European Journal of Operational Research 59 (1992) 231–247
North-Holland

Invited Review

The Traveling Salesman Problem: An overview of exact and approximate algorithms

Gilbert Laporte

Centre de recherche sur les transports, Université de Montréal, C.P. 6128, Station A, Montreal, Canada H3C 3J7

Received May 1991; received July 1991

Abstract: In this paper, some of the main known algorithms for the traveling salesman problem are surveyed. The paper is organized as follows: 1) definition; 2) applications; 3) complexity analysis; 4) exact algorithms; 5) heuristic algorithms; 6) conclusion.

Keywords: Traveling salesman problem; survey

Introduction

The *Traveling Salesman Problem* (TSP) is one of the most widely studied combinatorial optimization problems. Its statement is deceptively simple, and yet it remains one of the most challenging problems in Operational Research. Hundreds of articles have been written on the TSP. The book edited by Lawler et al. (1985) provides an insightful and comprehensive survey of all major research results until that date. The purpose of this survey paper is less ambitious. Our main objective is to present an integrated overview of some of the best exact and approximate algorithms so far developed for the TSP, at a level appropriate for a first graduate course in combinatorial optimization.

1. Definition

Let $G = (V, A)$ be a graph where V is a set of n vertices. A is a set of arcs or edges, and let $C = (c_{ij})$ be a *distance* (or *cost*) matrix associated with A. The TSP consists of determining a minimum distance circuit passing through each vertex once and only once. Such a circuit is known as a *tour* or *Hamiltonian circuit* (or *cycle*). In several applications, C can also be interpreted as a cost or travel time matrix. It will be useful to distinguish between the cases where C (or the problem) is *symmetrical*, i.e. when $c_{ij} = c_{ji}$ for all $i,j \in V$, and the case where it is *asymmetrical*. Also, C is said to satisfy the *triangle inequality* if and only if $c_{ij} + c_{jk} \geqslant c_{ik}$ for all $i,j,k \in V$. This occurs in *Euclidean* problems, i.e. when V is a set of points in $\mathbb{R}^2$ and c_{ij} is the straight-line distance between i and j.

2. Applications

The most common practical interpretation of the TSP is that of a salesman seeking the shortest tour through n clients or cities. This basic problem underlies several vehicle routing applications, but in this case a number of side constraints usually come into play (see Laporte, 1992). Several interesting permutation problems not directly associated with routing can also be described as TSPs. Here are some selected examples.

1. *Computer wiring* (Lenstra and Rinnooy Kan, 1975). Some computer systems can be described as modules with pins attached to them. It is often desired to link these pins by means of wires, so that exactly two wires are attached to each pin and total wire length is minimized.

2. *Wallpaper cutting* (Garfinkel, 1977). Here, n sheets must be cut from a roll of wallpaper on which a pattern of length 1 is repeated. For sheet i, denote by a_i and b_i the starting and ending points on the pattern, where $0 \leqslant a_i \leqslant 1$ and $0 \leqslant b_i \leqslant 1$. Then cutting sheet j immediately after sheet i results in a waste of

$$c_{ij} = \begin{cases} a_j - b_i & \text{if } b_i \leqslant a_j, \\ 1 + a_j - b_i & \text{if } b_i > a_j. \end{cases} \tag{1}$$

The objective is to order the n sheets so as to minimize total waste. In order to define the problem as a TSP, consider a dummy sheet $n+1$ with $c_{i,n+1} = 0$ and $c_{n+1,j} = 0$ for all $i,j = 1,\dots,n$. Alternatively, define b_{n+1} as the end of the roll position at the start of cutting and assume that after the last sheet, a final cut must be made to restore the row to its original position. Then $a_{n+1} = b_{n+1}$, and c_{ij} can be defined as in (1) for all $i,j = 1,\dots,n+1$.

3. *Hole punching* (Reinelt, 1989). In several manufacturing contexts, it is necessary to punch holes on boards or metallic sheets. The problem consists of determining a minimum-time punching sequence. Such a problem occurs frequently in metallic sheet manufacturing and in the construction of circuit boards. These problems are often of large scale and must be solved in real-time.

4. *Job sequencing.* Suppose n jobs must be performed sequentially on a single machine and that c_{ij} is the change-over time if job j is executed immediately after job i. Then again, by introducting a dummy job, this problem can be formulated as a TSP.

5. *Dartboard design* (Eiselt and Laporte, 1991). Dartboards are circular targets with concentric circles, and 20 sectors identified by the numbers 1 to 20. Players throw darts at target points on the board. In the most common version of the game, the objective is to reduce an initial value of 301 to zero by substracting scores. The game rewards accuracy in the throw and it is often more important to hit one's target that to merely register a large score. A natural objective for designing a dartboard is therefore to position the 20 numbers around the board so as to maximize players' risk. For fairly accurate players, it is reasonable to assume that the sector that is hit is always the targeted sector or its neighbour. Let $\pi = (\pi(1),\dots,\pi(20))$ be any permutation of the numbers $1,\dots,20$. In what follows, $\pi(k)$ must be interpreted as $\pi(k \bmod 20)$ whenever $k < 1$ or $k > 20$. Consider a player aiming at $\pi(k)$ and hitting $\pi(k \pm 1)$ with probability p, and $\pi(k)$ with probability $1 - 2p$. For this player, the expected deviation from the aimed score is equal to $p[\pi(k-1) - \pi(k)] + p[\pi(k+1) - \pi(k)]$. A possible objective is to maximize the expected sum of square deviations, i.e. $\Sigma_{k=1}^{20}\{p[\pi(k-1) - \pi(k)]^2 + p[\pi(k+1) - \pi(k)]^2\}$. Since p is a constant, this is equivalent to solving a TSP with $c_{ij} = (i - j)^2$.

6. *Crystallography* (Bland and Shallcross, 1989). In crystallography, some experiments consist of taking a large number of X-ray intensity measurements on crystals by means of a detector. Each measurement requires that a sample of the crystal be mounted on an apparatus and that the detector be positioned appropriately. The order in which the various measurements on a given crystal are made can be seen as the solution of a TSP. In practice, these problems are of large scale and obtaining good TSP solutions can reduce considerably the time needed to carry out all measurements.

3. Complexity

In order to study the complexity of the traveling salesman problem first consider the following well-known *decision problem*:

HAMILTONIAN CIRCUIT (HC)
Instance: A graph $G = (V, A)$.
Question: Does G contain a Hamiltonian circuit?

It is well known that HC is NP-complete (Garey and Johnson, 1979, p. 47). We show that TSP is

NP-hard by using the following transformation. Given any instance of HC relative to a graph $G^h = (V^h, A^h)$ with vertex set $V^h = \{1, \dots, n\}$ and arc set $A^h = \{(i, j)\}$, define a TSP instance having $V = V^h$, $A = \{(i, j): i, j = 1, \dots, n, i \neq j\}$ and $c_{ij} = 1$ if $(i, j) \in A^h$ and $c_{ij} = \infty$, otherwise. Then G contains a Hamiltonian circuit if and only if the optimal value of the TSP instance is equal to n.

A number of special cases of TSP are, however, solvable in polynomial time (see, for example, Gilmore, Lawler and Shmoys, 1985). Examples of such problems include:

1. TSPs where $C = (c_{ij})$ is an upper triangular matrix, i.e. $c_{ij} = 0$ for all $i \geqslant j$;
2. The wallpaper cutting problem described in Section 2;
3. A class of job sequencing problems defined by Gilmore and Gomory (1964). In this problem, there are $n - 1$ jobs to be processed sequentially in a kiln. Job i requires a starting temperature of a_i and must be finished at temperature b_i. Further assume that the initial kiln temperature is a_n and that the final temperature must be b_n. Then the problem can be formulated as a TSP with

$$c_{ij} = \begin{cases} \int_{b_i}^{a_i} f(x)\,\mathrm{d}x & \text{if } b_i \leqslant a_i, \\ \int_{a_i}^{b_i} g(x)\,\mathrm{d}x & \text{if } a_i \leqslant b_i \end{cases}$$

where f and g are cost density functions and $f(x) + g(x) \geqslant 0$ for all x since otherwise, it would be profitable to keep changing the kiln temperature.

4. Exact algorithms

A large number of exact algorithms have been proposed for the TSP. These can be best understood and explained in the context of integer linear programming (ILP). We examine in this section a number of ILP formulations and of algorithms derived from these formulations.

4.1. Integer linear programming formulations

One of the earliest formulations is due to Dantzig, Fulkerson and Johnson (DFJ) (1954). It associates one binary variable x_{ij} to every arc (i, j), equal to 1 if and only if (i, j) is used in the optimal solution, $i \neq j$. The DFJ formulation is (DFJ)

$$\text{Minimize} \quad \sum_{i \neq j} c_{ij} x_{ij} \tag{2}$$

$$\text{subject to} \quad \sum_{j=1}^{n} x_{ij} = 1, \quad i = 1, \dots, n, \tag{3}$$

$$\sum_{i=1}^{n} x_{ij} = 1, \quad j = 1, \dots, n, \tag{4}$$

$$\sum_{i,j \in S} x_{ij} \leqslant |S| - 1, \quad S \subset V,\ 2 \leqslant |S| \leqslant n - 2, \tag{5}$$

$$x_{ij} \in \{0, 1\}, \quad i, j = 1, \dots, n,\ i \neq j. \tag{6}$$

In this formulation, the objective function clearly describes the cost of the optimal tour. Constraints (3) and (4) are *degree constraints*: they specify that every vertex is entered exactly once (3) and left exactly once (4). Constraints (5) are *subtour elimination constraints*: they prohibit the formation of *subtours*, i.e. tours on subsets of less than n vertices. If there was such a subtour on a subset S of vertices, this subtour would contain $|S|$ arcs and as many vertices. Constraint (5) would then be violated for this subset since its left-hand side would be equal to $|S|$ and its right-hand side equal to $|S| - 1$. Because of degree constraints, subtours over one vertex (and hence, over $n - 1$ vertices) cannot occur. Therefore it is valid to define constraints (5) for $2 \leqslant |S| \leqslant n - 2$ only. Finally, constraints (6) impose binary conditions on the variables.

An alternative equivalent form of constraints (5) is

$$\sum_{i \in S} \sum_{j \in \bar{S}} x_{ij} \geqslant 1 \quad S \subset V,\ 2 \leqslant |S| \leqslant n - 2 \tag{5'}$$

where $\bar{S} = V \setminus S$. Constraints (5′) can be derived from (5) by noting that every vertex i of S is the origin of one arc to another vertex of S or to a vertex of $\bar{S}$. Since there are $|S|$ vertices, $|S| = \Sigma_{i,j \in S} x_{ij} + \Sigma_{i \in S} \Sigma_{j \in \bar{S}} x_{ij}$, and the equivalence of (5) and (5′) follows trivially. The geometric interpretation of *connectivity constraints* (5′) is that in every TSP solution, there must be at least one arc pointing from S to its complement, in other words, S cannot be disconnected.

This formulation contains $n(n-1)$ binary variables, $2n$ degree constraints and $2^n - 2n - 2$ subtour elimination constraints. Even for moderate values of n, it is unrealistic to solve DFJ directly by means of an ILP code. The model is usually relaxed, and solved by means of specialized algorithms.

Miller, Tucker and Zemlin (MTZ) (1960) have proposed an alternative formulation that reduces the number of subtour elimination constraints at the expense of extra variables u_i $(i = 2,\dots,n)$. The MTZ subtour elimination constraints can be expressed as

$$u_i - u_j + (n-1)x_{ij} \leqslant n-2 \quad i, j = 2,\dots,n,\ i \neq j, \tag{7}$$

$$1 \leqslant u_i \leqslant n-1 \quad i = 2,\dots,n. \tag{8}$$

Constraints (7) ensure that the solution contains no subtour on a set of vertices $S \subseteq V \backslash \{1\}$ and hence, no subtour involving less than n vertices. Constraints (8) ensure that the u_i variables are uniquely defined for any feasible tour. In order to see how constraints (7) operate, suppose there was a subtour $(i_1, i_2,\dots,i_k, i_1)$ with $k < n$. Writing constraints (7) for every arc of that subtour gives

$$u_{i_1} - u_{i_2} + (n-1) \leqslant n-2,$$
$$u_{i_2} - u_{i_3} + (n-1) \leqslant n-2,$$
$$\vdots$$
$$u_{i_k} - u_{i_1} + (n-1) \leqslant n-2.$$

Summing up these constraints yields $k(n-1) \leqslant k(n-2)$, a contradiction.

It has been observed recently (Desrochers and Laporte, 1991) that constraints (7) can be strengthened by introducing an extra term in their left-hand side to yield

$$u_i - u_j + (n-1)x_{ij} + (n-3)x_{ji} \leqslant n-2 \quad i, j = 2,\dots,n,\ i \neq j. \tag{9}$$

The validity of constraints (9) can be established as follows. Rewrite constraints (7) as

$$u_i - u_j + (n-1)x_{ij} + \alpha_{ji}x_{ji} \leqslant n-2 \tag{10}$$

where currently $\alpha_{ji} = 0$. We seek the largest possible value of α_{ji} so that (10) remains a valid inequality. In the optimal solution, x_{ji} can take only two values:

- if $x_{ji} = 0$, then (10) is satisfied for any α_{ji};
- if $x_{ji} = 1$, then $x_{ij} = 0$ (for $n > 2$) and $u_j + 1 = u_i$, so that $\alpha_{ji} \leqslant n-3$.

In spite of its relative compactness, the MTZ formulation is weaker than the DFJ formulation in the following sense. Denote by $z'(\text{DFJ})[z'(\text{MTZ})]$ the optimal value of the linear relaxation of DFJ [MTZ], i.e. the relaxation obtained by dropping integrality conditions. Then $z'(\text{MTZ}) \leqslant z'(\text{DFJ})$ (Wong, 1980). This result has not been proved for the modified MTZ formulation using constraints (9), but it is known that there exist cases where it produces a weaker linear relaxation than DFJ (Desrochers and Laporte, 1991).

Finally, a number of alternative formulations have been proposed and compared, but none of these seems to have a stronger linear relaxation than DFJ (Langevin, Soumis and Desrosiers, 1990).

4.2. The assignment lower bound and related branch-and-bound algorithms

Branch-and-bound (BB) algorithms are commonly used for the solution of TSPs. In the context of mathematical programming, they can best be viewed as initially relaxing some of the problem constraints, and then regaining feasibility through an enumerative process. The quality of a BB algorithm is directly related to the quality of the bound provided by the relaxation.

For the TSP, an initial lower bound can be obtained from the DFJ formulation by relaxing constraints (5). The resulting problem is an assignment problem (AP) which can be solved in $O(n^3)$ time (see Carpaneto, Martello and Toth, 1988). Thus, a valid lower bound on the value of the optimal TSP solution is the AP bound defined by (2)–(4) and the nonnegativity requirements on the variables.

Several authors have proposed BB algorithms for the TSP, based on the AP relaxation. These include Eastman (1958), Little et al. (1963), Shapiro (1966), Murty (1968), Bellmore and Malone (1971), Garfinkel (1973), Smith, Srinivasan and Thompson (1977), Carpaneto and Toth (1980), Balas and Christofides (1981) and Miller and Pekny (1991). We briefly describe the last three algorithms which are probably the best available.

In the Carpaneto and Toth algorithm, the problem solved at a generic node of the search tree is a modified assignment problem (i.e. x_{ii} is fixed at 0 for all i) in which some x_{ij} variables are fixed at 0 or at 1. If the AP solution consists of a unique tour over all vertices, it is then feasible for the TSP. Otherwise, it consists of a number of subtours. One of these subtours is selected and broken by creating subproblems in which all arcs of the subtour are in turn prohibited. We will use the following notation:

z^*: the cost of the best TSP solution so far identified;

z_h: the value of the objective function of the modified AP at node h of the search tree;

$\underline{z}_h$: a lower bound on z_h;

I_h: the set of included arcs (x_{ij} variables fixed at 1) at node h of the search tree;

E_h: the set of excluded arcs (x_{ij} variables fixed at 0) at node h of the search tree.

Step 1. (Initialization.) Obtain a first value for z^* by means of a suitable heuristic. Create node 1 of the search tree: set $I_1 := E_1 := \emptyset$, and obtain z_1 by solving the associated modified AP. If $z_1 \geqslant z^*$, stop: the heuristic solution is optimal. If the solution contains no illegal subtours, it constitutes the optimal tour: stop. Otherwise, insert node 1 in a queue.

Step 2. (Node selection.) If the queue is empty, stop. Otherwise, select the next node (node h) from the queue: here we use a *breadth first* rule, i.e. branching is always done on the pendant node having the lowest z_h.

Step 3. (Subproblem partitioning.) The solution obtained at node h is illegal and must be eliminated by partitioning the current subproblem into descendant subproblems h_r characterized by sets I_{h_r} and E_{h_r}. In order to create these subproblems, consider a subtour having the least number s of arcs not belonging to I_h. Let these arcs be $(i_1, j_1), \ldots, (i_s, j_s)$, in the order in which they appear in the subtours. Then create s subproblems with

$$I_{h_r} = \begin{cases} I_h, & r = 1, \\ I_h \cup \{(i_u, j_u) : u = 1, \ldots, r-1\}, \\ \quad r = 2, \ldots, s, \end{cases}$$

$$E_{h_r} = E_h \cup \{(i_r, j_r)\}, \quad r = 1, \ldots, s.$$

Execute Step 4–6 for $r = 1, \ldots, s$.

Step 4. (Bounding.) Compute a lower bound $\underline{z}_{h_r}$ on z_{h_r} by row and column reduction of the cost matrix. If $\underline{z}_{h_r} < z^*$, proceed to Step 5. Otherwise, consider the next r and repeat Step 4.

Step 5. (Subproblem solution.) Solve the subproblem associated with node h_r (a modified *AP* restricted by I_{h_r} and E_{h_r}). If $z_{h_r} \geqslant z^*$, consider the next r and proceed to Step 4.

Step 6. (Feasibility check). Check whether the current solution contains subtours. If it does, insert node h_r in the queue. Otherwise, set $z^* := z_{h_r}$ and store the tour, if $z^* = z_h$, go to Step 2.

Using their algorithm, Carpaneto and Toth have consistently solved randomly generated 240-vertex TSPs in less than one minute on a CDC 6600. The main limitation of this algorithm appears to be computed memory rather than CPU time.

The Balas and Christofides algorithm uses a stronger relaxation than the AP relaxation. Its description and the computational effort required for the lower-bound computations are much more involved, but the resulting search trees are smaller and the procedure is overall more powerful. Due to its complexity, it will only be sketched here. Interested readers are referred to the Balas and Christofides (1981) paper. In addition to constraints (3), (4) and (6), subtour elimination constraints (5), connectivity constraints (5′) and some positive linear combinations of these are considered, and introduced into the objective function in a Lagrangean fashion. Let T be the set of all such constraints and t a given constraint. Their generic expression can be written as

$$\sum_{i,j \in V} a_{ij}^t x_{ij} \geqslant a_0^t, \quad t \in T.$$

The Lagrangean objective is then

$$L(\lambda) = \min_x \left\{ \sum_{i,j \in V} c_{ij} x_{ij} - \sum_{t \in T} \lambda_t \times \left(\sum_{i,j \in V} a_{ij}^t x_{ij} - a_0^t \right) \right\}$$

where $\lambda = (\lambda_1, \ldots, \lambda_T)$ is the vector of Lagrange multipliers, and x is any modified AP solution. The strongest relaxation is given by $\max_{\lambda \geqslant 0} \{L(\lambda)\}$. As the number of components of λ is exponential, the required maximum is not determined by Lagrangean relaxation. Instead, a lower bound on its value is obtained by restricting λ to lie in the set $\{\lambda \geqslant 0$: $\exists u, v \in \mathbb{R}^n$, such that $u_i + v_j + \sum_{t \in T} \lambda_t \; a_{ij}^t = c_{ij}$ if (i, j) belongs to the AP solution; $u_i + v_j + \sum_{t \in T} \lambda_t \; a_{ij}^t \geqslant c_{ij}$, otherwise$\}$. Balas and Christofides propose an approximation procedure for computing λ. At a given node of the search tree, the procedure solves an AP with dual values given by u and v. It then computes λ. A lower bound on the value of the TSP tour associated to that node of the search tree is given by

$$\sum_{i \in V} u_i + \sum_{j \in V} v_j + \sum_{t \in T} \lambda_t a_t^0. \tag{12}$$

The authors then show how (12) can be computed for three particular classes of constraints (11). Using this procedure, Balas and Christofides report optimal solutions to randomly generated 325-vertex problems in less than one minute on a CDC 7600.

More recently, Miller and Pekny have proposed a new powerful BB algorithm based on the AP relaxation. Consider the dual AP:
(DAP)

$$\text{Maximize} \quad \sum_{i=1}^{n} u_i + \sum_{j=1}^{n} v_j \tag{13}$$

$$\text{subject to} \quad c_{ij} - u_i - v_j \geqslant 0$$
$$i, j = 1, \ldots, n, \; i \neq j. \tag{14}$$

Denote by $z^*(\text{TSP})$ the optimal TSP solution value, by $z^*(\text{AP})$ the optimal value of the AP linear relaxation, and by $z^*(\text{DAP})$ the optimal value of the dual AP linear relaxation. Clearly $z^*(\text{AP}) = z^*(\text{DAP})$. Moreover, note that $z^*(AP) + (c_{ij} - u_i - v_j)$ is a lower bound on the cost of an AP solution that includes arc (i, j). Miller and Pekny make use of this in an algorithm that initially removes from consideration all x_{ij} variables whose cost c_{ij} exceeds a threshold value λ. Consider a modified problem TSP$'$ with associated linear assignment relaxation AP$'$ and its dual DAP$'$, obtained by redefining the costs c_{ij} as follows:

$$c'_{ij} = \begin{cases} c_{ij} & \text{if } c_{ij} \leqslant \lambda, \\ \infty & \text{otherwise}. \end{cases} \tag{15}$$

The authors prove the following proposition which they use as a basis for their algorithm: an optimal solution for TSP$'$ is optimal for TSP if

$$z^*(\text{TSP}') - z^*(\text{AP}) \leqslant \lambda + 1 - u'_i - v'_{\max} \tag{16}$$

and

$$\lambda + 1 - u'_i - v'_{\max} \geqslant 0 \tag{17}$$

for $i = 1, \ldots, n$, where u' and v' are optimal solutions to DAP$'$, and $v'_{\max}$ is the maximum element of v'. The quantity $\lambda + 1 - u'_i - v'_{\max}$ underestimates the smallest reduced cost of any discarded variable. The algorithm is then:

Step 1 (Initialization). Choose λ.

Step 2 (TSP$'$ solution). Construct (c'_{ij}) and solve TSP$'$.

Step 3 (Termination check). If (16) and (17) hold, then $z^*(\text{TSP}') = z^*(\text{TSP})$: stop. Otherwise, double λ and go to Step 2.

The authors report that if λ is suitably chosen in Step 1 (e.g., the largest arc cost in a heuristic solution), there is rarely any need to perform a second iteration. To solve TSP$'$, the authors have developed a BB algorithm based on the AP relaxation. They have applied this procedure to randomly generated problems. Instances involving up to 5000 vertices were solved within 40 seconds on a Sun 4/330 computer. The largest problem reported solved by this approach contains 500000 vertices and required 12623 seconds of computing time on a Cray 2 supercomputer.

Finally, it is worth mentioning that in a different paper, Miller and Pekny (1989) describe a parallel branch-and-bound algorithm based on the AP relaxation. These authors report that randomly generated asymmetrical TSPs involving up to 3000 vertices have been solved to optimality using this approach.

4.3. *The shortest spanning arborescence bound and a related algorithm*

In a directed graph $G = (V, A)$, an r-arborescence is a partial graph in which the in-degree of each vertex is exactly 1 and each vertex can be reached from the root vertex r. The shortest

Shortest spanning arborescence é a mesma coisa que arvore geradora mínima. Porém é utilizado para grafos direcionados.

spanning r-arborescence problem (r-SAP) can be formulated as
(r-SAP)

$$\text{Minimize} \quad \sum_{i \neq j} c_{ij} x_{ij} \tag{18}$$

$$\text{subject to} \quad \sum_{\substack{i=1 \\ i \neq j}}^{n} x_{ij} = 1, \; j = 1, \dots, n, \tag{19}$$

$$\sum_{i \in S} \sum_{j \in \bar{S}} x_{ij} \geqslant 1, \; S \subset V; \; r \in S, \tag{20}$$

$$x_{ij} \geqslant 0, \; i, j = 1, \dots, n, \; i \neq j. \tag{21}$$

The problem of determining a minimum-cost r-arborescence on G can be decomposed into two independent subproblems: determining a minimum-cost arborescence rooted at vertex r, and finding the minimum-cost arc entering vertex r. The first problem is easily solved in $O(n^2)$ time (Tarjan, 1977). This relaxation can be used in conjunction with Lagrangian relaxation. However, on asymmetric problems, the AP relaxation would appear empirically superior to the r-arborescence relaxation (Balas and Toth, 1977).

An early reference to this lower bound is made by Held and Karp (1970). More recently, Fischetti and Toth (1991) have used it within a so-called 'additive bounding procedure' that combines five different bounds:

– the AP bound,
– the shortest spanning 1-arborescence bound,
– the shortest spanning 1-antiarborescence bound 1 – SAAP; (r-SAAP is defined in a manner similar to r-SAP but now it is required that vertex r should be reached from every remaining vertex),
– for $r = 1, \dots, n$, a bound r-SADP obtained from r-SAP by adding the constraint

$$\sum_{j \neq r} x_{rj} = 1, \tag{22}$$

– for $r = 1, \dots, n$, a bound r-SAADP obtained from r-SAAP by adding the constraint

$$\sum_{i \neq r} x_{ir} = 1. \tag{23}$$

The lower-bounding procedure described by Fischetti and Toth was embedded within the Carpaneto and Toth (1980) branch-and-bound algorithm on a variety of randomly generated problems and on some problems described in the literature. The success of the algorithm depends on the type of problem considered. For the easiest problem type, the authors report having solved 2000-vertex problems in an average time of 8329 seconds on an HP 9000/840 computer.

4.4. *The shortest spanning tree bound and related algorithms*

The AP-based algorithms described in the previous section are valid whether C is asymmetrical or symmetrical. However, in the latter case, the AP solutions will in general contain several subtours containing only two vertices, resulting in excessive computing times for their elimination. Symmetrical problems are better handled by specialized algorithms that exploit their structure. We now describe an ILP formulation for symmetrical TSPs.

Consider a TSP defined on $G = (V, E)$ where E is an *edge set*. Let x_{ij} be a binary variable equal to 1 if and only if edge (i, j) is used in the optimal solution. These variables are only defined for $i < j$. The formulation is then
(SYM)

$$\text{Minimize} \quad \sum_{i<j} c_{ij} x_{ij} \tag{24}$$

$$\text{subject to} \quad \sum_{i<k} x_{ik} + \sum_{j>k} x_{kj} = 2, \quad k = 1, \dots, n, \tag{25}$$

$$\sum_{i,j \in S} x_{ij} \leqslant |S| - 1, \quad S \subset V, \; 3 \leqslant |S| \leqslant n-3, \tag{26}$$

$$x_{ij} \in \{0, 1\}, \quad i, j = 1, \dots, n, \; i < j. \tag{27}$$

In this formulation, constraints (25) specify that every vertex has a degree of 2. Constraints (26) are subtour elimination constraints. They need not be defined for $|S| = 2$ and $n - 2$ since constraints (25) and (27) combined prevent the formation of subtours involving 2 (and thus $n - 2$) vertices. Constraints (27) impose binary conditions on the variables.

As for (DFJ), connectivity constraints equivalent to (26) can be derived. These can be written as

$$\sum_{\substack{i \in S, j \in \bar{S} \\ \text{or } j \in S, i \in \bar{S}}} x_{ij} \geqslant 2, \quad S \subset V, \; 3 \leqslant |S| \leqslant n-3. \tag{26'}$$

In order to show the equivalence of (26) and (26′), consider $D(S) = \sum_{k \in S}(\sum_{i<k} x_{ik} + \sum_{j>k} x_{kj})$, the sum of degrees of vertices of S. Every edge (i, j) for which $i,j \in S$ makes a contribution of 2 to $D(S)$, whereas edges (i, j) with $i \in S$, $j \in \bar{S}$ or $j \in S$, $i \in \bar{S}$ contribute by only one unit. Hence,

$$D(S) = 2 \sum_{i,j \in S} x_{ij} + \sum_{\substack{i \in S, j \in \bar{S} \\ \text{or } j \in S, i \in \bar{S}}} x_{ij},$$

and the equivalence follows.

A useful relaxation can be extracted from SYM by exploiting the following consideration. In any feasible solution, the degree of vertex 1 must be equal to 2, while the remaining vertices must all be connected. A valid lower bound on the optimal TSP solution value is therefore the length of the shortest 1-spanning tree (1-SST), i.e. the shortest tree having vertex set $V \setminus \{1\}$, together with two distinct edges at vertex 1. Formally, determining a least cost 1-SST is achieved by solving

(1-SST)

$$\text{Minimize} \quad \sum_{i<j} c_{ij} x_{ij} \tag{28}$$

$$\text{subject to} \quad \sum_{i<j} x_{ij} = n, \tag{29}$$

$$\sum_{j=2}^{n} x_{1j} = 2, \tag{30}$$

$$\sum_{\substack{i \in S, j \in \bar{S} \setminus \{1\} \\ \text{or } j \in S, i \in \bar{S} \setminus \{1\}}} x_{ij} \geqslant 1, \quad S \subset V \setminus \{1\},\ 1 \leqslant |S| \leqslant n-1, \tag{31}$$

$$x_{ij} \in \{0, 1\}. \tag{32}$$

However, in practice the 1-SST problem is better solved by means of a specialized algorithm (see Aho, Hopcroft and Ullman, 1974). This formulation is clearly a relaxation of SYM. Indeed, constraint (29) is obtained by taking half the sum of constraints (25); constraint (30) is constraint (25) for $k = 1$, and constraints (31) are a weaker form of (26′). The fact that these constraints are also imposed for $|S| = 1,2,n-2$ and $n-1$ is of no concern since it would have been valid (and redundant) to impose (27) for these values of $|S|$.

Christofides (1970) and Held and Karp (1971) were among the first to propose a TSP algorithm based on this relaxation. Improvements and refinements were later suggested by Helbig Hansen and Krarup (1974), Smith and Thompson (1977), Volgenant and Jonker (1982), Gavish and Srikanth (1986), and Carpaneto, Fischetti and Toth (1989). In the original Held and Karp algorithm, the 1-SST bound is reinforced by introducing constraints (25) in the objective function to yield the Lagrangean

$$\begin{aligned} L(\lambda) &= \min_x \left\{ \sum_{i<j} c_{ij} x_{ij} + \sum_{k \in V} \lambda_k \left(\sum_{i<k} x_{ik} + \sum_{j>k} x_{kj} - 2 \right) \right\} \\ &= \min_x \left\{ \sum_{i \in V} \sum_{j>i} (c_{ij} + \lambda_i + \lambda_j) x_{ij} \right\} - 2 \sum_{i \in V} \lambda_i, \end{aligned}$$

where x is a feasible 1-SST solution. By performing dual ascents, Held and Karp obtain a lower bound on the strongest Lagrangean relaxation $\max_\lambda\{L(\lambda)\}$. This bounding process is embedded in a branch-and-bound scheme in the following manner. First define

z^*: the cost of the best TSP solution so far identified;

z_h: the best lower bound on $\max_\lambda\{L(\lambda)\}$ derived at node h of the search tree;

λ_h: the value of λ yielding z_h;

I_h: the set of included edges at node h of the search tree;

E_h: the set of excluded edges at node h of the search tree.

Step 1 (initialization). Obtain a first value for z^* by means of a suitable heuristic. Create node 1 of the search tree: set $I_1 := E_1 := \emptyset$. Obtain z_1 and λ_1 by performing Lagrangean ascents, starting with $\lambda = (0)$. If $z_1 \geqslant z^*$, stop: the heuristic solution is optimal. If the solution consists of a tour, it is feasible and optimal: stop. Otherwise, insert node 1 in a queue.

Step 2 (Node selection). If the queue is empty, stop. Otherwise, select the next node (node h)

from the queue on with to branch: select h with the least lower bound z_h.

Step 3 (Subproblem partitioning). Rank the edges $e_1, \ldots, e_p$ of $E \setminus (I_h \cup E_h)$ according to the amount by which z_h would increase if the edge was excluded. In other words, if h_r was a node of the search tree with $I_{h_r} = I_h$ and $E_{h_r} = E_h \cup \{e_r\}$, then $z_{h_1} \geqslant z_{h_2} \geqslant \cdots \geqslant z_{h_p}$. The sets of included and excluded arcs in subproblems originating from node h are

$$\begin{array}{ll}
I_{h_1} = I_h, & E_{h_1} = E_h \cup \{e_1\}, \\
I_{h_2} = I_h \cup \{e_1\}, & E_{h_2} = E_h \cup \{e_2\}, \\
I_{h3} = I_h \cup \{e_1, e_2\}, & E_{h_3} = E_h \cup \{e_3\}, \\
\vdots & \vdots \\
I_{h_s} = I_h \cup \{e_1, \ldots, e_{s-1}\}, & E_{h_s} = E_h \cup R_i \cup R_j
\end{array}$$

where $s \leqslant p$ is the smallest index for which there exists a vertex i such that I_{h_s} does not contain two edges incident to i, but $I_{h_{s+1}}$ does; there may exist another vertex j possessing this property. R_i is the set of all edges incident to i and not in I_{h_s}. *Execute Step 4 and 5 for r = 1, . . . , s.*

Step 4 (Bounding). Compute z_{h_r}, starting with λ_h. If $z_{h_r} < z^*$, proceed to Step 5. Otherwise, consider the next r and repeat Step 4.

Step 5 (Feasibility check). If the solution obtained at node h_r is not a tour, insert node h_r in the queue. Otherwise, set $z^* := z_{h_r}$ and store the tour; if $z^* = z_h$, go to Step 2.

Using this procedure, Held and Karp (1971) have solved a number of classical TSPs with very limited branching. Helbig Hansen and Krarup (1974) have improved upon the original algorithm by a more judicious choice of parameters in the ascent procedure. Volgenant and Jonker (1982) have experimented with a new ascent procedure, upper-bound computations in the branch-and-bound tree, and new branching schemes. Gavish and Srikanth (1986) use fast sensitivity analysis techniques to increase the underlying graph sparsity and reduce the problem size. More recently, Carpaneto, Fischetti and Toth (1989) have suggested a number of further improvements to the Held and Karp lower bound by making use of an additive bounding procedure.

4.5. The 2-matching lower bound and related algorithms

The 2-matching relaxation of the TSP is extracted from SYM by omitting constraints (26). It provides a lower bound on the value of the optimal TSP solution. This relaxation can then be embedded in an optimization algorithm by first solving the linear relaxation of the 2-matching problem, and by then gradually introducing violated subtour elimination constraints and integrality constraints. The principles behind this method were first laid out in the seminal papers by Dantzig, Fulkerson and Johnson (1954, 1959). This work was followed by that of Martin (1966) and of Miliotis (1976, 1978). It was Miliotis who, to our knowledge, developed the first completely automatic procedure for solving symmetrical TSPs using this relaxation. He proposed several algorithms that differ in the order in which the violated constraints are introduced and in the way integer solutions are reached. One of these is a pure cutting planes algorithm:

Step 1 (First subproblem). Solve a first subproblem defined by (24), (25) and $0 \leqslant x_{ij} \leqslant 1$ ($i, j = 1, \ldots, n$, $i < j$). If the solution is feasible, stop. If the solution is infeasible but integer, go to Step 3.

Step 2 (Reoptimization). Regain optimality by introducing Gomory cutting planes (Gomory, 1963) and by reoptimizing. If the solution is feasible, it is also optimal: stop.

Step 3 (Subtour elimination). Eliminate one or several subtours by introducing the appropriate subtour elimination constraint(s). Reoptimize. If the solution is not integer, go to Step 2. If the solution is integer and contains subtours, repeat Step 3.

A variant of this problem consists of reaching integrality by branch and bound. Subtour elimination constraints are then introduced in individual subproblems through the branching process. These constraints remain, however, valid for the whole of the branch-and-bound tree.

As indicated by Miliotis, the two steps that consists of checking for integrality and then for violated subtour elimination constraints can be interverted, leading to a stronger algorithm. It must be observed that subtour elimination constraints can be applied to fractional components, i.e. connected subgraphs having fractional x_{ij} variables associated with some of their edges. Efficient procedures for generating violated subtour elimination constraints have been suggested by Gomory and Hu (1961), Crowder and Padberg (1980), and Padberg and Rinaldi (1990).

Miliotis' work was followed by that of Crowder and Padberg (1980), Padberg and Hong (1980), Padberg and Rinaldi (1987, 1990) and Grötschel and Holland (1991), among others. The basic idea behind this line of research consists of introducing several types of valid constraints before branching on fractional variables (on this subject, see Grötschel and Padberg, 1985, and Padberg and Grötschel, 1985). The effect of this is to increase the value of the LP relaxation and thus, to limit the growth of the search tree. These constraints are particularly powerful since they are facets of the polytope of integer solutions of SYM. Here are the most commonly used constraints (see Padberg and Rinaldi, 1990):

(a) *Subtour elimination constraints*,

$$\sum_{i,j \in S} x_{ij} \leqslant |S| - 1, \quad S \subset V,\ 3 \leqslant |S| \leqslant n-3, \tag{26}$$

(b) *2-matching inequalities*,

$$\sum_{i,j \in H} x_{ij} + \sum_{(i,j) \in E'} x_{ij} \leqslant |H| + \tfrac{1}{2}(|E'| - 1), \tag{33}$$

for all $H \subset V$ and all $E' \subset E$ satisfying

(i) $|\{i, j\} \cap H| = 1$ for all $(i, j) \in E'$;

(ii) $|\{i, j\} \cap \{k, l\}| = \emptyset$, $(i, j) \neq (k, l) \in E'$;

(iii) $|E'| \geqslant 3$ and odd.

The set H is called the *handle* and the edges of E' are called *teeth* (see Figure 1).

(c) *Comb inequalities*,

$$\sum_{i,j \in H} x_{ij} + \sum_{k=1}^{s} \sum_{i,j \in T_k} x_{ij} \leqslant |H| + \sum_{k=1}^{s} (|T_k| - 1) - \tfrac{1}{2}(s-1), \tag{34}$$

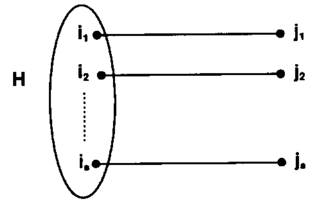

Figure. 1. 2-Matching inequalities. $E = \{(i_1, j_1), \ldots, (i_s, j_s)\}$

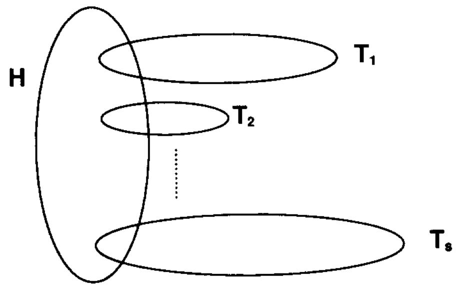

Figure 2. Comb inequalities

for all $H, T_1, \ldots, T_s \subset V$ satisfying

(i) $|T_k \cap H| \geqslant 1$ $(k = 1, \ldots, s)$;

(ii) $|T_k \setminus H| \geqslant 1$ $(k = 1, \ldots, s)$;

(iii) $T_k \cap T_l = \emptyset$ $(1 \leqslant k < l \leqslant s)$;

(iv) $s \geqslant 3$ and odd.

Again, H is called the handle and $T_1, \ldots, T_s$ are called the teeth of the comb (see Figure 2).

(d) *Clique tree inequalities*,

$$\sum_{k=1}^{r} \sum_{i,j \in H_k} x_{ij} + \sum_{l=1}^{s} \sum_{i,j \in T_l} x_{ij} \leqslant \sum_{k=1}^{r} |H_k| + \sum_{l=1}^{s} (|T_l| - t_l) - (s+1)/2, \tag{35}$$

for all $H_1, \ldots, H_r \subseteq V$, and all $T_1, \ldots, T_s \subseteq V$ which are handles and teeth, respectively, of a clique tree. A *clique tree* is a connected subgraph of the graph K_n derived from G by defined all $\frac{1}{2}n(n-1)$ potential edges. The cliques of a clique tree are partitioned into handles and teeth; they satisfy the following properties:

(i) no two teeth intersect;

(ii) no two handles intersect;

(iii) each tooth contains at least 2 and at most $n-2$ vertices, and at least one vertex not belonging to any handle;

(iv) each handle intersects an odd number ($\geqslant 3$) of teeth;

(v) if a tooth T and a handle H have a nonempty intersection, then $H \cap T$ is an *articulation set* of the clique tree, i.e. a minimal set S of vertices such that the subgraph of $G = (V, E)$ induced by $V \setminus S$ has more connected components than G.

Finally, in (35), t_l is the number of handles intersected by tooth T_l.

Padberg and Rinaldi (1990) and Grötschel and Holland (1991) suggest efficient procedures for identifying violated instances of these four types of constraints. By incorporating these procedures within an ILP code, these authors have solved to optimality TSPs containing between 17 and 2392 vertices. At time of writing, the 2392-vertex problem is believed to be the largest nonrandom symmetrical TSP ever solved to optimality.

5. Heuristic algorithms

Since the TSP is a NP-hard problem, it is natural to tackle it by means of heuristic algorithms. One stream of research has consisted of developing heuristics with a guaranteed *worst-case* performance. Most effort has, however, been devoted to the design of heuristics with good *empirical* performance. In this section, we examine these two streams.

5.1. Heuristics with guaranteed worst-case performance

Consider a symmetrical TSP defined on a graph G and where C satisfies the triangle inequality. Let z^* be the value of the optimal TSP solution. A simple way to derive a lower bound on z^* is to first compute the length of a shortest spanning tree T on G. As shown by Aho, Hopcroft and Ullman (1974), this can be done in $O(n^2)$ time. Denote by $l(T)$ the length of that tree. A possible strategy for visiting all vertices is to traverse the spanning tree along its edges in the following fashion:

Step 1 Consider any leaf i_0 (vertex of degree 1) of the spanning tree and set $i := i_0$.

Step 2

– If there is any untraversed edge (i, j) incident to vertex i, follow that edge to vertex j. Set $i := j$ and repeat Step 2.

– If all edges from vertex i have already been traversed, go back to the vertex k from which i was first reached. If $k = i_0$, stop; otherwise set $k := i$ and repeat Step 2.

In order to illustrate this procedure, consider the spanning tree depicted in Figure 3. Starting with $i_0 = 1$, one then proceeds to 3, 2, 3, 4, 5, 6, 5, 8, 5, 7, 5, 4, 3 and 1. It is easy to see that every edge of the spanning tree is then covered exactly twice. In general, this solution is not a tour. In

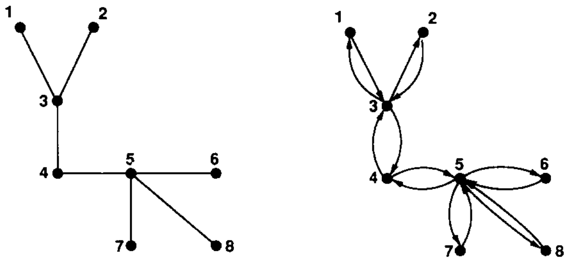

Figure 3. Optimal traversal of a graph using a shortest spanning tree

order to obtain a tour, the above procedure must be modified by using *shortcuts*. Follow the path obtained, but skip any already visited vertex. In our example, one would follow the path (1, 3, 2, 4, 5, 6, 8, 7, 1). Since the triangle inequality is satisfied, this never increases the total distance traveled and moreover, the resulting closed path is a tour of length not exceeding $2l(T) \leqslant 2z^*$.

Christofides (1976) has proposed an improvement to the above procedure. It is based on the following observation. The shortest spanning tree is not in general Eulerian. However, an Eulerian graph can be derived from it by linking its odd-degree vertices by means of a minimum-cost matching algorithm. This can be done in $O(n^3)$ time (Papadimitriou and Steiglitz, 1982). New edges corresponding to the optimal matching solution are then appended to the tree. Let $l(M)$ denote their total length. The resulting graph is Eulerian and its complete traversal requires a total distance of $l(T) + l(M)$. Again, using shortcuts, a tour of total length not exceeding $l(T) + l(M)$ can be obtained. This is illustrated in Figure 4. Figures 4a and 3a are identical. Figure 4b is obtained by solving a minimum-cost matching problem; the added edges are shown by the dashed lines. Traversing the resulting graph by

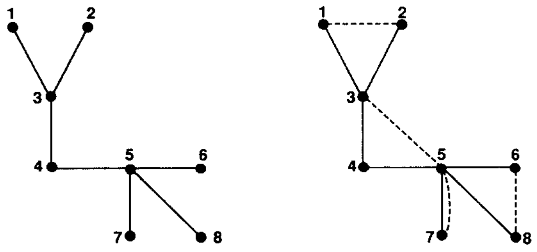

Figure 4. Optimal traversal of a graph using a shortest spanning tree and minimum matching

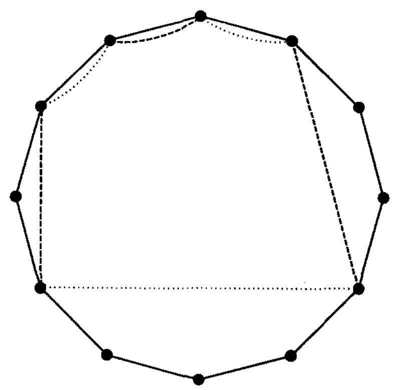
Figure 5. Christofides' heuristic

means of the above procedure can be done by following the path (1, 2, 3, 4, 5, 6, 8, 5, 7, 5, 3, 1). Using shortcuts yields the tour (1, 2, 3, 4, 5, 6, 8, 7, 1).

We now focus our attention on $l(M)$. Consider the sequence of vertices on an optimal TSP tour and link by shortcuts all vertex pairs corresponding to edges in the optimal matching solutions; delete all intermediate vertices. The total length of these edges is $l(M)$. Now close the tour by linking these edges in the same order as they appear on the tour (see Figure 5). The newly introduced edges have a total length of $l(M') \leqslant l(M)$ since the first matching is optimal. Since $l(M) + l(M') \leqslant z^*$, it follows that $l(M) \leqslant z^*/2$ and therefore, the heuristic yields a tour of length not exceeding $l(T) + l(M) \leqslant 3z^*/2$.

Finally, it should be mentioned that no heuristic with a guaranteed worst-case performance is known for the asymmetrical TSP.

5.2. *Heuristics with good empirical performance*

We now concentrate on a number of heuristics known to yield good TSP solutions in an empirical sense. Broadly speaking, TSP heuristics can be classical into *tour construction procedures* which involve gradually building a solution by adding a new vertex at each step, and *tour improvement procedures* which improve upon a feasible solution by performing various exchanges. The best methods are *composite algorithms* combining these two features. Most methods described in this section work on symmetrical and asymmetrical problems. There are, however, some exceptions that will be indicated. For further readings on this subject, see Rosenkrantz, Stearns and Lewis (1977), Golden and Stewart (1985) and Ong and Huang (1989).

5.2.1. *Tour construction procedures*

(a) *The nearest-neighbour algorithm* (Rosenkrantz, Stearns and Lewis, 1977).

In this method, a feasible tour is constructed by taking at each step the decision that is immediately the most advantageous. The main-interest of this 'myopic' algorithm lies in this simplicity.

Step 1. Consider an arbitrary vertex as a starting point.

Step 2. Determine the closest vertex to the last vertex considered and include it in the tour. If any vertex has not yet been considered, repeat Step 2.

Step 3. Link the last vertex of the tour to the first one.

The complexity of this procedure is $O(n^2)$. A possible modification is to consider in turn all n vertices as a starting point. The overall algorithm complexity is then $O(n^3)$, but the resulting tour is generally better.

(b) *Insertion algorithms* (Rosenkrantz, Stearns and Lewis, 1977; Stewart, 1977; Norback and Love, 1977, 1979; Or, 1976).

This category includes a number of similar algorithms whose basic structure can be summarized as follows:

Step 1. Construct a first tour consisting of two vertices.

Step 2. Consider in turn all vertices not yet in the tour. Insert in the tour a vertex chosen with respect to a given criterion, for example:
- the vertex yielding the least distance increment;
- the vertex closest to the current tour;
- the vertex furthest away from the tour;
- the vertex forming the largest angle with two consecutive vertices of the tour, etc.

Depending on the criterion that is used, the complexity of this type of procedure varies between $O(n^2)$ and $O(n \log n)$.

(c) *The patching algorithm for asymmetrical TSPs* (Karp, 1979).

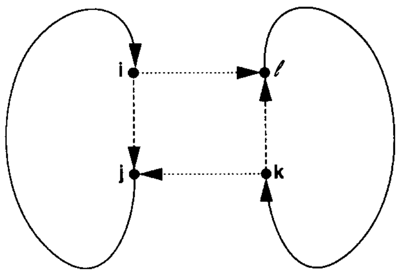

Figure 6. Merging two subtours in the patching algorithm

The following procedure was devised by Karp for asymmetrical TSPs. It exploits the fact that on problems for which the c_{ij}'s are uniformly distributed, the assignment relaxation of the TSP provides a near-optimal solution (see Balas and Toth, 1985).

Step 1. Solve the AP with cost matrix C.

Step 2. If the solution contains only one circuit, stop.

Step 3. Select the two circuits having the largest numbers of vertices. Select an arc (i, j) on the first circuit and an arc (k, l) on the second circuit that minimize the cost $c_{il} + c_{kj} - c_{ij} - c_{kl}$ of merging the two circuits (see Figure 6). Go to Step 2.

5.2.2. *Tour improvement procedures*

These methods are used to improve a tour obtained by any means. They will be classified into three main categories.

(a) *The r-opt algorithm* (Lin, 1965).

Step 1. Consider an initial tour.

Step 2. Remove r arcs from the tour and tentatively reconnect the r remaining chains in all possible ways. If any reconnection yields a shorter tour, consider this tour as a new initial solution and repeat Step 2. Stop when no improvement can be obtained.

This heuristic was originally devised for symmetrical TSPs. In this case, the number of candidate solutions at each step is of the order of n^r since there are $\binom{n}{r}$ ways to remove r arcs and $r!$ ways to reconnect the resulting undirected chains. Not all these reconnections are, however, feasible. In general, r is taken as 2 or 3. One interesting exception is the Christofides and Eilon (1972) implementation of this method with $r = 4$ and 5. Lin and Kerninghan (1973) have proposed an improvement to this method. Here, the value of r is modified dynamically throughout the algorithm. This procedure is considerably more difficult to code than the original Lin r-opt method, but it generally produces near-optimal solutions. Recently Johnson (1990) has developed a "randomized iterated Lin–Kerningham method" that produces near-optimal solutions. Or (1976) has proposed a simplified exchange procedure requiring only $O(n^2)$ operations at each step, but producing tours nearly as good on the average as those obtained with a 3-opt algorithm. Or's algorithm can be described as follows:

Step 1. Consider an initial tour and set $t := 1$ and $s := 3$.

Step 2. Remove from the tour a chain of s consecutive vertices, starting with the vertex in position t, and tentatively insert it between all remaining pairs of consecutive vertices on the tour.

- If the tentative insertion decreases the cost of the tour, implement it immediately, thus defining a new tour; set $t := 1$ and repeat Step 2.
- If no tentative insertion decreases the cost of the tour, set $t := t + 1$. If $t = n + 1$, then proceed to Step 3, otherwise repeat Step 2.

Step 3. Set $t := 1$ and $s := s - 1$. If $s > 0$, go to Step 2; otherwise stop.

Before closing this section, it is worth mentioning that Kanellakis and Papadimitriou (1980) have suggested an adaptation of the Lin and Kernighan procedure to the asymmetrical case. According to the analysis of Golden and Stewart (1985), further experiments are required to confirm that this procedure is indeed competitive.

(b) *Simulated annealing* (Kirkpatrick et al., 1983).

This successive improvements method is derived from an analogy with a material annealing process used in mechanics (Metropolis et al., 1953). In order to bring a material to a minimal-energy solid state, it is necessary to heat it until its particles are randomly distributed in the liquid state; then, to avoid local minima, its temperature is gradually reduced in steps, until the system reaches an equilibrium step for a given temperature level. At a high temperature T, all possible states can be reached but as the system cools down, the number of possibilities is reduced and the process converges to a stable state.

In combinatorial optimization, the aim is to move from a given initial solution to a minimum-cost solution, by performing gradual changes to the starting solution. Denote by T the state of the process (T corresponds to a temperature level in a physical system). In the beginning, the value of T is high and the number of allowed moves is also high. This number decreases with T, until no change to the solution is possible. A local minimum has then been reached.

For a given value of T, the algorithm is similar to the r-opt procedure: all solutions y in a neighbourhood of a solution x are examined. However, substituting x by y is sometimes allowed even if this results in a larger cost. This reduces the probability of becoming trapped in a local optimum. Simulated annealing can be applied to a large spectrum of combinatorial optimization problems. Formally, it can be summarized as follows:

Step 1. Consider an initial tour x of cost $F(x)$.

Step 2. Consider a solution y of cost $F(y)$ in the neighbourhood of x. If $F(y) < F(x)$, set $x := y$ and repeat Step 2. If $F(y) \geqslant F(x)$, define p_T exp $\{[F(x) - F(y)]/T\}$, where T is a parameter tending towards zero as the process evolves. Randomly select a number r in [0,1]. If $r \leqslant p_T$, set $x := y$ and repeat Step 2. Otherwise, repeat Step 2 with a new solution y in the neighbourhood of x, or stop if this neighbourhood has been completely examined.

Simulated annealing has been applied to the TSP by a number of authors including Bonomi and Lutton (1984), Rossier, Troyon and Liebling (1986), Golden and Skiscim (1986) and Nahar, Sahni and Shragowitz (1989), with apparently a mixed degree of success.

(c) *Tabu search* (Glover 1977, 1988, 1989, 1990; Glover and McMillan, 1986).

As in the previous two methods, successive neighbours of a solution x are examined and, as for simulated annealing, the objective is allowed to deteriorate in order to avoid local minima. In order to prevent cycling, solutions that have already been examined are forbidden and inserted in a constantly updated 'tabu list'. The method can be summarized in three steps:

Step 1. Consider an initial solution x of cost $F(x)$. Set the tabu list $T := \emptyset$.

Step 2. Let $N(x)$ be a neighbourhood of x. If $N(x) \setminus T = \emptyset$, go to Step 3. Otherwise, identify a least cost solution y in $N(x) \setminus T$ and set $x := y$. Update T and the best known solution.

Step 3. If the maximum number of allowed iteractions since the beginning of the process or since the last update has been reached, stop. Otherwise, go to Step 2.

The success of this method depends on the careful choice of a number of control parameters. For more details on this, see Soriano (1989). Several authors have applied tabu search to the TSP (see Knox, 1988, Malek, 1988, and Fiechter, 1990) with seemingly very positive results.

5.2.3. Composite algorithms

In recent years, two effective composite algorithms have been developed. The first is the Golden and Stewart (1985) CCAO heuristic. The second, GENIUS, is more recent. It was devised by Gendreau, Hertz and Laporte (1990).

(a) *The CCAO algorithm* (Golden and Stewart, 1985).

This heuristic was designed for symmetrical Euclidean TSPs. It exploits a well-known property of such problems, namely that in any optimal solution, vertices located on the convex hull of all vertices are visited in the order in which they appear on the convex hull boundary (Flood, 1956). The method can be summarized as follows:

Step 1 (C: convex hull). Define an initial (partial) tour by forming the convex hull of vertices.

Step 2 (C: cheapest insertion). For each vertex k not yet contained in the tour, identify the two adjacent vertices i_k and j_k on the tour such that $c_{i_k,k} + c_{k,j_k} - c_{i_k,j_k}$ is minimized.

Step 3 (A: largest angle). Select the vertex k^* that maximizes the angle between edges (i_k, k) and (k, j_k) on the tour, and insert it between i_{k^*} and j_{k^*}.

Step 4 Repeat Steps 2 and 3 until a Hamiltonian tour of all vertices is obtained.

Step 5 (O: Or-opt). Apply the Or-opt procedure to the tour and stop.

The rationale behind Steps 2 and 3 is that by selecting k^* so as to maximize the angle it makes with the tour, the solution remains as close as possible to the initial convex hull.

(b) *The GENIUS algorithm* (Gendreau, Hertz and Laporte, 1992).

One major drawback of the CCAO algorithm is that its insertion phase is myopic in the following sense: since insertions are executed sequentially without much concern for global optimality, they may result in a succession of bad decisions that the post-optimization phase will be unable to undo. GENIUS executes each insertion more carefully, by performing a limited number of local transformation of the tour, simultaneously with the insertion itself. It consists of two parts: a generalized insertion phase, followed by a post-optimization phase that successively removes vertices from the tour and reinserts them, using the generalized insertion rule.

The algorithm has been extensively tested on randomly generated problems and on problems taken from the literature; all these problems were symmetrical and Euclidean. Tests revealed that GENIUS produces in shorter computing times better solutions than CCAO, itself superior to all tour construction heuristics developed in this section. This algorithm also appears to compare favourably to tabu search and simulated annealing, although the number of comparisons was more limited in the case of these two methods.

6. Conclusion

The TSP occupies a central place in Operational Research. It underlies several practical applications and its study over the last 35 years or so has led to important theoretical developments. Problems involving a few hundred vertices can now be solved routinely to optimality. Instances involving more than 2000 vertices have also been solved exactly by means of constraint relaxation algorithms. A number of powerful heuristics have also been proposed: tabu search methods and generalized insertion algorithms appear to hold much potential.

Acknowledgements

This research was in part supported by the Canadian Natural Sciences and Engineering Research Council (grant OGP0039682), and by a Quebec-New – Brunswick interprovincial research grant. An extended version of this paper will appear in *Discrete Optimization Models*, H.A. Eiselt and C.L. Sandblom (eds.), De Gruyter, Berlin, 1993. Permission to publish this paper in EJOR is gratefully acknowledged. Finally, thanks are due to Paolo Toth for his valuable comments on a preliminary version of this paper.

References

Aho, A.V., Hopcroft, J.E., and Ullman, J.D. (1974), *The Design and Analysis of Computer Algorithms*, Addison-Wesley, Reading, MA.

Balas, E., Christofides, N. (1981), "A restricted lagrangean approach to the traveling salesman problem", *Mathematical Programming* 21, 19–46.

Balas, E., Toth, P. (1985), "Branch and bound methods", in: E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan and D.B. Shmoys (eds.), *The Traveling Salesman Problem. A Guided Tour of Combinatorial Optimization*, Wiley, Chichester, 361–401.

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

Bland, R.G., and Shallcross, D.F. (1989), "Large traveling salesman problems arising experiments in X-ray crystallography: A preliminary report on computation", *Operations Research Letters* 8, 125–128.

Bonomi, E., and Lutton, J.-L. (1984), "The *N*-city travelling salesman problem: Statistical mechanics an the Metropolis algorithm", *SIAM Review* 26, 551–568.

Carpaneto, G., Fischetti, M., and Toth, P. (1989), "New lower bounds for the symmetric travelling salesman problem", *Mathematical Programming* 45, 233–254.

Carpaneto, G., Martello, S., and Toth, P. (1988), "Algorithms and codes for the assignment problem", in: B. Simeone, P. Toth, G. Gallo, F. Maffioli, and S. Pallottino (eds.), *FORTRAN Codes for Network Optimization, Annals of Operations Research* 13, 193–223.

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

Christofides, N. (1970), "The shortest Hamiltonian chain of a graph", *SIAM Journal on Applied Mathematics* 19, 689–696.

Christofides, N. (1976), "Worst-case analysis of a new heuristic for the travelling salesman problem", Report 388, Graduate School of Industrial Administration, Carnegie-Mellon University, Pittsburgh, PA.

Christofides, N., and Eilon, S. (1972). Algorithms for large-scale travelling salesman problems, *Operational Research Quarterly* 23, 511–518.

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

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

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

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

Desrochers, M., and Laporte, G. (1991), "Improvements and extensions to the Miller–Tucker–Zemlin subtour elimination constraints", *Operations Research Letters* 10, 27–36.

Eastman, W.L. (1958), "Linear programming with pattern constraints", Ph.D. Thesis, Harvard University, Cambridge, MA.

Eiselt, H.A., and Laporte, G. (1991), "A combinatorial optimization problem arising in dartboard design", *Journal of the Operational Research Society* 42, 113–118.

Fiechter, C.-N. (1990), "A parallel tabu search algorithm for large scale traveling salesman problems", Working Paper 90/1, Départment de Mathématiques, École Polytechnique Féderale de Lausanne.

Fischetti, M., and Toth, P. (1991), "An additive bounding procedure for the asymmetric travelling salesman problem", *Mathematical Programming*, forthcoming.

Flood, M.M. (1956), "The traveling-salesman problem", *Operations Research* 4, 61–75.

Garey, M.R., and Johnson, D.S. (1979), "Computers and intractability: A guide to the theory of NP-completeness", Freeman, San Francisco.

Garfinkel, R.S. (1973), "On partitioning the feasible set in a branch-and-bound algorithm for the asymmetric traveling-salesman problem", *Operations Research* 21, 340–343.

Garfinkel, R.S. (1977), "Minimizing wallpaper waste, Part I: A class of traveling salesman problems", *Operations Research* 25, 741–751.

Gavish, B., and Srikanth, K.N. (1986), "An optimal solution method for large-scale multiple traveling salesman problems", *Operations Research* 34, 698–717.

Gendreau, M. Hertz, A., and Laporte, G. (1992), "New insertion and post-optimization procedures for the traveling salesman problem", *Operations Research*, forthcoming.

Gilmore, P.C., and Gomory, R.E. (1964), "Sequencing a one state-variable machine: A solvable case of the traveling salesman problem", *Operations Research* 12, 655–679.

Gilmore, P.C., Lawler, E.L., and Shmoys, D.B. (1985), "Well-solved special cases", in: E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, and D.B. Shmoys (eds.), *The Traveling Salesman Problem. A Guided Tour of Combinatorial Optimization*, Wiley, Chichester, 87–143.

Glover, F. (1977), "Heuristic for integer programming using surrogate constraints",*Decision Sciences 8*, 156–166.

Glover, F. (1988), "Tabu search", Report 88-3, Center for Applied Artificial Intelligence (CAAI), Graduate School of Business, University of Colorado.

Glover, F. (1989), "Tabu search, Part I", *ORSA Journal on Computing* 1, 190–209.

Glover, F. (1990), "Tabu search", Part II, *ORSA Journal on Computing* 2, 4–32.

Glover, F., and McMillan, C. (1986), "The general employee scheduling problem: An integration of MS and AI", *Computers & Operations Research* 13, 563–573.

Golden, B.L., and Skiscim, C.C. (1986), "Using simulated annealing to solve routing and location problems", *Naval Research Logistics Quarterly* 33, 261–280.

Golden, B.L., and Stewart, Jr., W.R. (1985), "Empirical analysis of heuristics", in: E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, and D.B. Shmoys (eds.), *The Traveling Salesman Problem. A Guided Tour of Combinatorial Optimization*, Wiley, Chichester, 207–249.

Gomory, R.E. (1963), "An algorithm for integer solutions to linear program", in: R.L. Graves, and P. Wolfe (eds). *Recent Advances in Mathematical Programming*, McGraw-Hill, New York, 269–302.

Gomory, R.E., and Hu, T.C. (1961), "Multi-terminal network flows", *SIAM Journal on Applied Mathematics* 9, 551–556.

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

Grötschel, M., and Padberg, M.W. (1985), "Polyhedral theory", in: E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, and D.B. Shmoys (eds.), *The Traveling Salesman Problem. A Guided Tour of Combinatorial Optimization*, Wiley, Chichester, 251–305.

Helbig Hansen, K., and Krarup, J. (1974), "Improvements of the Held-Karp algorithm for the symmetric traveling-salesman problem", *Mathematical Programming* 7, 87–96.

Held, M., and Karp., R.M. (1970), "The traveling salesman problem and minimum spanning trees", *Operations Research* 18, 1138–1162.

Held, M., and Karp, R.M. (1971), "The traveling salesman problem and minimum spaning trees: Part II", *Mathematical Programming* 1, 6–25.

Johnson, D.S. (1990), "Local optimization and the traveling salesman problem", in: M.S. Paterson (ed.), *Proceedings of the 17th International Colloquium on Automata, Languages and Programming*, Lecture Notes in Computer Science, Springer-Verlag, Berlin, 446–461.

Kanellakis, P.-C., and Papadimitriou, C.H. (1980), "Local search for the asymmetric traveling salesman problem", *Operations Research* 28, 1086–1099.

Karp, R.M. (1979), "A patching algorithm for the nonsymmetric traveling-salesman problem", *SIAM Journal on Computing* 8, 561–573.

Kirkpatrick, S., Gelatt, Jr., C.D., and Vecchi, M.P. (1983), "Optimization by simulated annealing", *Science* 220, 671–680.

Knox, J. (1988), "An application of TABU search to the symmetric traveling salesman problem", Ph.D. Thesis, Center for Applied Artificial Intelligence (CAAI), Graduate School of Business, University of Colorado.

Langevin, A., Soumis, F., and Desrosiers, J. (1990), "Classification of travelling salesman problem formulations", *Operations Research Letters* 9, 127–132.

Laporte, G. (1992), "The vehicle routing problem: An overview of exact and approximate algorithms", *European Journal of Operational Research*, 59, 345–358 (to appear).

Lawler, E.L., Lenstra, J.K., Rinnooy Kan, A.H.G., and Shmoys, D.B. (1985), *The Traveling Salesman Problem. A Guided Tour of Combinatorial Optimization*, Wiley, Chichester.

Lenstra, J.K., Rinnooy Kan, A.H.G. (1975), "Some simple applications of the travelling salesman problem", *Operational Research Quarterly* 26, 717–733.

Lin, S. (1965), "Computer solutions of the traveling salesman problem", *Bell System Computer Journal* 44, 2245–2269.

Lin, S., and Kernighan, B.W. (1973), "An effective heuristic algorithm for the traveling-salesman problem", *Operations Research* 21, 498–516.

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

Malek, M. (1988), "Search methods for traveling salesman problems", Working Paper, University of Texas, Austin, TX.

Martin, G.T. (1966), "Solving the traveling salesman problem by integer programming", Working Paper, CEIR, New York.

Metropolis, N., Rosenbluth, A.W., Rosenbluth, M.N., Teller, A.H., and Teller, E. (1953), "Equation of state calculations by fast computing machines", *Journal of Chemical Physics* 21, 1087–1091.

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

Miliotis, P. (1978), "Using cutting planes to solve the symmetric salesman problem", *Mathematical Programming* 15, 177–188.

Miller, D.L., and Pekny, J.F. (1989), "Results from a parallel branch and bound algorithm for solving large asymmetric traveling salesman problems", *Operations Research Letters* 8, 129–135.

Miller, D.L., and Pekny, J.F. (1991), "Exact solution of large asymmetric traveling salesman problems", *Science* 251, 754–761.

Miller, C.E., Tucker, A.W., and Zemlin, R.A. (1960), "Integer programming formulations and traveling salesman problems", *Journal of the Association for Computing Machinery* 7, 326–329.

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

Nahar, S., Sahni, S., and Shragowitz, E. (1989), "Simulated annealing and combinatorial optimization", *International Journal of Computer Aided VLSI Design* 1, 1–23.

Norback, J., and Love, R. (1977), "Geometric approaches to solving the traveling salesman problem", *Management Science* 23, 1208–1223.

Norback, J., and Love, R. (1979), "Heuristic for the Hamiltonian path problem in Euclidean two space", *Journal of the Operational Research Society* 30, 363–368.

Ong, H.L., and Huang, H.C. (1989), "Asymptotic expected performance of some TSP heuristics", *European Journal of Operational Research* 43, 231–238.

Or, I. (1976), "Traveling salesman-type combinatorial problems and their relation to the logistics of regional blood banking", Ph.D. Dissertation, Northwestern University, Evanston, IL.

Padberg, M.W., and Grötschel, M. (1985), "Polyhedral computations", in: E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, and D.B. Shmoys (eds.), *The Traveling Salesman Problem. A Guided Tour of Combinatorial Optimization*, Wiley, Chichester, 307–360.

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

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

Padberg, M.W., and Rinaldi, G. (1990), "Facet identification for the symmetric traveling salesman problem", *Mathematical Programming* 47, 219–257.

Papadimitriou, C.H., and Steiglitz, K. (1982), *Combinatorial Optimization: Algorithms and Complexity*, Prentice-Hall, Englewood Cliffs, NJ.

Reinelt, G. (1989), "Fast heuristics for large geometric traveling salesman problems", Report No. 185, Institut für Mathematik, Universität Augsburg.

Rosenkrantz, D.J., Stearns, R.E., and Lewis, II, P.M. (1977), "An analysis of several heuristics for the traveling salesman problem", *SIAM Journal on Computing* 6, 563–581.

Rossier, Y., Troyon, M., and Liebling, T.M. (1986), "Probabilistic exchange algorithms and the Euclidian traveling salesman problem", *Operations Research Spektrum* 8, 151–164.

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

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

Smith, T.H.C., and Thompson, G.L. (1977), "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.

Soriano, P. (1989), "Étude de nouvelles avenues de recherche proposées en optimisation combinatoire", Publication CRT-619, Centre de recherche sur les transports, Montréal.

Stewart, Jr., W.R. (1977), "A computationally efficient heuristic for the traveling salesman problem", *Proceedings of the 13th Annual Meetings of S.E. Tims*, 75–85.

Tarjan, R.E. (1977), "Finding optimum branchings", *Networks* 7, 25–35.

Volgenant, T., and Jonker, R. (1982), "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.

Wong, R.T. (1980), "Integer programming formulations of the travelling salesman problem", *Proceedings of the IEEE International Conference on Circuits and Computers*, 149–152.