonn, Germany.

Grötschel and Padberg (1985), Padberg and Grötschel (1985) — started with the seminal paper Dantzig, Fulkerson and Johnson (1954) and reached a temporary peak in Crowder and Padberg (1980), who solved a 318 city problem, the largest problem solved to optimality until recently.

About 1980 we decided to make more systematic use of the classes of facets known for the travelling salesman polytope and to design a more powerful cutting plane algorithm to solve TSP's to optimality. There were many stimuli for this project. Around this time the ellipsoid method came into focus (with a revival of cutting plane ideas), the importance of polynomial time separation algorithms was discovered (see, e.g., Grötschel, Lovász and Schrijver, 1981, 1988), and Padberg and Rao (1982) invented a fast separation algorithm for the perfect 2-matching problem. Moreover, the computing power available increased considerably so that we hoped to be able to at least double (in terms of the number of nodes) the size of problems that can be solved to optimality. Our goal was to get close to the 1000 city barrier.

For this reason, one of us made up a geographical problem with 666 cities, he thought to be very hard, as a challenge for our work. It turned out to be just that in many respects. For instance, not only the integrality stipulations caused difficulties. Some linear programs that arose were hard to solve, even for highly praised commercial LP-codes like IBM's MPSX. But we finally managed to overcome all these obstacles.

We report here about the result of our work over these last years. Of course, this was not a continuous effort, and there are much longer periods of neglect of the problem than actual design and coding phases. We now feel that a stage is reached where our code has attained its limits. We have achieved our initial goal to solve travelling salesman problems up to 1000 cities. This, though, is still not a routine matter and requires a substantial amount of computing time on large computers; see Section 5. Some of the initial steps of our code design, restricted to certain relaxations of the TSP, can be found in Grötschel and Holland (1985, 1987). A much more detailed documentation of the code and its design is Holland (1987).

Let us mention here that Manfred W. Padberg and Giovanni Rinaldi have, in the recent years, developed a cutting plane code for the TSP that is based on the same approach and uses very similar ideas, many of which have been outlined in Padberg and Grötschel (1985). The design and the "tricks" of their code have not been documented completely yet, though some of the important ingredients appeared in Padberg and Rinaldi (1987, 1990a,b). In fact, the announcement in Padberg and Rinaldi (1987a) that a 2392 city problem was solved to optimality is really breath-taking.

Before describing our work we would like to add a few "philosophical" remarks concerning the questions: "What are all these efforts good for?" "Isn't it better to stick to heuristics?"

Everybody knows that travelling salesman problems may come up in tremendous sizes in practice. For instance, Bland and Shallcross (1987) report about problems

from crystallography with up to 15 000 cities; we know drilling problems (for printed circuit boards) of up to 60 000 cities. These problems seem far out of reach for our present (exact) algorithmic machinery. For the time being, these sizes can only be handled (approximately) with fast heuristics. We also do not advocate to solve, e.g., certain practical 1000 city drilling problems by running our code for a couple of hours in order to save one minute of drilling time. But there are some large-scale instances where knowing exact optima is important.

Optimization tools should not be applied blindly. One has to estimate whether or not it pays to use them, whether exact or approximate methods are the appropriate tools. We view our work mainly as a contribution to the state of the art of exact problem solving using LP-techniques and cutting plane procedures combined with heuristics and branch & bound.

Beautiful structural and algorithmic theory has been developed in the recent years. If one considers mathematical programming as a branch of applied mathematics, this should not remain just theory, it has to be put to work. The implementation process is more than straightforward and — at times — frustrating work. Often new interesting and challenging theoretical problems arise that have to be solved. But most of all it is the justification and validation of our scientific approach. The challenge of our time is large scale and we have to enlarge our algorithmic toolbox in various ways. We should not only confine ourselves to simple heuristics. Even for really large scale problems exact optimization is sometimes possible (and necessary). In addition, if only good approximate solutions are needed, the approach described here can be used heuristically in many ways to obtain excellent upper and lower bounds.

There is a further reason for our work. When starting this project, we had in mind to show that polyhedral combinatorics is not only nice theory but also a powerful algorithmic approach. We believe that the findings presented in this paper and the computational results of many similar projects completed in the recent years corroborate our point of view.

1. Notation

We will briefly mention a few symbols and definitions needed in the sequel.

We denote *graphs* by $G=(V, E)$, where V is the *node set* and E the *edge set*. All our graphs are *simple*, i.e., contain no loops and no multiple edges. An edge e with *endnodes* i and j is denoted by $e=ij$. The (up to isomorphism) unique graph on n nodes where every two nodes are adjacent is called *complete* and is denoted by K_n. The node set of a complete subgraph of a graph is called a *clique*. If G is a connected graph and W is a node set such that its removal disconnects G then W is called an *articulation set*. A Hamiltonian cycle (a cycle that contains every node of the graph exactly once) is also called a *tour*.

For a graph $G=(V, E)$ and $W \subseteq V$, we write

$$\delta(W) := \{ij \in E \mid i \in W, j \in V \setminus W\} \quad (=\delta(V \setminus W)),$$

$$E(W) := \{ij \in E \mid i, j \in W\}.$$

The edge set $\delta(W)$ is called the *cut* induced by W. If $W=\{v\}$ we write $\delta(v)$ instead of $\delta(\{v\})$.

If E is a finite set, then $\mathbb{R}^E$ denotes the set of functions from E to $\mathbb{R}$. This set is a real vector space and can be viewed as the set of vectors $x=(x_e)_{e \in E}$ where each component is indexed by an element of E. If $x \in \mathbb{R}^E$ and $F \subseteq E$ we write $x(F)$ to denote the sum $\sum_{e \in F} x_e$. The *incidence vector* $\chi^F \in \mathbb{R}^E$ of $F \subseteq E$ is the vector defined by $\chi_e^F = 1$ if $e \in F$, $\chi_e^F = 0$ if $e \notin F$.

A set $P \subseteq \mathbb{R}^E$ is a polytope if it is the convex hull of finitely many points. An inequality $a^T x \leqslant \alpha$ is *valid* with respect to P if $P \subseteq \{x \in \mathbb{R}^E \mid a^T x \leqslant \alpha\}$. A valid inequality $a^T x \leqslant \alpha$ defines a *facet* of P if $H := \{x \in P \mid a^T x = \alpha\}$ has dimension one less than P. An important fact from polyhedral theory is the following. If $P \subseteq \mathbb{R}^E$ is a polytope then there are an equation system $Ax=b$ and an inequality system $Dx \leqslant d$ such that $P=\{x \in \mathbb{R}^E \mid Ax=b, Dx \leqslant d\}$, A has full row rank and each inequality of $Dx \leqslant d$ defines a facet of P.

Finally, the (symmetric) travelling salesman problem is the following. Given a complete graph $K_n=(V, E)$ and distances c_{ij} for each edge $ij \in E$. Find a tour T with $c(T)$ as small as possible. Without loss of generality, we will assume throughout the paper that *all distances c_{ij} are integral.*

2. A short summary of some polyhedral results

To avoid some trivial technicalities let us assume from now on that the number n of cities (or nodes of the complete graph K_n) is at least 6.

Given a complete graph $K_n=(V, E)$, the (symmetric) *travelling salesman polytope* Q_T^n is the convex hull of all incidence vectors of tours of K_n. Thus

$$Q_T^n = \text{conv}\{\chi^T \in \mathbb{R}^E \mid T \subseteq E \text{ is a tour}\}.$$

The interest in this polytope derives from the fact that the symmetric travelling salesman problem can be solved by solving $\min\{c^T x \mid x \in Q_T^n\}$ which — in some sense — is a linear program. The polytope Q_T^n has been the subject of intensive investigations. A quite complete summary of the results on Q_T^n published to date can be found in Grötschel and Padberg (1985). (Let us mention, though, that very recently D. Naddef and G. Rinaldi and S. Boyd and B. Cunningham (personal communication) have discovered large new classes of facet-defining inequalities for Q_T^n.) We will briefly describe those equations and inequalities valid for Q_T^n that will be used in the sequel.

The affine hull of Q_T^n is defined by the linearly independent equations

$$x(\delta(v))=2 \quad \text{for all } v \in V. \tag{2.1}$$

Thus $\dim(Q_T^n) = |E| - |V|$. The *trivial inequalities*

$$0 \leqslant x_e \leqslant 1 \quad \text{for all } e \in E \tag{2.2}$$

also define facets of Q_T^n as well as the *subtour elimination constraints* (see Grötschel and Padberg, 1979) introduced by Dantzig, Fulkerson and Johnson (1954)

$$x(E(W)) \leqslant |W| - 1 \quad \text{for all } W \subseteq V,\ 3 \leqslant |W| \leqslant n-3. \tag{2.3}$$

Using (2.1) one can see that an inequality $x(E(W)) \leqslant |W| - 1$ is equivalent (defines the same facet) to $x(E(V \setminus W)) \leqslant |V \setminus W| - 1$. This in turn is equivalent to the *cut constraint* $x(\delta(W)) = x(\delta(V \setminus W)) \geqslant 2$. So the system of cut constraints

$$x(\delta(W)) \geqslant 2 \quad \text{for all } W \subseteq V,\ 3 \leqslant |W| \leqslant n-3, \tag{2.4}$$

defines the same facets of Q_T^n as the system (2.3).

Let $H, T_1, \ldots, T_s$ be a system of subsets of V. The inequality

$$x(E(H)) + \sum_{i=1}^{s} x(E(T_i)) \leqslant |H| + \sum_{i=1}^{s} (|T_i| - 1) - \lceil \tfrac{1}{2} s \rceil \tag{2.5}$$

is called a *2-matching constraint* (introduced in Edmonds (1965) to give a complete description of the 2-matching polytope) if $H, T_1, \ldots, T_s$ satisfy

$$|T_i \cap H| = 1, \quad i = 1, \ldots, s, \tag{2.6a}$$

$$|T_i \setminus H| = 1, \quad i = 1, \ldots, s. \tag{2.6b}$$

(2.5) is called *comb constraint* if $H, T_1, \ldots, T_s$ satisfy

$$|T_i \cap H| \geqslant 1, \tag{2.6a'}$$

$$|T_i \setminus H| \geqslant 1. \tag{2.6b'}$$

Grötschel and Padberg (1979b) proved that a 2-matching constraint or a comb constraint defines a facet of Q_T^n if, in addition, the node sets $H, T_1, \ldots, T_s$ satisfy

$$s \geqslant 3 \cdot \text{and } s \text{ odd}, \tag{2.6c}$$

$$T_i \cap T_j = \emptyset, \quad 1 \leqslant i < j \leqslant s. \tag{2.6d}$$

The following class of valid inequalities for Q_T^n, which contains all nontrivial facet-defining inequalities listed above, was introduced by Grötschel and Pulleyblank (1986).

A *clique tree* is a connected graph C composed of cliques that satisfy the following properties (in the following we shall always consider clique trees as subgraphs of K_n):

(i) The cliques are partitioned into two sets, the set of *handles* and the set of *teeth*.

(ii) No two teeth intersect.

(iii) No two handles intersect.

(iv) Each tooth contains at least two and at most $n-2$ nodes and at least one node not belonging to any handle.

(v) The number of teeth that each handle intersects is odd and at least three.

(vi) If a tooth T and a handle H have a nonempty intersection, then $H \cap T$ is an articulation set of the clique tree.

Grötschel and Pulleyblank (1986) showed that, for every clique tree C with handles $H_1, \ldots, H_r$ and teeth $T_1, \ldots, T_s$, the following *clique tree inequality* defines a facet of Q_T^n,

$$\sum_{i=1}^{r} x(E(H_i)) + \sum_{j=1}^{s} x(E(T_j))$$

$$\leqslant \sum_{i=1}^{r} |H_i| + \sum_{j=1}^{s} (|T_j| - t_j) - \tfrac{1}{2}(s+1) =: s(C), \tag{2.7}$$

where, for every tooth T_j, the integer t_j denotes the number of handles that intersect T_j. Note that the facet-defining comb inequalities are exactly the clique tree inequalities associated with clique trees with only one handle.

Setting

$$Q_S^n := \{x \in \mathbb{R}^n \mid x \text{ satisfies (2.1), (2.2), (2.3)}\}, \tag{2.8a}$$

$$Q_{2M}^n := \{x \in \mathbb{R}^n \mid x \text{ satisfies (2.1), (2.2), and the 2-matching constraints (2.5), (2.6a,b)}\}, \tag{2.8b}$$

$$Q_C^n := \{x \in \mathbb{R}^n \mid x \text{ satisfies (2.1), (2.2), and the comb constraints (2.5), (2.6a',b')}\}, \tag{2.8c}$$

$$Q_{CT}^n := \{x \in \mathbb{R}^n \mid x \text{ satisfies (2.1), (2.2), (2.7)}\}, \tag{2.8d}$$

we see that $Q_T^n \subseteq Q_{CT}^n \subseteq Q_C^n \subseteq Q_{2M}^n$ and $Q_T^n \subseteq Q_{CT}^n \subseteq Q_S^n$.

Our approach to solving $\min c^T x$, $x \in Q_T^n$, is to use linear programming relaxations that can be defined by the polyhedra Q_S^n, Q_{2M}^n, Q_C^n and Q_{CT}^n. We will see later that the linear program

$$\min\{c^T x \mid x \in Q_S^n \cap Q_{2M}^n\}$$

which has a number of constraints that is exponential in n can be solved in polynomial time (in theory) by the ellipsoid method. In practice it can be solved by a simplex-based cutting plane procedure with reasonable efficiency. We do not know how to solve linear programs of the form $\min\{c^T x \mid x \in Q_C^n\}$, $\min\{c^T x \mid x \in Q_{CT}^n\}$, or $\min\{c^T x \mid x \in Q_T^n\}$ in theory or practice efficiently but we are able to generate some comb and some clique tree inequalities through separation heuristics (see Section 4). Thus or LP-based attack on the TSP ends with an optimum solution x^* of a linear program

$$\min\{c^T x \mid x \in Q\},$$

where Q is a polytope that contains Q_T^n and is contained in $Q_S^n \cap Q_{2M}^n$. If x^* is the incidence vector of a tour, we are done; otherwise we resort to branch & bound.

3. Outline of the code

We have explained the "philosophy" of our polyhedral approach to the TSP above. It is, however, a nontrivial and time consuming task to make this idea work.

We sketch now the important basic ingredients of our code. Details on cutting plane generation will be presented in Section 4. We assume that an instance of the symmetric travelling salesman problem is given by the number n of cities and by distances $c_{ij} \in \mathbb{Z}$, $1 \leq i < j \leq n$. The code has the four stages indicated in Figure 3.1.

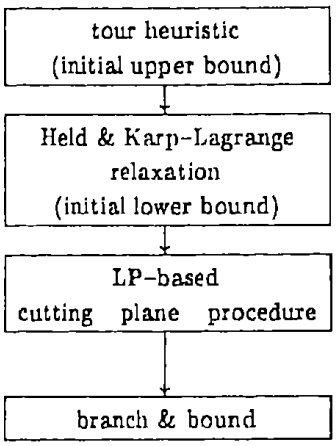

Fig. 3.1.

3.1. *Preprocessing*

The first two stages are mere preprocessing phases that help to speed up the cutting plane and the branch & bound phase. Their role is the following.

(1) *Tour heuristic.* By running heuristic procedures we generate several tours. The best tour found is later used to set up a starting basis for the initial LP of the cutting plane phase. The set $\hat{E}$ of edges that appear in at least one of the tours generated is stored. It is used to set up the initial linear program. Moreover, the length U of the best tour is utilized in the branch and bound phase as an upper bound.

We have chosen the following heuristic procedure. We first run the next neighbor heuristic 50 times starting with randomly chosen nodes. Each of the (different) tours produced this way is used as a starting tour for our implementation of the Lin-Kernighan heuristic — see Lin and Kernighan (1973). These methods are well-known and it is not necessary to describe the details here.

We would like to point out, however, that the Lin-Kernighan heuristic is rather time consuming. Running the procedure described in (1) in the standard way requires as much and sometimes even more time than the whole cutting plane and branch & bound method to be described later. Considerable parameter adjustments and coding efforts are necessary to make this heuristic run in acceptable time. On the

other hand, we (and of course others as well) have noticed that a good upper bound is as crucial as a good lower bound for a successful branch & bound phase. Thus, the quality of the upper bound achieved in this stage does have a serious impact on the overall performance of the algorithm.

Recall that our goal was to create a code that can solve symmetric travelling salesman problems of up to 1000 cities to optimality. This requires handling linear programs of up to half a million variables. We do not know any LP-code that can solve such linear programs. Thus, it is necessary to reduce the dimensions considerably. The reduction must, of course, be done in such a way that global optimality can still be proved. A first reduction step is based on the following procedure.

(2) *Held and Karp-Lagrange relaxation.* We solve the 1-tree relaxation described in Held and Karp (1970, 1971), using a standard subgradient algorithm. The largest value found in this procedure gives a lower bound L for the optimum tour length.

In theory, the solution of the Lagrange relaxation of the 1-tree problem gives the optimum value of $\min\{c^T x \mid x \in Q_S''\}$. In practice, however, this value is rarely obtained through this procedure. By making good choices in the step length parameter etc. of the subgradient algorithm, good lower bounds for $\min\{c^T x \mid x \in Q_S''\}$ can, in fact, be computed quickly.

(3) *First variable reduction.* Using the upper bound U found in (1), the lower bound L and the best 1-tree T found in (2) we can eliminate some edges based on a standard reduced cost criterion. For instance, if $e \in E \setminus (T \cap \delta(1))$, where 1 is the special node of the 1-tree, we know that $T \cup \{e\}$ contains exactly one cycle C not containing node 1. Suppose $\bar{c}_{ij} = c_{ij} + \lambda_i + \lambda_j$ is the cost of edge e, where λ_i, $i \in V$, is the set of best Lagrange multipliers. If f is an edge with largest cost $\bar{c}_f$ among the edges in $C \setminus \{e\}$ then edge e can be eliminated if $\bar{c}_e - \bar{c}_f > U - L - 1$. A similar criterion can be used for the edges $e \in \delta(1) \setminus T$.

Let $\bar{E} \subseteq E$ be the set of edges that can be eliminated due to these criteria, and let H be the tour giving the upper bound U. Then $E' := (E \setminus \bar{E}) \cup H$ is a set of edges that is guaranteed to contain at least one optimum tour of the original TSP. Thus we can restrict ourselves to considering the subgraph $G = (V, E')$ of K_n.

This finishes the description of the preprocessing phases. The variable elimination procedure described above is quite effective. Table 3.1 gives an overview of the results achieved in this way. The columns have the following meaning. "Problem (NUM)" is our name for the instance of the TSP. The number appearing in the name gives the number of cities of the TSP. (Details about the problems can be found in the appendix.) The second column shows the lower bound computed by the method described in (2). The third column reports the upper bound found in (1). The fourth column "%GAP" shows the maximal possible deviation (in percent) of the length U of the tour found in (1) from the optimum tour length. It is computed by means of the formula $(U - L) * 100/L$. The fifth column "Problem variables"

Table 3.1
Heuristic upper and lower bounds, reduction of variables

Problem (NUM)	Lower bound	Upper bound	%GAP	Problem variables	%VAR
17	2 047.7274	2 085	1.82	40	29.41
21	2 696.6135	2 707	0.39	31	14.76
24	1 265.2850	1 272	0.53	41	14.85
42	684.1273	699	2.17	188	21.84
48	4 953.3513	5 046	1.87	210	18.62
48H	11 425.9497	11 461	0.31	84	7.45
57	12 758.6891	12 985	1.77	332	20.80
70	669.0735	675	0.89	223	9.23
96	54 544.2939	55 209	1.22	557	12.21
100A	20 920 6091	21 282	1.73	681	13.76
100B	21 736.3089	22 141	1.86	847	17.11
100C	20 460.2047	20 749	1.41	619	12.51
100D	20 999.9530	21 294	1.40	617	12.46
100E	21 770.0528	22 068	1.37	602	12.16
100R	9 653.7772	9 690	0.38	181	3.66
120	6 902.3900	6 951	0.70	528	7.39
137	68 926.6767	69 853	1.34	1 196	12.84
200R	9 550.3396	9 653	1.07	721	3.62
202	39 502.0277	40 214	1.80	7 249	35.71
229	133 180.3029	134 666	1.12	3 879	14.86
300R	10 282.9372	10 424	1.37	1 742	3.88
318	31 182.3861	31 404	0.71	2 753	5.46
400R	9 496.0431	9 617	1.27	2 520	3.16
431	170 121.4302	171 778	0.97	24 354	26.28
442	5 038.7512	5 083	0.89	7 990	8.20
500R	9 161.2755	9 271	1.20	3 503	2.81
532	27 357.0196	27 829	1.73	43 282	30.64
600R	9 571.4626	9 732	1.68	6 654	3.70
666	292 188.0715	296 371	1.43	66 913	30.22
700R	10 120.1419	10 305	1.83	10 052	4.11
800R	10 094.6417	10 286	1.90	13 406	4.19
900R	9 995.6538	10 233	2.38	20 544	5.08
1000R	9 962.3615	10 156	1.94	20 774	4.16

contains the number $|E'|$ of variables remaining after the elimination procedure (3) has been executed. The sixth column "%VAR" shows the percentage of the number of remaining variables $|E'|$ compared to the number $|E|$ of original variables. E.g., in problem 500R only 2.81% of the variables are left for the final optimization step, while in problem 202 still 35.71% of the variables remain to be processed.

3.2. *The cutting plane phase*

This phase is the core of our algorithm. We enter it from the preprocessing phase with a subgraph $G' = (V, E')$ of $K_n = (V, E)$, a tour H whose length gives the upper bound U and with the set $\hat{E} \subseteq E'$ of edges that appeared in at least one of the

tours generated by the heuristic of (1). A flow chart of our cutting plane procedure can be found in Figure 3.2.

The aim of this phase is twofold. We want to produce a "very good" lower bound for the optimum tour value by LP-techniques and we want to set up a linear program whose 0/1-solutions contain the incidence vector of an optimum tour. To do this

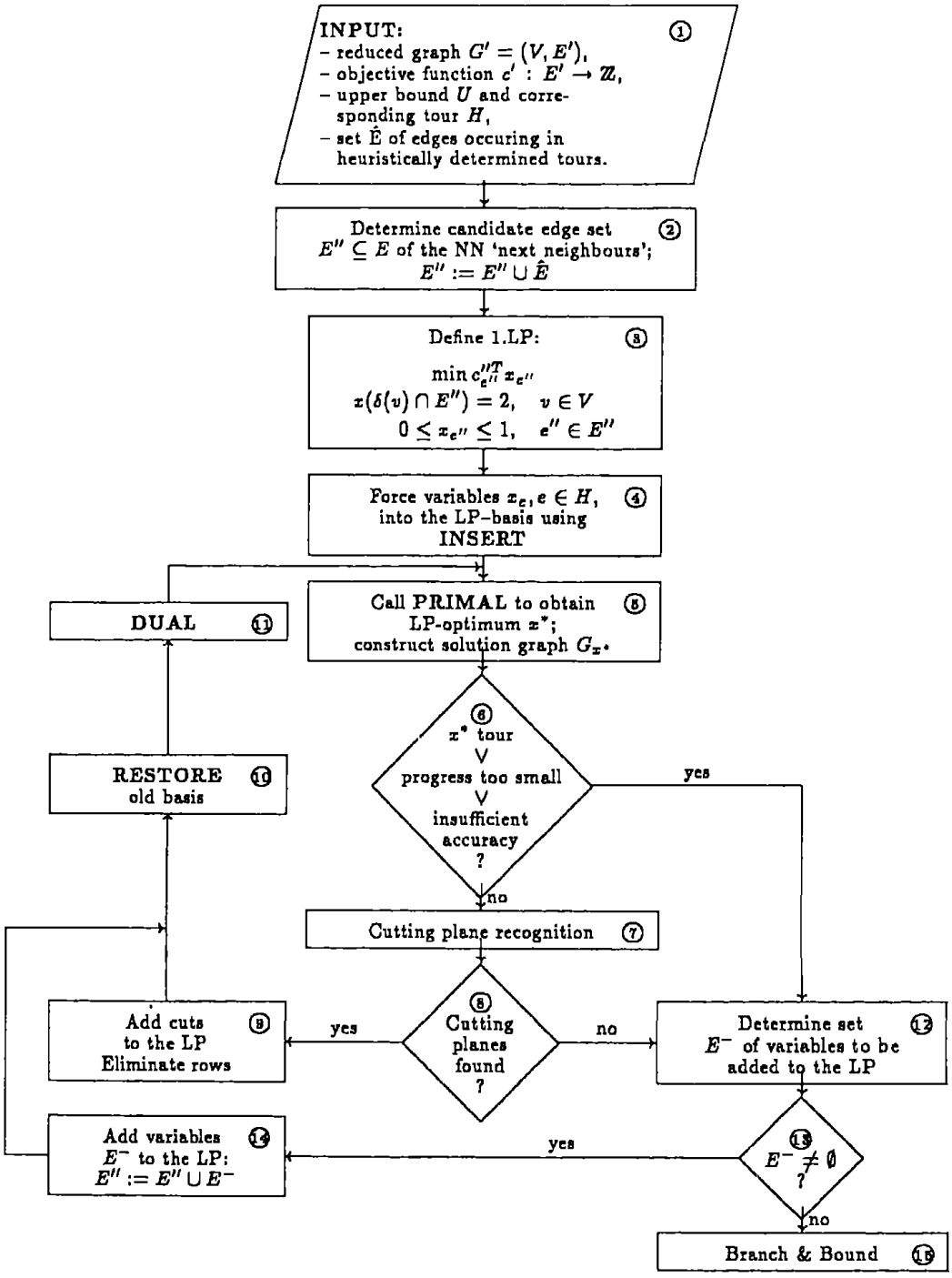

Fig. 3.2.

we first generate a set of edges $E'' \subseteq E$ and an inequality system $A''x'' \leqslant b''$, $x'' \in \mathbb{R}^{E''}$, such that the value of the linear program

$$\begin{aligned} &\text{minimize} && c''^{\mathrm{T}}x'' \\ &\text{subject to} && A''x'' \leqslant b'' \end{aligned} \tag{3.1}$$

is a true (and good) lower bound for the length of a shortest tour (c'' is the restriction of the vector c to the components E'').

The matrix A'' and the edge set E'' are not defined in advance. They are a result of the row and column generation scheme to be explained below. A'' has (by construction) the property that it can be extended in a canonical way to a matrix A with $|E|$ columns such that all inequalities of the system $Ax \leqslant b$ are either equations of the form (2.1) or define facets of Q_T^n. Moreover, $\min\{c''^{\mathrm{T}}x'' \,|\, A''x'' \leqslant b''\} = \min\{c^{\mathrm{T}}x \,|\, Ax \leqslant b\}$ holds.

Secondly A'' and E'' have the property that, by using reduced cost criteria, an extension of A'' to a matrix $\tilde{A}$ and an extension of E'' to a set of variables $\tilde{E}$ is possible such that the solution set of the 0/1-linear program

$$\begin{aligned} &\text{minimize} && \tilde{c}^{\mathrm{T}}\tilde{x} \\ &\text{subject to} && \tilde{A}\tilde{x} \leqslant \tilde{b}, \\ & && \tilde{x} \in \{0,1\}^{\tilde{E}}, \end{aligned} \tag{3.2}$$

contains the incidence vector of a shortest tour of the original problem. The 0/1-program (3.2) is the input to the branch & bound part of the algorithm, unless the solution (3.1) is the incidence vector of a tour (and we do not call branch & bound).

To achieve the goals described above, it would seem sensible to choose the edge set E' (of remaining variables) as the set of variables E'' to set up the linear program (3.1). Although our elimination procedure (3) is quite successful (see column %VAR of Table 3.1) the number of variables $|E'|$ is, in general, still much too large (e.g., $|E'| = 66\,913$ for the 666-city problem), even for fast commercial LP-solvers. So we decided to do the following.

(4) *Selection of initial variables.* We initially select a "candidate set" E'' of edges (of which we hope that they will contain an optimum tour) as follows. Depending on the parameter NN (we have used $0 \leqslant \mathrm{NN} \leqslant 10$) to be set before execution of the algorithm, we determine, for each node $v \in V$, a subset E_v of edges of $\delta(v)$ with cardinality NN having smallest length among all edges in $\delta(v)$ and set

$$E'' := \bigcup_{v \in V} E_v \cup \hat{E},$$

where $\hat{E}$ is the set of edges occurring in heuristically determined determined tours, see Section 3.1. This procedure is indicated in Box 2 of Figure 3.2. $\hat{E}$ is added to the "next neighbour edges" $\bigcup_{v \in V} E_v$ to guarantee the existence of a tour in

$G'' = (V, E'')$. The cutting plane procedure is initialized with the variable set E'', see Box 3 of Figure 3.2.

We now outline the LP solution techniques, the basics, and the main loop of the cutting plane part of our algorithm. These are indicated in Boxes $3, 4, \ldots, 11$ of Figure 3.2.

To solve the linear programs coming up we used IBM's package MPSX/370. This contains a quite fast LP-solver, though, for the application to be described in this paper, it does have some drawbacks that will be discussed later.

(5) *Initial LP and initial basis.* The initial linear program is defined in the standard fashion. We generate all degree constraints (2.1) and the upper and lower bounds (2.2). As outlined before we restrict the LP to the initial variables E'' defined in (4). Thus our first LP is of the form

$$\begin{aligned} &\text{minimize} && c''^{\mathrm{T}}x \\ &\text{subject to} && x(\delta(v) \cap E'') = 2 \text{ for all } v \in V, \\ &&& 0 \leqq x_{e''} \leqq 1 \text{ for all } e'' \in E''. \end{aligned} \tag{3.3}$$

It is well known that, for every tour of the graph $G'' = (V, E'')$, one can determine a basis of (3.3) with the given tour as associated basic solution. We initialize our LP-solver by introducing the best tour H known at present as a starting basis using the MPSX-routine INSERT. (By the choice of E'', we have $H \subseteq E''$.) This process is indicated in Boxes 3 and 4 of Figure 3.2.

We now enter the main loop through Boxes $5, 6, \ldots, 11$ of Figure 3.2 and describe a general step.

To call the MPSX-routine PRIMAL in Box 5 we have to know a basis of our present LP. This is at hand in the first call due to (5). In a general step, a basis will be part of the output of the routine DUAL called in Box 11. PRIMAL determines an optimum solution x^* of the present LP. To process and analyse x^* we generate the graph $G_{x^*} := (V, E_{x^*})$ defined by

$$E_{x^*} := \{e \in E'' \mid x_e^* > 0\}. \tag{3.4}$$

The next step, Box 6 of Figure 3.2, consists of a couple of tests. We first check whether x^* is the incidence vector of a tour. If this is the case we go to the variable generation procedure of Box 12. Then we check whether the cutting plane procedure has "tailed off". We do this in the following way. Every tenth time we enter Box 6 we compare the present optimum LP-value γ^* with the optimum value γ of the linear program solved ten iterations (of the main loop) before. If $\gamma^* - \gamma \leqq 1$ we feel that further cutting plane generations will not pay and exit from the loop to Box 12 of Figure 3.2. In a third test we check for numerical accuracy. MPSX offers some parameters to do this. If we feel that the present accuracy is insufficient, we leave