ADDIS ABABA UNIVERSITY

SCHOOL OF GRADUATE STUDIES

DEPARTMENT OF MATHEMATICS

GRADUATE PROJECT REPORT ON SYMMETRIC TRAVELLING SALESMAN PROBLEM

(Submitted in partial fulfillment of the M.Sc.degree in mathematics)

Compiled by: Adugna Kefale

Advisor: Dr.Berhanu Guta

June, 2014

Addis Ababa.

Abstract

Symmetric travelling sales man problem is a well known combinatorial optimization problem .It is a problem of findining the shortest Hamiltonian cycle in a given weighted graph. In this project paper the symmetric travelling sales man problem, together with its mathematical formulation and solution method ,especially the exact solution (Branch and bound approach) and an approximate(Heuristics approach) have been covered.

Acknowledgement

I would like to take this movement to thank all the people who have been Crucial in aiding me during my graduate work. In particular, I must thank my parents. Their support condense in me have served as a source of inspiration.

I would like to express my heartfelt gratitude to my Advisors Dr.BerhanuGuta ,my lecturers, relatives and friends especially my best friends , Ato SemahegnAbayneh,AtoBahiruTsegaye, for their encouragement and co-operations in every aspects for the completion of this project.

I must also thank the Mathematics department of Addis Ababa University for giving me the opportunity to do my graduate work and teaching me far more about life than Mathematics.

Table of contents

ContentsPage

Chapter One

1. Symmetric Travelling sales man problem

1.1 Introduction

In this project paper, the travelling sales man problem, the problem of finding the shortest Hamiltonian cycle in a given weighted graph, together with its mathematical formulation, and solution method will be considered.

1.2. Symmetric Travelling sales man problem

What is travelling sales man problem?

Suppose a sales man has to visit n-cities. He wishes to start from city 1, visits each city once and then returns to his starting point. Then we call this travelling salesman problem. Here we will restrict ourselves to symmetric travelling salesman problem that is the distance from city i to city j is the same as distance from city j to city i. A travelling sales man problem has solution if the following assumptions holds 1) $c_{ik} \leq c_{ij} + c_{jk}$ for all $i, j \; and \; k$(triangle inequality must holds) 2) every node must be connected

The objective is to select a route that minimizes his total travelling time. Clearly starting from a given city a sales man will have $\frac{(n-1)!}{2}$ possible routes.

Historical development of travelling sales man problem

In 1736 Leonard ever studied the problem of finding around trip through seven bridges in Konigsberg In 1832, a hand book was published for German travelling sales man, which included example of tour.

In 18^{th} century travelling salesman were studied by Mathematician from Ireland named Sir William row am Hamilton and by the British Mathematician Thomas Penyngton Kirkman

Detailed discussion about the work of Hamilton and Kirkman can be seen from the book titled graph theory.(Biggs et al .1976)

In mid-1940 the travelling sales man problem was studied by several statisticians etc.

There are several applications of the TSP. Some of these applications are:

Work sequencing (Garfunkel, 1985).The solution of the TSP is used to determine the optimal sequence of the production works that minimizes the total preparative time of the preliminary works. This extension of the TSP optimization model is applicable in cases when the production works can be processed in any order.

Vehicle routing (Christofides, 1985). The optimal solution of the TSP is used to determine for a fleet of vehicles which customers should be visited by which vehicles, and in which order each vehicle should visit its customers. This variant of the TSP optimization model usually contains additional time restrictions for the customers and capacity constraints for the vehicles.

Symmetric travelling salesman problem is given by a graph $G = (V, E)$ and a weight vector $C \in IR^{|E|}$ where $V = \{1 \dots m\}$ and set $E = \{e_1, e_2 \dots e_n\}$ whose elements are a subset of V of size two that is $e_k = \{i, j\}$, where i, j is an element of V. The element of V are called nodes and elements of E are called edges. Graphs are useful models of many of the problems considered in combinatorial optimization especially for symmetric traveling sales man problems. Because of this I' will consider some concepts of graph theory.

A Graph G is a pair of sets (V, E) where V is a set of vertices and E is a set of edges.A Graph is said to be complete if it contains all possible edges. Let the graph G= (V, E), $U \subseteq V$, let $E(U) = \{(i,j) \backslash (i,j) \in E,\ i \in U, j \in U\}$. E (U) is the set of edges with both end points in U.

If $V' \subseteq V$ and $E' \subset E$, then $G' = (V', E')$ is said to be sub graph of $G = (V, E)$ .The graph G= (V,E) where E= {e/e $\notin E$} is called the complement of G.

G is spanning sub graph if $V = V'$.

A node sequence $v_0, v_1 \dots v_k, k \geq 1$ is called a V_0 - V_K walk if $(v_{i-1}, v_i) \in E$ for $i = 1 \dots k$

Node V_0 is called Origin and node V_k is called destination and node$V_1 \ \dots \ V_{k-1}$ are called intermediate nodes. We can also represent a walk by its edges $e_1, e_2 \dots e_k$ where, $e_{i=(}v_{i-1}, v_{i)}$) for $i = 1 \dots k$. The length of a walk$v_0 \dots v_k$ or $e_1 \dots e_k$ is k , the number of edges in it.

A Walk is called a path if there are no node repetitions. or in non-directed graph G a sequence p of zero or more edges of the form $\{v_o,v_1\},\{v_1,v_2\},\ldots\{v_{n-1},v_n\}$is called a path from v_0 to v_n.

A Path p is simple if all edges and vertices on the path are distinct except possibly the end points. A Path of length ≥ 1 with no repeated edges and whose end points are equal are called circuit. A circuit may have repeated vertex other than the end points. A cycle is a circuit with no other repeated vertices except its end points thus a cycle is simple circuit. A loop is a cycle of length 1. A $v_o - v_k$ walk is called closed if $v_{0=}v_{k.}$

A closed walk is said to be a cycle if k $\geq$3 and $v_{o,}v_1 \ldots v_{k-1}$ is a path. A graph G is said to be acyclic if it doesn't contain any cycle.

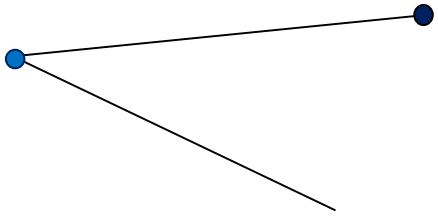
If G is directed graph (digraph) the elements of E are order Pairs of vertices. In this case edge (u, v) is said to be from u to v. If G is non-directed graph the elements of E are unordered pairs or sets of vertices .In this case an edge {u, v} is said to join u and v. An edge that is between a vertex and itself is called self-loop. If G is a graph $V(G)$ and $E(G)$ denotes sets of vertices and edges respectively. Ordinarily V (G) is assumed to be finite set in which case $E(G)$ must also be finite and we say that G is finite. If G is finite $|V(G)|$ denotes the number of vertices in G, and is called the order of G. Similarly if G is finite | E (G)| denotes the number of edges in G and is called the size of G.If one allows more than one edge to join a pair of vertices, the result is then called a multigraph. Graphs may be expressed by diagrams in which each vertex is represented by a point in the plane and each edge by a curve joining the points.

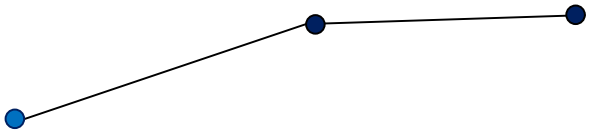
In symmetric travelling salesman problem we use non- directed graph in which for every edge (u, v) between two vertices in one direction there is also an edge (v, u) between the same vertexes in the other direction

Connected graph –A graph G is called connected if there is at least one path between every pairs of vertices in G

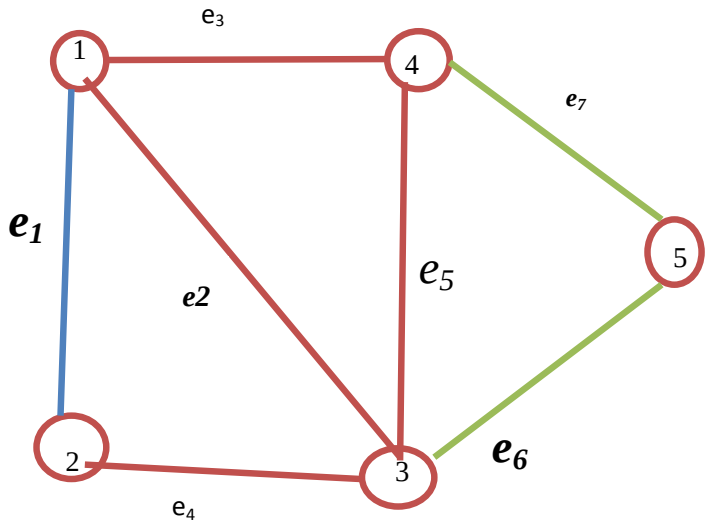
A Tree is connected graph without cycle.

fig (1) tree





Spanning Tree-Let G= (V, E) a connected graph T is said to be a spanning tree of G if T is acyclic connected sub graph of G and contains all nodes of G. Minimum spanning tree is a



spanning tree of Minimum weight

Fig (1) –non-directed graph

Example consider $G = (V, E)$ with V={1, 2, 3, 4, 5} and E={e_1=(1,2), e_2=(1,3), e_3=(1,4), e_4=(2,3), e_5=(3,4), e_6=(3,5) and e_7=(4,5).

1, 3, 4, 5---is a1-5 path.

1, 3, 5, 4, 1---is a cycle of length 4

1, 2, 3, 1, 4, 5 or {1, 2},{2, 3},{3, 1},{1, 4},{4, 5} is a walk but not path. But the sequence1, 2, 3, 4, 5 is a path

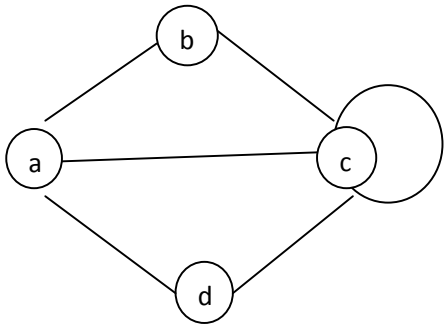


Fig (2)-non simple graph

In the above figure the path {c, c} is a cycle of length 1. While the sequence of edges {a, b}, {b, c} {c, a} and {a, d}, {d, c}, {c, a} forms a cycle of length 3. The path {a, b}, {b, c}, {c, d}, {d, a} is a cycle of length 4. More over {a, b}, {b, c}, {c, c}, {c, a} is a circuit of length 4. It is not cycle because the sequence of vertices a-b-c-c-a includes more than one repeated vertex.

Chapter two

2. Mathematical formulation of symmetric travelling sales man problem

An integer programming formulation of symmetric travelling salesman problem.

Symmetric travelling salesman can be formulated as integer programming problem.

Here we translate problem descriptions in to mathematical expressions To do this notice the three basic elements of optimizations models

i) Decisions variables –the decision variables describe the decision to be made

ii) Constraints-are restrictions on the value of decisions variables.

iii) Objective functions-the function we wish to maximize or minimize

So to formulate symmetric travelling salesman problem we have to consider the above three elements of optimizations,

1) **The decisions variable** here we define the necessary variables

Let x_e be **decision variable** defined as $x_e = \begin{cases} 1, & if e \in T \\ 0, & otherwise \end{cases}$

2) Constraints using decision variables we define set of required constraints. In symmetric Travelling sales man problem, we have two restrictions 1) each node must have degree two (degree constraints) and 2) a sales man must visit all cities (sub tour eliminations constraints)

Degree constraints degree of node i is defined as the number of edges incident to node i .

For node one only two of its decision variables take the value 1 all other decision variables take the value 0 and it is true for all other nodes.

For n nodes the degree constraints can be formulated as

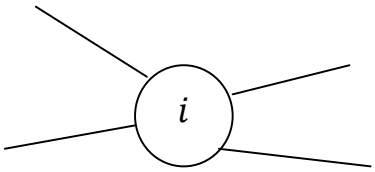
$x_{12} + x_{13} + x_{14} + \cdots + x_{1n} = 2$ *Node one* degree *constraints*

$x_{21} + x_{23} + x_{24} + \cdots x_{2n} = 2$ *Node 2* degree *constraints*

… … ….……..

… … … ….…….

$x_{n1} + x_{n2} + x_{n3}$ … ….$x_{n-1n} = 2$ *Node n* degree *constraints*

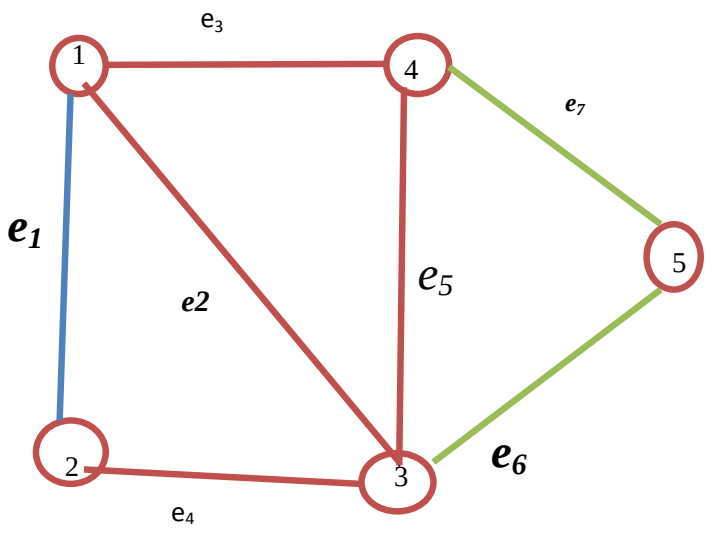


Let δ(i)-be set of edges which has one end in i since each edge node has degree 2 then the degree *constraint* becomes

$$\sum_{e \in \delta(i)} x_e = 2 \; for i \in V$$

Degree constraints *ensures that for a tour there must be two edges incident to every node*

Example consider the following non-*directed graph*



Ex

$x_{12} + x_{14} + x_{13} = 2$ *Node one constraints*

$x_{12} + x_{23} = 2$ *Node 2 constraints*

$x_{23} + x_{31} + x_{34} + x_{35} = 2$*Node 3 constraints*

$x_{41} + x_{43} + x_{45} = 2$*Node 4 constraints*

$x_{54} + x_{53} = 2$*Node 5 constraints*

Since each edge has two end points this implies that every n variables are allowed to take the value 1 so the degree constraints alternatively formulated as

$$\sum_{e \in E_T} x_e = n \quad for\ i \in V$$

A tour T of G is a cycle that contains all the nodes.

A sub tour is a cycle that does not contain all the nodes.

If $T = (V, E_T)$ then each node of T is of degree 2.

T is connected and $|E_T| = |V|$

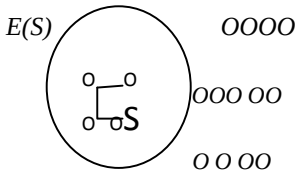
Our reasons for studying tours is that it is feasible solution for symmetric travelling salesman problem

The objective is to find a tour of minimum weight.

The weight of a tour T with edge set $E_T \subset E$ is given by

$$\sum_{e \in E_T} c_e \, .$$

ii) Sub tour eliminations constraints-states that for a specific subset S of V number of edges connecting vertices in S have to be less than $|S|$ at least by 1



Let E(S) -be set of edges which has both ends in S then the sub tour eliminations constraints become

$$\sum_{e \in \mathrm{E(S)}} x_e \leq |S| - 1$$

For every tour number of nodes =number of arcs therefore to prevent from forming sub tour ,number of edges must be less than number of cities for every subset and for any partitions S and V/S ,there must at least be two edged connecting S and V/S.

Therefore the sub tour eliminations constraints alternatively formulated as

$$\sum_{e \in \mathrm{E(S)}} x_e \geq 2$$

3) **Objective function**-let c_e be costs on edge e so our objective in **symmetric travelling salesman problem**s is to minimize

$$\sum_{e \in E} c_e \, x_e$$

Hence the mathematical formulation of symmetric travelling salesman problem is given by

$$min \sum_{e \in E} c_e \, x_e$$

Subject to the following constraint;

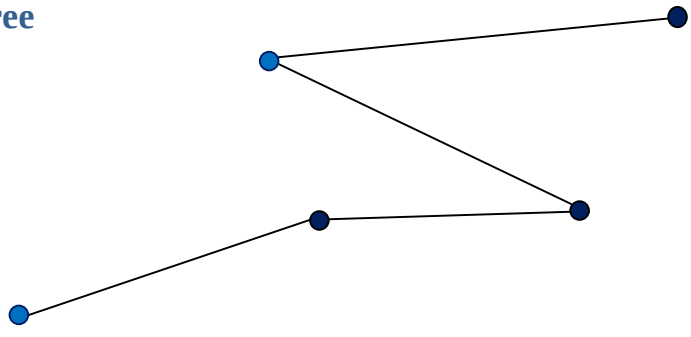
$$\sum_{e \in \delta(i)} x_e = \; 2 \; for \; i \in V$$

$$\sum_{e \in E(S)} x_e \leq |S| - 1 \; forall \; S \subseteq V$$

Tree based formulations

A Tree is connected graph without cycle.

fig (1) tree



Now the important concept of a 1-tree may be defined.

A 1-tree for a graph G = (V, E) is a spanning tree on the node set V\ {1} combined with two edges from E incident to node 1.

Minimum 1-tree gives lower bound for symmetric travelling salesman

Tree based formulation

$$min \sum_{e \in E} c_e \, x_e$$

Subject to the following constraint;

$$\sum_{e \in \delta(i)} x_e = 2 \; for \; i \in V$$

$x \in$ 1- tree here we don't bother about sub tour eliminations constraints because every node is connected

Chapter three

3. Methods of solving symmetric travelling sales man problem

3.1. Branch and bound approach

3.2. Heuristics

3.1.1. Exact algorithms

The *exact* algorithms are guaranteed to find the optimal solution in a bounded number of steps. Today one can find exact solutions to symmetric problems with a few hundred nodes, although there have been reports on the solution of problems with thousands of nodes.

The most effective exact algorithms are branch and bound, cutting-plane or facet-finding algorithms. These algorithms are quite complex, with codes on the order of 10,000 lines. In addition, the algorithms are very demanding of computer power. For example, the exact solution of a symmetric problem with 2392 cities was determined over a period of more than 27 hours on a powerful super computer It took roughly 3-4 years of CPU time on a large network of computers to determine the exact solution of the 7397-city problem.
Symmetric problems are usually more difficult to solve than asymmetric problems today the 7397-city problem is the largest (nontrivial) symmetric problem that has been solved. In comparison, the optimal solution of a 500,000-city asymmetric problem has been reported

3.1.1.1. Branch and bound approach

The branch and bound technique involves a well-structured systematic search of the space of all feasible solution of constrained optimization problem that have finite number of feasible solutions. The general principle of the branch and bound technique remains constant from one algorithm to next; however, the procedure for branching, bounding and pruning may vary considerably.

At any point in a branch and bound algorithm where a branching decision must be made ,any subset with a bound that is less than the least upper bound for all feasible solutions discovered to date is a possible candidate for branching (minimization problem) .There are basically two branching decisions rules used in all branch and bound algorithm ; namely,

Branch from lowest bound and Branch from newest active bound.

The branch from lowest bound decision rule says that the next branching should be from the subset of possible solution that has the lowest bound on the optimal solution .In general this policy has disadvantages that it examines fewer sub problems than the other rule, but has disadvantages of requiring more computer storage because more intermediate data must be stored. The branch from newest active bound rule chooses from the subsets most recently generated the subset with the lowest bound to branch from next. This rule generally requires many more branching operations than the decisions rule but requires much less storage.

Branch and Bound techniques involves branching from one problem to a set of sub problems and establishing bounds on the new sub problems. A procedure for branching to a set of sub problems and bounding them is branch and bound algorithms.

Branch and bound uses two relaxations.

1) Assignment relaxations

2) 1- tree relaxations

1. Assignment relaxations-Assignment relaxations use the following algorithms

1. Set all $c_{ii}= \infty$

2. Solve assignment problem to get a lower bound

3. If there are no sub tours: finished with an optimal TSP solution

4. If there are sub tours: Find an upper bound

5. Take one sub tour and eliminate it by branching fewest numbers of arcs.

6. Solve the AP corresponding to each sub tour. Kill any branch whose lower bound is greater than upper bound.

Since we solve symmetric traveling salesman problem using Assignment problem, we consider some concept of assignment problem.

Assignment problem is a particular class of transportation linear programming problem when supplies and demand equal to integers (often=1).

Methods of solving Assignment Problem

Assignment problem can be solved by

Hungarian method,

Simplex method and

Transportation method .So for this case only Hungarian method is used.

Hungarian method-The Hungarian method of assignment problem provides us an efficient method of finding the optimal solution without having to make a direct comparison of every solution .It works on the principle of reducing the given cost matrix of opportunity cost Opportunity cost show the relative penalties' of associated with assigning resources to an activity opposed to making the best or least cost assignment

Theorem: - In an assignment problem with cost c_{ij} ,if all $c_{ij} \geq 0$ then the feasible solution x_{ij} which satisfies $\sum_j^n \sum_i^n c_{ij}\, x_{ij} = 0$ is an optimal solution for the problem.

Proof:-since all c_{ij} ,≥ 0 the objective function Z =$\sum_j^n \sum_i^n c_{ij}\, x_{ij}$cannot be negative. The minimum possible value that Z can attain is 0 .thus any feasible solution x_{ij} that satisfies $\sum_j^n \sum_i^n c_{ij}\, x_{ij} = 0$ will be optimal.

The Hungarian method follows the following steps

Step1:-For each row subtract the minimum number in the row from all numbers in that row

Step2:-Then for each column subtract the minimum number in that column from all the numbers in that column

Step3:-Draw the minimum number of lines to cover all zeros if this number =n stop

An assignment can be made otherwise go to step4

Step4:- determine the minimum uncovered number (call it d)

a) Subtract d from uncovered numbers

b) Add d to numbers covered by two lines

c) Number covered by one lines remains the same then go to step3

Finding the minimum number of lines and determining the optimal solutions

1 Find a row or column with only one unlined zero and circle it .If all rows and /columns have two or more zeros choose an arbitrary zero and circle it

2 If the circle in a row with one zero draw line throw its columns.

if the circle in the column with one zero then draw one line throw its row

3 repeat step2 until the entire circle is lined.

This determines the minimum number of lines if this minimum number equal's n this circle provides the optimal assignment.

Let us consider the following example: which is symmetric;

A Salesman starts from city A wants to visit city B, C, D and E. Finally he returns back to City A. The distance between each of the cities is given in (Table 1) below. Then what order of visiting the cities minimizes the total distance travelled?

Table 1: The distance between each of the cities.

	A	B	C	D	E
City 1(A)	0	132	217	164	58
City 2(B)	132	0	290	201	79
City 3 (C)	217	290	0	113	303
City 4(D)	164	201	113	0	196
City 5(E)	58	79	303	196	0

Then using Eastman algorithm we can solve the above problem

Step1

- Consider the problem as if it were an assignment problem two methods are available Hungarian and Branch and bound.
- Now using Hungarian method we solve the assignment problem as follows

Step 1.1- Subtract the smallest element in each row from every element of the same row of cost matrix then we get the following reduced cost matrix as given bellow.

	A	B	C	D	E
A:	M	53	159	106	0
B:	53	M	211	122	0
C:	104	156	M	0	190
D:	51	67	0	M	83
E:	39	0	245	138	M

Step1. 2:- Subtract the smallest element in each column from every element of the same column of the reduced cost matrix as below.

	A	B	C	D	E
A:	M	53	159	106	0
B:	53	M	211	122	0
C:	104	156	M	0	190
D:	51	67	0	M	83
E:	0	0	245	138	M

Step1. 3:-We check the rows and mark these rows where is exactly one zero

A	B	C	D	E

A: M 53 159 106 < 0>

B: 53 M 211 122 ∅

C: 104 156 M <0> 190

D: 51 67 < 0 > M 83

E: 0 0 245 138 M

Step 1.4:-We cross all zeros in the columns where is a mark

	A	B	C	D	E
A:	M	53	159	106	<0>
B:	53	M	211	122	∅
C:	104	156	M	0	190
D:	51	67	< 0 >	M	83
E:	0	0	245	138	M

Step 1.5:-We check the columns and mark these columns where is exactly one remaining zero.

We mark it and cross all other zeros in this row.

	A	B	C	D	E
A:	M	53	159	106	< 0>
B:	53	M	211	122	ø
C:	104	156	M	<0>	190
D:	51	67	< 0 >	M	83
E:	< 0 >	ø	245	138	M

Now if we have all rows and all columns have marked zeros then we have an optimal table but in this case we have a new situation that not all row and columns have a marked zero. So we go to the next step.

Step 1.6-we mark the row where is no marked zero and we mark that column which contain a zero from a marked row. We mark also the row which contains the marked zero in marked column.

				↓	
	A	B	C	D	E
A:	M	53	159	106	< 0 >
→B:	53	M	211	122	ø
C:	104	156	M	< 0 >	190
D:	51	67	< 0 >	M	83
E:	< 0 >	ø	245	138	M

Step 1.7-We draw lines throw the non-marked rows and the marked columns

				↓
A	B	C	D	E

A: M 53 159 106 < 0 >

→ B: 53 M 211 122 Ø

C: 104 156 M < 0 > 190

D: 51 67 < 0 > M 83

E: < 0 > Ø 245 138 M

Step 1.8- From the non-cover elements we subtract the minimal and we add to those numbers which lies in the cross points of the drawn lines then we get the following reduced matrix.

	A	B	C	D	E
A:	M	53	159	106	53
B:	0	M	158	69	0
C:	104	156	M	0	243
D:	51	67	0	M	136
E:	0	0	245	138	M

Step1. 9-repeat all the above steps in cost matrix given in step 8 then we get optimal assignment.

	A	B	C	D	E
A:	M	< 0 >	106	53	0
B:	0	M	158	69	< 0 >
C:	104	156	M	00	243
D:	51	67	0	M	136
E:	0	0	245	138	M

The optimal solution is $x_{12} = x_{25} = x_{34} = x_{43} = x_{51} = 1$ of distance =495 and this solution contain two sub tours (1-2-5-1) and (3-4-3) and cannot be the optimal solution to salesman problem because it is not a tour.

Step2

Since the solution of the assignment problem in step1 did not yield a tour, 495 is a lower bound of all feasible solution of symmetric travelling salesman problem

Step3

The sub tour with smallest number of links is clearly 3-4-3since it has two edges, while 1-2-5-1has 3 edges thus k=2

Step4

Branch in to sub problems, one with c_{34}=M and other with c_{43}=M

Step5

Let c_{34}=M and solve the resulting assignment problem.

Table 2- The cost matrix for sub problem C_{34}=M

	City 1	City 2	City 3	City 4	City 5
City 1	M	132	217	164	58
City 2	132	M	290	201	79
City 3	217	290	M	M	303
City 4	164	201	113	M	196
City 5	58	79	303	196	M

The optimal solution to sub problem c_{34}=M is Z=652,$x_{14} = x_{25} = x_{31=}x_{43} = x_{52} = 1$ this solution include the sub tours 1-4-3-1and 2-5-2with total distance of 652.

Again repeat step 3with smallest number of edges .Branch in to sub problems one with c_{25}=M other with c_{52}=M

Let c_{25}=M and solve the resulting assignment problem

	City 1	City 2	City 3	City 4	City 5
City 1	M	132	217	164	58
City 2	132	M	290	201	M
City 3	217	290	M	M	303
City 4	164	201	113	M	196
City 5	58	79	303	196	M

Then we get the optimal solution Z= 668 that is x_{15} =x_{24} = x_{31}=x_{45}=x_{52} =1. This solution contains no sub tour and yields a candidate solution with Z= 668

Following LIFO rule, we next solve sub problems applying Hungarian method to the following cost Matrix with sub problems c_{52}=M.

Table 4: cost Matrix for sub problem c_{52}=M

	City 1	City 2	City 3	City 4	City 5
City 1	M	132	217	164	58
City 2	132	M	290	201	79
City 3	217	290	M	M	303
City 4	164	201	113	M	196
City 5	58	M	303	196	M

Then the optimal solution for sub problems is Z = 704, $x_{14} = x_{43} = x_{32} = x_{25} = x_{51} = 1$. This solution is a tour, but Z = 704 $>$ 668. This sub problem may be eliminated from consideration Cost matrix for sub problem with

	City 1	City 2	City 3	City 4	City 5
City 1	M	132	217	164	58
City 2	132	M	290	201	79
City 3	217	290	M	113	303
City 4	164	201	M	M	196
City 5	58	79	303	196	M

$x_{13} = x_{25} = x_{34} = x_{41} = x_{52} = 1$, of distance =652.This solution contains the sub tours 1-3-4-1 and 2-5-2. Because $652 < 668$, however, it is still possible for sub problem 3 to yield a solution with no sub tours that beats Z = 668. Thus we now branch on sub problem 3 in an effort to exclude the sub tours. So the smallest number of arcs is 2-5-2 branch on sub problem with one C_{25}=M or C_{52}=M.

Cost matrix for sub problem with C_{25}=M

	City 1	City 2	City 3	City 4	City 5
City 1	M	132	217	164	M
City 2	132	M	290	201	79
City 3	217	290	M	113	303
City 4	164	201	113	M	196
City 5	58	79	303	196	M

Solving the above assignment problem we get a tour (1-5-2-3-4-1) of distance 704

This solution contains no sub tours, but its Z-value of 704 is inferior to the candidate solution from sub problem 4 so sub problem 6 cannot yield the optional solution to the problem.

The only remaining sub problems is sub problem 7 the optimal solution to the sub problem 7 is $x_{13} = x_{25} = x_{31} = x_{42} = x_{54} = 1$, Z=910, Again *Z*= 910 is inferior to *Z*=668, so sup problem 7 of cannot yield the optimal solution.

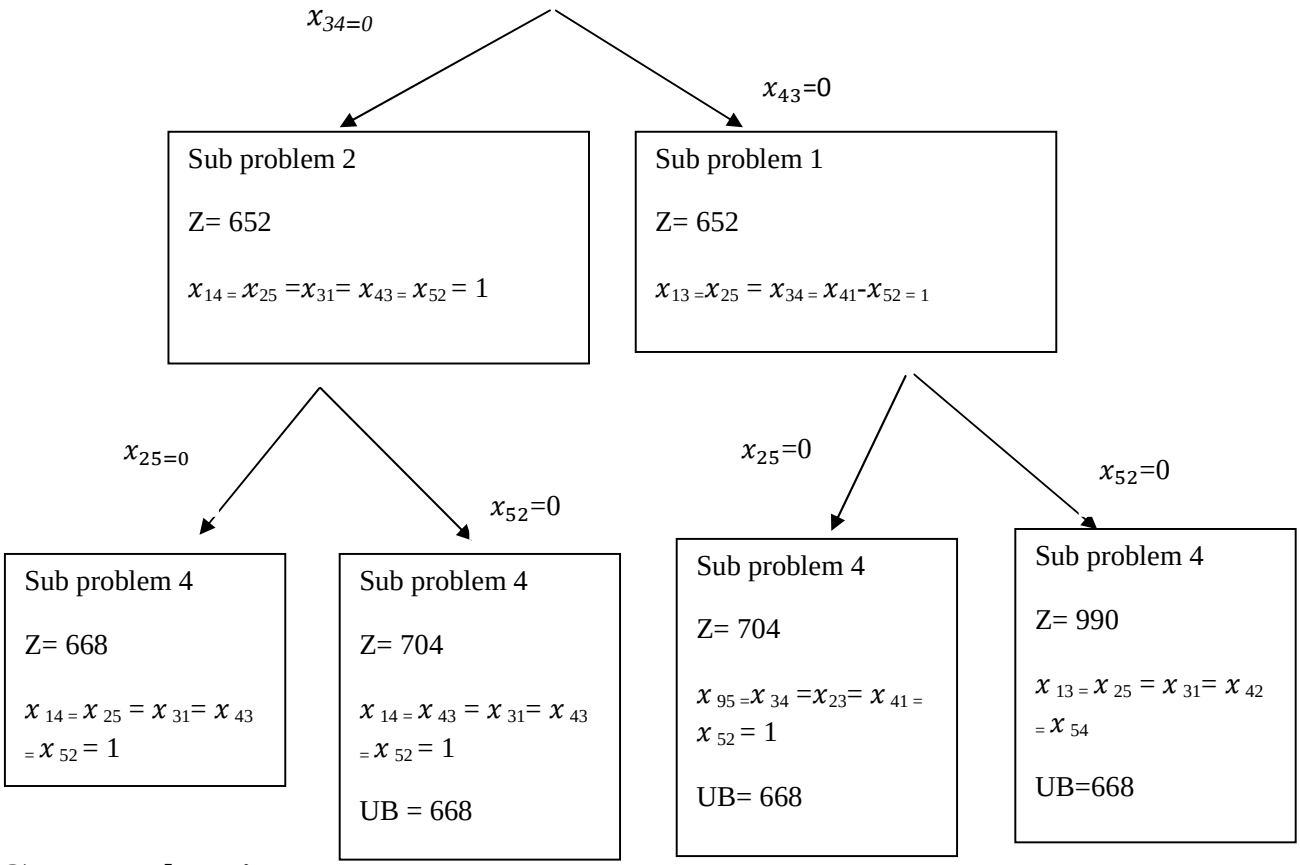
Sup problem 4 yields thus the optimal solution: A salesman should travel from *1* to *5, 5* to *2, and 2* to *4, 4* to *3* and from *3* to *1* so salesman travels a total distance of 668 miles .This problem can be also solved by Heuristics.

Fig (3)-Branch and bound tree

Sub problem 1

Z= 495

$x_{15} = x_{21} = x_{34} = x_{43} = x_{52} = 1$



2)1 tree relaxations

Branch and Bound for TSP– *Using the 1-tree* relaxation

A B&B algorithm for a minimization problem consists of three main

Components:

1. A*bounding function* providing for a given subspace of the solution space a lower bound for the best solution value obtainable in the subspace,

2. A *strategy for selecting* the live solution subspace to be investigated in the current iteration, and

3. A *branching* rule to be applied if a subspace after investigation cannot be discarded, hereby subdividing the subspace considered into two or more subspaces to be investigated in subsequent iterations

Bounds

One way to identify a bound for the TSP is by relaxing constraints. This could be to allow

Sub tours. This bound is although known to be rather weak. An alternative is the 1-tree relaxation

The 1-tree bound

Identify a special vertex 1 (this can be any vertex of the graph).

1 and all edges incident with 1 are removed from G.

For the remaining graph determine the minimum spanning tree T.

Now the two smallest edges e_1 and e_2 incident with 1 are added to T producing T1 (called a 1-tree)

Strengthening the bound.

Starting from the calculated 1-tree if the solution is a tour we have found the solution of our sub problem

Otherwise a vertex of degree at least 3 exists and we have to perform a branching

We can branch-and-bound by including or excluding set of edges.

The sub-problems are also TSP problems and the corresponding 1-tree problems give us the lower bound. And we can improve lower bound using problem transformations

Substantially by making a simple transformation of the original cost matrix.

The transformation is based on the following observations:

(1) Every tour is a 1-tree. Therefore the length of a minimum 1-tree is a lower bound on the length of an optimal tour.

(2) If the length of all edges incident to a node are changed with the same amount, π, any optimal tour remains optimal. Thus, if the cost matrix $C= (c_{ij})$ is transformed to $D = (d_{ij})$, where $d_{ij} = c_{ij} + \pi_i + \pi_j$, then an optimal tour for the D is also an optimal tour for C.

The length of every tour is increased by $2\Sigma\pi_i$. The transformation leaves the TSP invariant, but usually changes the minimum 1-tree.

(3) If T_π is a minimum 1-tree with respect to D, then its length, $L(T_\pi)$, is a lower bound on the length of an optimal tour for D. Therefore $w(\pi) = L(T_\pi) - 2\Sigma\pi_i$ is lower bound on the length of an optimal tour for C

The aim is now to find a transformation, $C \rightarrow D$, given by the vector $\pi = (\pi_1, \pi_2, ..., \pi_n)$, that maximizes the lower bound $w(\pi) = L(T_\pi) - 2\Sigma\pi_i$.

If $T\pi$ becomes a tour, then the exact optimum has been found. Otherwise, it appears, at least intuitively, that if $w(\pi) > w(0)$, the values computed from D are better estimates of edges being optimal the values computed from C.

Usually, the maximum of $w(\pi)$ is close to the length of an optimal tour. Computational experience has shown that this maximum typically is less than 1 percent below optimum. However, finding maximum for $w(\pi)$ is not a trivial task. The function is piece-wise linear and concave, and therefore not differentiable everywhere.

A suitable method for maximizing $w(\pi)$ is *sub gradient optimization* (a sub gradient is a generalization of the gradient concept). It is an iterative method in which the maximum is approximated by stepwise changes of π.

At each step π is changed in the direction of the sub gradient, i.e., $\pi_{k+1} = \pi_k + t_k v_k$, where v_k is a sub gradient vector, and t_k is a positive scalar, called the *step size*.

For the actual maximization problem it can be shown that $v_k = d_k - 2$ is a sub gradient vector, where d_k is a vector having as its elements the degrees of the nodes in the current minimum 1-tree. This sub gradient makes the algorithm strive towards obtaining minimum 1-trees with node degrees equal to 2, i.e., minimum 1-trees that are tours. Edges incident to a node with degree 1 are made shorter. Edges incident to a node with degree greater than 2 are made longer. Edges incident to a node with degree 2 are not changed.

The π-values are often called *penalties*. The determination of a (good) set of penalties is called an *ascent*.

Figure shows a sub gradient algorithm for computing an approximation W for the maximum of $w(\pi)$.

1. Let $k = 0$, $\pi_0 = 0$ and $W = -\infty$.
2. Find a minimum 1-tree, $T\pi_k$.
3. Compute $w(\pi_k) = L(T\pi_k) - 2\Sigma\pi_i$.
4. Let $W = \max(W, w(\pi_k))$.
5. Let $v_k = d_k - 2$, where d_k contains the degrees of nodes in $T\pi_k$.
6. If $v_k = 0$ ($T\pi_k$ is an optimal tour), or a stop criterion is satisfied, then stop.
7. Choose a step size, t_k.
8. Let $\pi_{k+1} = \pi_k + t_k v_k$.

9. Let k = k + 1 and go to Step 2.

Example consider the following symmetric matrix below then to find minimum sales man problem

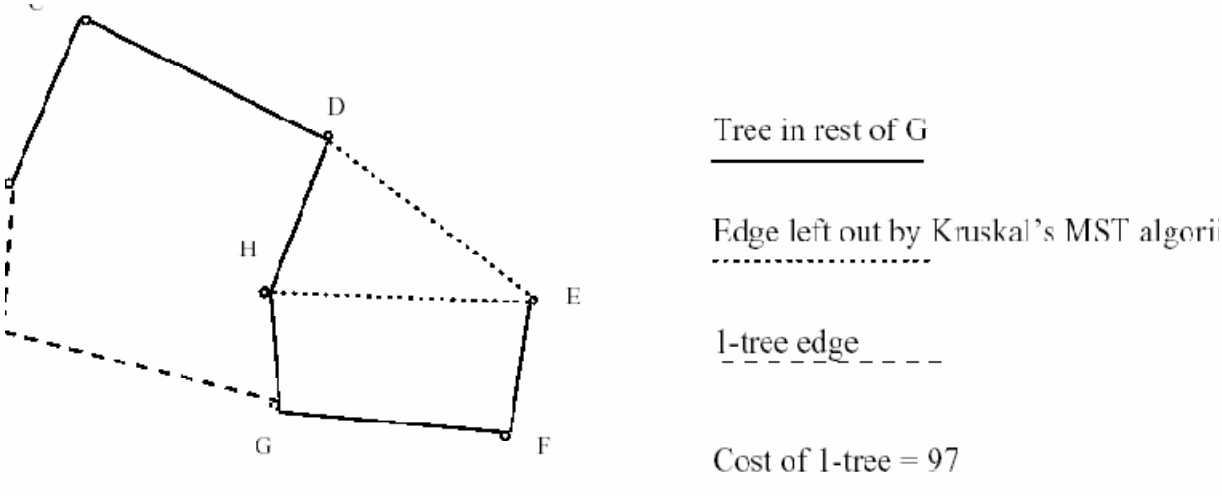
	A	B	C	D	E	F	G	H
A	0	11	24	25	30	29	15	15
B	11	0	13	20	32	37	17	17
C	24	13	0	16	30	39	29	22
D	25	20	16	0	15	23	18	12
E	30	32	30	15	0	9	23	15
F	29	37	39	23	9	0	14	21
G	15	17	29	18	23	14	0	7
H	15	17	22	12	15	21	7	0

First we transformed the cost matrix to get better lower bound

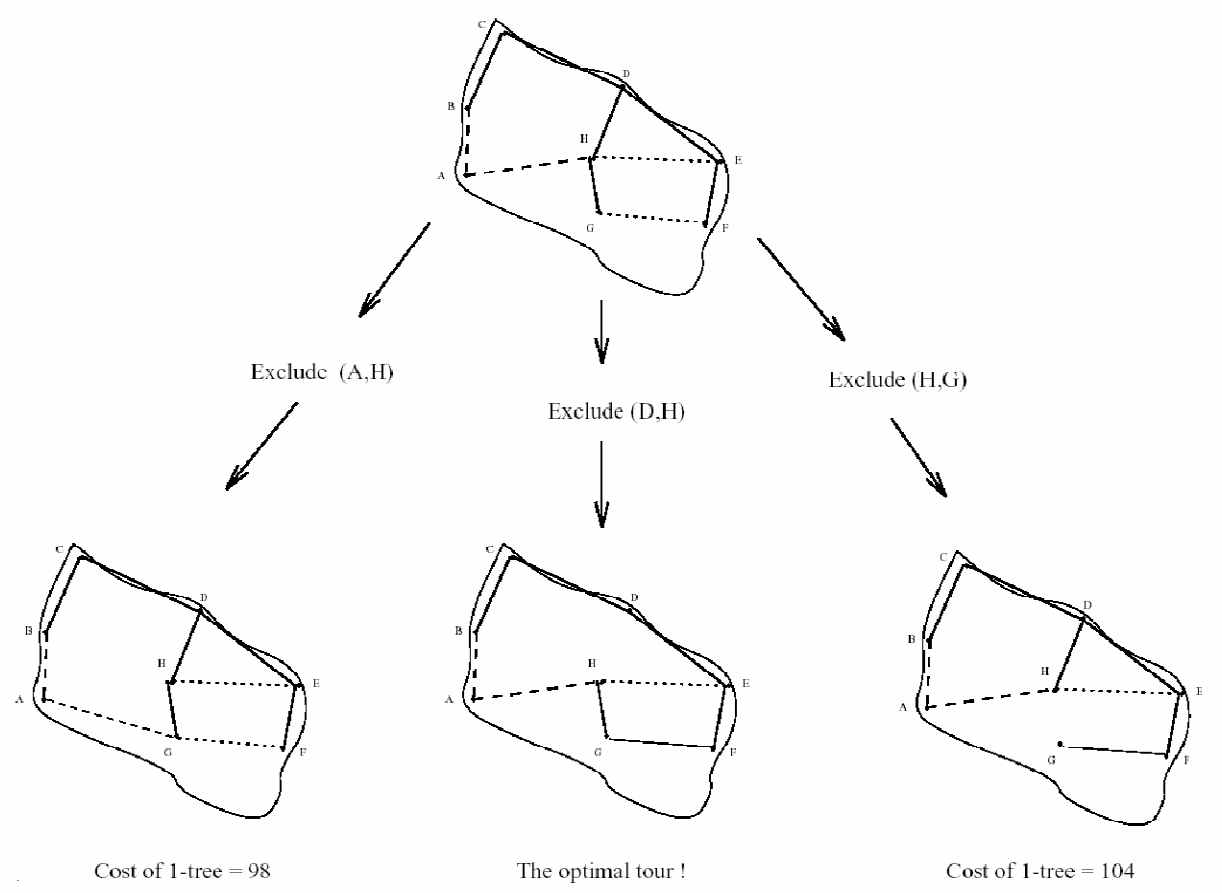
Then we we get the following cost matrix bellow

	A	B	C	D	E	F	G	H
A	0	11	24	25	29	29	16	15
B	11	0	13	20	31	37	18	17
C	24	13	0	16	29	39	30	22
D	25	20	16	0	14	23	19	12
E	29	31	29	14	0	8	23	14
F	29	37	39	23	8	0	15	21
G	16	18	30	19	23	15	0	8
H	15	17	22	12	14	21	8	0

Then solve using minimum one tree using kruskuals algorithm we get the following minimum 1 tree



This process can be repeated to search for a better πso to obtain the large bound as possible. Next it is integrated in a B&B algorithm



Heuristics for TŚPs

Approximate approaches

The *approximate* algorithms obtain good solutions but do not guarantee that optimal solutions will be found. These algorithms are usually very simple and have (relative) short running times. Some of the algorithms give a solution that in average differs only by a few percent from the optimal solution. Therefore, if a small deviation from optimum can be accepted, it may be appropriate to use an approximate algorithm.

The class of approximate algorithms may be subdivided into the following three classes:

• Tour construction algorithms

• Tour improvement algorithms

The *tour construction algorithms* gradually build a tour by adding a new city at each step. The *tour improvement algorithms* improve upon a tour by performing various exchanges. A simple example of a tour construction algorithm is the so-called *nearest neighbor algorithm* Start in an arbitrary city. As long as there are cities, that have not yet been visited, visit the nearest city that still has not appeared in the tour. Finally, return to the first city.

This approach is simple, but often too greedy. The first distances in the construction process are reasonably short, whereas the distances at the end of the process usually will be rather long. A lot of other construction algorithms have been developed to remedy this problem

The tour improvement algorithms, however, have achieved the greatest success.

A simple example of this type of algorithm is the so-called *2-opt algorithm*: Start with a given tour. Replace 2 links of the tour with 2 other links in such a way that the new tour length is shorter. Continue in this way until no more improvements are possible.

Approximation

Solving even moderate size of the TSP optimally takes huge computational time; therefore there is a room for the development and application of approximate algorithms, or heuristics. The approximate approaches never guarantee an optimal solution but gives near optimal solution in a reasonable computational effort. So far, the best known approximate algorithm available is due to (Arora, 1998)..

Tour construction approaches

All tour construction algorithms stops when a solution is found and never tries to improve it. It is believed that tour construction algorithms find solution within 10-15% of optimality. Few of the tour construction algorithms available in published literature are described below.

When using branch and bound methods to solve symmetric travelling salesman problem with many cities a large amount of computer time may be required. For this reason heuristic method which quickly leads a good (but not necessarily optimal solution to a TSP) are often used Heuristics is a method used to solve a problem by trial and error when an algorithmic approach is impractical. Computational speed is not the only reasons for using heuristics.

A fuller set of reasons including this one is as follows.

1) Optimization may require prohibitive amounts of computer time and storage.

2) Heuristic is easier to understand .This may seems a strong reasons but it can make it more acceptable to non-technical users

3) Heuristics can often be incorporated within optimization procedure (example branch and bound) to good effect. Heuristic algorithms should be used when: the problem dimensions exceed the maximum dimensions which can be dealt with exact approaches in acceptable computation times the problem, although of limited dimensions, must be solved in a very short computation time the problem data are approximated (seeking for optimality is not worthy) there are problem variations for which exact approach cannot be easily extended the problem is dynamic We now discuss four heuristics for travelling salesman problem.

Nearest –Neighbor heuristics-to apply the nearest neighbor's heuristics we begin at any cities and then visit the nearest city. Then we go to the unvisited city closest to the city we most recently visited. Continue in this fashion until a tour is obtained. After applying this procedure beginning at each city, we take the best tour found.

The steps of the nearest neighbor's heuristics are as follows

1 selects a random city

2 find the nearest unvisited city and go there.

3 are there any unvisited city left if yes repeat step 2

4 return to the first city

Cheapest –insertion heuristics-we begin at any city and find its closest neighbor. Then we create a sub tour joining those two cities. Next we replace an arc in the sub tour [say arc $(i,$ j)] by combination of two arcs$-(i,k)$ and(k,j) where k is not in the current sub tour –that will increase the length of sub tour by the smallest (cheapest) amount. Let c_{ij} be the length of arc (i,j).Note that if arc (i ,j)is replaced by arcs (i ,k) and (k,j),then a length a $c_{ik+}c_{kj-}c_{ij}$ is added to the sub tour.Then we continue with this procedure until a tour is obtained .After applying this procedure beginning with each city we take the best tour found.

Steps of an insertion heuristics are select the shortest edge and make a sub tour

1Selects a city not in the sub tour having the shortest distance to any one of the cities in the sub tour

2 Find an edge, in the sub tour such that the cost of inserting the selected city between the edges cities will be minimal

3 Repeat step 2 until no more cities remain

Greedy heuristics-The greedy heuristic gradually constructs a tour by repeatedly selecting the shortest edge and adding it to the tour as long as it does not create a cycle with less than N edges or increase the degree of any node to more than 2.

We must not add the same edge twice of course

Steps of greedy algorithms are

1 Sort all edges

2 Select the shortest edge and add it to our tour if it does not violate any of the above constraint

3 Do we have N edges in our tour? If no repeat step

.

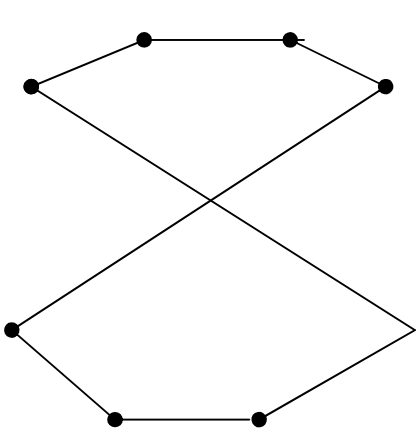
Tour improvement

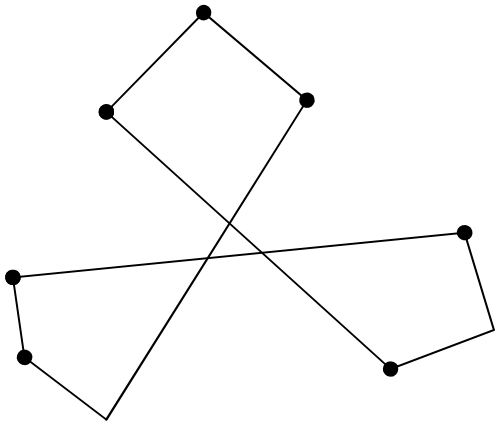
After generating the tour using any tour construction heuristic, an improvement heuristic can be further applied to improve the quality of the tour generated. Popularly, 2-opt and 3-optexchange heuristic is applied for improving the solution. The performance of 2-opt or 3-optheuristic basically depends on the tour generated by the tour construction heuristic. Other ways of improving the solution is to apply meta-heuristic approaches such as tabu search orsimulated annealing using 2-opt and 3-opt.

2-opt and 3-opt.

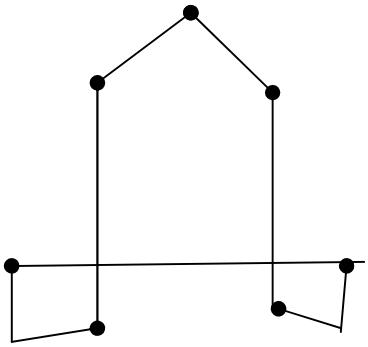
The 2-opt algorithm removes randomly two edges from the already generated tour, and reconnects the new two paths created. This is referred as a 2-opt move. The reconnecting is done such a way to keep the tour valid (see figure (a)). This is done only if the new tour is shorter than older. This is continued till no further improvement is possible. The resulting tour is now 2 optimal. The 3-opt algorithm works in a similar fashion, but instead of removing the two edges it removes three edges. This means there are two ways of reconnecting the three paths into a valid tour (see figure (b) and figure (c)). Search is completed when no more 3-opt moves can improve the tour quality. If a tour is 3 optimal it is also 2 optimal (Helsgaun). Running the 2-opt move often results in a tour with a length less than 5% above the Held-Karp bound. The improvements of a 3-opt move usually generates a tour about 3% above the Held-Karp bound (Johnson &McGeoch, 1995). Pt and 3-opt

The fig below shows a 2-opt move and a 3- opt move





(a) **(b)**



(c)

k-opt

In order to improve the already generated tour from tour construction heuristic, k-opt move can be applied (2-opt and 3-opt are special cases of k-opt exchange heuristic) but exchange heuristic having k>3 will take more computational time. Mainly one 4-opt move is used, called “the crossing bridges” (see Figure 2). This particular move cannot be sequentially constructed using 2-opt moves. For this to be possible two of these moves would have to be illegal (Helsgaun).

Lin-Kernighan

Lin & Kernighan constructed an algorithm making it possible to get within 2% of the Held-Karp lower bound. The Lin-Kernighan heuristic (LK) is a variable k-way exchange heuristic. It decides the value of suitable k at each iteration. This makes an improvement heuristic quite complex, and few have been able to make improvements to it. The time complexity of LK is approximately ( ) 2.2 On (Helsgaun), making it slower than a simple 2-opt implementation.

The Held-Karp lower bound

this lower bound if the common way of testing the performance of any new TSP heuristic. Held-Karp (HK) bound is actually a solution to the linear programming relaxation of the integer formulation of TSP (Johnson et al. 1996). A HK lower bound averages about 0.8% below the optimal tour length (Johnson et al., 1996). For more details regarding the HK lower bound, paper by (Johnson et al., 1996) can be referred.

We can also solve the previous examples using Heuristics

In the cheapest –insertion heuristics we begin at any city and find it its closest neighbor .then we create a sub tour joining those two cities. Next we replace an arcin the sub tour [say arc (i, j)] by combination of two arcs$-(i,k)$ and$(k,j)$where k is not in the current sub tour –that will increase the length of sub tour by the smallest (cheapest) amount. Letc_{ij} be the length of arc (i,j).Note that if arc (i,j)is replaced by arcs (i,k) and (k,j),then a length a $c_{ik+}c_{kj-}c_{ij}$ is added to the sub tour .Then we continue with this procedure until a tour is obtained .Suppose we begin the CIH at city 1 City 5 is closest to city 1,so we begin with the sub tour (1,5)-(5,1).Then we could replace arc (1,5) by (1,2) -(2,5) ,(1,3)-(3,5) or(1,4)-(4,5).we could also replace arc(5,1) by (5,2)- (2,1) ,(5,3) -(3,1) or,(5,4) -(4,1) .The calculations used to determine which arc of (1,5) -(5,1) should be replaced are given in table 7(* indicates the correct replacement) As seen in the table we may replace either (1,5) or (5,1).We arbitrary choose to replace arc (1,5) by arcs (1,2) and(2,5).We currently have the sub tour (1, 2)-(2, 5)-(5, 1). We must know replace an arc (i, j) of this sub tour by the arcs (i, k) and (k, j), where k = 3 or 4 as shown in (Table 8). We now replace (1, 2) by arcs (1, 4) and (4, 2). This yields the sub tour (1, 4)-(4, 2)-(2, 5)-(5, 1). An arc (i, j) in this sub tour must now be replaced by arcs (i, 3) and (3, j) as shown in (Table 9). We now replace arc (1, 4) by (1, 3) and (3, 4). This yields the tour (1,3) -(3,4) -(4,2) -(2,5) -(5,1). In this example the cheapest-insertion heuristics yields an optimal tour –but in general the cheapest –insertion heuristic does not necessarily do so.

Table 7

Determining which arc of (1, 5) - (5, 1) is replaced

Arc replaced	Arc Added to sub tour	Added Length
(1, 5)*	(1, 2) - (2, 5)	$c_{12+}c_{25-}c_{15=153}$
(1, 5)	(1, 3) - (3, 5)	$c_{13+}c_{35-}c_{15=462}$
(1, 5)	(1, 4) - (4, 5)	$c_{14+}c_{45-}c_{15=302}$
(5, 1)*	(5, 2) - (2, 1)	$c_{52+}c_{21-}c_{51=153}$
(5, 1)	(5, 3) - (3, 1)	$c_{53+}c_{31-}c_{51=462}$
(5, 1)	(5, 4) - (4, 1)	$c_{54+}c_{41-}c_{51=302}$

Table 8

Determining which arc of (1, 2) - (2, 5) – (5, 1) is replaced

Arc replaced	Arc Added	Added Length
(1, 2)	(1, 3) - (3, 2)	$c_{13+}c_{32-}c_{12=375}$
(1, 2)*	(1, 4) - (4, 2)	$c_{14+}c_{42-}c_{12=233}$
(2, 5)	(2, 3) - (3, 5)	$c_{23+}c_{35-}c_{25=514}$
(2, 5)	(2, 4) - (4, 5)	$c_{24+}c_{45-}c_{25=318}$
(5, 1)	(5, 3) - (3, 1)	$c_{53+}c_{31-}c_{51=462}$
(5, 1)	(5, 4) - (4, 1)	$c_{54+}c_{41-}c_{51=302}$

Table 9

Determining which arc of (1, 4)- (4,2) –(2,5) –(5,1) is replaced

Arc replaced	Arc Added	Added Length
(1, 4)*	(1, 3) - (3,4)	$c_{13+}c_{34-}c_{14=166}$
(4, 2)	(4, 3) - (3, 2)	$c_{43+}c_{32-}c_{42=202}$
(2, 5)	(2, 3) - (3, 5)	$c_{23+}c_{35-}c_{25=514}$
(5, 1)	(5, 3) - (3, 1)	$c_{53+}c_{31-}c_{51=462}$

Example; A salesman travels from city A to nine other cities. The distance is shown below and for i≠j,$w_{ij=}w_{ji}$ then what order of visiting the cities minimizes his distance?

City	B	C	D	E	F	G	H	I	J
1 (A)	96	105	50	41	86	46	29	56	70
2 (B)		78	49	94	21	64	63	41	37
3 (C)			60	84	61	54	86	76	51
4 (D)				45	35	20	26	17	18
5 (E)					80	36	55	59	64
6 (F)						46	50	28	8
7 (G)							45	37	30
8 (H)								21	45
9 (I)									25
10(J)									

Then the above problem can be solved by heuristics as follows

1) - Nearest Neighbor starting at city A. This yields the tour (1, 8 9, 4, 7, 10, 6, 2, 3, 5, 1) of distance 29+21+17+20+30+8+ 21+ 18+ 84+ 41 = 349

2)- Greedy feasible-this yields the edges set (6, 10), (4, 9), (4,10), (2, 6), (8, 9), (1, 8), (5, 7), (1, 5), (3, 7), (2, 3) and the tour (1, 8, 9, 4, 10, 6, 2, 3, 7, 5, 1) of distance 323.

3)- Nearest insertion beginning with the triangle (4,9,10,4) The successive sub tour s are (4, 9, 6, 10, 4), (4, 9, 6, 10, 7, 4), (4, 9, 6, 2, 10, 7, 4), (4, 8, 9, 6, 2, 10, 7, 4), (4, 1, 8, 9, 6, 2, 10, 7, 4) the resulting tour (1, 8, 9, 6, 2, 10, 3, 7, 5, 4, 1) has weight 372.

4) - 2-Interchange beginning with the tour produced by nearest insertion. We find the following sequence Improving tours (1, 8, 9, 2, 6, 10, 3, 7, 5, 4, 1) of weight 353 (1, 8, 9, 2, 6, 10, 3, 7, 4, 5, 1) of weight 328 and (1, 8, 9, 2, 6, 10, 3, 4, 7, 5, 1) of weight 325.

Evaluation of Heuristics

The following three methods have been suggested for evaluating heuristics

Performance guaranties

Probabilistic analysis

Empirical analysis

A performance guarantee for a heuristics

Gives a worst-case bound on how far away from optimality a tour constructed by the heuristics can be. For the NNH; it can be shown that for any number r aTSP can be constructed such that the NHH yields a tour that is r times as large as the optimal tour. Thus in a worst-case scenario, the NHH fare poorly. For symmetricTSP satisfying the triangle inequality that is for $c_{ij=}c_{ji}$and$c_{ik} < c_{ij+}c_{jk}$ for all, $i\ j, k$ it has been shown that the length of the optimal tour obtained by the cheapest –insertion heuristic cannot exceed twice the length of the optimal tour.

In probabilistic analysis a heuristic is evaluated by assuming that the location of cities follows some known probability distribution. For example we might assume that the cities are

independent random variables that are uniformly distributed on a cube of unit length width and height. Then for each heuristics, we would compute the following ratio.

Expected length of the path found by herustic divides by expected length of an optimal tour

The closes the ratio is to 1, the better the heuristic.

For empirical analysis, Heuristics are compared to the optimal solution for a number of problems for which the optimal tour is known. As an illustration, for five 100-city TSPS Golden, Badin, Doyle and Stewart (1980) found that nearest-neighbor heuristic-taking the best of all solutions found when the nearest-neighbor heuristic was applied beginning at each city produced tours that average 15% longer than the optimal tour. For the same set of problems, it was found that the cheapest-insertion heuristic (again applying the best solution to obtained by applying cheapest-insertion heuristic to all cities) produced tour that also average 15% longer than the optimal tour.

1.6 Alternatives approach for solving symmetric travelling sales man problem

1.6.1Dynamic programming approach

Richard Bellman invented the technique of dynamic programming in 1950 as, a general method for optimizing multistage decision process.

The word programming here stands for planning .It is often possible to divide an instance of problem in to sub instances to solve these sub instances and then combine the solution of the sub instances to solve the original instance. The natural way of dividing an instance may lead us to consider several over lapping sub instances

If we solve each of these sub instances independently they will in turn create a large number of identical sub instances.

It is likely that we will be duplicating certain pieces of computation resulting in an inefficient algorithm .If on the other hand we take advantages of the duplication and solve each sub instances only once solving the solution for latter use then a more efficient algorithm will result. The underlining idea of dynamic programming is avoiding calculating the same thing twice usually by keeping the table of known result which we fill up as the sub instances are solved.

Dynamic programming is- bottom -up technique we usually start with the smallest and hence the simplest sub instances by combing there solutions we obtain answers to sub instances of increasing size until we arrive at the solution of the original instances. So this symmetric travelling sales man problem can be solved by this technique. In short dynamic programming is a technique that can be used to solve many optimization problems.

In most application Dynamic programming obtains solutions by working back ward recursion, this breaking up a large un widely problem in to a series of smaller more tractable problems.

References

1. GEORGE L. NEMHAUSER and LAURENCE A.WOLSEY, Integer and Combinatorial optimization

2. WAYNEL .WINSTON, Operation Research Applications and Algorithms, Fourth Edition

3. K.V.MITAL .Optimization methods in Operation Research and system analysis, second Edition

4. C.K.MUSTAFI, Operations research, second Edition

5. G.V.SHENY, Linear Programming Methods and application

6. P.K.GUPTA, MAN MOHAN, Operations Research and statically Analysis.

7. ABRAHAM KANDEL and THEODORE P. BAKER .Discrete Mathematics for Computer Scientists and Mathematicians, Second edition

8. S.K BASU.Design Methods and Analysis of Algorithms.

9. MARSHALL.FISHER, Management science1981, volume 27, No.1 page 1-18

10. G .DANTIZING, D.FULKERSON,and S.JOHNSON .solution of large scale travelling salesman problem .Operation Reasearch2:393-410, 1954.11. ABRAHAM P.PUNNEN and Y.PANEJA, The journal of the Operation Research Society, 112: Held, M., & Karp, R., "The Travelling-Salesman Problem and Minimum Spanning Trees" *Ops Research*, v.18,