BRANCH AND BOUND METHODS FOR THE TRAVELING SALESMAN PROBLEM

by

Egon Balas
Carnegie-Mellon University

and

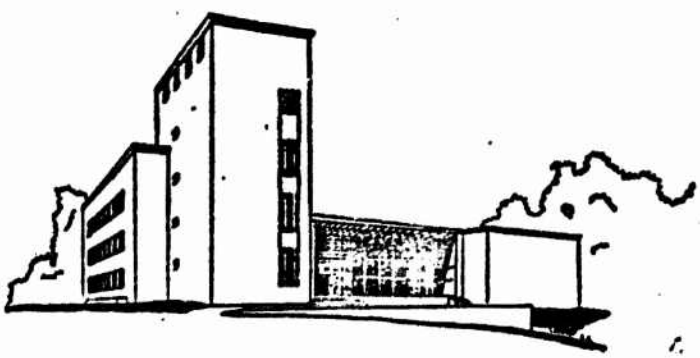
Paolo Toth
University of Florence

Carnegie-Mellon University

PITTSBURGH, PENNSYLVANIA 15213

GRADUATE SCHOOL OF INDUSTRIAL ADMINISTRATION

WILLIAM LARIMER MELLON, FOUNDER



W.P.#45-82-83

Management Science Research Report No. MSRR 488

BRANCH AND BOUND METHODS FOR THE TRAVELING SALESMAN PROBLEM

by

Egon Balas
Carnegie-Mellon University

and

Paolo Toth
University of Florence

March 1983

The research of the first author was supported by Grant ECS-8205425 of the National Science Foundation and Contract N00014-75-C-0621 NR 047-048 with the U.S. Office of Naval Research; and that of the second author by the Centro Nazionale delle Ricerche of Italy.

Management Science Research Group
Graduate School of Industrial Administration
Carnegie-Mellon University
Pittsburgh, Pennsylvania 15213

ABSTRACT

This paper reviews the state of the art in enumerative solution methods for the traveling salesman problem (TSP). The introduction (Section 1) discusses the main ingredients of branch and bound methods for the TSP. Sections 2, 3 and 4 discuss classes of methods based on three different relaxations of the TSP: the assignment problem with the TSP cost function, the 1-tree problem with a Lagrangean objective function, and the assignment problem with a Lagrangean objective function. Section 5 briefly reviews some other relaxations of the TSP, while Section 6 discusses the performance of some state of the art computer codes. Besides material from the literature, the paper also includes the results and statistical analysis of some computational experiments designed for the purposes of this review.

BRANCH AND BOUND METHODS
FOR THE TRAVELING SALESMAN PROBLEM

by

Egon Balas and Paolo Toth

1. Introduction

Since the first attempt to solve traveling salesman problems by an enumerative approach, apparently due to Eastman [1958], many such procedures have been proposed. In a sense the TSP has served as a testing ground for the development of solution methods for discrete optimization, in that many procedures and devices were first developed for the TSP and then, after successful testing, extended to more general integer programs. The term "branch and bound" itself was coined by Little, Murty, Sweeney and Karel [1963] in conjunction with their TSP algorithm.

Enumerative (branch and bound, implicit enumeration) methods solve a discrete optimization problem by breaking up its feasible set into successively smaller subsets, calculating bounds on the objective function value over each subset, and using them to discard certain subsets from further consideration. The bounds are obtained by replacing the problem over a given subset with an easier (relaxed) problem, such that the solution value of the latter bounds that of the former. The procedure ends when each subset has either produced a feasible solution, or was shown to contain no better solution than the one already in hand. The best solution found during the procedure is a global optimum.

For any problem P, we denote by $v(P)$ the value of (an optimal solution to) P. The essential ingredients of any branch and bound procedure for a discrete optimization problem P of the form $\min\{f(x) \mid x \in S\}$ are

(i) a relaxation of P, i.e. a problem R of the form $\min\{g(x) \mid x \in T\}$, such that $S \subseteq T$ and for every $x, y \in S$, $f(x) < f(y)$ implies $g(x) < g(y)$.

(ii) a branching or separation rule, i.e. a rule for breaking up the feasible set S_i of the current subproblem P_i into subsets $S_{i1}, \ldots, S_{iq}$, such that $\bigcup_{j=1}^{q} S_{ij} = S_i$;

(iii) a lower bounding procedure, i.e. a procedure for finding (or approximating from below) $v(R_i)$ for the relaxation R_i of each subproblem P_i; and

(iv) a subproblem selection rule, i.e. a rule for choosing the next subproblem to be processed.

Additional ingredients, not always present but always useful when present, are

(v) an upper bounding procedure, i.e. a heuristic for finding feasible solutions to P; and

(vi) a testing procedure, i.e., a procedure for using the logical implications of the constraints and bounds to fix the values of some variables (reduction, variable fixing) or to discard an entire subproblem (dominance tests).

For more information on enumerative methods in integer programming see, for instance, Chapter 4 of Garfinkel and Nemhauser [1972], and/or the surveys by Balas [1975], Balas and Guignard [1979], Beale [1979], Speilberg [1979].

Since by far the most important ingredient is (i), we will classify the branch and bound procedures for the TSP according to the relaxation that they use.

The integer programming formulation of the TSP that we will refer to when discussing the various solution methods is defined on a complete directed graph $G = (V,A)$ on n nodes, with node set $V = \{1,\ldots,n\}$, arc set $A = \{(i,j) | i,j = 1,\ldots,n\}$, and nonnegative costs c_{ij} associated with the arcs. The fact that G is

complete involves no restriction, since arcs that one wishes to ignore can be assigned the cost $c_{ij} = \infty$. In all cases $c_{ii} = \infty$, $\forall\, i \in V$. The TSP can be formulated, following Dantzig, Fulkerson and Johnson [1954], as the problem

$$(1) \qquad \min \sum_{i \in V} \sum_{j \in V} c_{ij} x_{ij}$$

s.t.

$$(2) \qquad \sum_{j \in V} x_{ij} = 1 , \quad i \in V$$

$$\sum_{i \in V} x_{ij} = 1 , \quad j \in V$$

$$(3) \qquad \sum_{i \in S} \sum_{j \in S} x_{ij} \leq |S| - 1 , \qquad \forall\, S \subsetneq V,\ S \neq \emptyset$$

$$(4) \qquad x_{ij} = 0 \text{ or } 1, \quad i,j \in V.$$

where $x_{ij} = 1$ if arc (i,j) is in the solution, $x_{ij} = 0$ otherwise.

The subtour elimination inequalities (3) can also be written as

$$(5) \qquad \sum_{i \in S} \sum_{j \in V \setminus S} x_{ij} \geq 1 , \qquad \forall\, S \subsetneq V,\ S \neq \emptyset$$

A very important special case is the symmetric TSP, in which $c_{ij} = c_{ji}$, $\forall i,j$. The symmetric TSP can be defined on a complete undirected graph $G = (V,E)$ on n nodes, with node set V, edge set E, and arbitrary costs c_{ij}. It can be stated as

$$(6) \qquad \min \sum_{i \in V} \sum_{j > i} c_{ij} x_{ij}$$

s.t.

$$(7) \qquad \sum_{j < i} x_{ji} + \sum_{j > i} x_{ij} = 2 , \qquad i \in V$$

$$(8) \qquad \sum_{i \in S} \sum_{\substack{j \in S \\ j>i}} x_{ij} \leq |S|-1, \qquad \forall\, S \subsetneq V,\ S \neq \emptyset$$

$$(9) \qquad x_{ij} = 0 \text{ or } 1, \qquad i,j \in V,\ j>i$$

where the subtour elimination inequalities (8) can also be written as

$$(10) \qquad \sum_{i \in S} \sum_{\substack{j \in V \setminus S \\ j>i}} x_{ij} + \sum_{i \in V \setminus S} \sum_{\substack{j \in S \\ j>i}} x_{ij} \geq 2, \qquad \forall\, S \subsetneq V,\ S \neq \emptyset$$

Next we outline two versions of a branch and bound procedure for the TSP. Prior to using any of these versions, a relaxation R of the TSP must be chosen. Both versions carry at all times a list of active subproblems. They differ in that version 1 solves a (relaxed) subproblem R_k only when node k is selected and taken off the list, while version 2 solves each (relaxed) subproblem as soon as it is created, i.e. before it is placed on the list. Although the branch and bound procedures used in practice differ among themselves in many details, nevertheless all of them can be viewed as variants of one of these two versions.

Branch and bound method for the TSP

Version 1

1. (Initialization). Put TSP on the list (of active subproblems). Initialize the upper bound at $U = \infty$, and go to 2.
2. (Subproblem selection). If the list is empty, stop: the tour associated with U is optimal (or, if $U = \infty$, TSP has no solution). Otherwise choose a subproblem TSP_i according to the subproblem selection rule, remove TSP_i from the list, and go to 3.

3. (Lower bounding). Solve the relaxation R_i of TSP_i or bound $v(R_i)$ from below, and let L_i be the value obtained.

   If $L_i \geq U$, return to 2.

   If $L_i < U$ and the solution defines a tour for TSP, store it in place of the previous best tour, set $U \leftarrow L_i$, and go to 5.

   If $L_i < U$ and the solution does not define a tour, go to 4.

4. (Upper bounding: optional). Use a heuristic to find a tour in TSP. If a better tour is found than the current best, store it in place of the latter and update U. Go to 5.

5. (Reduction: optional). Remove from the graph of TSP_i all the arcs whose inclusion in a tour would raise its value above U, and go to 6.

6. (Branching). Apply the branching rule to TSP_i, i.e. generate new subproblems $TSP_{i1},\ldots, TSP_{iq}$, place them on the list, and go to 2.

Version 2

1. (Initialization). Like in version 1, but solve R before putting TSP on the list.

2. (Subproblem selection). Same as in version 1.

3. (Upper bounding: optional). Same as step 4 of version 1, with "go to 5" replaced by "go to 4."

4. (Reduction: optional). Same as step 5 of version 1, with "go to 6" replaced by "to go 5."

5. (Branching). Use the branching rule to define the set of subproblems $TSP_{i1},\ldots,TSP_{iq}$ to be generated from the current subproblem TSP_i, and go to 6.

6. (Lower bounding). If all the subproblems to be generated from TSP_i according to the branching rule have already been generated, go to 2. Otherwise generate the next subproblem TSP_{ij} defined by the branching rule, solve the relaxation R_{ij} of TSP_{ij} or bound $v(R_{ij})$ from below, and let L_{ij} be the value obtained.

If $L_{ij} \geq U$, return to 6.

If $L_{ij} < U$ and the solution defines a tour for TSP, store it in place of the previous best tour, set $U \leftarrow L_{ij}$, and go to 6.

If $L_{ij} < U$ and the solution does not define a tour, place TSP_{ij} on the list and return to 6.

In both versions, the procedure can be represented by a rooted tree (search or branch and bound tree) whose nodes correspond to the subproblems generated, with the root node corresponding to the original problem, and the successor nodes of a given node i associated with TSP_i corresponding to the subproblems $TSP_{i1},\ldots,TSP_{iq}$ defined by the branching rule.

It is easy to see that under very mild assumptions on the branching rule and the relaxation used, both versions of the above procedure are finite (see Exercise 1).

Next we discuss various specializations of the procedure outlined above, classified according to the relaxation that they use. When assessing and comparing the various relaxations, one should keep in mind that a "good" relaxation is one that (i) gives a strong lower bound, i.e. yields a small difference $v(TSP) - v(R)$; and (ii) is easy to solve. Naturally, these are often conflicting goals, and in such cases one has to weigh the tradeoffs.

2. Relaxation I: The Assignment Problem with the TSP Cost Function

The most straightforward relaxation of the TSP, and historically the first one to have been used, is the problem obtained from the integer programming formulation (1), (2), (3), (4) by removing the constraints (3), i.e. the assignment problem (AP) with the same cost function as TSP. It was used, among others, by Eastman [1958], Little, Murty, Sweeney and Karel [1963], Shapiro [1966], Bellmore and Malone [1971], Smith, Srinivasan and Thompson [1977], Carpaneto and Toth [1980].

An assignment (i.e., a solution to AP) is a union of directed cycles, hence either a tour, or a collection of subtours. There are $n!$ distinct assignments, of which $(n-1)!$ are tours. Thus on the average one in every n assignments is a tour. Furthermore, in the current context only those assignments are of interest that contain no diagonal elements (i.e., satisfy $x_{ii} = 0$, $i=1,\cdots,n$), and their number is $n!/e$ rounded to the nearest integer, i.e. $\lfloor n!/e + 1/2 \rfloor$ (see, for example, Hall [1967], p. 10). Thus on the average one in every n/e "diagonal-free" assignments is a tour. This relatively high frequency of tours among assignments suggests that $v(AP)$ is likely to be a pretty strong bound on $v(TSP)$, and computational experience with AP-based solution methods supports such a view. To test how good this bound actually is for randomly generated problems, we performed the following experiment. We generated 400 problems with $50 \leq n \leq 250$ with the costs independently drawn from a uniform distribution of the integers over the intervals [1,100] and [1,1000], and solved both AP and TSP. We found that on the average $v(AP)$ was 99.2% of $v(TSP)$. Furthermore, we found the bound to improve with problem size, in that for the problems with $50 \leq n \leq 150$ and $150 \leq n \leq 250$ the outcomes were 98.8% and 99.6%, respectively.

AP can be solved by the Hungarian method (Kuhn [1955]; for a more recent treatment, see Christofides [1975] or Lawler [1976]) in at most $O(n^3)$ steps. The assignment problems AP_i to be solved at every node of the search tree differ from the initial assignment problem AP in that some arcs are excluded (forbidden) from, while other arcs are included (forced) into the solution. These modifications do not present any difficulty. Since the Hungarian method provides at every iteration a lower bound on v(AP), the process of solving a subproblem can be stopped whenever the lower bound meets the upper bound U. More importantly, in the typical case (see the branching rules below), the assignment problem AP_j to be solved at node j of the search tree differs from the problem AP_i solved at the parent node i only in that a certain arc belonging to the optimal solution of AP_i is excluded from the solution of AP_j, and possibly some other arcs are required to maintain the same position (or out) with respect to the solution of AP_j, that they have with respect to the solution of AP_i. Whenever this is the case, the problem AP_j can be solved by the Hungarian method starting from the optimal solution of the problem at the parent node (or at a brother node), in at most $O(n^2)$ steps (see Exercise 2 or Bellmore and Hong [1971]). For an efficient implementation of this version of the Hungarian method, which uses on the average considerably less than $O(n^2)$ steps, see Carpaneto and Toth [1980]. The primal simplex method for the assignment problem has been used in a parametric version to solve efficiently this sequence of related assignment problems by Smith, Srinivasan and Thompson [1977].

The lower bound v(AP) can be slightly improved by addition of a penalty. This can be calculated as the minimal change in the objective function caused either by a first simplex pivot that eliminates some arc from the solution, or by a first iteration of the Hungarian method that accomplishes the same thing. Furthermore, the arc to be included in the solution by the pivot can be

restricted to a cutset defined by some subtour of the AP solution. Computational experience indicates, however, that the impact of such a penalty tends to decrease with problem size and is negligible for anything but small problems. In the computational experiment involving the 400 randomly generated problems that we ran, the addition of a penalty to v(AP) raised the value of the lower bound on the average by 0.03%, from 99.2% to 99.23% of v(TSP).

Branching rules

Several branching rules have been used in conjunction with the AP relaxation of the TSP. In assessing the advantages and disadvantages of these rules one should keep in mind that the ultimate goal is to solve the TSP by solving as few subproblems as possible. Thus a "good" branching rule is one that (a) generates few successors of a node of the search tree, and (b) generates strongly constrained subproblems, i.e. excludes many solutions from each subproblem. Again, these criteria are usually conflicting and the merits of the various rules depend on the tradeoffs.

We will discuss the various branching rules in terms of sets of arcs excluded (E_k) from, and included (I_k) into the solution of subproblem k. In terms of the variables x_{ij}, the interpretation of these sets is that subproblem k is defined by the conditions

$$(11) \qquad x_{ij} = \begin{cases} 0, & (i,j) \in E_k \\ 1, & (i,j) \in I_k \end{cases}$$

in addition to (1), (2), (3), (4). Thus the relaxation of subproblem k is given by (11) in addition to (1), (2), (4). We abbreviate Branching Rule by BR.

BR 1. (Little, Murty, Sweeney and Karel [1963]). Given the current relaxed subproblem AP_k and its reduced costs $\bar{c}_{ij} = c_{ij} - u_i - v_j$, where u_i and v_j are optimal dual variables, for every arc (i,j) such that $\bar{c}_{ij} = 0$ define the penalty

$$p_{ij} = \min \left\{ \bar{c}_{ih} : h \in V \setminus \{j\} \right\} + \min \left\{ \bar{c}_{hj} : h \in V \setminus \{i\} \right\}$$

and choose $(r,s) \in A$ such that

$$p_{rs} = \max \left\{ p_{ij} : \bar{c}_{ij} = 0 \right\} .$$

Then generate two successors of node k, nodes $k+1$ and $k+2$, by defining

$$E_{k+1} = E_k \cup \{(r,s)\} , \qquad I_{k+1} = I_k$$

and

$$E_{k+2} = E_k , \qquad I_{k+2} = I_k \cup \{(r,s)\} .$$

This rule does not use the special structure of the TSP (indeed, it applies to any integer program), and has the disadvantage that it leaves the optimal solution to AP_k feasible for AP_{k+2}.

The following rules are based on disjunctions derived from the subtour elimination inequalities (3) or (5).

BR 2. (Eastman [1958], Shapiro [1966]). Let x^k be the optimal solution to the current relaxed subproblem AP_k, and let $A_S = \{(i_1,i_2),\ldots,(i_t,i_1)\}$ be the arc set of a minimum cardinality subtour of x^k involving the node set $S = \{i_1,\ldots,i_t\}$. Constraint (3) for S implies the inequality

$$(3') \qquad \sum_{(i,j) \in A_S} x_j \leq |S| - 1,$$

which in turn implies the disjunction

$$(12) \qquad x_{i_1 i_2} = 0 \ \vee \ \ldots \ \vee \ x_{i_t i_1} = 0 .$$

Generate t successors of node k, defined by

$$\left.\begin{aligned} E_{k+r} &= E_k \cup\{(i_r, i_{r+1})\} \\ I_{k+r} &= I_k \end{aligned}\right\} \quad r = 1,\ldots,t \tag{13}$$

with $i_{t+1} = i_1$.

Now x^k is clearly infeasible for all AP_{k+r}, $r = 1,\ldots,t$, and the choice of a shortest subtour for branching keeps the number of successor nodes small. However, the disjunction (12) does not define a partition of the feasible set of AP_k, and thus different successors of AP_k may have solutions in common. This shortcoming is remedied by the next rule, which differs from BR 2 only in that it strengthens the disjunction (12) to one that defines a partition.

BR 3. (Murty [1968], Bellmore-Malone [1971], Smith, Srinivasan and Thompson [1977]). The disjunction (12) can be strengthened to

$$(x_{i_1 i_2} = 0) \vee (x_{i_1 i_2} = 1,\ x_{i_2 i_3} = 0) \vee \ldots \vee (x_{i_1 i_2} = \ldots = x_{i_{t-1} i_t} = 1,\ x_{i_t i_1} = 0). \tag{14}$$

and accordingly (13) can be replaced by

$$\left.\begin{aligned} E_{k+r} &= E_k \cup\{(i_r, i_{r+1})\} \\ I_{k+r} &= I_k \cup\{(i_1, i_2), \ldots, (i_{r-1}, i_r)\} \end{aligned}\right\} \quad r = 1,\ldots,t \tag{15}$$

with $i_{t+1} = i_1$.

A slightly different version of BR 3 (as well as of BR 2) is to replace the edge set A_S of a minimum-cardinality subtour with that of a subtour with a

minimum number of free edges (i.e. edges of $E\setminus E_k \cup I_k$). This rule is used in Carpaneto and Toth [1980].

BR 4. (Bellmore and Malone [1971]). Let x^k and S be as before. Constraint (5) implies the disjunction

$$(16) \qquad (x_{i_1 j} = 0,\ j\in S) \vee (x_{i_2 j} = 0,\ j\in S) \vee \ldots \vee (x_{i_t j} = 0,\ j\in S).$$

Generate t successors of node k, defined by

$$(17) \qquad \left.\begin{aligned} E_{k+r} &= E_k \cup \{(i_r, j) : j\in S)\} \\ I_{k+r} &= I_k \end{aligned}\right\} \quad r = 1,\ldots,t$$

Like in the case of BR 2, Br 4 makes x^k infeasible for all successor problems of AP_k, but again (16) does not partition the feasible set of AP_k. This is remedied by the next rule, which differs from BR 4 only in that it defines a partition.

BR 5. (Garfinkel [1973]). The disjunction (16) can be strengthened to

$$(18) \qquad (x_{i_1 j} = 0,\ j\in S) \vee (x_{i_1 j} = 0,\ j\in V\setminus S;\ x_{i_2 j} = 0,\ j\in S) \vee \ldots$$
$$\vee (x_{i_r j} = 0,\ j\in V\setminus S, r = 1,\ldots,t-1;\ x_{i_t j} = 0,\ j\in S)$$

and accordingly (17) can be replaced by

$$(19) \qquad \left.\begin{aligned} E_{k+r} &= E_k \cup \{(i_r, j):\ j\in S\} \cup \{(i_q, j):\ q=1,\ldots,r-1,\ j\in V\setminus S\} \\ I_{k+r} &= I_k \ . \end{aligned}\right.$$

The two rules BR 2 and BR 4 (or their strengthened variants, BR 3 and BR 5), based on the subtour-elimination constraints (3′) and (5), respectively, generate the same number of successors of a given node k. However, the rule based on inequality (5) generates more tightly constrained subproblems, i.e., excludes a greater number of assignments from the feasible set of each successor problem, than the rule based on inequality (3′). Indeed, with $|S| = k$, we have

Theorem 1. (Bellmore and Malone [1971]). Each inequality (3′) eliminates $\lfloor (n-k)!/e + 1/2 \rfloor$ diagonal-free assignments, whereas each inequality (5) eliminates $\lfloor (n-k)!/e + 1/2 \rfloor \cdot \lfloor k!/e + 1/2 \rfloor$ diagonal-free assignments.

Proof. Each inequality (3′) eliminates those diagonal-free assignments that contain the subtour with arc set A_S. There are as many such assignments as there are diagonal-free assignments in the complete graph defined on node set $V \backslash S$, and the number of these is $(n-k)!/e$ rounded to the nearest integer, i.e., $\lfloor (n-k)!/e + 1/2 \rfloor$ (see section 2).

On the other hand, each inequality (5) eliminates those diagonal-free assignments consisting of the union of two such assignments, one in the complete graph defined on S, the other in the complete graph defined on $V \backslash S$. Since the number of the latter is $\lfloor (n-k)!/e + 1/2 \rfloor$ and that of the former is $\lfloor k!e + 1/2 \rfloor$, the number of diagonal-free assignments eliminated by each inequality (5) is as stated in the theorem.‖

Nevertheless, both Smith, Srinivasan and Thompson [1977] and Carpaneto and Toth [1980] found their respective implementations of BR 3 more efficient than BR 4 or BR 5, both in terms of total computing time and number of nodes generated. We have no good explanation for this.

Other features

The subproblem selection rule used by many branch and bound algorithms is the one known as "depth first" or LIFO (last in first out). It amounts to choosing one of the nodes generated at the last branching step (in order, for instance, of nondecreasing penalties, like in Smith, Srinivasan and Thompson [1977]); and when no more such nodes exist, backtracking to the parent node and applying the same rule to its brother nodes. This rule has the advantage of modest storage requirements and easy bookkeeping. Its disadvantage is that possible erroneous decisions (with respect to arc exclusion or inclusion) made early in the procedure cannot be corrected until late in the procedure.

The alternative extreme is known as the "breadth first" rule, which amounts to always choosing the node with the best lower bound. This rule has the desirable feature of keeping the size of the search tree as small as possible, (see Exercise 3), but on the other hand requires considerable storage space. In order to keep simple the passage from one subproblem to the next one, this rule must be embedded in a procedure patterned after version 2 of the outline in the introduction, which solves each assignment problem as soon as the corresponding node is generated, and places on the list only those subproblems TSP_{ij} with $L_{ij} < U$. The procedure of Carpaneto and Toth [1980] uses this rule, and it chooses the subproblems to be processed (successors of a given node) in the order defined by the arc adjacencies in the subtour that serves as a basis for the branching.

As mentioned earlier, the high frequency of tours among assignments makes AP a relatively strong relaxation of TSP, which in the case of random (asymmetric) costs provides an excellent lower bound on v(TSP). However, in the case of the symmetric TSP, the bound given by the optimal AP solution is substantially weaker. An experiment that we ran on 140 problems with $40 \leq n \leq 100$ and with symmetric costs independently drawn from a uniform distribution of the integers

in the interval [1, 1000], showed v(AP) to be on the average 82% of v(TSP), while the addition of a penalty raised the bound to 85%. The explanation of the relative weakness of this bound is pretty straightforward: in the symmetric case, there is a tendency towards a certain symmetry also in the solution, to the effect that if $x_{ij} = 1$, then (since $c_{ij} = c_{ji}$), one tends to have $x_{ji} = 1$ too; and thus the optimal AP solution usually contains a lot of subtours of length 2, irrespective of the size of n. Thus as a rule, a much larger number of subtours has to be eliminated before finding an optimal tour in the symmetric case than in the asymmetric one. This makes the AP a poor relaxation for the symmetric TSP.

3. Relaxation II: The 1-Tree Problem with Lagrangean Objective Function

This relaxation was successfully used for the symmetric TSP first by Held and Karp [1970, 1971] and Christofides [1970], and subsequently by Helbig Hansen and Krarup [1974], Smith and Thompson [1977], Volgenant and Jonker [1982].

Consider the symmetric TSP and the undirected (complete) graph $G = (V,E)$ associated with it. The problem of finding a connected spanning subgraph H of G with n edges, that minimizes the cost function (6), is obviously a relaxation of the symmetric TSP. Such a subgraph H consists of a spanning tree of G, plus an extra edge. We may further restrict H to the class $\mathcal{T}$ of subgraphs of the above type in which some arbitrary node of G, say node 1, has degree 2 and is contained in the unique cycle of H. For lack of a better term, the subgraphs of this class $\mathcal{T}$ are called 1-trees. To see that finding a 1-tree that minimizes (6) is a relaxation of the TSP, it suffices to realize that the constraint set defining the family $\mathcal{T}$ is (9) and

$$(20)\qquad \sum_{i\in S}\sum_{\substack{j\in V\setminus S\\ j>i}} x_{ij} + \sum_{i\in V\setminus S}\sum_{\substack{j\in S\\ j>i}} x_{ij} \geq 1, \qquad \forall\, S \subsetneq V,\ S\neq\emptyset$$

$$(21)\qquad \sum_{i\in V}\sum_{j>i} x_{ij} = n$$

$$(22)\qquad \sum_{j\in V} x_{1j} = 2$$

Here (20) is a weakening of (10), (21) is the sum of all equations (7) divided by two, and (22) is the first equation (7).

The minimum-cost 1-tree problem is easily seen to be decomposable into two independent problems:

(α) to find a minimum-cost spanning tree in $G - \{1\}$; and

(β) to find two smallest-cost edges among those incident in G with node 1.

The $n-2$ edges of the spanning tree found under (α), together with the 2 edges found under (β), form a minimum-cost 1-tree in G.

Solving problem (β) requires $O(n)$ comparisons, whereas problem (α) can be efficiently solved by the algorithms of Dijkstra [1959] or Prim [1957], of complexity $O(n^2)$, or by the algorithm of Kruskal [1956], of complexity $O(|E| \log |E|)$. Since the $\log |E|$ in the last expression comes from sorting the edges, a sequence of subproblems that requires only minor resorting of the edges between two members of the sequence can be more efficiently solved by Kruskal's procedure than by the other two.

The number of 1-trees in the complete undirected graph G on n nodes can be calculated as follows: the number of distinct spanning trees in $G - \{1\}$ is $(n-1)^{n-3}$ (Cailey's formula), and from each spanning tree one can get $\binom{n-1}{2}$ distinct 1-trees by inserting two edges joining node 1 to the tree. Thus the number of

1-trees in G is $\frac{1}{2}(n-2)(n-1)^{n-2}$, which is much higher than the number of solutions to AP. Since G has (n-1)! tours, on the average the number of tours among the 1-trees of a complete undirected graph is one in every $\frac{1}{2}(n-2)(n-1)^{n-3}/(n-2)!$, and hence the minimum-cost 1-tree problem with the same objective function as the TSP is a rather weak relaxation of the TSP. In the above mentioned computational experiment on 140 randomly generated symmetric problems, we also solved the corresponding 1-tree problems and found the value of an optimal 1-tree to be on the average only 63% of v(TSP). However, this relaxation can be considerably strengthened by taking the equations (7) into the objective function in a Lagrangean fashion, and then maximizing the Lagrangean as a function of the multipliers.

The problem

$$(23) \qquad L(\lambda) = \min_{x \in X(\mathcal{T})} \sum_{i \in V} \sum_{j>i} c_{ij}x_{ij} + \sum_{i \in V} \lambda_i \Big(\sum_{j<i} x_{ji} + \sum_{j>i} x_{ij} - 2 \Big)$$

$$= \min_{x \in X(\mathcal{T})} \sum_{i \in V} \sum_{j>i} (c_{ij} + \lambda_i + \lambda_j)x_{ij} - 2 \sum_{i \in V} \lambda_i ,$$

where λ is any n-vector and $X(\mathcal{T})$ is the set of incidence vectors of 1-trees in G, i.e., the set defined by (9), (20), (21), (22), is a Lagrangean relaxation of the TSP. From the last expression in (23) and the fact that $X(\mathcal{T})$ contains all tours, it is easy to see that for any λ, $L(\lambda) \leq v(TSP)$. (For surveys of Lagrangean relaxation in a more general context see Geoffrion [1974], Fisher [1981], Shapiro (1979].) The strongest Lagrangean relaxation is obviously given by $\lambda = \bar{\lambda}$ such that

$$(24) \qquad L(\bar{\lambda}) = \max_{\lambda} L(\lambda) .$$

Problem (24) is sometimes called a Lagrangean dual of the TSP.

Now (24) is a much stronger relaxation than the 1-tree problem with the TSP cost function. Indeed, computational experience with randomly generated problems has produced on the average values of $L(\bar{\lambda})$ of about 99% of v(TSP) according to Christofides [1979] (p. 134), and of about 99.7% of v(TSP) according to Volgenant and Jonker [1982].

However, solving (24) is a lot more difficult than solving a 1-tree problem. The objective function of (24), i.e. the function $L(\lambda)$ of (23), is piecewise linear and concave in λ. Thus $L(\lambda)$ is not everywhere differentiable. Held and Karp [1971], who first used (24) as a relaxation of the TSP, have tried several methods, and found that an iterative procedure akin to the relaxation method of Agmon [1954] and Motzkin and Schoenberg [1954] was the best suited approach for this type of problem. The method, which turned out to have been theoretically studied in the Soviet literature (see Polyak [1967] and others) became the object of extensive investigations in the Western literature under the name of subgradient optimization, as a result of its successful use by Held and Karp in conjunction with the TSP (for surveys of subgradient optimization in a more general context see Held, Wolfe and Crowder [1974], Sandi [1979]).

The subgradient optimization method for solving (24) starts with some arbitrary $\lambda = \lambda^0$ (say the zero vector) and at iteration k updates λ^k as follows. Find $L(\lambda^k)$, i.e. solve problem (23) for $\lambda = \lambda^k$. Let $H(\lambda^k)$ be the optimal 1-tree found. If $H(\lambda^k)$ is a tour, or if $v(H(\lambda^k)) \geq U$, stop. Otherwise, for $i \in V$, let d_i be the degree of node i in $H(\lambda^k)$. Then the n-vector with components $d_i^k - 2$, $i \in V$, is a subgradient of $L(\lambda)$ at λ^k (see Exercise 4). Set

$$(25) \qquad \lambda_i^{k+1} = \lambda_i^k + t^k(d_i^k - 2), \qquad i \in V$$

where t^k is the "step length" defined by

$$(26) \qquad t^k = \alpha(U - L(\lambda^k))/\sum_{i \in V}(d_i^k - 2)^2$$

with $0 < \alpha \leq 2$. Then set $k \leftarrow k+1$ and repeat the procedure.

It can be shown (see any of the surveys mentioned above) that the method converges if $\sum_{k=1}^{\infty} t^k = \infty$ and $\lim_{k \to \infty} t^k = 0$. These conditions are satisfied if one starts with $\alpha = 2$ and periodically reduces α by some factor.

Example 1.

Consider the 8-city symmetric TSP whose graph is shown in Fig. 1 (only arcs with finite cost are present). Initially $U = 25$, $\alpha = 2$, $\lambda_i^o = 0$ for $i = 1,\ldots,8$.

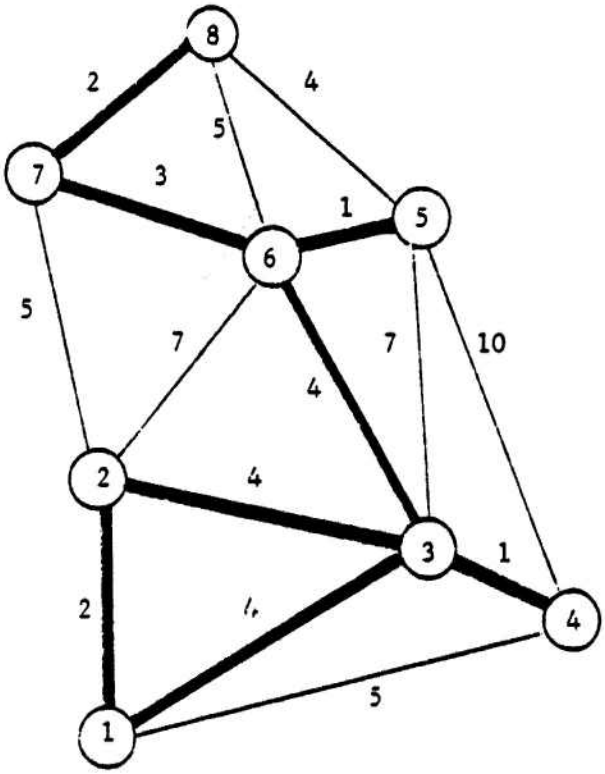


Fig. 1. Initial graph G = (V,E)

The optimal 1-tree, shown in heavy lines in Fig. 1, has a weight of $L(\lambda^0) = 21$.

At iteration 0 we have:

$$d_i^0 = (2, 2, 4, 1, 1, 3, 2, 1);$$

$$t^0 = 2(25-21)/8 = 1;$$

$$\lambda_i^1 = (0, 0, 2, -1, -1, 1, 0, -1).$$

The updated arc costs $(c_{ij} + \lambda_i^1 + \lambda_j^1)$ and the corresponding optimal 1-tree, having a weight of $L(\lambda^1) = 24$, are shown in Fig. 2.

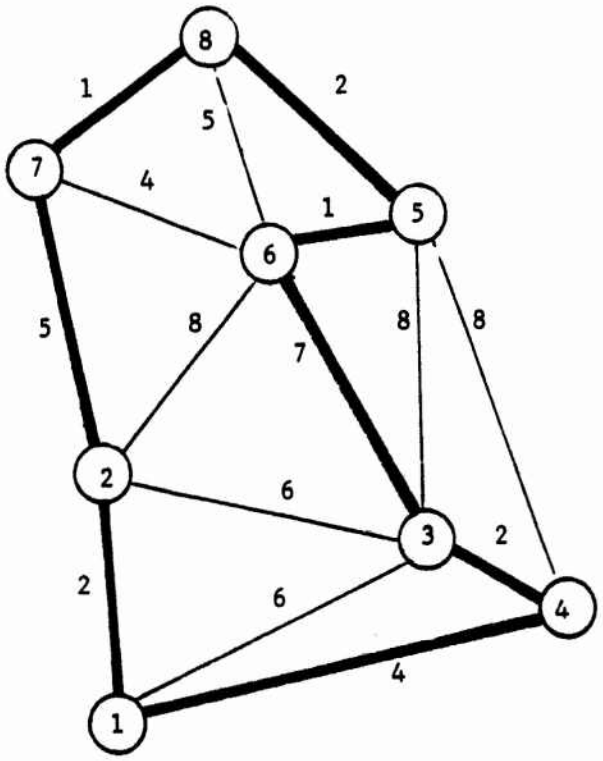


Fig. 2. Updated graph G = (V,E)

We have $d_i^1 = 2$ for $i = 1,\ldots,8$; thus a tour has been found and the procedure stops.

Held and Karp [1971] pointed out that if λ^o is taken to be, instead of 0, the vector defined by

$$\lambda_i^o = -(u_i + v_i)/2 , \qquad i \in V ,$$

where (u, v) is an optimal solution to the dual of the assignment problem with costs $c_{ij} = c_{ji}$, $\forall$ i,j, then one always has $v(H(\lambda^o)) \geq v(AP)$. Indeed, for this choice of λ^o one has from (23)

$$\begin{aligned} L(\lambda^o) &= \min_{x \in X(\mathcal{T})} \sum_{i \in V} \sum_{j>i} (c_{ij} + \lambda_i^o + \lambda_j^o) x_{ij} - 2 \sum_{i \in V} \lambda_i^o \\ &= \min_{x \in X(\mathcal{T})} \sum_{i \in V} \sum_{j>i} \frac{1}{2}[(c_{ij} - u_i - v_j) + (c_{ji} - u_j - v_i)] + \sum_{i \in V} (u_i + v_i) \\ &\geq v(AP), \end{aligned}$$

since $v(AP) = \sum_{i \in V} (u_i + v_i)$ and $c_{ij} - u_i - v_j \geq 0$, $\forall$ i,j.

This kind of initialization requires of course that one solve AP prior to addressing problem (24).

Helbig Hansen and Krarup [1974] and Smith and Thompson [1977] distinguish between the application of the subgradient procedure at the root node of the search tree and at subsequent nodes, by using different starting vectors λ^o and different stopping rules.

Volgenant and Jonker [1982] use an updating formula for λ^k, and an expression for t^k, different from (25) and (26), respectively. Namely, they take t^k to be a positive scalar decreasing according to a series with a constant second order difference, i.e.

$$t^{k+1} - 2t^k + t^{k-1} = \text{constant}, \tag{27}$$

and define λ^{k+1} by setting for $i \in V$,

$$(28) \qquad \lambda_i^{k+1} = \begin{cases} \lambda_i^k & \text{if } d_i^k = 2 \\ \lambda_i^k + 0.6t^k(d_i^k - 2) + 0.4t^k(d_i^{k-1} - 2) & \text{otherwise} \end{cases}$$

It should be mentioned that none of the versions of this subgradient optimization method can be guaranteed to solve (24) in polynomial time with a prespecified degree of accuracy. However, the stopping rules are such that after a certain number of iterations the procedure terminates with an approximation to an optimal λ, which gives a (usually good) lower bound on $L(\bar{\lambda})$.

Branching rules

BR 6. (Held and Karp [1971]). At node k, let the free edges of the current 1-tree (i.e. those in $E \setminus E_k \cup L_k$) be ordered according to nonincreasing penalties, and let the first q elements of this ordered set be $J = \{(i_1,j_1),\ldots,(i_q,j_q)\}$, where q will be specified below. Define q new subproblems by

$$(29) \qquad \begin{aligned} L_{k+r} &= L_k \cup \{(i_h,j_h) : h = 1,\ldots,r-1\} && r = 1,\ldots,q \\ E_{k+r} &= E_k \cup \{(i_r,j_r)\}, && r = 1,\ldots,q-1 \\ E_{k+q} &= E_k \cup \{(i,j) \notin L_{k+q} : i = p \text{ or } j = p\} \end{aligned}$$

Here $p \in V$ is such that L_k contains at most one edge incident with p, while L_{k+q} contains two such edges; and q is the smallest subscript of an edge in J for which a node with the properties of p exists.

This rule partitions the feasible set, and makes the current 1-tree infeasible for each of the new subproblems generated, but the number q of the new subproblems is often larger than necessary.

BR 7. (Smith and Thompson [1977]). Choose a node whose degree in the current 1-tree is not 2, and a maximum-cost edge (i,j) among those incident with the chosen node. Then generate two new subproblems defined by

$$E_{k+1} = E_k \cup \{(i,j)\}, \qquad I_{k+1} = I_k$$
$$E_{k+2} = E_k, \qquad I_{k+2} = I_k \cup \{(i,j)\}. \tag{30}$$

This rule generates only two successors of each node k of the search tree, but the minimum 1-tree in subproblem k remains feasible for subproblem $k + 2$.

BR 8. (Volgenant and Jonker [1982]). Choose a node p whose degree in the current 1-tree exceeds 2. Such a node is incident with at least two free edges, say (i_1,j_1) and (i_2,j_2) (otherwise I_k contains two edges incident with p, hence the remaining edges incident with p belong to or should belong to E_k). Generate three new subproblems defined by

$$E_{k+1} = E_k, \qquad I_{k+1} = I_k \cup \{(i_1,j_1), (i_2,j_2)\};$$
$$E_{k+2} = E_k \cup \{(i_2,j_2)\}, \qquad I_{k+2} = I_k \cup \{(i_1,j_1)\} \tag{31}$$
$$E_{k+3} = E_k \cup \{(i_1,j_1)\}, \qquad I_{k+3} = I_k$$

If p is incident with an edge in I_k, then node k+1 is not generated.

This rule also partitions the feasible set and makes the 1-tree at node k infeasible for each of the successor nodes, while the number of successors of each node is at most 3.

Other features

Held and Karp [1971] and Smith and Thompson [1977] use a depth first subproblem selection rule, while Volgenant and Jonker [1982] have implemented both a depth first and a breadth first rule, with computational results that indicate a slight advantage for the depth first rule (in their implementation).

Extension to the asymmetric TSP

The basic ideas of the 1-tree relaxation of the symmetric TSP carry over to the asymmetric case (Held and Karp [1970]), in that the 1-tree in an undirected graph can be replaced by a 1-arborescence in the directed graph $G = (V,A)$, defined as an arborescence (directed tree) rooted at node 1, plus an arc $(i,1)$ joining some node $i \in V \setminus \{1\}$ to node 1. The constraints defining a 1-arborescence, namely (4) and

$$(32) \qquad \sum_{i \in S} \sum_{j \in V \setminus S} x_{ij} \geq 1 , \qquad \forall\, S \subsetneq V : \{1\} \in S$$

$$(33) \qquad \sum_{i \in V} \sum_{j \in V} x_{ij} = n$$

$$(34) \qquad \sum_{i \in V} x_{i1} = 1$$

are easily seen to be a relaxation of the constraint set (2), (4), (5) of the TSP.

The problem of finding a minimum-cost 1-arborescence can again be decomposed into two independent problems, namely (α) finding a minimum-cost arborescence in G rooted at node 1, and (β) finding a minimum-cost arc $(i,1)$ in G. Problem (α) can be solved by the polynomial time algorithms of Edmonds [1967] or Fulkerson [1974], or by the $O(n^2)$-time algorithm of Tarjan [1977].

To obtain the Lagrangean version of the 1-arborescence relaxation, one forms the function

$$(35) \qquad L(\lambda) = \min_{x \in X(\mathcal{G})} \sum_{i \in V} \sum_{j \in V} c_{ij} x_{ij} + \sum_{i \in V} \lambda_i \Big(\sum_{j \in V} x_{ij} - 1 \Big)$$

$$= \min_{x \in X(\mathcal{G})} \sum_{i \in V} \sum_{j \in V} (c_{ij} + \lambda_i) x_{ij} - \sum_{i \in V} \lambda_i ,$$

where $X(\mathcal{G})$ is the set of incidence vectors of $\mathcal{G}$, the family of 1-arborescences in G. Again, the strongest lower bound on $v(TSP)$ is of course given by $\lambda = \bar{\lambda}$ such that

$$(36) \qquad L(\bar{\lambda}) = \max_{\lambda} L(\lambda) ,$$

and subgradient optimization can be used to solve problem (36). However, computational experience with this relaxation (see Smith [1975]) shows it to be inferior (for asymmetric problems) to the AP relaxation, even when the latter uses the original objective function of the TSP.

4. Relaxation III: the Assignment Problem with Lagrangean Objective Function

This relaxation was used for the asymmetric TSP by Balas and Christofides [1981]. It is a considerable strengthening of the relaxation consisting of the AP with the original cost function, involving a significant computational effort, which however seems amply justified by the computational results that show this approach to be the fastest currently available method for this class of problems.

Consider the asymmetric TSP defined on the complete directed graph $G = (V,A)$, in the integer programming formulation (1), (2), (4), plus the subtour-elimination constraints. The latter can be written either as (3) or as (5), but for reasons

to be explained later, we include <u>both</u> (3) and (5), as well as some positive linear combinations of such inequalities, and write the resulting set of subtour-elimination inequalities in the generic form

$$(37) \qquad \sum_{i \in V} \sum_{j \in V} a_{ij}^t x_{ij} \geq a_o^t , \qquad t \in T .$$

Thus our integer programming formulation of the TSP consists of (1), (2), (4) and (37). To construct a Lagrangean relaxation of TSP, we denote by X the feasible set of AP, and associate a multiplier w_t, $t \in T$, with every inequality in the system (37). We then have

$$(38) \qquad L(w) = \min_{x \in X} \sum_{i \in V} \sum_{j \in V} c_{ij} x_{ij} - \sum_{t \in T} w_t \left(\sum_{i \in V} \sum_{j \in V} a_{ij}^t x_{ij} - a_o^t \right)$$

$$= \min_{x \in X} \sum_{i \in V} \sum_{j \in V} \left(c_{ij} - \sum_{t \in T} w_t a_{ij}^t \right) x_{ij} + \sum_{t \in T} w_t a_o^t ,$$

where $w = (w_t)$. Clearly, the strongest such relaxation is given by $w = \bar{w}$ such that

$$(39) \qquad L(\bar{w}) = \max_{w \geq 0} L(w)$$

The Lagrangean dual (39) of the TSP could be solved by subgradient optimization, like in the case of the 1-tree relaxation of the symmetric TSP. However, in this case the vector w of multipliers has an exponential number of components, and until an efficient way is found to identify the components that need to be changed at every iteration, such a procedure seems computationally expensive. Balas and Christofides [1981] therefore replace (39) by the "restricted" Lagrangean dual

$$(40) \qquad \max_{w \in W} L(w) ,$$

where

$$W = \left\{ w \;\middle|\; \begin{array}{l} w \geq 0 \text{ and there exists } u,v \in R^n \text{ such that} \\ u_i + v_j + \sum_{t \in T} w_t a_{ij}^t \begin{cases} = c_{ij} & \text{if } \bar{x}_{ij} = 1 \\ \leq c_{ij} & \text{if } \bar{x}_{ij} = 0 \end{cases} \end{array} \right\}$$

and $\bar{x}$ is the optimal solution found for the AP.

In other words, (40) restricts the multipliers w_t to values that, together with appropriate values u_i, v_j, form a feasible solution to the dual of the linear program given by (1), (2), (37) and $x_{ij} \geq 0$, $i,j \in V$. This may cause the value of (40) to be less than that of (39), but it leaves the optimal solution $\bar{x}$ to AP, also optimal for the objective function (38). Thus (40) can be solved without changing $\bar{x}$. While no good method is known for the exact solution of (40), Balas and Christofides [1981] give a polynomially bounded sequential noniterative approximation procedure, which yields multipliers $\hat{w}_t$ such that $L(\hat{w})$ typically comes close to v(TSP): for randomly generated asymmetric TSP's, $L(\hat{w})$ was found to be on the average 99.5% of v(TSP) (Christofides [1979], p. 139-140).

The procedure starts by solving AP for the costs c_{ij}, $\forall$ i,j, and taking u_i, v_j to be the components of the optimal solution to the dual of AP. It then assigns values to the multipliers w_t sequentially, without changing the values assigned earlier. We say that an inequality (37) admits a positive multiplier, if there exists a $w_t > 0$ which, together with the multipliers already chosen, satisfies the constraints of W. At any stage, v(TSP) is bounded from below by

$$(41) \qquad \sum_{i \in V} u_i + \sum_{j \in V} v_j + \sum_{t \in T} w_t a_t^o ,$$

since (u, v, w) is a feasible solution to the dual of the linear program defined by (1), (2), (37) and $x_{ij} \geq 0$, $\forall$ i,j.

The bounding procedure successively identifies valid inequalities that

(i) are violated by the AP solution $\bar{x}$, and

(ii) admit a positive multiplier.

Such inequalities are included into $L(w)$ in the order in which they are found, with the largest possible multiplier w_t. The inclusion of each new inequality strengthens the lower bound $L(w)$. We denote by $\bar{c}_{ij}$ the reduced costs defined by the optimal dual variables u_i, v_j and the multipliers w_t, i.e., $\bar{c}_{ij} = c_{ij} - u_i - v_j - \sum_{t\in T} w_t a^t_{ij}$.

At any given stage, the admissible graph $G_o = (V,A_o)$ is the spanning subgraph of G containing those arcs with zero reduced cost, i.e.

$$A_o = \left\{ (i,j) \in A \,\middle|\, u_i + v_j + \sum_{t\in T} w_t a^t_{ij} = c_{ij} \right\},$$

where T is the index set of the inequalities included so far in $L(w)$. The inclusion of each new inequality into the Lagrangean function adds at least one new arc to the set A_o. Furthermore, as long as G_o is not strongly connected, the procedure is guaranteed to find a valid inequality satisfying (i) and (ii). Thus the number of arcs in A_o steadily grows; and when no more inequalities can be found that satisfy (i) and (ii), G_o is strongly connected. Finally, if at some point G_o becomes Hamiltonian and a tour H is found in G_o whose incidence vector satisfies (37) with equality for all $t\in T$ such that $w_t > 0$, then H is an optimal solution to TSP (see Exercise 5).

Three types of inequalities, indexed by T_1, T_2 and T_3, respectively, are used in three noniterative bounding procedures applied in sequence. We will denote the three components of w corresponding to these three inequality classes, by $\lambda = (\lambda_i)_{i\in T_1}$, $\mu = (\mu_i)_{i\in T_2}$ and $\gamma = (\gamma_i)_{i\in T_3}$, respectively.

Bounding procedure 1

This procedure uses the inequalities (5) satisfying conditions (i) and (ii). For any $S \subset V$, the set of arcs $(S, V\setminus S) = \{(i,j)\in A \mid i\in S,\ j\in V\setminus S\}$ is called a directed cutset. The inequalities (5) corresponding to the node sets S_t, $t\in T$, can be represented in terms of the directed cutsets $K_t = (S_t, V\setminus S_t)$, as

$$(42) \qquad \sum_{(i,j)\in K_t} x_{ij} \geq 1, \qquad t \in T_1 .$$

At any stage of the procedure, the inequality corresponding to cutset K_t is easily seen to satisfy conditions (i) and (ii) if and only if

$$(43) \qquad K_t \cap A_o = \emptyset .$$

To find a cutset K_t satisfying (43), one chooses a node $i \in V$ and forms its reachable set $R(i) = \{j\in V \mid \text{there is a directed path from } i \text{ to } j\}$ in G_o. If $R(i) = V$, there is no cutset K_t with $i \in S_t$ satisfying (43), so one chooses another node. If $R(i) \neq V$ for some $i \in V$, then $K_t = (R(i), V\setminus R(i))$ satisfies (43), and the largest value that one can assign to the corresponding multiplier λ_t without violating the constraints of W is $\bar{\lambda}_t = \min_{(i,j)\in K_t} \bar{c}_{ij}$. Thus the inequality (42) corresponding to K_t is included in L(w) by setting the reduced costs to $c_{ij} \leftarrow \bar{c}_{ij} - \lambda_t$, $(i,j)\in K_t$, $\bar{c}_{ij} \leftarrow \bar{c}_{ij}$ otherwise. This adds to A_o all arcs for which the minimum in the definition of $\bar{\lambda}_t$ is attained. The search is then started again for a new cutset; and the procedure ends when the reachable set of every node is V. At that stage G_o is strongly connected, and $K\cap A_o \neq \emptyset$ for all directed cutsets K in G. Also, from (41) and the fact that $a_o^t = 1$, $\forall\ t\in T_1$, it follows that procedure 1 improves the lower bound on v(TSP) by $\Sigma_t \lambda_t$, i.e., at the end of procedure 1 the lower bound is

$$B_1 = v(AP) + \sum_{t\in T_1} \lambda_t .$$

One can show that bounding procedure 1 generates at most (h-1)(h+2)/2 cutsets, where h is the number of subtours in $\bar{x}$ (see Exercise 6). The computational effort required to find a cutset satisfying (43) or showing that none exists is $O(n|A|)$.

Example 2.

Consider the 8-city TSP whose cost matrix is shown in Table 1.

Table 1

	1	2	3	4	5	6	7	8
1	x	2	11	10	8	7	6	5
2	6	x	1	8	8	4	6	7
3	5	12	x	11	8	12	3	11
4	11	9	10	x	1	9	8	10
5	11	11	9	4	x	2	10	9
6	12	8	5	2	11	x	11	9
7	10	11	12	10	9	12	x	3
8	7	10	10	10	6	3	1	x

Table 2 shows the optimal solution $\bar{x}$ to AP ($\bar{x}_{ij} = 1$ for (i,j) boxed in, $\bar{x}_{ij} = 0$ otherwise), the optimal solution $(\bar{u},\bar{v})$ to the dual of AP (the numbers on the rim), and the reduced costs $\bar{c}_{ij}$. The solution value is 17. The corresponding admissible graph G_o is shown in Fig. 3.

Bounding procedure 1. Cutset $K_1 = (\{1, 2, 3, 7, 8\}, \{4, 5, 6\})$ admits $\lambda_1 = \bar{c}_{8,6} = 2$, and cutset $K_2 = (\{4, 5, 6\}, \{1, 2, 3, 7, 8\})$ admits $\lambda_2 = \bar{c}_{6,3} = 3$. The lower bound becomes 17 + 2 + 3 = 22. The new reduced cost matrix is shown in Table 3 and the corresponding admissible graph G_o in Fig. 4. Note that G_o of Fig. 4 is strongly connected.‖

Table 2

	1	2	3	4	5	6	7	8	
1	x	[0]	9	8	6	5	4	3	2
2	3	x	[0]	7	7	3	5	6	1
3	[0]	9	x	8	5	9	0	8	3
4	8	8	9	x	[0]	8	7	9	1
5	7	9	7	2	x	[0]	8	7	2
6	8	6	3	[0]	9	x	9	7	2
7	5	8	9	7	6	9	x	[0]	3
8	4	9	9	9	5	2	[0]	x	1
	2	0	0	0	0	0	0	0	

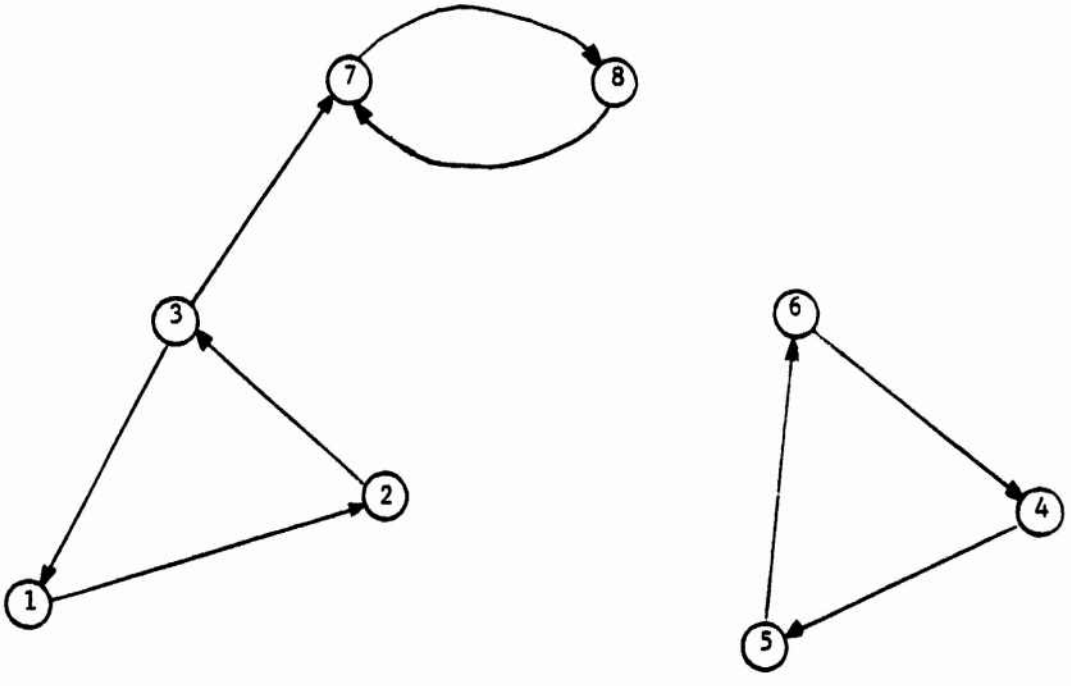


Fig. 3. Graph G_o defined by the AP solution

Table 3

	1	2	3	4	5	6	7	8
1	x	[0]	9	6	4	3	4	3
2	3	x	[0]	5	5	1	5	6
3	[0]	9	x	6	3	7	0	8
4	5	5	6	x	[0]	8	4	6
5	4	6	4	2	x	[0]	5	4
6	5	3	0	[0]	9	x	6	4
7	5	8	9	5	4	7	x	[0]
8	4	9	9	7	3	0	[0]	x

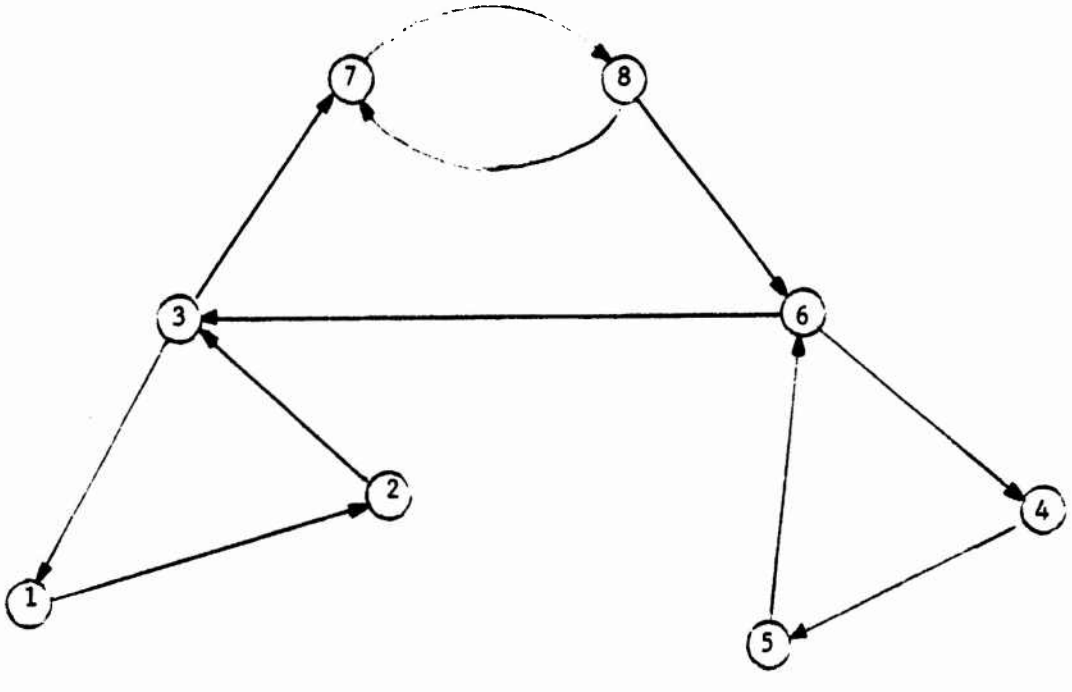


Fig. 4. Graph G_o after bounding procedure 1

Bounding procedure 2

This procedure uses the inequalities (3) that satisfy conditions (i) and (ii), i.e. are violated by $\bar{x}$ and admit a positive multiplier. To write these inequalities in the general form (37), we restate them as

$$(44) \qquad -\sum_{i\in S_t}\sum_{j\in S_t} x_{ij} \geq 1 - |S_t|, \qquad t \in T_2 .$$

The subtour elimination inequalities (3) (or (44)) are known to be equivalent to (5) (or (42)). Nevertheless, an inequality (44) may admit a positive multiplier when the corresponding inequality (42) does not, and vice versa.

If $S_1,\ldots,S_h$ are the node sets of the h subtours of $\bar{x}$, every inequality (44) defined by S_t, $t=1,\ldots,h$, is violated by $\bar{x}$; but a positive multiplier μ_t can be applied without violating the condition that $\bar{x}_{ij} = 1$ implies $\bar{c}_{ij} = 0$, only by changing the values of some u_i and v_j, and this in turn can only be done if a certain condition is satisfied. Roughly speaking, we have to find a set of rows I and columns J such that, by adding to each u_i, $i\in I$ and v_j, $j\in J$ the same amount $\mu_t > 0$ that is being added to $\bar{c}_{ij}$, $(i,j)\in(S_t, S_t)$, we obtain a new set of reduced costs $\bar{c}'_{ij}$ such that $\bar{c}'_{ij} \geq 0$ for all (i,j), and $\bar{c}'_{ij} = 0$ for all those (i,j) such that $\bar{x}_{ij} = 1$. The condition for this is best expressed in terms of the assignment tableau of the Hungarian algorithm whose rows and columns are called <u>lines</u>, and whose row/column intersections are called <u>cells</u>. Cells correspond to arcs of G and are denoted the same way.

Let S_t be the node set of a subtour of $\bar{x}$, and

$$A_t = \{(i,j)\in A_o \mid i,j\in S_t\}, \quad A'_t = \{(i,j)\in A_t \mid \bar{x}_{ij} = 1\}$$

Theorem 2 (Balas and Christofides [1981]). Inequality (44) admits a positive multiplier if and only if there exists a set C of lines such that

(α) every $(i,j)\in A'_t$ is covered by exactly one line in C,

(β) every $(i,j)\in A_t \setminus A'_t$ is covered by at most one line in C,

(γ) no $(i,j)\in A_o \setminus A_t$ is covered by any line in C.

If such a set C exists, and it consists of row set I and column set J, then the maximum applicable multiplier is

$$\mu_t = \min_{(i,j)\in M} \bar{c}_{ij}$$

where

$$M = (I,J) \cup (I, V\setminus S_t) \cup (V\setminus S_t, J) .$$

Proof. Sufficiency. Suppose line set C, consisting of row set I and column set J, satisfies (α), (β), (γ). Then adding $\mu_t > 0$ to $\bar{c}_{ij}$ for all $(i,j)\in(S_t, S_t)$, as well as to all u_i, $i\in I$ and v_j, $j\in J$, produces a set of reduced costs $\bar{c}'_{ij}$ such that $\bar{c}'_{ij} = 0$ for $(i,j)\in A'_t$, since $C = I \cup J$ satisfies (α). Further, since C satisfies (β) and (γ), $\bar{c}'_{ij} \geq \bar{c}_{ij} = 0$ for all $(i,j)\in A_t \setminus A'_t$, and $\bar{c}'_{ij} = \bar{c}_{ij} = 0$ for all $(i,j)\in A_o \setminus A_t$. The only reduced costs that are diminished as a result of the above changes, are those corresponding to arcs in one of the three sets (I,J), $(I, V\setminus S_t)$, $(V\setminus S_t, J)$ whose union is the set M of the theorem. Hence setting μ_t equal to the minimum reduced cost over M provides a positive multiplier that can be applied to the arcs in (S_t, S_t).

Necessity. Suppose a multiplier $\mu > 0$ can be applied to the arc set (S_t, S_t). In order to prevent the $\bar{c}_{ij}$ for $(i,j)\in A'_t$ from becoming positive, one must increase $u_i + v_j$ by μ for all $(i,j)\in A'_t$. If this can be done, it can be done by adding μ

to u_i or v_j (but not to both) for $(i,j)\in A'_t$; and the corresponding index sets I and J form a set $C = I \cup J$ that satisfies (α). Let $\mathcal{C}$ be the collection of all sets C obtained in this way. Now take any $C\in\mathcal{C}$. If C violates (β), then $\bar{c}'_{ij} = \bar{c}_{ij} + \mu - 2\mu < \bar{c}_{ij} = 0$ for some $(i,j)\in A_t \setminus A'_t$, and if it violates (γ), then $\bar{c}'_{ij} < \bar{c}_{ij} = 0$ for some $(i,j)\in A_o \setminus A_t$. Since by assumption $\mu > 0$ can be applied to (S_t, S_t), there exists at least one set $C\in\mathcal{C}$ that satisfies both (β) and (γ).‖

To check whether for a given subtour-node-set S_t there exists a set of lines C satisfying conditions (α), (β), (γ), we proceed as follows.

First we construct a set R^- of rows that cannot belong to C, and a set K^+ of columns that must belong to C, if conditions (α), (β), (γ) are to be satisfied. To do this, we start with $K^+ = \emptyset$ and in view of (γ), put into R^- all rows i for which there exists a cell $(i,j)\in A_o$ with $j\in V\setminus S_t$. Then we apply recursively the following two steps, until no more additions can be made to either set:

If a row i was put into R^-, then to satisfy (α) we put into K^+ every column j such that $(i,j)\in A'_t$.

If a column j was put into K^+, then to satisfy (β) we put into R^- every row h such that $(h,j)\in A_t$.

To state the procedure formally, we set $K^+_o = \emptyset$,

$$R^-_o = \{i\in S_t \mid \exists (i,j)\in A_o \text{ with } j\in V\setminus S_t\},$$

and define recursively for $r = 1,\ldots,\bar{r}$,

$$K^+_r = K^+_{r-1} \cup \{j\in S_t \mid \exists (i,j)\in A'_t \text{ with } i\in R^-_{r-1}\}$$

$$R^-_r = R^-_{r-1} \cup \{i\in S_t \mid \exists (i,j)\in A_t \text{ with } j\in K^+_r\}\ .$$

Here $\bar{r}$ is the smallest r for which $K^+_r = K^+_{r-1}$ or $R^-_r = R^-_{r-1}$.

Next we use a perfectly analogous procedure to construct a set R^+ of rows that must belong to C and a set K^- of columns that cannot belong to C, if (α), (β), (γ) are to hold. In other words, we set $R_o^+ = \emptyset$,

$$K_o^- = \{j \in S_t \mid \exists (i,j) \in A_o \text{ with } i \in V \setminus S_t\},$$

and define recursively for $s = 1, \ldots, \bar{s}$,

$$R_s^+ = R_{s-1}^+ \cup \{i \in S_t \mid \exists (i,j) \in A_t' \text{ with } j \in K_{s-1}^-\}$$

$$K_s^- = K_{s-1}^- \cup \{j \in S_t \mid \exists (i,j) \in A_t \text{ with } i \in R_s^-\}$$

Here $\bar{s} = \min\{\bar{s}_1, \bar{s}_2\}$, where $\bar{s}_1$ is the smallest s such that $R_s^+ = R_{s-1}^+$ or $K_s^- = K_{s-1}^-$, and $\bar{s}_2$ is the smallest s such that $R_s^+ \cap R_{\bar{r}}^- \neq \emptyset$ or $K_s^- \cap K_{\bar{r}}^+ \neq \emptyset$.

If $\bar{s} = \bar{s}_2$, then some row or some column that cannot belong to C, must belong to C for (α), (β), (γ) to hold; hence there exists no set C of lines satisfying (α), (β), (γ), and no positive multiplier can be applied to the inequality (44) corresponding to S_t.

If $\bar{s} = \bar{s}_1$, then the set of lines $C = I \cup J$, where $I = S_t \setminus R_{\bar{r}}^-$ and $J = K_{\bar{r}}^+$, satisfies conditions (α), (β), (γ). Thus we include the inequality (44) corresponding to S_t into L(w) with the multiplier $\mu_t > 0$ defined in Theorem 2, and set the reduced costs to $\bar{c}_{ij} \leftarrow \bar{c}_{ij} - \mu_t$, $(i,j) \in M$, $\bar{c}_{ij} \leftarrow \bar{c}_{ij}$ otherwise. (Here M is the set defined in Theorem 2.)

In both cases, we then choose another subtour, until all subtours have been examined. If h is again the number of subtours, bounding procedure 2 requires $O(h \cdot |A|)$ steps. It can be shown (see Exercise 7) that this procedure improves the lower bound on v(TSP) by $\Sigma_t \mu_t$, i.e., at the end of procedure 2 the lower bound is

$$B_2 = v(AP) + \sum_{t \in T_1} \lambda_t + \sum_{t \in T_2} \mu_t .$$

Example 2 (continued).

Bounding procedure 2. The subtours of $\bar{x}$ are (1, 2, 3), (4, 5, 6) and (7, 8)(see Table 3 and Fig. 4).

For $S_1 = \{1, 2, 3\}$, $R_o^- = \{3\}$, $K_1^+ = \{1\}$; $K_o^- = \{3\}$, $R_1^+ = \{2\}$. Thus $C = I \cup J$, where $I = \{1, 2\}$, $J = \{1\}$, and $\mu_1 = \bar{c}_{2,6} = 1$. For $S_2 = \{4, 5, 6\}$, $R_o^- = \{6\}$, $K_1^+ = \{4\}$; $K_o^- = \{6\}$, $R_1^+ = \{5\}$, and $C = I \cup J$, with $I = \{4, 5\}$, $J = \{4\}$, and $\mu_2 = \bar{c}_{5,4} = 2$. Finally, for $S_3 = \{7, 8\}$, $R_o^- = \{8\}$, $K_1^+ = \{7\}$; $K_o^- = \{7\}$, and since $K_1^+ \cap K_1^- = \{7\} \neq \emptyset$, the inequality corresponding to subtour (7, 8) does not admit a positive multiplier.

The lower bound becomes $B_2 = B_1 + \mu_1 + \mu_2 = 22 + 1 + 2 = 25$. The new reduced costs are shown in Table 4, and the corresponding admissible graph G_o in Fig. 5.‖

Table 4

	1	2	3	4	5	6	7	8
1	x	[0]	9	3	3	2	3	2
2	2	x	[0]	2	4	0	4	5
3	[0]	9	x	4	3	7	0	8
4	2	3	4	x	[0]	8	2	4
5	1	4	2	0	x	[0]	3	2
6	4	3	0	[0]	9	x	6	4
7	4	8	9	3	4	7	x	[0]
8	3	9	9	5	3	0	[0]	x

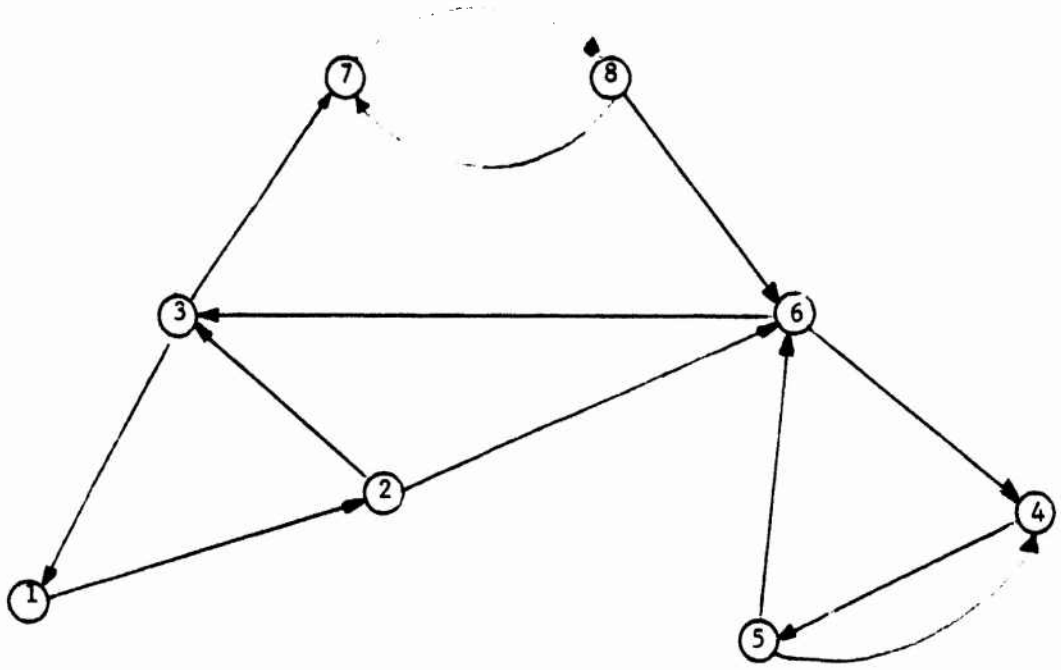


Fig. 5. Graph G_o after bounding procedure 2

Bounding procedure 3

The class of inequalities used in this procedure is defined as follows. Suppose G_o has an articulation point, i.e. a node k such that $G_o - \{k\}$ has more than one component. Let one of the components have node set S_t, and denote $W_t = V \setminus S_t \cup \{k\}$. Then every tour contains an arc of at least one of the cutsets $K'_t = (S_t, W_t)$ and $K''_t = (W_t, S_t)$, hence the incidence vector x of any tour satisfies the inequality

$$(45) \qquad \sum_{(i,j) \in K'_t \cup K''_t} x_{ij} \geq 1 .$$

Furthermore, (45) satisfies condition (i), i.e. is violated by the AP solution.

Bounding procedure 3 uses those inequalities (45) that also satisfy condition (ii). Although every inequality (45) is the combination of some inequalities (3) and equations (2) (see Exercise 8), nevertheless it is possible to find inequalities (45) that satisfy condition (ii), i.e., admit a positive multiplier, when no inequality (3) (i.e., (44)) satisfies it. Indeed, it is not hard to see, that if k is an articulation point of G_o and S_t is the node set of one of the components of $G_o - \{k\}$, then $K_t' \cap A_o = K_t'' \cap A_o = \emptyset$ and a positive multiplier given by

$$(46) \qquad \nu_t = \min_{(i,j)\in K_t' \cup K_t''} \bar{c}_{ij}$$

can be applied to the arc set $K_t' \cup K_t''$. On the other hand, if G_o has no articulation point, then for any choice of the node k, the minimum in (46) is 0 and thus no inequality (45) admits a positive multiplier.

Thus bounding procedure 3 checks for every $i \in V$ whether it is an articulation point, and if so, it takes the corresponding inequality (45) into L(w) with the multiplier ν_t given by (46). This is done by setting $\bar{c}_{ij} \leftarrow \bar{c}_{ij} - \nu_t$, $(i,j) \in K_t' \cup K_t''$, $\bar{c}_{ij} \leftarrow \bar{c}_{ij}$ otherwise. Since G_o has n nodes, and testing for connectivity requires $O(|A|)$ steps, bounding procedure 3 requires $O(n|A|)$ steps.

In view of (41) and the fact that (45) has a righthand side of 1, at the end of bounding procedure 3 one has the following lower bound on v(TSP):

$$B_2 = v(AP) + \sum_{t\in T_1} \lambda_t + \sum_{t\in T_2} \mu_t + \sum_{t\in T_3} \nu_t .$$

Example 2 (continued).

Vertex 6 is an articulation point of G_o (see Fig. 5). The corresponding cutsets are K_1' = ({4, 5}, {1, 2, 3, 7, 8}) and K_1'' = ({1, 2, 3, 7, 8}, {4, 5}),

and the arc set $K_1' \cup K_1''$ admits the multiplier $\nu_1 = \bar{c}_{5,1} = 1$. There is no other articulation point, and the procedure stops with the lower bound $B_3 = B_2 + \nu_1 = 25 + 1 = 26$. The new reduced costs are shown in Table 5, and the corresponding G_o in Fig. 6.‖

Table 5

	1	2	3	4	5	6	7	8
1	x	[0]	9	2	2	2	3	2
2	2	x	[0]	1	3	0	4	5
3	[0]	9	x	3	2	7	0	8
4	1	2	3	x	[0]	8	1	3
5	0	3	1	0	x	[0]	2	1
6	4	3	0	[0]	9	x	6	4
7	4	8	9	3	4	7	x	[0]
8	3	9	9	5	3	0	[0]	x

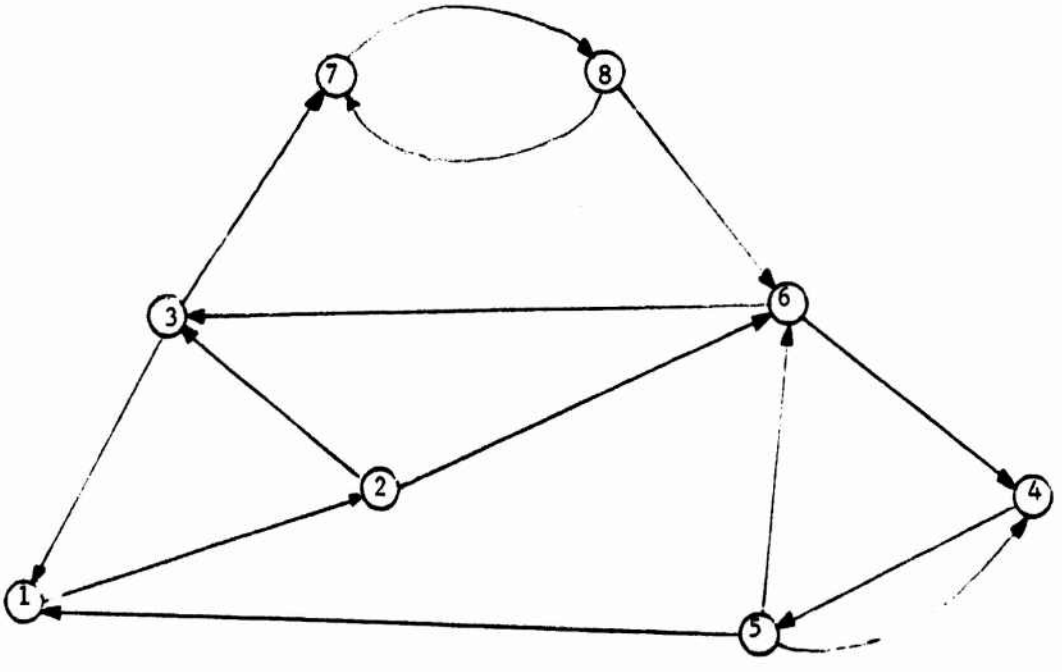


Fig. 6. Graph G_o after bounding procedure 3

Additional bounding procedures

At the end of bounding procedure 3, G_o is strongly connected and without articulation points. At that stage an attempt is made to find a tour in G_o. For that purpose a specialized implicit enumeration technique is applied, with a cut-off rule. If a tour $\hat{H}$ is found whose incidence vector $\bar{x}$ satisfies with equality all those inequalities (37) such that $w_t > 0$, then $\hat{H}$ is optimal for the current subproblem (this follows from elementary Langrangean theory).

Example 2 (continued). The following tour can be identified by inspection in G_o of Fig. 6: $H = \{(1, 2), (2, 3), (3, 7), (7, 8), (8, 6), (6, 4), (4, 5), (5, 1)\}$. The value of H is 26, equal to $L(w) = B_3$, the lower bound at the end of procedure 3. The tour H contains exactly one arc of each cutset associated with a positive λ_t, namely arc (8, 6) of $K_1 = (\{1, 2, 3, 7, 8\}, \{4, 5, 6\})$, and arc (5, 1) of $K_2 = (\{4, 5, 6\}, \{1, 2, 3, 7, 8\})$. Thus the incidence vector of H satisfies with equality the two inequalities (42) corresponding to K_1 and K_2, as required. Further, H contains exactly $|S_1| - 1 = 2$ arcs of the subtour with node set $S_1 = \{1, 2, 3\}$, namely, (1, 2) and (2, 3); and exactly $|S_2| - 1 = 2$ arcs of the subtour with node set $S_2 = \{4, 5, 6\}$, namely (6, 4) and (4, 5). Thus the complementarity condition is also satisfied for the two inequalities (44) corresponding to S_1 and S_2. Finally, it contains exactly one arc of the set $K_1' \cup K_1''$, where $K_1' = (\{4, 5\}, \{1, 2, 3, 7, 8\})$, $K_1'' = (\{1, 2, 3, 7, 8\}, \{4, 5\})$, namely (5, 1): so the complementarity condition also holds for the inequality (45) corresponding to $K_1' \cup K_1''$. In conclusion, H is optimal.||

If, after bounding procedure 3, a tour $\hat{H}$ is found such that $\hat{x}$ violates this complementarity condition for some $t \in T$, then attempts are made to replace those

inequalities (37) that are "out of kilter," i.e., for which the complementarity condition is violated, by "in kilter" inequalities (of the same type), i.e., inequalities that are tight for $\hat{x}$ and thus admit positive multipliers satisfying the complementarity condition. These attempts consist of a sequence of three additional bounding procedures, called 4, 5 and 6, one for each type of inequality (42), (44) and (45), respectively. Bounding procedure 4 takes in turn each inequality (42) which has a positive multiplier λ_t and yet is slack for $\hat{x}$, and performs an exhaustive search for other inequalities of type (42) that could replace the inequality in question (with new multipliers) and which are tight for $\hat{x}$. If the search is successful, the in kilter inequalities with their new multipliers replace the out of kilter inequality and one proceeds to the next out of kilter inequality of type (42). Procedures 5 and 6 perform the same function for out of kilter inequalities of type (44) and (45), respectively. These procedures are described in detail in Balas and Christofides [1981]. When procedures 4, 5 and 6 are not successful in replacing all out of kilter inequalities (and thus proving $\hat{H}$ to be an optimal tour), they nevertheless strengthen the lower bound on v(TSP).

Each of the six bounding procedures is polynomially bounded. This (worst case) bound is $O(n^4)$ for procedure 1, $O(n^3)$ for each of the other procedures. The mean times are considerably shorter, and on the average procedure 2 (the only one that changes the dual variables u_i, v_j) takes the longest. The general algorithm of course remains valid if any subset of the six bounding procedures is used in place of the full set, but computational testing indicates that using all 6 procedures is more efficient (i.e. results in smaller search trees and shorter overall computing times) than using any proper subset.

Branching rules and other features

Before branching, all arcs (i,j) such that $\bar{c}_{ij} \geq U - L(w)$ are deleted from G. This "reduction" has removed on a set of 120 randomly generated problems (Balas and Christofides [1981]), on the average 96-97% of the arcs in problems with up to 150 variables, and 98% in problems with 175-325 variables.

The AP relaxation with Lagrangean objective function can of course be used with any of the branching rules BR1 - BR5 described in the context of the AP relaxation with objective function (1). Balas and Christofides [1981] use two rules intermittently, namely BR3 (partitioning on the basis of a subtour elimination inequality (3)), and another rule based on a disjunction from a conditional bound, introduced earlier in the context of set covering (Balas [1980]). This latter rule is motivated by the following considerations.

Let $\hat{H}$ be the current tour and $\hat{x}$ its incidence matrix. Remove from L(w) all those inequalities (37) that are slack while the associated multiplier is positive. Let $\hat{c}_{ij}$ be the reduced costs, and $L(\hat{w})$ the lower bound, resulting from this removal.

Theorem 3. Let $S \subseteq \hat{H}$, $S = \{(i_1, j_1),\ldots,(i_p, j_p)\}$ be such that

$$(47) \qquad \sum_{r=1}^{p} \hat{c}_{i_r j_r} \geq U - L(\hat{w}),$$

and let the arc sets $Q_r \subset A$, $r = 1,\ldots,p$, satisfy

$$(48) \qquad \sum_{r \mid (i,j) \in Q_r} \hat{c}_{i_r j_r} \leq \hat{c}_{ij} \quad , \quad (i,j) \in A$$

Then every solution x to TSP such that $cx < U$ satisfies the disjunction

$$(49) \qquad \bigvee_{r=1}^{p} (x_{ij} = 0, \ (i,j) \in Q_r).$$

Proof. $L(\hat{w})$ is the value of an optimal solution to the dual of the linear program LP defined by (1), (2), $x_{ij} \geq 0$, $(i,j) \in A$, and those inequalities (37) with a positive multiplier. Now let x be a feasible solution to LP that violates (49). Then x satisfies

$$(50) \qquad \sum_{(i,j)\in Q_r} x_{ij} \geq 1 \quad , \quad r = 1,\ldots,p.$$

Let LP_+ be the linear program obtained by adding to LP the constraints (50). From (48), if we assign the values $\hat{c}_{i_r j_r}$, $r = 1,\ldots,p$ to the dual variables associated with the inequalities (50), we obtain a feasible solution to the dual of LP_+. But then the objective function value of this solution is $L(\hat{w}) + \sum_{r=1}^{p} \hat{c}_{i_r j_r}$, and hence from (47)

$$cx \geq L(\hat{w}) + \sum_{r=1}^{p} \hat{c}_{i_r j_r} \geq U.$$

Thus every solution x to TSP such that $cx < U$ satisfies (49).‖

The branching rule can now be stated as follows.

BR9. Choose a minimum-cardinality set $S \subseteq \hat{H}$, $S = \{(i_1, j_1),\ldots,(i_p, j_p)\}$, satisfying (47). Next construct a $p \times |A|$ 0-1 matrix $D = (d_{ij}^r)$ (where r is the row index and (i,j) the column index), with as many 1's in each column as possible, subject to the condition (48) and $(i_r, j_r) \in Q_r$, $r = 1,\ldots,p$, where

$$Q_r = \{(i,j) \in A \mid d_{ij}^r = 1\}.$$

Generate the p new subproblems defined by the disjunction (49), where the r-th subproblem is given by

$$(51) \qquad \left.\begin{aligned} E_{k+r} &= E_k \cup Q_r \\ I_{k+r} &= I_k \end{aligned}\right\} \quad r = 1,\ldots,p \ .$$

The branching rule BR9 is used intermittently with BR3 because at different nodes the ranking of the two rules (in terms of strength) may be different. The choice is based on certain indicators of relative strength.

As to subproblem selection, the Balas-Christofides algorithm uses a mixture of depth first and breadth first: a successor of the current node is selected whenever available; otherwise the algorithm chooses a node k that minimizes the function

$$E(k) = (L(w)_k - v(AP)) \frac{s(k) - 1}{|s(0) - s(k)|},$$

where $L(w)_k$ is the value of L(w) at node k, v(AP) is the value of the initial AP, while s(0) and s(k) are the number of subtours in the solutions to the initial AP and the one at node k, respectively.

5. Other Relaxations

For the same reasons as in the case of the AP relaxation with the original objective function, the AP relaxation with the Lagrangean objective function is inefficient (weak) in the case of the symmetric TSP. Limited computational experience indicates that on the average the bound L(w) attains about 96% of v(TSP), which compares unfavorably with the bound obtained from the 1-tree relaxation.

On the other hand, the main reason for the weak performance of AP-based relaxations in the case of symmetric problems, namely the high frequency of subtours of length 2 in the optimal AP solution, can be eliminated if AP is replaced by the 2-matching problem in the undirected graph G = (V,E).

The 2-matching relaxation

The problem of minimizing the function (6) subject to constraints (7) and (9) is known in the literature as the 2-matching problem, and is obviously a relaxation of the TSP. Bellmore and Malone [1971] have used it for the symmetric TSP in a way that parallels their use of the AP-relaxation for the asymmetric TSP. A 2-matching is either a tour or a collection of subtours, and the branching rules BR2 - BR5 based on the subtour-elimination inequalities (3) and (5) for the asymmetric TSP have their exact parallels in branching rules based on the subtour elimination inequalities (8) and (10) for the symmetric TSP.

The objective function (6) can be replaced, just like in the case of the AP relaxation, with a Lagrangean function using the inequalities (8) and/or (10). The Lagrangean dual of the TSP formulated in this way is as hard to solve exactly as in the asymmetric case, but it can be approximated by a procedure similar to the one used by Balas and Christofides [1981] with the AP-relaxation. Further facet defining inequalities, beyond (8) and (10), based on the work of Grötschel and Padberg [1979], can be used to enrich the set drawn upon in constructing the Lagrangean function.

Although the 2-matching problem is polynomially solvable (Edmonds [1965]), the main impediment in the development of an efficient branch and bound procedure based on the 2-matching relaxation has so far been the absence of a good implementation of a weighted 2-matching algorithm. However, as this difficulty is likely to be overcome soon, the 2-matching relaxation with a Lagrangean objective function will in all likelihood provide bounds for the symmetric TSP comparable to those obtained from the 1-tree relaxation.

The n-path relaxation

The problem of minimizing (1) subject to the constraint that the solution x be the incidence matrix of a directed n-path starting and ending at a fixed node v (where "path" is used in the sense of walk, i.e., with possible repetitions of nodes, and n denotes the length of the path) is clearly a relaxation of the TSP. An analogous relaxation of the symmetric TSP can be formulated in terms of n-paths in the associated undirected graph. Furthermore, the constraints (2) in the asymmetric case, or (7) in the symmetric case, can be used to replace the objective function (1) or (6), respectively, by a Lagrangean function of the same type as the one used with the 1-arborescence and 1-tree relaxations. This family of relaxations of the TSP was introduced by Houck, Picard, Queyranne and Vemuganti [1977]. The (directed or undirected) n-path problems involved in this relaxation can be solved by a dynamic programming recursion in $O(n^3)$ steps. Computational experience with this approach seems to indicate (Christofides [1979], p. 142) that the quality of the bound obtained is comparable to the one obtained from the 1-arborescence relaxation in the asymmetric case, but slightly weaker than the bound obtained from the 1-tree relaxation in the symmetric case. Since solving the 1-tree and 1-arborescence problems is computationally cheaper than solving the corresponding n-path problems, this latter relaxation seems to be dominated (for the case of the "pure" TSP) by the 1-tree or 1-arborescence relaxation. However, the n-path relaxation can easily accommodate extra conditions which the 1-tree and 1-arborescence relaxations cannot, and which often occur in problems closely related to the TSP (traveling salesman problems with side constraints appear in vehicle routing (see Chapter 12 of this book) and other practical contexts.)

A substantial generalization of the n-path relaxation, due to Christofides, Mingozzi and Toth [1981] and called state-space relaxation, has the same desirable characteristics of being able to easily accommodate side constraints.

The LP with cutting planes as a relaxation

Excellent computational results have been obtained recently by Crowder and Padberg [1980] for the symmetric TSP by a cutting plane/branch and bound approach. It applies the primal simplex method to the linear program defined by (6), (7), $x_{ij} \geq 0$, $\forall i,j$, and an unspecified subset of the inequalities defining the convex hull of incidence vectors of tours, generated as needed to avoid fractional pivots. The procedure uses mostly inequalities of the form (10), but also other facet inducing inequalities from among those introduced by Grötschel and Padberg [1979]. When the search for the next inequality needed for an integer pivot fails, the procedure branches. Since the main feature of this approach is the identification of appropriate inequalities to be added to the linear program at each step, it is being reviewed in the chapter on cutting plane methods.

6. Performance of State of the Art Computer Codes

In this section we review the performance of some state of the art branch and bound codes for the TSP, by comparing and analyzing the computational results reported by the authors of these codes.

The asymmetric TSP

The three fastest currently available computer codes for the asymmetric TSP seem to be those of Balas and Christofides [1981], Carpaneto and Toth [1980] and Smith, Srinivasan and Thompson [1977], to be designated in the following by BC, CT and SST, respectively. The main characteristics of these codes are summa-

rized in Table 6. Table 7 describes the computational results reported by the authors of the codes. Each of the codes was run on a set of (different) asymmetric TSP's whose costs were independently drawn from a uniform distribution of the integers in the interval [1,1000]. The entries of the table represent averages for 5 problems (SST), 20 problems (CT) and 10 problems (BC), respectively, in each class. The number of nodes in the SST column is not strictly comparable with that in the CT and BC columns, since it is based on counting only those nodes that were selected for branching and processed. Also, the computing times are not strictly comparable without a correction, since the CDC 7600 is about 3 times faster than the UNIVAC 1108 and the CDC 6600 (Computer Review, GML Corp., Lexington, MA, 1979). The picture that emerges, however, by comparing the figures within each column, for any of these three codes, is a pattern of growth in computational effort with problem size, that seems rather modest for a problem usually viewed as "notoriously intractable". We will discuss the functional relationship between problem size and computational effort in some detail further below.

For problems in the range $40 \leq n \leq 180$, the number of nodes generated by the BC algorithm is considerably smaller than the corresponding numbers for the other two algorithms, although CT uses a "breadth first" branching strategy, meant to minimize the number of nodes generated, at the cost of increased storage requirements. The reason for this is that the Lagrangean bounding function used by BC changes the ranking of tours among the assignments, removing from consideration many assignment problems whose value in terms of the original objective function is higher than that of the optimal TSP, and which therefore must be processed by the CT algorithm. On the other hand, in the range $200 \leq n \leq 240$,

Table 6. Summary description of three codes for the asymmetric TSP

	SST	CT	BC
Relaxation	AP with TSP objective	AP with TSP objective	AP with Lagrangean objective
Lower bounding	v(AP), obtained by parametric simplex method, plus penalty	v(AP), obtained by Hungarian method (post-optimizing version)	lower bound on Lagrangean, obtained by approximation procedures
Branching rule	BR3	BR3	BR3 + BR9
Subproblem selection	depth first	breadth first	depth first upon forward step, breadth first upon backtracking
Upper bounding	no special procedure	no special procedure	tour-finding heuristic
Variable fixing	no	no	yes

Table 7. Computational results on randomly generated asymmetric TSP's

n	Nodes of the search tree			Computing time (seconds)		
	SST(1)	CT(2)	BC(2)	SST(3)	CT(4)	BC(5)
40	26	27	-	2.9	0.9	
50	11	-	12	1.7	-	0.2
60	39	24	-	9.3	2.2	-
70	32	-	-	8.5	-	-
75	-	-	27	-	-	0.3
80	32	42	-	13.8	6.6	-
90	82	-	-	42.0	-	-
100	87	56	39	53.0	10.4	0.7
110	24	-	-	22.3	-	-
120	65	61	-	62.9	16.2	-
125	-	-	43	-	-	1.1
130	97	-	-	110.1	-	-
140	130	57	-	165.2	19.0	-
150	50	-	46	65.3	-	2.0
160	70	73	-	108.5	32.8	-
170	98	-	-	169.8	-	-
175	-	-	58	-	-	4.2
180	215	69	-	441.4	29.2	-
200	-	58	63	-	35.7	6.1
220	-	43	-	-	46.7	-
225	-	-	84	-	-	10.4
240	-	63	-	-	53.4	-
250	-	-	89	-	-	13.7
275	-	-	106	-	-	21.7
300	-	-	124	-	-	38.4
325	-	-	142	-	-	49.7

(1) Number of nodes that were explored; (2) total number of nodes; (3) UNIVAC 1108; (4) CDC 6600; (5) CDC 7600

BC seems to generate more nodes than CT; the reason for this may be that at this point the advantage of the "breadth first" strategy used by CT outweighs that of the stronger bounding procedures used by BC. This seems to suggest that an algorithm based on the Lagrangean bounding procedures of BC, but using the "breadth first" node selection strategy of CT, will generate fewer nodes for any problem size, than either the CT or the BC algorithms. This is undoubtedly true, but notice that at the current state of the art, the limiting factor in the use of both algorithms is not computing time (which has never exceeded 1.5 minutes for any problem), but (in core) storage space.

The symmetric TSP

The fastest currently available branch and bound codes for the symmetric TSP seem to be those of Smith and Thompson [1977] and Volgenant and Jonker [1982], to be designated in the following by ST and VJ, respectively. Table 8 summarizes their main characteristics, while Table 9 reports on their computational performance.

Again, each of the two codes was run on a set of (different) symmetric TSP's whose costs were independently drawn from a uniform distribution of the integers in the interval [1,1000]. The entries of the table represent averages for 15 problems (except for n=80, where the entry for SST is the average for 5 problems only). The CYBER 750 is about 3 times faster than the UNIVAC 1108.

Both codes were also tested on randomly generated symmetric Euclidean TSP's, which required for each code a greater computational effort (e.g., for n = 60 the average number of subgradient iterations was 3049 for ST and 1034 for VJ).

Table 8. Summary description of two codes for the symmetric TSP

	ST	VJ
Relaxation	1-tree with Lagrangean objective	1-tree with Lagrangean objective
Lower bounding	subgradient optimization	subgradient optimization with convex combination of subgradients
Branching rule	BR7	BR8
Subproblem selection	depth first	depth first
Upper bounding	no special procedure	no special procedure
Variable fixing	no	yes

Table 9. Computational results on randomly generated symmetric TSP's

n	Nodes of the search tree		Subgradient iterations		Computing time (seconds)	
	ST(1)	VJ(2)	ST	VJ	ST(3)	VJ(4)
50	17		526	-	22.1	-
60	15		572	352	34.1	4.7
70	19		760	-	61.6	-
80	15		764	702	83.0	15.5
100	-		-	1664	-	53.2

(1) Number of nodes that were explored; (2) not reported; (3) UNIVAC 1108; (4) CYBER 75?.

Average performance as a function of problem size

TSP is well known to be NP-complete, hence in all likelihood there is no polynomial time TSP algorithm, i.e. no algorithm guaranteed to solve every instance of TSP in a number of steps polynomial in n. However, this statement refers to the worst case behavior of algorithms, and does not exclude the existence of algorithms whose performance, though exponential in the worst case, is on the average polynomial in n. To make the colloquial term "on the average" more precise, assume the costs c_{ij} of TSP are random numbers drawn independently from a uniform distribution over the unit interval. Whether the expected time required to solve such a problem is an exponential or polynomial function of n, is at present an open question, on which the opinion of experts is divided (see, for instance, Bellmore and Malone [1971], and Lenstra and Rinnooy Kan [1978]).

While the theoretical issue remains unsolved, it is not irrelevant to examine from this point of view the empirical performance of some of the more efficient algorithms on randomly generated TSP's. In a recent study, Balas, McGuire and Toth [1983] have fitted three different approximating curves to the data of Table 7 for each of the three codes SST, CT and BC for the asymmetric TSP, in an attempt to determine which of the three types of functions describes best the behavior of each algorithm. The data of Table 7 were corrected for the difference in speed between the CDC 7600 and the other two computers by multiplying by 3 the computing times reported for the Balas-Christofides code. The functions examined were:

$$f(n) = \alpha n^{\beta} \qquad \text{(polynomial)},$$

$$f(n) = \alpha n^{\beta \log n} \qquad \text{(superpolynomial)}.$$

$$f(n) = \alpha e^{\beta n} \qquad \text{(exponential)},$$

where log stands for the natural logarithm and e for its base.

Each of the three functions was expressed in logarithmic form, and a simple regression of log f(n) was run on log n (in the case of the polynomial function), on $\log^2 n$ (in the case of the superpolynomial function), and on n (in case of the exponential function), in order to find the best fitting values of α and β for each case. The outcome is shown in Tables 10, 11 and 12.

Table 10. Statistical analysis of the Smith-Srinivasan-Thompson algorithm

$50 \leq n \leq 180$

Type of function	Best fit	Standard error of estimation	Coefficient of determination
Polynomial	$0.38 \times 10^{-5} \times n^{3.473}$	0.505	0.883
Superpolynomial	$0.105 \times 10^{-1} \times n^{0.3771\log n}$	0.519	0.877
Exponential	$1.19 \times e^{0.0326n}$	0.595	0.838

Table 11. Statistical analysis of the Carpaneto-Toth algorithm

$40 \leq n \leq 240$

Type of function	Best fit	Standard error of estimation	Coefficient of determination
Polynomial	$0.26 \times 10^{-3} \times n^{2.261}$	0.193	0.978
Superpolynomial	$0.47 \times 10^{-1} \times n^{0.2421\log n}$	0.255	0.962
Exponential	$1.05 \times e^{0.0184n}$	0.488	0.860

Table 12. Statistical analysis of the Balas-Christofides algorithm

$50 \leq n \leq 325$

Type of function	Best fit	Standard error of estimation	Coefficient of determination
Polynomial	$0.5 \times 10^{-6} \times n^{3.114}$	0.361	0.962
Superpolynomial	$0.87 \times 10^{-3} \times n^{0.320\log n}$	0.260	0.980
Exponential	$0.85 \times 10^{-1} \times e^{0.0205n}$	0.199	0.989

These results suggest that in the limited range of n for which the algorithms were tested ($40 \leq n \leq 180$ for SST, $40 \leq n \leq 240$ for CT, and $50 \leq n \leq 325$ for BC), their behavior can be almost equally well described by any of the three types of functions considered. Although the rankings given by the coefficient of determination seem to be polynomial/superpolynomial/exponential for SST and CT, versus exponential/superpolynomial/polynomial for BC, the differences between the coefficients of determination for the three function types are too small in comparison to the differences between the same coefficients for the different algorithms, in order to attach much significance to these rankings. Further caution and reservations are in order because of the considerable differences in the range of n over which the three codes were tested.

In an attempt to obtain a more meaningful ranking of the three types of approximation curves, the range of n for each of the three algorithms was then broken up into two approximately equal parts, and the same three function types were fitted separately to the data in the lower half and in the upper half of the range of n. The results, shown in Tables 13, 14, 15, yield the same rankings

Table 13. The Smith-Srinivasan-Thompson algorithm, with splitting of the range of n

Type of function	Best fit	Standard error of estimation	Coefficient of determination
	$50 \leq n \leq 110$		
Polynomial	$0.22 \times 10^{-5} \times n^{3.645}$	0.582	0.748
Superpolynomial	$0.58 \times 10^{-2} \times n^{0.419 \log n}$	0.599	0.732
Exponential	$0.368 \times e^{0.0457n}$	0.664	0.672
	$120 \leq n \leq 180$		
Polynomial	$0.14 \times 10^{-4} \times n^{3.243}$	0.5183	0.400
Superpolynomial	$0.43 \times 10^{-1} \times n^{0.327 \log n}$	0.5155	0.406
Exponential	$4.48 \times e^{0.0225n}$	0.5041	0.432

Table 14. The Carpaneto-Toth algorithm, with splitting of the range of n

Type of function	Best fit	Standard error of estimation	Coefficient of determination
	$40 \leq n \leq 120$		
Polynomial	$0.45 \times 10^{-4} \times n^{2.689}$	0.116	0.990
Superpolynomial	$0.12 \times 10^{-1} \times n^{0.317 \log n}$	0.120	0.989
Exponential	$0.33 \times e^{0.0364n}$	0.237	0.959
	$140 \leq n \leq 240$		
Polynomial	$0.22 \times 10^{-1} \times n^{1.406}$	0.141	0.698
Superpolynomial	$0.9 \times n^{0.134 \log n}$	0.138	0.708
Exponential	$9.0 \times e^{0.0073n}$	0.128	0.749

Table 15. The Balas-Christofides algorithm, with splitting of the range of n

Type of function	Best fit	Standard error of estimation	Coefficient of determination
	$50 \leq n \leq 175$		
Polynomial	$0.14 \times 10^{-4} \times n^{2.407}$	0.288	0.937
Superpolynomial	$0.74 \times 10^{-2} \times n^{0.267 \log n}$	0.242	0.956
Exponential	$0.54 \times 10^{-1} \times n^{0.0245n}$	0.120	0.989
	$200 \leq n \leq 325$		
Polynomial	$0.5 \times 10^{-1} \times n^{.379}$	0.100	0.984
Superpolynomial	$0.11 \times 10^{-3} \times n^{0.395 \log n}$	0.095	0.986
Exponential	$0.199 \times e^{0.0170n}$	0.087	0.988

as before for the lower half of the range of n, but almost completely reverse the rankings for the upper half of the range: ignoring differences of less than 0.01 in the coefficient of determination, the exponential function ranks first over this range for both the SST and CT algorithms, with the polynomial and superpolynomial functions tied for second place; whereas for the BC algorithm, all three functions are now tied. To the cautionary note voiced earlier, we should now add the fact that the coefficient of determination for this range of n (i.e., the upper half) is considerably weaker for SST (0.40-0.43) and CT (0.70-0.75) than for the full range of n, while for BC it is about the same, i.e., rather strong (0.98-0.99). The findings listed above are supported by additional statistical evidence, for which, as well as for the methodological details of the analysis, the reader is referred to Balas, McGuire and Toth [1983].

The conclusions that we draw from this statistical analysis are as follows.

First, over the limited range of n for which data are available, the performance of the three algorithms analyzed can be described almost equally well by each of the three function types considered: polynomial, superpolynomial and exponential. Second, while the best fitting polynomial functions are of a moderate degree (ranging between 1.4 and 4.4), the best fitting exponential functions have a base very close to 1 (ranging between $e^{0.046} = 1.079$ and $e^{0.007} = 1.012$). Note that an exponential function of this type is very different from the function e^n. While the value of the latter increases more than twice whenever the variable goes from n to n + 1, the value of 1.012^n increases only by 1.2 percent when the variable goes from n to n + 1.

EXERCISES

1. Show that, if the relaxation R of TSP used in the branch and bound procedure of section 1 (either version) has a finite solution set, and the branching rule is such that at every node i at least one solution to the relaxed problem R_i becomes infeasible for all the successor problems $R_{i1},\ldots,R_{iq}$, then the procedure is finite. For the rooted tree representation of the branch and bound procedure discussed in section 1, what is the maximum depth of the tree, i.e., the maximum length of a path joining any node to the root? Give a bound on the number of nodes of the rooted tree.

2. Let $x^* = (x^*_{ij})$ be an optimal solution to the assignment problem AP, and let AP_1 be the assignment problem obtained from AP by adding the constraint $x_{i_o j_o} = 0$ for some (i_o, j_o) such that $x^*_{i_o j_o} = 1$. Describe a version of the Hungarian method that starts with x^* and finds an optimal solution to AP_1 in $O(n^2)$ steps. (Hint: show that only one labeling is required.)

3. Show that the "breadth first" rule of always choosing the node with the best lower bound produces a search tree with a minimum number of nodes, if (i) every node selection is uniquely determined, i.e., there are no ties for the best lower bound; and (ii) the branching and bounding at any given node is done only on the basis of information generated on the path from the root of the tree to the given node. Construct examples to show that neither (i) nor (ii) is sufficient by itself (Fox, Lenstra, Rinnooy Kan and Schrage [1978]). Assuming that conditions (i), (ii) are not satisfied, describe a subproblem selection rule that combines some of the advantages of "breadth first" with some of those of "depth first" (Forrest, Hirst and Tomlin [1974], Balas [1975]).

4. A subgradient of a convex function $f(x)$ at $x = x^k$ is a vector s such that

$$f(x) - f(x^k) \geq s(x-x^k) \quad , \qquad \forall\, x,$$

and the subdifferential of $f(x)$ at x^k is the set of all subgradients of $f(x)$ at $x = x^k$.

Let $\mathcal{T}$ be the family of 1-trees in $G = (V,E)$ introduced in section 3, and let $X(\mathcal{T})$ denote the set of incidence vectors of 1-trees. Show that, if $H(\lambda^k)$ is a 1-tree whose incidence vector x^k minimizes the function

$$\sum_{i\in V} \sum_{j>i} (c_{ij} + \lambda_i^k + \lambda_j^k)x_{ij}$$

on $X(\mathcal{T})$, and d_i^k is the degree of node i in $H(\lambda^k)$, then the n-vector whose components are $d_i^k - 2$, $i\in V$, is a subgradient of $L(\lambda)$ at λ^k. Identify the subdifferential of $L(\lambda)$ at λ^k.

5. Let $G_o = (V,A_o)$ be the admissible graph defined in section 4 with respect to (u, v, w), and let $\hat{x}$ be the incidence vector of a tour $H(\hat{x})$ in G_o. Show that $H(\hat{x})$ is an optimal tour in G if $\hat{x}$ satisfies inequality (37) with equality for all $t\in T$ such that $w_t > 0$. Is this sufficient condition also necessary? (Hint: use the optimality conditions for the linear program defined by (1), (2), (37) and $x_{ij} \geq 0$, $\forall$ i,j, and its dual.)

6. Show that bounding procedure 1 of section 4 generates at most $(h-1)(h+2)/2$ cutsets, where h is the number of subtours in the optimal solution $\bar{x}$ to AP. (Hint: use the following facts: (i) any node of a strongly connected component, hence of a subtour, is reachable from any other node; (ii) every directed cutset that is generated adds to A_o at least one new arc joining some subtour to some other subtour; and (iii) when two subtours are joined by arcs in both directions, they form a strongly connected component.)

7. Show that, if B_1 is the lower bound on v(TSP) obtained by bounding procedure 1, the lower bound generated by bounding procedure 2 is

$$B_2 = B_1 + \sum_{t \in T_2} \mu_t .$$

(Hint: use the fact that if the cost of each arc in the set $\bigcup_{t \in T_2} (S_t, S_t)$ is increased by μ_2, then the value of the solution $\bar{x}$ (hence of the solution (u,v) to the dual of AP obtained at the end of procedure 2) is $v(AP) + \sum_{t \in T_2} |S_t| \mu_t$.)

8. Let k be an articulation point of the admissible graph G_o, let S_t be the node set of one of the components of $G_o - \{k\}$, and consider the two directed cutsets

$$K_t' = \{S_t, V \backslash S_t \cup \{k\}\}, \quad K_t'' = \{V \backslash S_t \cup \{k\}, S_t\}.$$

Show that the inequality

$$\sum_{(i,j) \in K_t' \cup K_t''} x_j \geq 1$$

is the sum of the inequality (5) for $S = S_t \cup \{k\}$, the inequality (3) for $S = S_t$, and the equations (2) for all $i \in S_t$ and $j \in S_t$.

9. Formulate the n-path relaxation of the TSP discussed in section 5 for both the asymmetric and the symmetric cases, with a Lagrangean function involving the equations (2) (in the asymmetric case) or (7) (in the symmetric case). Give some examples of side constraints, i.e., extra conditions, that this relaxation of TSP can accommodate but the 1-arborescence or 1-tree relaxations can not.

References

Agmon S., The relaxation method for linear inequalities. Canadian Journal of Mathematics, 6, 382-392, (1954).

Balas, E., Bivalent programming by implicit enumeration. In J. Belzer, A. G. Holzman and A. Kent (editors), Encyclopedia of Computer Science and Technology, Vol. 2, M. Dekker, New York, 479-494, (1975).

Balas, E., Cutting planes from conditional bounds: a new approach to set covering. Mathematical Programming Study 12, 19-36, (1980).

Balas, E. and Christofides, N., A restricted Lagrangean approach to the traveling salesman problem. Mathematical Programming 21, 19-46, (1981).

Balas, E. and Guignard, M., Branch and bound/implicit enumeration. Annals of Discrete Mathematics 5, 185-191, (1979).

Balas, E., McGuire, T. W. and Toth, P., Statistical analysis of some traveling salesman algorithms. GSIA, Carnegie-Mellon University (1983).

Beale, E. M. L., Branch and bound methods for mathematical programming systems, Annals of Discrete Mathematics 5, 201-220, (1979).

Bellmore, M. and Malone, J.C., Pathology of traveling salesman subtour-elimination algorithms. Operations Research, 19, 278-307, (1971).

Carpaneto, G. and Toth, P., Some new branching and bounding criteria for the asymmetric travelling salesman problem. Management Science, 26, 736-743, (1980).

Christofides, N., "Graph Theory - An Algorithmic Approach," Academic Press, London, (1975).

Christofides, N., The shortest hamiltonian chain of a graph. J. SIAM (Applied Mathematics), 19, 689-696, (1970).

Christofides, N., The travelling salesman problem. In N. Christofides, A. Mingozzi, P. Toth and C. Sandi (editors), Combinatorial Optimization, J. Wiley, 131-149, (1979).

Christofides, N., Mingozzi, A. and Toth, P., State-Space relaxation procedures for the computation of bounds to routing problems. Networks, 11, 145-164, (1981).

Crowder, H. and Padberg, M.W., Solving large scale symmetric traveling salesman problems to optimality. Management Science, 26, 495-509 (1980).

Dantzig, G.B., Fulkerson, D.R. and Johnson, S.M., Solution of a large traveling salesman problem, Operations Research, 2, 393-410, (1954).

Dantzig, G.B., Fulkerson, D.R. and Johnson, S.M., On a linear programming, combinatorial approach to the traveling salesman problem. Operations Research, 7, 58-66, (1959).

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

Eastman, W.L., Linear Programming with pattern constraints. Ph.D. Dissertation, Harvard University, (1958).

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

Fisher, M. L., The Lagrangean relaxation method for solving integer programming problems, Management Science, 27, 1-18, (1981).

Forrest, J. J. H., Hirst, J. P. H. and Tomlin, J. A., Practical solution of large and complex integer programming problems with UMPIRE. Management Science, 20, (1974).

Fox, B. L., Lenstra, J. K., Rinnooy Kan, A. H. G. and Schrage, L. E., Branching from the largest upper bound: folklore and facts. European Journal of Operational Research, 2, 191-194, (1978).

Fulkerson, D. R., Packing rooted directed cuts in a weighted directed graph. Mathematical Programming, 6, 1-13, (1974).

Garfinkel, R. S. and Nemhauser, G. L., Integer programming. Wiley, (1972).

Geoffrion, A.M., Lagrangean relaxation and its uses in integer programming. Mathematical Programming Study 2, 82-114, (1974).

Grötschel, M. and Padberg, M.W., On the symmetric traveling salesman problem I: Inequalities, II: Lifting theorems and facets. Mathematical Programming, 16, 265-280 and 281-302, (1979).

Hall, M., Combinatorial Theory. Blaisdell Publishing Co., (1967).

Helbig Hansen, K. and Krarup, J., Improvements of the Held-Karp algorithm for the symmetric traveling-salesman problem. Mathematical Programming, 4, 87-96, (1974).

Held, M. and Karp, R., The traveling-salesman problem and minimum spanning tree. Operations Research, 18, 1138-1162, (1970).

Held, M. and Karp, R., The traveling-salesman problem and minimum spanning tree: Part II. Mathematical Programming 1, 6-25, (1971).

Held, M., Wolfe, P. and Crowder, H.P., Validation of subgradient optimization, Mathematical Programming, 6, 62-88, (1974).

Houck, D., Picard, J.C., Queyranne, M. and Vemuganti, R.R., The traveling salesman problem and shortest n-paths., Tecn. Rep. University of Maryland (1977).

Kruskal, J.B., On the shortest spanning subtree of a graph and the traveling salesman problem. Proc. Amer. Math. Soc., 2, 48-50, (1956).

Kuhn, H. W., The Hungarian method for the assignment problem. Naval Research Logistics Quarterly, 2, 83-97, (1955).

Lawler, E. L., Combinatorial optimization: networks and matroids. Holt, Rinehart and Winston, (1976).

Lenstra, J. K. and Rinnooy Kan, A. H. G., On the expected performance of branch and bound algorithms. Operations Research, 26, 347-350, (1978).

Little, J.D.C., Murty, J.G., Sweeney, D.W. and Karel, C., An algorithm for the traveling salesman problem. Operations Research, 11, 972-989, (1963).

Motzkin, T. and Schoenberg, I.J., The relaxation method for linear inequalities. Canadian Journal of Mathematics, 6, 393-404, (1954).

Murty, K., An algorithm for ranking all the assignments in order of increasing cost. Operations Research, 16, 682-687, (1968).

Polyak, B.T., A general method of solving extremum problems. Soviet Mathematics Doklady, 8, 593-597 (Translation of Doklady Akademii Nauk SSSR, 174), (1967).

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

Sandi, C., Subgradient optimization. In N. Christofides, A. Mingozzi, P. Toth, and C. Sandi (editors), Combinatorial Optimization, J. Wiley, 73-91, (1979).

Shapiro, D.M., Algorithms for the solution of the optimal cost and bottleneck traveling salesman problems. Sc. D. Thesis, Washington University, St. Louis, (1966).

Shapiro, J.F., A survey of Lagrangean techniques for discrete optimization. Annals of Discrete Mathematics 5, 113-138, (1979).

Smith, T.H.C., A LIFO implicit enumeration algorithm for the asymmetric traveling salesman problem using a one-arborescence relaxation. Ph.D. Dissertation Chapter, Carnegie-Mellon University, Pittsburgh, (1975).

Smith, T.H.C. and Thompson, G.L., A LIFO implicit enumeration search algorithm for the symmetric traveling salesman problem using Held and Karp's 1-tree relaxation. Annals of Discrete Mathematics, 1, 479-493, (1977).

Smith, T.H.C., Srinivasan, V. and Thompson, G.L., Computational performance of three subtour elimination algorithms for solving asymmetric traveling salesman problems, Annals of Discrete Mathematics, 1, 495-506, (1977).

Spielberg, K., Enumerative methods in integer programming. Annals of Discrete Mathematics 5, 139-183, (1979).

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

Volgenant, T. and Jonker, R., A branch and bound algorithm for the symmetric traveling salesman problem based on the 1-tree relaxation. European Journal of Operational Research, 9, 83-89, (1982).

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: MSRR 488
2. GOVT ACCESSION NO.
3. RECIPIENT'S CATALOG NUMBER

4. TITLE (and Subtitle): Branch and Bound Methods for the Traveling Salesman Problem
5. TYPE OF REPORT & PERIOD COVERED: Technical Report March 1983
6. PERFORMING ORG. REPORT NUMBER

7. AUTHOR(s): Egon Balas and Paolo Toth
8. CONTRACT OR GRANT NUMBER(s): N00014-75-C-0621 NR 047-048

9. PERFORMING ORGANIZATION NAME AND ADDRESS: Graduate School of Industrial Administration, Carnegie-Mellon University, Pittsburgh, PA 15213
10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS

11. CONTROLLING OFFICE NAME AND ADDRESS: Personnel and Training Research Programs, Office of Naval Research (Code 434), Arlington, VA 22217
12. REPORT DATE: March 1983
13. NUMBER OF PAGES: 65

14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)
15. SECURITY CLASS. (of this report): Unclassified
15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

This document has been approved for public release and sale; its distribution is unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

branch and bound, implicit enumeration, traveling salesman problem, combinatorial optimization, algorithm performance

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This paper reviews the state of the art in enumerative solution methods for the traveling salesman problem (TSP). The introduction (Section 1) discusses the main ingredients of branch and bound methods for the TSP. Sections 2, 3, and 4 discuss classes of methods based on three different relaxations of the TSP: the assignment problem with the TSP cost function, the 1-tree problem with a Lagrangean objective function, and the assignment problem with a Lagrangean objective function. Section 5 briefly reviews some other relaxations of the TSP, while Section 6 discusses the performance of some state of the art

(cont'd)

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601

Unclassified
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

computer codes. Besides material from the literature, the paper also includes the results and statistical analysis of some computational experiments designed for the purposes of this review.