Chapter 10

The Traveling Salesman Problem

10.1 Introduction

The traveling salesman problem consists of a salesman and a set of cities. The salesman has to visit each one of the cities starting from a certain one (e.g. the hometown) and returning to the same city. The challenge of the problem is that the traveling salesman wants to minimize the total length of the trip.

The traveling salesman problem can be described as follows:
TSP = $\{(G, f, t)$: $G = (V, E)$ a complete graph,
f is a function $V \times V \rightarrow$ Z,
$t \in$ Z,
G is a graph that contains a traveling salesman tour with cost that does not exceed $t\}$.

Example:

Consider the following set of cities:

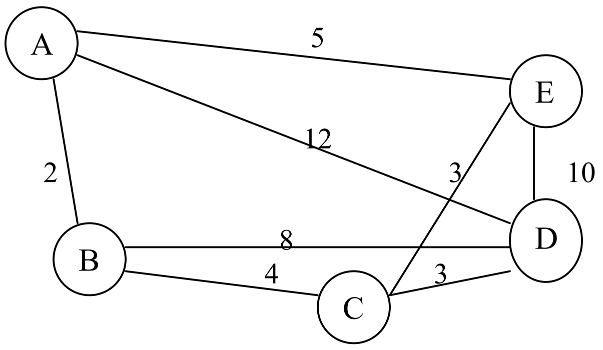

Figure 10.1 A graph with weights on its edges.

The problem lies in finding a minimal path passing from all vertices once. For example the path Path1 {A, B, C, D, E, A} and the path Path2 {A, B, C, E, D, A} pass all the vertices but Path1 has a total length of 24 and Path2 has a total length of 31.

Definition:

A Hamiltonian cycle is a cycle in a graph passing through all the vertices once.

Example:

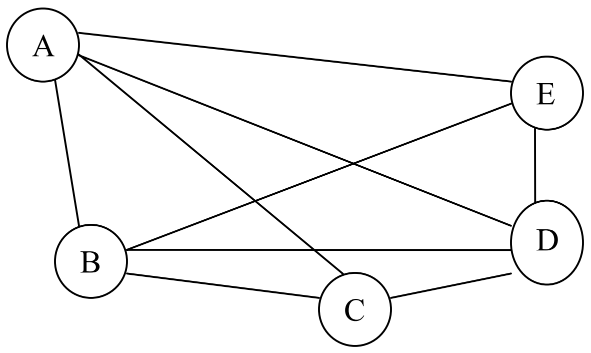

Figure 10.2 A graph with various Hamiltonian paths.

P = {A, B, C, D, E} is a Hamiltonian cycle.
The problem of finding a Hamiltonian cycle in a graph is NP-complete.

Theorem 10.1:
The traveling salesman problem is NP-complete.

Proof:
First, we have to prove that TSP belongs to NP. If we want to check a tour for credibility, we check that the tour contains each vertex once. Then we sum the total cost of the edges and finally we check if the cost is minimum. This can be completed in polynomial time thus TSP belongs to NP.
Secondly we prove that TSP is NP-hard. One way to prove this is to show that Hamiltonian cycle $\leq_P$ TSP (given that the Hamiltonian cycle problem is NP-complete). Assume $G = (V, E)$ to be an instance of Hamiltonian cycle. An instance of TSP is then constructed. We create the complete graph $G' = (V, E')$, where $E' = \{(i, j): i, j \in V$ and $i \neq j$. Thus, the cost function is defined as:

$$t(i ,,j) = \begin{cases} 0 \text{ if } (i, j) \in E, \\ 1 \text{ if } (i, j) \notin E. \end{cases} \qquad 10.1$$

Now suppose that a Hamiltonian cycle h exists in G. It is clear that the cost of each edge in h is 0 in G' as each edge belongs to E. Therefore, h has a cost of 0 in G'. Thus, if graph G has a Hamiltonian cycle then graph G' has a tour of 0 cost.
Conversely, we assume that G' has a tour h' of cost at most 0. The cost of edges in E' are 0 and 1 by definition. So each edge must have a cost of 0 as the cost of h' is 0. We conclude that h' contains only edges in E.
So we have proven that G has a Hamiltonian cycle if and only if G' has a tour of cost at most 0. Thus TSP is NP-complete.

10.2 Methods to solve the traveling salesman problem

10.2.1 Using the triangle inequality to solve the traveling salesman problem

Definition:
If for the set of vertices $a, b, c \in V$, it is true that t $(a, c) \leq t(a, b) + t(b, c)$ where t is the cost function, we say that t satisfies the triangle inequality.

First, we create a minimum spanning tree the weight of which is a lower bound on the cost of an optimal traveling salesman tour. Using this minimum spanning tree we will create a tour the cost of which is at most 2 times the weight of the spanning tree. We present the algorithm that performs these computations using the MST-Prim algorithm.

Approximation-TSP
Input: A complete graph $G (V, E)$
Output: A Hamiltonian cycle

1.select a "root" vertex $r \in V [G]$.
2.use MST-Prim (G, c, r) to compute a minimum spanning tree from r.
3.assume L to be the sequence of vertices visited in a preorder tree walk of T.
4.return the Hamiltonian cycle H that visits the vertices in the order L.
The next set of figures show the working of the proposed algorithm.

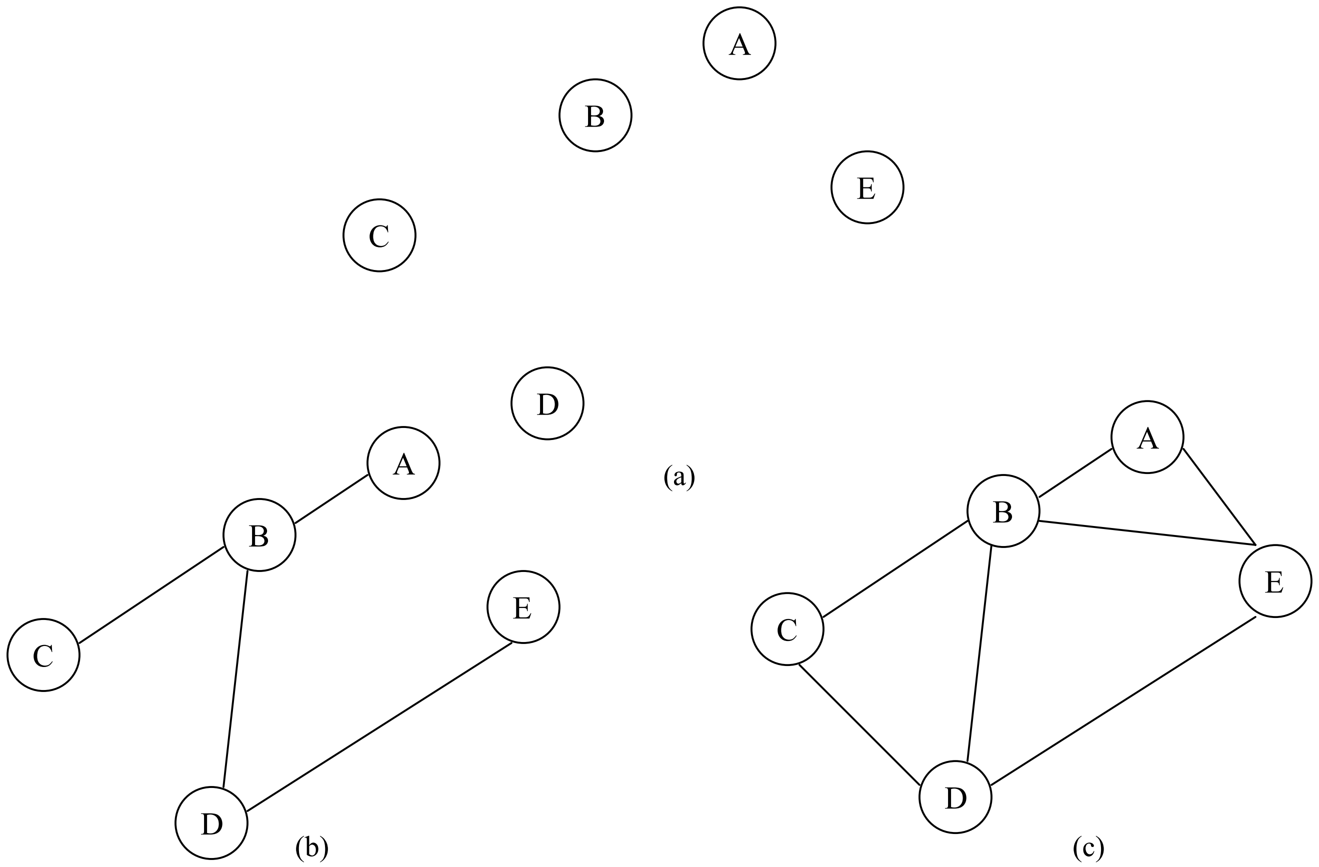

Figure 10.3 A set of cities and the resulting connection after the MST-Prim algorithm has been applied..

In Figure 10.3(a) a set of vertices is shown. Part (b) illustrates the result of the MST-Prim thus the minimum spanning tree MST-Prim constructs. The vertices are visited like {A, B, C, D, E, A) by a preorder walk. Part (c) shows the tour, which is returned by the complete algorithm.

Theorem 10.2:
Approximation-TSP is a 2-approximation algorithm with polynomial cost for the traveling salesman problem given the triangle inequality.

Proof:
Approximation-TSP costs polynomial time as was shown before.

Assume H^* to be an optimal tour for a set of vertices. A spanning tree is constructed by deleting edges from a tour. Thus, an optimal tour has more weight than the minimum-spanning tree, which means that the weight of the minimum spanning tree forms a lower bound on the weight of an optimal tour.

$$c(t) \leq c(H^*). \qquad 10.2$$

Let a full walk of T be the complete list of vertices when they are visited regardless if they are visited for the first time or not. The full walk is *W*. In our example:
W = A, B, C, B, D, B, E, B, A,.
The full walk crosses each edge exactly twice. Thus, we can write:

$$c(W) = 2c(T). \qquad 10.3$$

From equations 10.2 and 10.3 we can write that

$$c(W) \leq 2c(H^*), \qquad 10.4$$

Which means that the cost of the full path is at most 2 time worse than the cost of an optimal tour. The full path visits some of the vertices twice which means it is not a tour. We can now use the triangle inequality to erase some visits without increasing the cost. The fact we are going to use is that if a vertex a is deleted from the full path if it lies between two visits to b and c the result suggests going from b to c directly.
In our example we are left with the tour: A, B, C, D, E, A. This tour is the same as the one we get by a preorder walk. Considering this preorder walk let H be a cycle deriving from this walk. Each vertex is visited once so it is a Hamiltonian cycle. We have derived H deleting edges from the full walk so we can write:

$$c(H) \leq c(W) \qquad 10.5$$

From 10.4 and 10.5 we can imply:

$$c(H) \leq 2\ c(H^*). \qquad 10.6$$

This last inequality completes the proof.

10.2.2 The general traveling salesman problem

Definition:
If an NP-complete problem can be solved in polynomial time then P = NP, else P ≠ NP.

Definition:
An algorithm for a given problem has an approximation ratio of $\rho(n)$ if the cost of the S solution the algorithm provides is within a factor of $\rho(n)$ of the optimal S^* cost (the cost of the optimal solution). We write:

$$\max(S/S^*, S^*/S) \le \rho(n). \qquad 10.7$$

If the cost function t does not satisfy the triangle inequality then polynomial time is not enough to find acceptable approximation tours unless P = NP.

Theorem 10.3:
If P≠NP then there is no approximation algorithm with polynomial cost and with approximation ratio of ρ for any ρ≥1 for the traveling salesman problem.

Proof:
Let us suppose that there is an approximation algorithm A with polynomial cost for some number $\rho \ge 1$ with approximation ratio ρ. Let us assume that ρ is an integer without loss of generality. We are going to try to use A to solve Hamiltonian cycle problems. If we can solve such NP-complete problems then P = NP.
Let us assume a Hamiltonian-cycle problem $G = (V, E)$. We are going to use algorithm A to determine whether A contains Hamiltonian cycles. Assume $G' = (V, E')$ to be the complete graph on V. Thus:

$$E' = \{(a, b): a, b \in V \text{ and } a \ne b\} \qquad 10.8$$

Each edge in E' is assigned an integer:

$$\mathrm{t}(a, b) = \begin{cases} 1 \text{ if } (\mathrm{a}, \mathrm{b}) \in E, \\ p\,|V| + 1 \mathit{else} \end{cases} \qquad 10.9$$

Consider the traveling salesman problem (G', t). Assume there is a Hamiltonian cycle H in the graph G. Each edge of H is assigned a cost of 1 by the cost function t. Hence (G', t) has a tour of cost $|V|$. If we had assumed that there is not a Hamiltonian cycle in the graph G, then a tour in G' must contain edges that do not exist in E. Any tour with edges not in E has a cost of at least

$$\begin{aligned}(\rho|V| + 1) + (|V| - 1) &= \rho|V| + |V| \\ &> \rho|V|\end{aligned} \qquad 10.10$$

Edges that do not exist in G are assigned a large cost the cost of any tour other than a Hamiltonian one is incremented at least by $|V|$.
Let us use the approximation algorithm described in 10.2.1 to solve the traveling salesman problem (G', t).
A returns a tour of cost no more than ρ times the cost of an optimal tour. Thus, if G has a Hamiltonian cycle then A must return it. If G does not have a Hamiltonian cycle, A returns a tour

whose cost is more than $\rho|V|$. It is implied that we can use A to solve the Hamiltonian-cycle problem with a polynomial cost. Therefore, the theorem is proved by contradiction.

10.2.3 A heuristic solution proposed by Karp

According to Karp, we can partition the problem to get an approximate solution using the divide and conquer techniques. We form groups of the cities and find optimal tours within these groups. Then we combine the groups to find the optimal tour of the original problem. Karp has given probabilistic analyses of the performance of the algorithms to determine the average error and thus the average performance of the solution compared to the optimal solution. Karp has proposed two dividing schemes. According to the first, the cities are divided in terms of their location and only. According to the second, the cities are divided into cells that have the same size. Karp has provided upper bounds of the worst-case error for the first dividing scheme, which is also called Adaptive Dissection Method. The working of this method is explained below.

Let us assume that the n cities are distributed in a rectangle. This rectangle is divided in $B = 2^k$ sub rectangles. Each sub rectangle contains at most t cities where $k = \log_2[(N-1)/(t-1)]$. The algorithm computes an optimum tour for the cities within a sub-rectangle. These 2^k optimal tours are combined to find an optimal tour for the N cities. Let us explain the working of the division algorithm. Assume Y to be a rectangle with *num* being the number of cities. The rectangle is divided into two rectangles in correspondence of the $[num/2]_{th}$ city from the shorter side of the rectangle. This city is true that belongs to the common boundary of the two rectangles. The rest of the division process is done recursively.
The results for this algorithm are presented below.
Assume a rectangle X containing N cities and t the maximum allowed number of cities in a sub-rectangle. Assume W to be the length of the walk the algorithm provides and L_{opt} to be the length of the optimal path. Then the worst-case error id defined as follows:

$$W\text{-}L_{opt} \leq 3/2 \sum^{2k}_{i=1} Per(Y_i) \qquad 10.11$$

Where $Per(Y_i)$ is the perimeter of the i_{th} rectangle.
If a,b are the dimensions of the rectangle we can imply an upper bound for $\sum^{2k}_{i=1} Per(Y_i)$:

$$\sum^{2k}_{i=1} Per(Y_i) \leq 3/2\ 2*2^{a+b/2}\,(2^{\lceil k/2\rceil+\lfloor k/2\rfloor}) \qquad 10.12$$

Now we can write:

$$W\text{-}L_{opt} \leq 3/2\ 2*2^{a+b/2}\,(2^{\lceil k/2\rceil+\lfloor k/2\rfloor}) \qquad 10.13$$

Where $2^{\alpha} = a$ and $2^{\beta} = b$.

If a = b then we can imply that:

$$W\text{-}L_{opt} \leq 3/2[2a(2^{\lceil k/2\rceil+\lfloor k/2\rfloor})] \qquad 10.14$$

There are two possibilities for k:

$$\text{If k even } W\text{-}L_{opt} \leq 3a2^{k/2+1} \qquad 10.15$$

$$\text{If k odd } W\text{-}L_{opt} \leq 3a3/\sqrt{2} * 2^{k/2} \qquad 10.16$$

Assuming that $\log_2(N-1)/(t-1)$ is an integer we can express this equation in terms of N and t:

If k even $W\text{-}L_{opt} \leq 3a2\sqrt{(N-1)/(t-1)}$ 10.17

If k odd $W\text{-}L_{opt} \leq 3a\, 3\Big/\sqrt{2}\ \sqrt{(N-1)/(t-1)}$ 10.18

Observation: The points distribution does not affect the result.
It should be noted however that these results only hold for uniform distributions. We now assume random distributions to generalize the results.
Let us assume a rectangle X of area $v(X)$, within which there are randomly distributed N cities, following a uniform distribution. Let us denote the length of an optimal tour through the N cities to a random variable $T_N(X)$.Thus there exists a positive constant β such as that $\forall\varepsilon>0$

$$\text{Prob}\,\{\lim_{N\to\infty}(T_N(X)/\sqrt{Nv(X)}\ -\beta>\varepsilon\} = 0 \qquad 10.19$$

This result from equation 10.12 shows that the relative error between a spanning walk and an optimal tour can be estimated as:

$$\forall\varepsilon>0\ \text{Prob}\,\{\lim_{N\to\infty}(W-\ T_N(X)/\ T_N(X) - S/\sqrt{t}\ )>\varepsilon\} = 0 \qquad 10.20$$

Where S>0.

Let us assume a rectangle $X[a, b]$ with $ab = 1$.
Let $T_t(X)$ be an optimal tour through t<N cities in the rectangle. We are going to compare the average length of this tour to the average length of an optimal tour through all the N cities.

$\beta_x(t)$ is defined as: $\beta_x(t) = E(T_t(X))/\sqrt{t}$.

From equation 10.20 we get that $\lim_{t\to\infty}\beta_x(t) = \beta$.
Thus, there exists the following bound:

$\beta_x(t) - \beta \leq 6(a+b)/\ \sqrt{t}$.
We can say that the length of a tour through $t<N$ cities tends to be almost the same as the length of the optimal tour.

10.2.4 Trying to solve the traveling salesman problem using greedy algorithms

Assume the asymmetric traveling salesman problem. We use the symbol of (K_n,c) whre c is the weight function and n is the number of vertices. We assume the symmetric traveling salesman problem to be defined in the same way but K_n symbols a complete undirected graph. If we try to find an approximate solution to an NP-hard problem using heuristics, we need to compare the solutions using computational experiments. There is a number called domination number that compares the performance of heuristics. A heuristic with higher domination number is a better choice than a heuristic with a lower domination number.

Definition:
The domination number for the TSP of a heuristic A is an integer such as that for each instance I of the TSP on *n* vertices A produces a tour T that is now worse than at least *d(n)* tours in I including T.
If we evaluate the greedy algorithm and the nearest neighbor algorithm for the TSP, we find that they give good results for the Euclidean TSP but they both give poor results for the asymmetric and the symmetric TSP. We analyze below the performance of the greedy algorithm and the nearest neighbor algorithm using the domination number.

Theorem 10.4:
The domination number of the greedy algorithm for the TSP is 1.

Proof:
We assume an instance of the ATSP for which the greedy algorithm provides the worst tour. Let *nmin{i, j}* be the cost of each arc(i, j). We assume the following exceptions: $c(i, i+1) = in$, for i = 1,2,..,n-1, $c(i, 1) = n^2 - 1$ for i = 3,4,…,n-1 and $c(n,1) = n^3$.
We observe that the cheapest arc is (1,2). Thus the greedy algorithm returns the tour (1,2,…,n,1). We can compute the cost of T as:

$$\sum_{i=1}^{n-1} in + c(n,1). \qquad 10.21$$

Assume a tour *H* in the graph such as that $c(H) \geq c(T)$. The arc (*n*,1) must be contained within the tour *h* as

$$c(n,1) > n \max\{c(i,j) : 1 \leq i \neq j \leq n, (i,j) \neq (n,1)\}. \qquad 10.22$$

It is implied that there is a Hamiltonian path P from 1 to n with a cost of $\sum_{i=1}^{n-1} in$
Let e_i be an arc of P with a tail *i*. It is true that $c(e_i) \leq in+1$. P mut have an arc (e_k) that goes to an edge with an identification number smaller than the number of the edge it starts from. We can now write:

$$c(e_k) \leq (k-1)n + 1 \text{ and} \qquad 10.23$$
$$c(P) \leq \sum_{i=1}^{n-1} in + (n-1) - n. \qquad 10.24$$

The theorem is proven by contradiction.

Theorem 10.5:
Let $n \geq 4$. The domination number of the nearest neighbor algorithm for the asymmetric traveling salesman problem is at most n-1 and at least n/2.

Proof:
Assume all arcs such as that (i,i+1) $1 \leq i < n$, have a cost of *iN*, all arcs such as that (i,i+2) $1 \leq i \leq n-2$ have a cost of *iN*+1, all the other arcs that start from an edge with an identification number smaller than that of the edge hey end(forward arcs) have a cost of iN+2 and all the other arcs that start from an edge with a greater identification number than that of the edge they end (backward arcs) have a cost of *(j-1)N.*
If nearest neighbor starts at i, which is neither 1 nor *n*, it has a cost of

$$l = \sum_{k=1}^{n-1} kN - N + 1. \qquad 10.25$$

If the algorithm starts at 1 we have a cost of $\sum_{k=1}^{n-1} kN > l$

Any tour has a cost of at least l. Let us define the length of a backward arc as i-j. Let *F* be the set of tours described above and *T1* a tour not in *F*. T1 is a tour, so the cost of *T1* is at most

$$\sum_{k=1}^{n-1} iN + 2 - qN - |B|N, \qquad 10.26$$

where *B* is the set of backward arcs and q is the sum of length of the arcs in *B*.
We conclude that the cost of *T1* is less than l, which would mean that *T1* belongs to *F*. So all cycles that do not belong to *F* have a cost less than those who belong to *F*.
We assume that nearest neighbor does not have a domination number of at least *n*/2. Nearest neighbor constructs n tours. By assumption the number of cities is at least 4, so we have at least 3 tours that may coincide. Let $F = x_1x_2x_nx_1$ be a tour such as that $F = F_i = F_j = F_k$. We could assume that i=1 and $2<j\leq 1+(n/2)$. Foe every m with $j<m\leq n$ let C_m be the tour provided by deleting consecutive arcs and adding backward arcs. We should note that
$c(C_m)\geq c(C_f)$ since $c(x_i,x_{i+1})\leq c(x_j,x_{j+1})$.
This is true as we used nearest neighbor from x_j to construct F_j. and
$c(x_m,x_{m+1})\leq c(x_m,x_{i+1})$ since nearest neighbor chose the (x_m,x_{m+1}) on F_j when the arc (x_m,x_{i+1}) was available.
We can state now that the domination number is at least
$n-j+1\leq n/2$.
The theorem is proven by contradiction.

Definition:
A tour $x_1\ x_2\ x_3\ x_n\ x_1$, $x_1 = 1$ in a symmetric traveling salesman problem is called pyramidal if $x_1<x_2<x_n<x_1\leq x_{k+1} >\ldots>x_n$.
The number of pyramidal tours in a symmetric traveling salesman problem is:
2^{n-3}.

Theorem 10.6:
Let n≥4. The domination number of nearest neighbor for the symmetrical traveling salesman problem is at most 2^{n-3}.

Proof:
We consider an instance of symmetric traveling salesman problem, which proves that nearest neighbor has a domination number of 2^{n-3}.
Let all edges $\{i, i+1\}$, $1\geq i < n$ have cost of *iN*.
Let all edges $\{i,i+2\}$ have a cost of $iN+1$
Let all the remaining edges $\{i,j\}$, $i<j$, cost $iN+2$.
Let us assume that C_{NN} is the cost of the cheapest tour provided by the nearest neighbor algorithm. It is then clear that

$$C_{NN} = c(12\ldots n1) = \sum_{i=1}^{n-1} iN + N + 2. \qquad 10.27$$

Assume a tour on the graph. Let it be $x_1\ x_2\ x_n\ x_1$.
We assume a directed cycle T' which is constructed by orienting all the edges on the tour. For a backward arc in the cycle *e(j,i),* we define its length as *q(e) = j-i.*
We express the sum of the lengths of the backward arcs in the cycle as *q(T')*.
Assume the most expensive non-pyramidal tour *T*. Let C_{max-} be the cost of this tour.
We have to show that

$$C_{max} < C_{nn}, \qquad 10.28$$

as there are 2^{n-3} pyramidal tours.
It is true that *q(T')* $\geq$ n for every *T'*.
Assume a non pyramidal tour H of cost C_{max} and e_i = (*i,j*) be an arc of *H'*.
If e_1 is forward then $c(e_1) \leq iN + 2$.
If e_1 is backward then $c(e_1) \leq jN + 2 - q(e_i)N$.
Thus we can write:

$$C_{max} \leq \sum_{i=1}^{n}(iN+2) - q(H')N \leq \sum_{i=1}^{n-1} iN + 2n \qquad 10.29$$

As *q(H')* $\geq$ *n*. From the preceding equations, we conclude that indeed

$$C_{max} < C_{nn}$$

Thus, the theorem is proven.

10.2.5 The branch and bound algorithm and its complexity

The branch and bound algorithm converts the asymmetric traveling salesman problem into an assignment problem. Consider a graph V that contains all the cities. Consider Π being the set of all the permutations of the cities, thus covering all possible solutions. Consider a permutation of this set $\pi \in \Pi$ in which each city is assigned a successor, say *i*,for the π_i city. So a tour might be (1, $\pi(1)$, $\pi(\pi(1)),\ldots,$1). If the number of the cities in the tour is n then the permutation is called a cyclic permutation. The assignment problem tries to find such cyclic permutations and the asymmetric traveling salesman problem seeks such permutations but with the constraint that they have a minimal cost. The branch and bound algorithm firstly seeks a solution of the assignment problem. The cost to find a solution to the assignment problem for n cities is quite large and is asymptotically $O(n^3)$.
If this is a complete tour, then the algorithm has found the solution to the asymmetric traveling salesman problem too. If not, then the problem is divided in several sub-problems. Each of these sub-problems excludes some arcs of a sub-tour, thus excluding the sub-tour itself. The way the algorithm chooses which arc to delete is called branching rules. It is very important that there are no duplicate sub-problems generated so that the total number of the sub-problems is minimized. Carpaneto and Toth have proposed a rule that guarantees that the sub-problems are independent. They consider the included arc set and select a minimum number of arcs that do not belong to that set. They divide the problem as follows. Symbolize as E the excluded arc set and as I the included arc set. The I is to be decomposed. Let t arcs of the selected sub-tour $x_1 x_2 \ldots x_n$ not to belong to I. The problem is divided into t children so that the j_{th} child has E_j excluded arc set and I_j included arc set. We can now write:

$$\left.\begin{array}{l} \mathrm{Ej} = \mathrm{E} \cup \{x_j\} \\ \mathrm{Ii} - \mathrm{I} \cup \{x_1, x_2, \ldots, x_{j-1}\} \end{array}\right\} k = 1,2,..,j \qquad 10.30$$

But x_j is an excluded arc of the j_{th} sub-problem and an included arc in the $(j+1)_{st}$ problem. This means that a tour produced by the $(j+1)_{st}$ problem may have the x_j arc but a tour produced by the j_{th} problem may not contain the arc. This means that the two problems cannot generate the same tours, as they cannot contain the same arcs. This guarantees that there are no duplicate tours.

Complexity of the branch and bound algorithm.

There has been a lot of controversy concerning the complexity of the branch and bound algorithm. Bellmore and Malone have stated that the algorithm runs in polynomial time. They have treated the problem as a statistical experiment assuming that the i_{th} try of the algorithm is successful if it finds a minimal tour for the I_{th} sub-problem. They assumed that the probability of the assignment problem to find the solution to the asymmetric traveling salesman problem is e/n where n is the number of the cities. Under other assumptions, they concluded that the total number of sub-problems is expected to be:

$$\sum_{i=1}^{\infty} ipi \prod_{j=0}^{i-1} 1 - pj \leq \sum_{i=1}^{\infty} ipo\,(1-\mathrm{p_o})^{i-1} = 1\backslash \mathrm{p_o} = O(n). \qquad 10.31$$

Smith concluded that under some more assumptions the complexity of the algorithm is $O(n^3 \ln(n))$

The assumptions made to reach this result are too optimistic. Below it will be proven that they do not hold and that the complexity of the branch and bound algorithm is not polynomial.

Definition:
The assignment problem of a cost matrix with $c_{i,j} = \infty$ is called a modified assignment problem.
Lemma 10.1:
Assume a n x n random cost matrix. Consider the assignment problem that has s<n excluded arcs and t included arcs. Let q(n,s,t) be the probability that the solution of the assignment problem is a tour. Then q(n,s,t) is asymptotically

$$e/n - o(1/n) < q(n,s,t) + o(1/n) \text{ if } t=0, \qquad 10.32$$

$$q(n,s,t) - \lambda/(n-t) + o(1/n) \text{ if } t>0, \qquad 10.33$$
where λ, $<\lambda<e$ is a constant.

Proof:
See [7]
Lemma 10.2:
Consider branch and bound select two nodes that are independent. Assume that the probability that a non-root node in the search tree is a leaf is p. Let p_o be the possibility that the node is the root. There exists a constant $0<\delta<l-1/e$ for a non-root node so that if $t<\delta n$ then $p<p_o$, where n is the number of cities.

Proof:
Assume the search tree and a node of it say *Y*, which is constructed from 10.33.Y, has some included and some excluded arcs. Suppose the number of included arcs is t and the number of excluded arcs is s, Observe that s is the depth of the node in the search tree. Let the path from the root to the node is is *Y0,Y1,Y2,...,Y*. We can say that Yi has I excluded arcs. The probability of *Y* creating a complete tour is that none of its ancestors provides a solution to the assignment problem thus they do not provide a complete tour—but *Y* does. The probability that Y's parent does not provide a complete tour is (1-q(n,s-1,t_{s-1})). Consequently the probability that *p* and *Y* exists and is a leaf taking into account the independence assumption is:

$$p = q(n.s.t) = \prod_{i=0}^{s-1}(1-q(n,i,ti)) . \qquad 10.34$$

Using lemma 10.1, we find that:

$$p = (\lambda/(n-t) + o(1/n)) \prod_{i=0}^{s-1} 1-\frac{\lambda}{n-ti} =$$

$$\frac{\lambda}{n-t}(1-\sum_{i=0}^{s-1}\frac{\lambda i}{n-ti} + o(1/n), \qquad 10.35$$

where $\lambda_o > \lambda_1 > \lambda_2 > \lambda_{s-1} > \lambda > 1$ are constants.
We can now show that

$$\sum_{i=0}^{s-1}\frac{\lambda i}{n-ti} = \frac{\lambda' s}{n-t'} \qquad 10.36$$

where $\lambda' = ½ \sum_{i=0}^{s-1}\lambda i$

and $t' = \frac{\sum_{i=0}^{s-1}\lambda iti}{\sum_{i=0}^{s-1}\lambda i}$

It is true that $0 < t' < t$.
Now we can write the probability as:

$$p = \frac{\lambda}{n-t}(1-\frac{\lambda' s}{n-t'}) + o(1/n) \qquad 10.37$$

Now we assume that the lemma does not hold thus $p \geq p_o$ where $p_o = e/n$. Let $\delta = (e-\lambda)/e$. we know that $1 < \lambda < e$ and $0 < \delta < 1-1/e$. Now it can be shown that:

$$t' > n + \frac{\lambda\lambda' sn}{(e-\lambda)n-et} > n \qquad 10.38$$

but $n \geq t$ so $t' > t$ which is a contradiction. Thus, the lemma is proven.

Lemma 10.3:

Assume a n x n random matrix. Assume a solution in a modified assignment problem. The expected number of sub-tours is asymptotically less than ln(n) and the expected number of arcs in a sub-tour is greater than n/ln(n).

Proof:

See [7]

The number of children constructed when we choose a sub-tour with the minimum number of arcs is O(n/ln(n)), as is proven in lemma 10.1.

The nodes at the first depth have t = O(n/ln(n)) included arcs. But as it it shown above $t < \delta n$.

This means that all nodes on the first depth asymptotically follow the inequality: $t < \delta n$.

Equally all nodes in the i_{th} depth follow the same inequality except the ones that I is greater than O(ln(n)). Assume a node with no included arcs. Its ancestors do not have included arcs either. Using lemma 10.1 we can state that the probability that the node or one of its ancestors being a solution to the assignment problem is e/n. We can now generalize this and say that the probability that a node with no included arcs exists at d_{th} depth and is a leaf node is:

$$p = e/n(1-e/n)^d. \qquad 10.39$$

Observe that this probability is less than e/n or the probability that the root node is a leaf.. It is therefore true that if we consider nodes whose depth is no greater than O(ln(n)) the probability that they are leaf nodes is less than the probability of the root being a leaf itself.

The probability that a sub-problem chosen by branch and bound will be solved by the assignment problem and will produce an optimal tour is less than the probability that the search will become a leaf node. Consider the node that generates the optimal tour. If its depth is greater than ln(n) then we have to expand ln(n) nodes each one of which has a probability of producing the optimal tour less than p_o. If the depth of the node is lesser than ln(n) and if we need to expand only a polynomial number of nodes according to Bellmore and Malone, then the expanded nodes have a probability less than p_o of creating the optimal tour. This statement contradicts the polynomial assumption. Therefore we can state that the branch and bound algorithm expands more than ln(n) nodes to find the optimal tour. This means that the algorithm cannot finish in polynomial time.

10.2.6 The k-best traveling salesman problem, its complexity and one solution using the branch and bound algorithm

Consider a graph $G = (V, E)$. The number of the n cities in the graph has to be at least 3. Consider an edge $e \in E$. The length of this edge is described by l(e). Consider a vector that contains the lengths of the edges of the initial graph. Let us call this vector l. We can now create a weighted graph, which consists of the pairs (G, d). Consider a set S of edges. The length of this set is described as $L_l(S)$. Consider the set H of all the Hamiltonian tours in the G. We assume that G has at least on Hamiltonian tour.

Definition:

Let $1 \leq k \leq |H|$. Any set $H(k)$ satisfying

$L_l(H_1) \leq L_l(H_2) \leq \ldots \leq L_l(H_k) \leq L_l(H)$ for all H

is called a set of k-best tours.

In other words the k-best tour problem is the problem of finding a set of k tours such that the length of each tour is at least equal to the length of the greater tour in the set.

Complexity of the k-best traveling salesman problem

Theorem 10.7:
The k-best TSP is NP-hard for k≥1 regardless if k is fixed or variable.

Proof:
Consider the case that k is variable. This means that k is included in the input. We have to solve TSP itself to find a solution to the k-best TSP. Since the TSP which is NP-hard is part of the k-best TSP then the k-best TSP is NP-hard too.
Consider the case that k is fixed. This means that k is not included in the input. It is clear that a shortest path can be determined if we know k-best tours. So we can conclude that k-best TSP is NP-hard itself too.

Solutions of the k-best TSP

Solutions provided by partitioning algorithms

Definition:
For $I, O \in E$, the set $\{H: I \subseteq H \subseteq E \backslash O\}$ is called a restricted set of tours.
To solve the problem using partitioning algorithms we use the following idea. We partition the tours into sets such as that each set is a restricted one. We apply algorithms for solving the traveling salesman problem for each one of these restricted sets. We combine the optimal solutions for the sets to find the k-best solution.
There are two known partitioning algorithms for the k-best TSP. The one of them is the Lawler algorithm and the other one is the Hamacher and Queyranne algorithm. The difference in these two algorithms lies in the way they partition the solutions. They both call an algorithm to solve the problem for a restricted set. Their complexity cannot be easily determined since the k-best TSP is NP-hard. A clue to figure out which algorithm is the best of the two would be to check how many times they call the algorithm to find a solution for the restricted set.

Using the branch and bound method to derive solutions for the k-best traveling salesman problem

Since the branch and bound method is used for solving the classic traveling salesman problem (although in greater time than polynomial) it is worthy to modify it to solve the k-best TSP. Initially the branch and bound tree contains only one node, the root node. As the algorithm proceeds each node of the tree expands taking into computation edges from the graph. At a given moment the branch and bound tree contains information about what are the best tours so far. As an analogue to the original branch and bound, which contains information, what is the best candidate for the optimal path. When the algorithm ends, the initially empty tree has information about the set of k-best tours.
We considered a restricted set of tours as is defined above. Let us assume a node in the branch and bound tree with this restricted set. First of all the algorithm determines a lower bound which we will express as $LB(I, O) : L_l(H) \geq LB(I, O)$ for every tour H.
It is true that if $LB(I, O) \geq U$ then we should not take into account any tour that exists in the tree. The algorithm continues until the above holds for all the tours ; that is to say that we cannot take into account any tour in the tree. At that moment we should say that the tree is equal to $H(k)$.
If $LB(I, O) < U$ then we have to distinguish two cases. If the graph contains k tours, then the longest of them is removed. The tour from the tree is removed from the tree and added to the

graph. At that point, the information about the longest tour is updated. If the graph contains less than k tours, then we do not have to remove any tour. The longest tour from the tree is added to the graph and the information about the longest tour is updated.
Below is the formal expression of the algorithm. It uses a recursive procedure named EXPLORE that runs through all the nodes in the branch and bound tree and performs the computations we have explained. It searches the tree at a depth-first way. It has three parameters *I, O* and the graph. The I is the partial tour. The algorithm starts by taking the empty sets *I ,O* and the graph and calling the procedure EXPLORE for these sets.

Modified Branch and bound for finding k-best tours for the traveling salesman problem
Input: (*G, d*), the set of tours *H* and an integer so that $1 \leq k \leq |H|$
Output: A set *H(k)* of best tours in the (*G, d*).

```
Procedure EXPLORE ($I, O, G$)
 1 Begin
 2   Find ($LB(I, O)$) for ($G, d$)
 3    If ($LB(I, O) < U$))
 4     Then if  $|I| = n-1$
 5       Then begin
 6          $H$ is the completion of the partial tour I
 7          If($|G| = k$)
 8           Then
 9             Remove one of the longest tours in $G$
10             $G = G \cup \{H\}$
11             If ($|G| = k$)
12               Then
13               $U$ is the length of a longest tour in $G$
14     End
15     Else begin
16       Find a branching edge $E$
17       EXPLORE ($I \cup \{e\}, O, G$)
18       Find $LB(I, O \cup \{e\}$ taking into account ($G, d$)
19       If ($LB(I, O \cup \{e\}) < U$)
20         Then
21           EXPLORE ($I, O \cup \{e\}, G$)
22      End
23 Begin
24    U <= $\infty$
25    $H(k)$ <= 0
26    EXPLORE (0, 0, $H(k)$)
27    End
```

It has been shown by experiments that the complexity of the branch and bound algorithm increases dramatically as k gets larger. It is a result that we should expect as the branch and bound algorithm gets much slower as the number of cities increase in the classic traveling salesman problem.

10.3 Geometric solutions

10.3.1 A heuristic solution based on continuous approximation

We assume that the cities are distributed over a region that has the shape of a circular sector. The starting point is on the sector vertex. It is not obligatory that this point is actually the point at which the traveling salesman starts the tour, but we need to ensure that the tour visits the vector. We also assume that the distribution of the cities is uniform to be able to state that the probability of finding a city does not depend on the location of the city. Let us assume that we have *N* points *C* of which are cities the salesman visits and *N* = *C* + 1 are the cities plus the starting city. The way the tour is constructed is explained below.
We divide the circular vector into three parts. There are two inner circular sectors that share the vertex as the border and the remaining ring sector. Figure 10.4 shows the division of the circular sector.

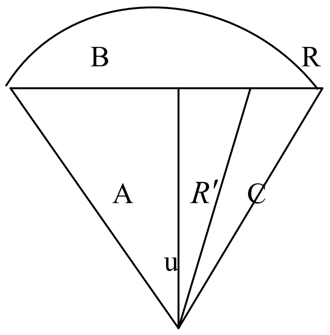

Figure 10.4 The circular sector after it has been divided in three regions.

We name the inner sectors *A* and *C* and the ring sector *B*. The division can be described completely by *R'*. We can write now:

$$p = \frac{R'}{R} \qquad 10.40$$

We visit each one of the regions and construct paths in these regions. In the final step, we combine the paths. Figures 10.5 and 10.6 show the working of the algorithm.

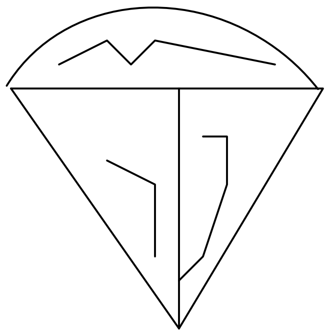
Figure 10.5 The cities in the three sections have been connected

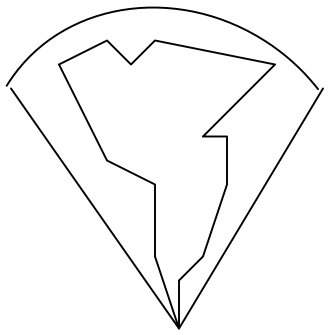
Figure 10.6 All the cities have been connected

From the preceding figures, one can see only the tour and not the starting point. This depends entirely on the partition.
The average length of the tour is estimated below.
The distance between two points of polar coordinates (r, u) is:

$$d(P1,P2) = \min(r1,r2)\text{abs}(u1-u2)+\text{abs}(r1-r2) = d_u + d_r \quad 10.41$$

We assume the length of the radial links and the length of the ring links. We express the total tour length as the sum of all the path lengths either radial or ring on all sections. We can now write:

$$l = 2(l_{A,r} + l_{A,u}) + l_{B,r} + l_{B,,u} \quad 10.42$$

The two multiplier is there because section A and C are the same. We approximate the length of the radial links as $l_{A,r} = p.$
We can also approximate the length of the ring links in A or C by the number of the points in these two regions times the average length of the ring between two points. So we can write:

$$l_{A,u} = n_A d_{A,u} \quad 10.43$$

By assumption the point are distributed uniformly in all the regions so we can state that:

$$n_A = Cp^2/2 \quad 10.44$$

The average ring length for all the points can be approximated by:

$$d_{A,u} = \int_0^p \frac{ru}{6} dr \; = pu/12 \quad 10.45$$

So the length of the ring in A or C becomes:

$$l_{A,\mathrm{u}} = p^3 uC/24 \quad 10.46$$

Taking into consideration that the radial distribution has a probability density of f(r) = 2r we conclude that the average distance in sector B is:

$$\mathrm{p_{avg}} = \int_{p}^{1} rf(r)dr = \frac{2(pp+p+1)}{3(1+p)} \qquad 10.47$$

let us make an assumption that we have a uniform radial distribution so as to simplify the above expression. Now we can write:

$$p = \frac{1+p}{2} \qquad 10.48$$

Now we can compute the length of the ring links in B. We find that

$$L_{B,u} = \frac{(1+p)u}{2} \qquad 10.49$$

So the expected radial distance between two points found in B can be expressed as:

$$d_{B,r} = 2\int_{p}^{1}\int_{p}^{r}(r-s)f(r)f(s)dsdr = \frac{4(1-p)(pp+3p+1)}{15(1+p)(1+p)} \qquad 10.50$$

Once again, we assume a uniform radial distribution and can write that:

$$d_{B,\mathrm{r}} = \frac{1-p}{3} \qquad 10.51$$

We compute the length of the radial links in B as the number of points times the expected distance between two points. Thus we can write:

$$l_{B,r} = n_{B,}d_{B,r} = \mathrm{C}(1-p^2)(1-p)/3 \qquad 10.52$$

From the above expressions, we can find the average tour length:

$$l = 2p + \frac{pppuC}{12} + \frac{(1+p)u}{2} + \frac{C}{3}(1-p^2)(1-p). \qquad 10.53$$

The above expression has a drawback. The results it produces are pessimistic if p is close to 1. This is the case when the circular sector is divided into two identical sections, thus having an angle of *u*/2. Now there is no ring *B* but there still exists a ring that connects the outermost paths of *A* and *C*. This is the cause that makes the estimation pessimistic.
We can instead substitute the expression that gives the length of the radial links in *B* by the more accurate:

$$l_{B,u} = \frac{(1+p)u}{2}(1-p/2) \qquad 10.54$$

We can take more precise estimations for the total length of the tour by this expression:

$$l = 2p + \frac{pppuC}{12} + \frac{(1+p)u}{2}(1\text{-}p/2) + \frac{C}{3}(1\text{-}p^2)(1\text{-}p). \quad 10.55$$

This expression is immune to very low values of p (approaching 1) and gives a very accurate estimation of the total tour length.

10.3.2 Held – Karp lower bound

Definition:
A 1-tree problem on n cities is the problem of finding a tree that connects n cities with the first city connecting to two cities.

When we try to find a lower bound for the 1-tree problem we try to find a minimum 1-tree. We apply the 1-tree problem to the traveling salesman problem by considering that a tour is a tree whose each vertex has a degree of two. Then a minimum 1-tree is also a minimal path. Figure 10.7 shows a simple 1-tree.

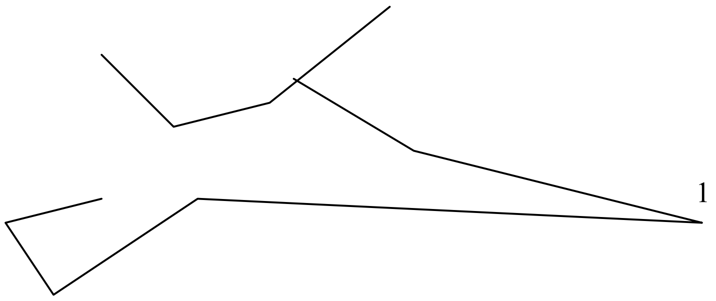

Figure 10.7 A simple 1-tree.

Let us consider the geometric traveling salesman problem. We denote to e_{ij} the length of the path from the i city to the j city. We assume that each city has a weight of π_i. So we can say that each edge has a cost of

$$c_{ij} = e_{ij} + \pi_i\ \pi_j \quad 10.56$$

We now compute a new minimum 1-tree taking into account the edge costs. It is clear that the new 1-tree we will construct id different from the original 1-tree. Let us consider a set of different tours V. Let U be the set of 1-trees constructed by each tour from V. Recall that a tour is a 1-tree with each vertex having a degree of 2. This means that the set of tours is included in the set of 1-trees. Let us express a tour with T and the cost of a tour as $L(c_{ij}, T)$ if we take in account the cost of the edges. Therefore, it is true that

$$\min L_{T\in U}(c_{ij}, T) \leq \min L_{T\in V}(c_{ij}, T) \quad 10.57$$

From equation 3.17 we can write that:

$$L(c_{ij}, T) = L(e_{ij}, T) + \sum_{i=1}^{n} \pi_i d^T{}_i \quad 10.58$$

With d^T_i we symbol the degree of i vertex in the 1-tree.
Consider T to be a tour. This means that $d^T_i = 2$. So we can now write that:

$$L(c_{ij}, T) = L(e_{ij}, T) + \sum_{i=1}^{n} \pi_i 2 \qquad 10.59$$

We assume a minimal tour T'. Equation 3.18 is then transformed as:

$$\min L_{T\in U}(c_{ij}, T) \leq \min L(c_{ij}, T') - \sum_{i=1}^{n} \pi_i 2 \qquad 10.60$$

Let us express the length of the optimal tour as $c' = L(e_{ij}, T')$.Then from equation 10.59 and 10.60 we can get:

$$\min_{T\in U}\{c + \sum_{i=1}^{n} \pi_i d^T_i\} \leq c' + \sum_{i=1}^{n} \pi_i 2 \qquad 10.61$$

This is transformed into:

$$\min_{T\in U}\{c + \sum_{i=1}^{n} \pi_i (d^T_i - 2)\} \leq c' \qquad 10.62$$

We can finally write that:

$$w(\pi) = \min_{T\in U}\{c + \sum_{i=1}^{n} \pi_i (d^T_i - 2)\} \qquad 10.63$$

Hence the lower bound for Held-Karp is

$$\text{Held-Karp}_{\text{lower-bound}} = \max(w(\pi)). \qquad 10.64$$

It has been shown that Held Karp is a very good estimate for the minimum tour length although it does not give the exact result.

References:
[1] Corman H. Thomas, Leiserson E. Charles, Rivest L. Ronald, Stein Clifford "*Introduction to Algorithms*" Second Edition McGrawHill Book Company
[2] Cesari Giovanni *"Divide and Conquer Strategies for Parallel TSP Heuristics"*, Computers Operations Research , Vol.23, No.7, pp 681-694, 1996
[3] del Castillo M. Jose *"A heuristic for the traveling salesman problem based on a continuous approximation"*, Transportation Research Part B33 (1999) 123-152
[4] Gutin Gregory,eo Anders, Zverovich Alexey *"Traveling salesman should not be greedy: domination analysis of greedy-type heuristics for the TSP" ,* Discrete Applied Mathematics 117 (2002), 81-86
[5] van der Poort S. Edo, Libura Marek, Sierksma Gerard, vander Veen A. A. Jack *"Solving the k-best traveling salesman problem"*, Computers & Operations Research 26 (1999) 409-425
[6] Valenzuela I. Christine, Jones J. Antonia *"Estimating the Held-Karp lower bound for the geometric TSP* , European Journal of Operational Research 102(1997) 157-175
[7] Zhang Weixiong "*A note on the complexity of the Assymetric Traveling Salesman Problem",* Operation Research Letters 20 (1997) 31-38