Combinatorial Optimization: Solution Methods of Traveling Salesman Problem

Hülya Demez

Submitted to the
Institute of Graduate Studies and Research
in partial fulfillment of the requirements for the Degree of

Master of Science
in
Applied Mathematics and Computer Science

Eastern Mediterranean University
January 2013
Gazimağusa, North Cyprus

Approval of the Institute of Graduate Studies and Research

Prof. Dr. Elvan Yılmaz
Director

I certify that this thesis satisfies the requirements as a thesis for the degree of Master of Science in Applied Mathematics and Computer Science.

Prof. Dr. Nazım Mahmudov
Chair, Department of Mathematics

We certify that we have read this thesis and that in our opinion it is fully adequate in scope and quality as a thesis for the degree of Master of Science in Applied Mathematics and Computer Science.

Asst. Prof. Dr. Arif Akkeleş
Supervisor

Examining Committee

1. Asst. Prof. Dr. Arif Akkeleş
2. Asst. Prof. Dr. Mehmet Bozer
3. Asst. Prof. Dr. Müge Saadetoğlu

ABSTRACT

Traveling Salesman Problem has been one of the most interesting and challenging problem in the literature. It is include a large area in combinatorial optimization problem. A variety of Exact and Heuristic Algorithms are usable algorithms for solving TSP. Branch and Bound Algorithm is an exact algorithm that is developed for solving TSP type problems. Furthermore, Genetic Algorithm is one of the extensively algorithm within the Heuristic Algorithm. In this work, we looked into symmetric and asymmetric matrices to solve TSP. We used Genetic and Branch-and-Bound Algorithms as the solution methods to get the shortest path.

Keywords: Traveling Salesman Problem, Heuristic Algorithm, Exact Algorithm, Branch and Bound Algorithm, Genetic Algorithm

ÖZ

Gezgin Satıcı Problemi, literatürdeki en ilginç ve en iddealı problem olarak çalışılan, kombinasyonel eniyileme problemlerinin başında gelmektedir. Çözümü için birçok Sezgisel ve Kesin Çözüm Yöntemleri geliştirilmektedir. Dal ve Sınır Algoritmaları, gezgin satıcı ve benzer yapıdaki problemlerin çözümü için geliştirilen Kesin Çözüm Yöntemi olmakla birlikte, Genetik Algoritmalar da Sezgisel Yöntemlerin başında gelmektedir. Bu çalışmada Gezgin Satıcı Problemlerinin çözümü için simetrik ve asimetrik matrisler ele alınmıştır. En kısa turları elde etmek için de Dal ve Sınır ve Genetik Algoritmaları kullanılmaktadır.

Anahtar Kelimeler: Gezgin Satıcı Problemi, Sezgisel Yöntem, Kesin Çözüm Yöntemi, Dal ve Sınır Algoritması, Genetik Algoritma

To My Mother

ACKNOWLEDGMENT

I am heartily thankful to my dear supervisor, Asst. Prof. Dr. Arif Akkeleş for his unlimited patience, guidance throughout this dissertation. Without the time and energy that he brought, this thesis would not have been possible. I think, I owe him so much. So, thank you very much sir!

Secondly, I would like to express my sincere gratitude towards Asst. Prof. Dr. Müge Saadetoğlu, Dr. Ersin Kuset, Asst. Prof. Dr. Mehmet Bozer and Dr. Fatoş Bayramoğlu Rızaner for their valuable support and advices.

Special thanks are extended to my lovely friends who are Cemaliye Kürt, Doğukan Demir. In addition, thanks to my friend İlke Çetin, Meral Selimi, Simruy Hürol and Mustafa Hasanbulli to be with me.

Finally, my deepest gratitude goes to the Department of Mathematics, all my teachers, friends and my dear family for their understanding and endless love for his continuous help, valuable suggestions and encouragement during preparation of this dissertation.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

AP: Assignment Problem

ATSP: Asymmetric Traveling Problem

FP: Function Problems

GA: Genetic Algorithm

LB: Lower Bound

QAP: Quadratic Assignment Problem

STSP: Symmetric Traveling Salesman Problem

TSP: Traveling Salesman Problem

UB: Upper Bound

GSP: Gezgin Satıcı Problemi

Chapter 1

INTRODUCTION

Traveling Salesman Problem has been of the great interest for many years. It plays an important role in different solving algorithms such as Branch and Bound and Genetic Algorithm. In hand, it has a simple modeling, on the other hand it has a difficult solution way. Therefore, users should know the type of algorithms and their solution ways very well to overcome of these problems. In this sense, the thesis includes the solution algorithms of the TSP.

In Chapter 2, we search the progress of the TSP, and also some topics are collected with the relevant definitions and main information about it. Throughout this chapter, we point out the mathematical formulation and an example of TSP.

In the next chapter, we consider the Branch & Bound Algorithm which is the exact algorithm of TSP. The goal of this chapter is giving definitions, rules and how does it work on the examples.

In Chapter 4, we take the Genetic Algorithm that is the another solving algorithm of the TSP. In general, the procedure of genetic algorithm is explained with the related definitions and examples as well.

In the rest of the chapter, we talk about the hold results of this study and we give suggestions for the future studies.

Chapter 2

THE TRAVELING SALESMAN PROBLEM

2.1 History

Before starting to give information about the TSP, I would like to state a brief history of the TSP.

The story of TSP is starting with Euler. TSP emerged from his studies "Studied the Knight's tour Problem" in 1766. After Euler, the important well-known study was carried out in 18th century by the mathematicians Sir William Rowaw Hamilton and Thomas Penyngton Kirkman from Ireland and Britain respectively [1]. It was about finding paths and circuits on the dodecahedral graph, satisfying certain conditions [2]. Most of research into TSP history as a whole was done in the period from 1800 to 1900. We can meet Kirkman's, Menger's or Tucker's studies in ancient history as well. Then, Lawler, Lenstr, Rinnoy Kan and Shmoys didn`t say anything for TSP. Karl Menger give attention to his colleagues in 1920. After that, Merrill Meeks Flood submitted result related with TSP in 1940. It was the year 1948 when Flood publicized the travelling salesman problem by presenting it at the RAND Corporation. The RAND Corporation is a non-profit organization that is the focus of intellectual research and development within the United States. In its early days, RAND provided research and analysis to the United States armed forces, but then expanded to provide such services for the

government and other organizations [3]. After those years the TSP became more popular. This popularity was probably caused by a few factors, one of which is the prestige of the RANK Corporation [3]. The Linear Programming was investigated as a vital force in computing solutions to combinatorial optimization problems in 1950. As mentioned, this is one of the reasons why the TSP was in the interest of RAND[1]. Later on Dantzig, Fulkerson, and Johnson find a method for solving the TSP in 1950. They proved the effectiveness of their method by solving a 49-city instance. However, it became evident, as early as the mid 1960's, that the general instance of the TSP could not be solved in polynomial time using Linear Programming techniques. Finally, these categories of problem became known as NP-hard[2] [3]. Great progress was made in the late 1970's and 1980, when Grötschen, Padberg, Rinaldi and others managed to exactly solve instance with up to 2392 cities, using cutting planes and branch-and-bound [4].

2.2 Definition

Traveling Salesman Problem is an extremely important problem in operational research. We first define the problem and then we study the methods and algorithms to solve the TSP.

---

[1] Rand is a function which can generate a random number between 0 and 1.

[2] For any problem P is NP-Hard if a polynomial time algorithm for P would imply a polynomial-time algorithm for every problem in NP.

TSP or Hamiltonian tour is a type of classic and problem which shows one from to solve the more complex ones. Hamiltonian tour starts with a given edge, visits each of the specific groups of edges and then returns to the original point of departure. TSP is the shortest walk in a circuit provided that it passes from all vertices only once.

In mathematical formulation, there is a group of distinct cities $\{V_1, V_2, \dots, V_N\}$, and there is a corresponding edge for each pair of cities $\{V_i, V_j\}$ and a closed path $V_\alpha = \{V_{\alpha(1)}, V_{\alpha(2)}, \dots, V_{\alpha(N)}\}$. The objective then is to find an ordering α of cities such that the total time for the salesman is minimized. The lowest possible time is called the optimal time. The objective function is given as:

$$\sum_{i=1}^{N-1} d\left(V_{\alpha(i)}, V_{\alpha(i+1)}\right) + d\left(V_{\alpha(N)}, v_{\alpha(1)}\right)$$

[5].

In other words, TSP of NP-Hard problem class is known as one of the well known combinatorial optimization problems. This means for TSP, the solution techniques have not been improved in polynomial time. Thats why to solve TSP, there are many intuitive techniques. More precisely, it is complete for the complexity class (FP^{NP})[3], and the

[3] The complexity class NP is the set of decision problems that can be verified in polynomial time.

decision problem version is NP-complete. If an efficient algorithm is found for the TSP problem, then efficient algorithms could be found for all other problems in the NP-complete class. Although it has been shown that, theoretically, the Euclidean TSP is equally hard with respect to the general TSP, it is known that there exists a sub exponential time algorithm for it. The most direct solution for a TSP problem would be to calculate the number of different tours through V cities. Given a starting city, it has $V-1$ choices for the second city, $V-2$ choices for the third city, etc. Multiplying these together one gets $(V-1)!$ for one city and $V!$ for the V cities. Another solution is to try all the permutations (ordered combinations) and see which one is cheapest. At the end, the order is also factorial of the number of cities. Briefly, the solutions which appear in the literature are quite similar. The factorial algorithm's complexity motivated the research in two attack lines: exact algorithms or heuristics algorithms. The exact algorithms search for an optimal solution through the use of branch-and-bound, linear programming or branch-and-bound plus cut based on linear programming techniques. Heuristic solutions are approximation algorithms that reach an approximate solution (close to the optimal) in a time fraction of the exact algorithm. TSP heuristic algorithms might be based on genetic and evolutionary algorithms, simulated annealing, Tabu search, neural network, ant system, among some states in [6].

The reason of attention on TSP is that it has a wide range of applications area. Problems like, telecommunication networks, circuit board designing, logistic and many others can

be modeled by the TSPs. Furthermore, the TSP can be used to formulate other combinatorial optimization problems such as QAP[4].

The aim of TSP is to find the shortest tour that visits all the vertices exactly once. At the same time, the main purpose of the TSP is to minimize the total weight in TSP tour.

We can categorize TSP in two classes: Symmetric TSP and Asymmetric TSP. In STSP, the distance between two edges is always equal in opposite orientation. For ATSP, this condition may not be satisfied for all direction. Sometimes another factor like different distance or different departure time will affect our problem.

2.2.1 Mathematical Formulation of TSP

The TSP can be defined on a complete undirected graph $G = (V, E)$ if it is symmetric or on a directed graph $G = (V, A)$ if it is asymmetric. The set $V = \{1, \dots, n\}$ is the vertex set, $E = \{(i, j): i, j \in V, i < j\}$ is an edge set and $E = \{(i, j): i, j \in V, i \neq j\}$ is an arc set. A cost matrix $C = \left(c_{ij}\right)$ is defined on E or on A. The cost matrix satisfies the triangle inequality whenever $c_{ij} \leq c_{ik} + c_{kj}$, for all i, j, k. In particular, this is the case of a planar problem for which the vertices are points $P_i = (X_i, Y_i)$ in the plane, and $c_{ij} =$

[4] QAP is one of the hardest combinatorial optimization problem and also it is in the class of NP-Hard Problems. The aim of this problem is to minimize the total weighted cost.

$\sqrt{(X_i - X_j)^2 + (Y_i - Y_j)^2}$ is the Euclidean distance. The triangle inequality is also satisfied if c_{ij} is the length of the shortest path from i to j on G [1].

2.3 Symmetric Travelling Salesman Problem

Let $V = \{v_1, \dots, v_2\}$ be a set of cities, $A = \{(r, s): r, s \in V\}$ be the edge set, and $d_{rs} = d_{sr}$ be a cost measure associated with edge $(r, s) \in A$.

The STSP is the problem of finding a closed tour of minimal length that visits each city once. In the case where cities $v_i \in V$ are given by their coordinates (x_i, y_i) and d_{rs} is the Euclidean distance between r and s, we have an Euclidean TSP [1].

2.3.1 Integer Programming Formulation of STSP

This formulation associates a binary variable x_{ij} with each edge (i, j), which is equal to 1 if and only if the latter appears in the optimal tour. The formulation of TSP is as follows.

Minimize

$$\sum_{i<j} c_{ij} x_{ij} \qquad (2.1)$$

Subject to

$$\sum_{i<k} x_{ik} + \sum_{j>k} x_{kj} = 2 \qquad (k \in V) \qquad (2.2)$$

$$\sum_{i,j\in S} x_{ij} \leq |S| - 1 \quad (S \subset V, 3 \leq |S| \leq n-3) \tag{2.3}$$

$$x_{ij} = 0 \; or \; 1 \qquad (i,j) \in E \tag{2.4}$$

In this formulation, constraints (2.2), (2.3) and (2.4) are referred to as degree constraints, sub-tour elimination constraints and integrality constraints, respectively. In the presence of (2.2), constraint (2.3) is algebraically equivalent to the connectivity constraint

$$\sum_{i\in S, j\in V\setminus S, j\in S} x_{ij} \geq 2 \quad (S \subset V, 3 \leq |S| \leq n-3).$$

[1].

2.4 Asymmetric Travelling Salesman Problem

If $d_{rs} \neq d_{sr}$ for at least one (r,s), then the TSP becomes an ATSP [1].

2.4.1 Integer Programming Formulation of ATSP

Here x_{ij} is a binary variable, associated with arc (i,j) and equals to 1 if and only if the arc appears in the optimal tour. The formulation is as follows.

Minimize

$$\sum_{i\neq j} c_{ij} x_{ij}$$

Subject to

$$\sum_{j=1}^{n} x_{ij} = 1 \qquad (i \in V, i \neq j)$$

$$\sum_{i=1}^{n} x_{ij} = 1 \qquad (j \in V, j \neq i)$$

$$\sum_{i,j \in S} x_{ij} \leq |S| - 1 \quad (S \subset V, 2 \leq |S| \leq n - 2)$$

$$x_{ij} = 0 \; or \; 1 \qquad (i,j) \in A \qquad [1].$$

2.5 Example: Travelling Salesman Problem

In this section, branch and bound algorithm will be showed as a solution algorithm of the TSP in the following numerical example. I used to basic method to solve this example and throughout the next chapter; the technique of implicit enumeration[5] will be discussed on a couple example.

Case 1: Let us consider the symmetric matrix,

[5] Implicit enumeration can be applied within the family of optimization problems mainly if all variables have discrete nature.

	x_1	x_2	x_3	x_4	x_5
x_1	∞	10	8	9	7
x_2	10	∞	10	5	6
x_3	8	10	∞	8	9
x_4	9	5	8	∞	6
x_5	7	6	9	6	∞

Step 1: The lower bound of total distance is

$$LB = \sum_{i=1}^{5} \min d(x_i) = 7 + 5 + 8 + 5 + 6 = 31$$

and the tour has to travel at least 31. Optimal solution will have to be more than or equal to the lower bound of 31.

Step 2: We create $(n-1) = (5-1) = 4$ branches and say $x_{12} = 1, x_{13} = 1, x_{14} = 1,\ x_{15} = 1$. We do not have $x_{11} = 1$ becouse x_{11} is a subtour of length 1. The same thing holds for all x_{ii} which means x_{11}, x_{22}, x_{33}, x_{44} and x_{55} are all sub-tours.

Step 3: Fixing x_1 and x_2, by leaving out the first row and the second column to get additional minimum distance, we obtain

	x_1	x_3	x_4	x_5
x_2	10	10	5	6
x_3	8	∞	8	9
x_4	9	8	∞	6
x_5	7	9	6	∞

The minimum travelling is

$$x_1 \to x_2,\ x_2 \to x_4,\ x_3 \to x_1 \text{ or } x_4,\ x_4 \to x_5,\ x_5 \to x_4.$$

Then the lower bound is

$$10 + 5 + 8 + 6 + 6 = 35.$$

Repeating the same calculation, x_1 to x_3 will be $8 + 5 + 6 + 5 + 6 = 32$, x_1 to x_4 will be $9 + 6 + 8 + 5 + 6 = 34$ and x_1 to x_5 will be $7 + 5 + 8 + 5 + 6 = 31$.

Step 4: We want to minimize the total distance travelling so we branch further from 31 because it has the minimum value of lower bound. Therefore, we have to create three more branches as $x_{21} = 1, x_{23} = 1, x_{24} = 1$.

Step 5: We will find out the new lower bound. Now we will free x_1 to x_5 and x_2 to x_1 then we calculate the minimum travelling value. Here x_5 to x_2 is a sub-tour so we will choose it as ∞.

	x_2	x_3	x_4
x_3	10	∞	8
x_4	5	8	∞
x_5	∞	9	6

The minimum travelling is

$$x_1 \to x_5, \quad x_2 \to x_1, \quad x_3 \to x_4, \quad x_4 \to x_2, \quad x_5 \to x_4.$$

Then the lower bound is

$$7 + 10 + 8 + 5 + 6 = 36.$$

Repeating the same calculation, x_2 to x_3 will be $7 + 10 + 8 + 5 + 6 = 36$, x_2 to x_4 will be $7 + 5 + 8 + 8 + 6 = 34$.

Step 6: Evaluate up to this point, find the minimum value and then draw new branches. In the solution 32 is the new lower bound so spread in branches from this node. We create three new branches here like $x_{21} = 1, x_{24} = 1, x_{25} = 1$.

Step 7: Now we will go back to find out the new lower bound. For the first one we put $x_1 \to x_3$ and $x_2 \to x_1$. When $x_1 \to x_3$ and $x_2 \to x_1$ go, $x_3 \to x_2$ will be sub-tour.

	x_2	x_4	x_5
x_3	∞	8	9
x_4	5	∞	6
x_5	6	6	∞

The minimum travelling is

$$x_1 \to x_3,\ x_2 \to x_1,\ x_3 \to x_4,\ x_4 \to x_2,\ x_5 \to x_2 \text{ or } x_4.$$

Then the lower bound is

$$8 + 10 + 8 + 5 + 6 = 37.$$

As the same calculation, x_2 to x_4 will be $8 + 5 + 8 + 6 + 7 = 33$, x_2 to x_5 will be $8 + 6 + 8 + 5 + 6 = 33$.

Step 8: Out of these values we will try again to find the one with the smallest in term. We will branch from either x_2 to x_4 or x_2 to x_5. Let us continue from x_2 to x_4, we get $x_{32} = 1$ and $x_{35} = 1$ branches. Again x_{31} and x_{34} are sub-tours.

Step 9: We fix three terms and when we have only two more terms to go, we don't look for the lower bound but the upper one. This upper bound becomes the feasible solution. Therefore the way of feasible solution for x_3 to x_2 is

$$x_1 \to x_3,\ x_3 \to x_2,\ x_2 \to x_4,\ x_4 \to x_5,\ x_5 \to x_1.$$

Then the feasible solution is

$$z = 8 + 10 + 5 + 6 + 7 = 36.$$

When we get the first feasible solution for this branch and bound, we will check the other solution. The lower bound must be less than the upper bound. Therefore only then we can continue from x_3 to x_5 and x_2 to x_5.

The way of feasible solution for x_3 to x_5 is

$$x_1 \to x_3,\ x_3 \to x_5,\ x_5 \to x_2,\ x_2 \to x_4,\ x_4 \to x_1.$$

Then the feasible solution is

$$z = 8 + 9 + 6 + 5 + 9 = 37.$$

Step 10: Another possible branch solution for x_2 to x_5 will be $x_{32} = 1$ and $x_{34} = 1$.

The way of feasible solution for x_3 to x_2 is

$$x_1 \to x_3,\ x_3 \to x_2,\ x_2 \to x_5,\ x_5 \to x_4,\ x_4 \to x_1.$$

Then the feasible solution is

$$z = 8 + 10 + 6 + 6 + 9 = 39.$$

The way of feasible solution for x_3 to x_4 is

$$x_1 \to x_3,\ x_3 \to x_4,\ x_4 \to x_2,\ x_2 \to x_5,\ x_5 \to x_1.$$

Then the feasible solution is

$$z = 8 + 8 + 5 + 6 + 7 = 34.$$

Answer: All the nodes move around so we have four feasible solutions for this question.

The best optimal solution is

$$x_1 \to x_3 \to x_4 \to x_2 \to x_5 \to x_1$$

which is the 34.

Note: We will have the same solution for

$$x_1 \to x_5 \to x_2 \to x_4 \to x_3 \to x_1.$$

It gives us the same value 34 because we have the symmetric matrix.

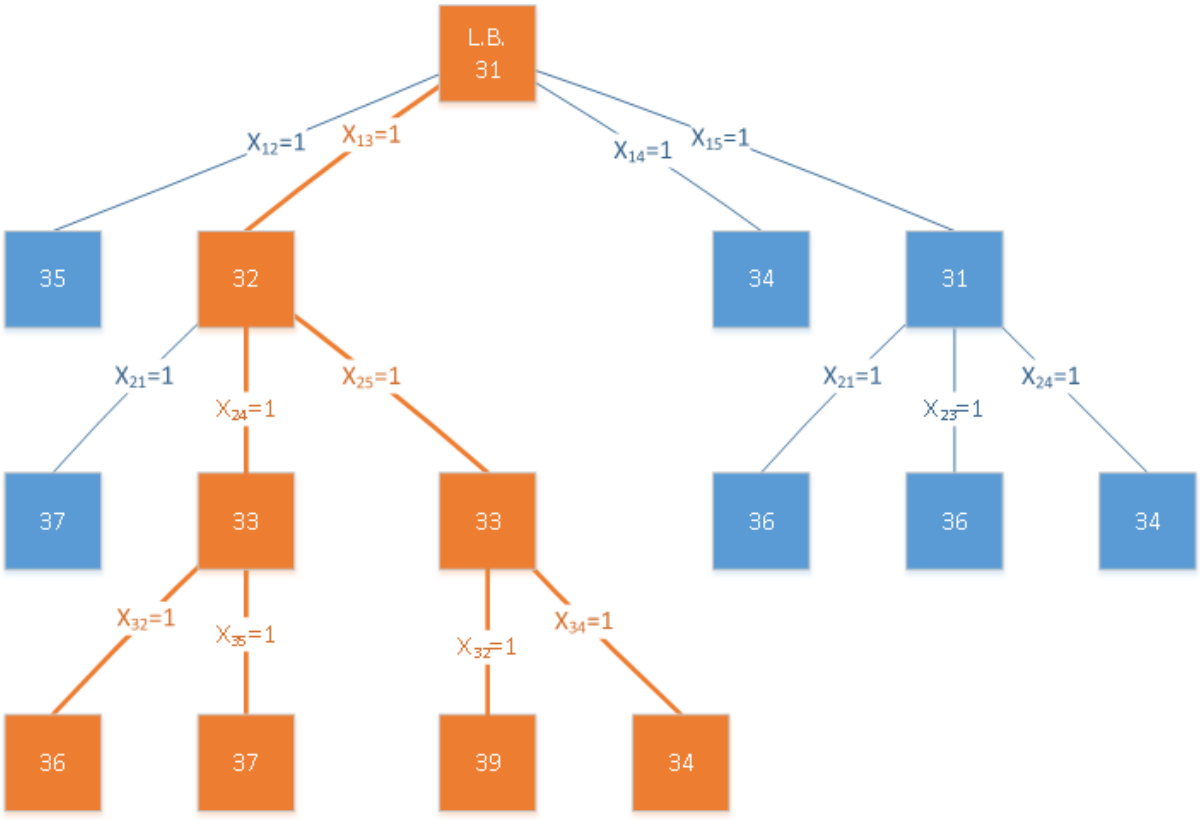


Figure 2.1: The branch and bound diagram with optimal solution at node 14

Case 2: Let us consider the asymmetric matrix,

	x_1	x_2	x_3	x_4	x_5
x_1	∞	2	5	7	1
x_2	6	∞	3	8	2
x_3	8	7	∞	4	7
x_4	12	4	6	∞	5
x_5	1	3	2	8	∞

Step 1: The lower bound of total distance is

$$LB = \sum_{i=1}^{5} \min d(x_i) = 1 + 3 + 4 + 4 + 1 = 13$$

and the tour has to travel at least 13. Optimal solution will have to be more than or equal to the lower bound of 13.

Step 2: We create $(n-1) = (5-1) = 4$ branches and say $x_{12} = 1, x_{13} = 1, x_{14} = 1,\ x_{15} = 1$. We do not have $x_{11} = 1$ becouse x_{11} is a subtour of length 1. The same thing holds for all x_{ii} which means x_{11}, x_{22}, x_{33}, x_{44} and x_{55} are all sub-tours.

Step 3: Fixing x_1 and x_2, by leaving out the first row and the second column to get additional minimum distance, we obtain

	x_1	x_3	x_4	x_5
x_2	6	3	8	2
x_3	8	∞	4	7
x_4	12	6	∞	5
x_5	1	2	8	∞

The minimum travelling is

$$x_1 \to x_2,\ x_2 \to x_5,\ x_3 \to x_4\ ,\ x_4 \to x_5,\ x_5 \to x_1.$$

Then the lower bound is

$$2 + 2 + 4 + 5 + 1 = 14.$$

Repeating the same calculation, x_1 to x_3 will be $5 + 2 + 4 + 4 + 1 = 16$, x_1 to x_4 will be $7 + 2 + 7 + 4 + 1 = 21$ and x_1 to x_5 will be $1 + 3 + 4 + 4 + 1 = 13$.

Step 4: We want to minimize the total distance travelling so we branch further from 13 because it has the minimum value of lower bound. Therefore, we have to create three more branches as $x_{21} = 1, x_{23} = 1, x_{24} = 1$.

Step 5: We will find out the new lower bound. Now we will free x_1 to x_5 and x_2 to x_1 then we calculate the minimum travelling value. Here x_5 to x_2 is a sub-tour so we will choose it as∞.

	x_1	x_2	x_3	x_4
x_2	6	∞	3	8
x_3	8	7	∞	4
x_4	12	4	6	∞
x_5	1	∞	2	8

The minimum travelling is

$$x_1 \to x_5, \quad x_2 \to x_1, \quad x_3 \to x_4, \quad x_4 \to x_2, \quad x_5 \to x_3.$$

Then the lower bound is

$$1 + 6 + 4 + 2 + 2 = 15.$$

Repeating the same calculation, x_2 to x_3 will be $1 + 3 + 4 + 4 + 1 = 13$, x_2 to x_4 will be $1 + 8 + 7 + 4 + 1 = 21$.

Step 6: Evaluate up to this point, find the minimum value and then draw new branches. In the solution 13 is the new lower bound so spread in branches from this node. We create two new branches here like $x_{31} = 1, x_{34} = 1$.

Step 7: Now we will go back to find out the new lower bound. For the first one we put $x_1 \to x_5$ and $x_2 \to x_3$.

	x_1	x_2	x_4
x_3	8	7	4
x_4	12	4	∞
x_5	1	3	8

The minimum travelling is

$$x_1 \to x_5,\ x_2 \to x_3,\ x_3 \to x_1,\ x_4 \to x_2,\ x_5 \to x_2$$

Then the lower bound is

$$1 + 2 + 8 + 4 + 3 = 19.$$

As the same calculation, x_3 to x_4 will be $1 + 3 + 4 + 4 + 1 = 13$.

Step 9: We fix three terms and when we have only two more terms to go, we don't look for the lower bound but the upper one. This upper bound becomes the feasible solution. Therefore the way of feasible solution for x_3 to x_4 is

$$x_1 \to x_5,\ x_2 \to x_3,\ x_3 \to x_4,\ x_4 \to x_2,\ x_5 \to x_1.$$

Then there are two sub-tours in this solution so we have to move one link from one to other

$$x_1 \to x_5, \qquad x_5 \to x_3, \qquad x_3 \to x_4, \qquad x_4 \to x_2, \qquad x_2 \to x_1$$

Thus the feasible solution is

$$z = 1 + 2 + 4 + 6 + 6 = 19.$$

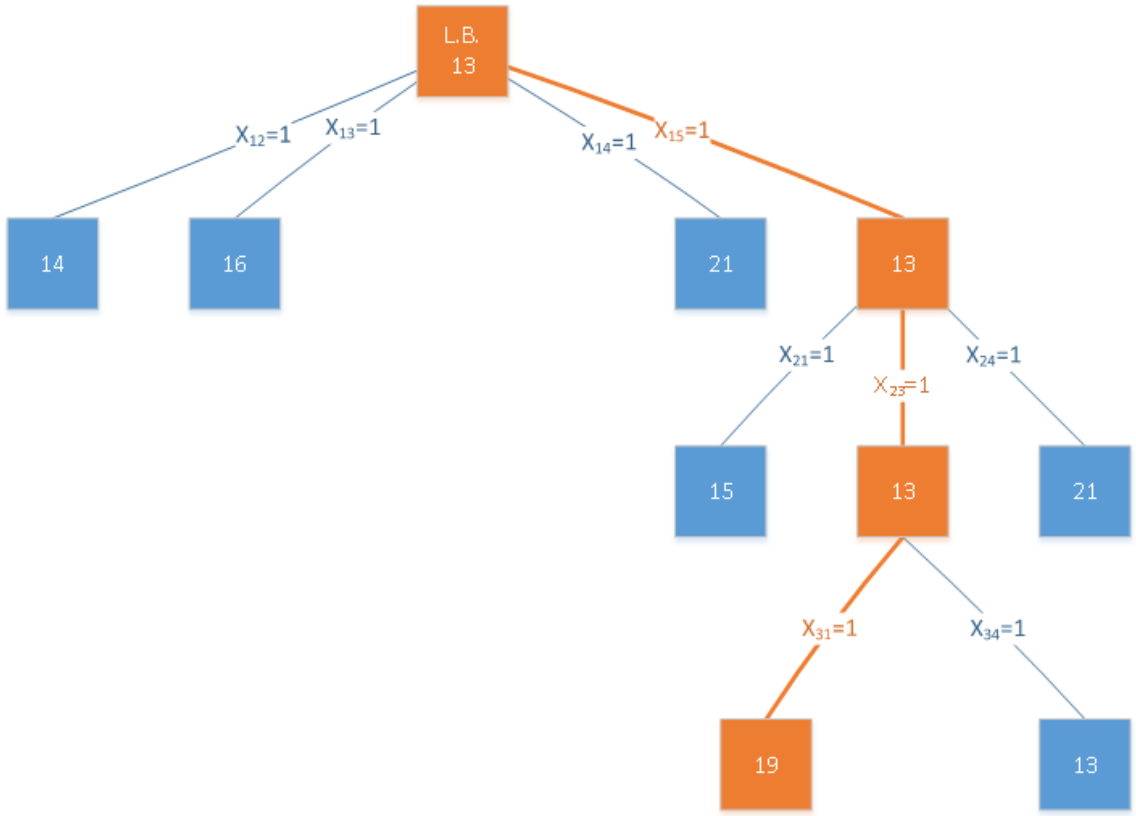


Figure 2.2: The branch and bound diagram with optimal solution at node 9

Chapter 3

BRANCH AND BOUND ALGORITHM

3.1 Introduction

In this section, we talk about the basic principle of branch and bound algorithms.

Since the first attempt to solve travelling salesman problems by an enumerative approach, apparently due to Eastman (1958), many such procedures have been proposed. In a sense the TSP has served as a testing ground for the development of solution methods for discrete optimization, in that many procedures and, devices were first developed for the TSP and then, after successful testing, extended to more general integer programs. The term "branch and bound" itself was coined by Little, Murty, Sweeney and Karel (1963) in conjunction with their TSP algorithm. [7]

Enumerative[6] (branch and bound, implicit enumeration) methods solve a discrete optimization problem by breaking up its feasible set into successively smaller subsets,

[6] Enumerative methods are investigating many cases only in an implicit way. This means that huge majority of the cases are dropped based on consequences obtained from the analysis of the particular numerical problem.

calculating bounds on the objective function value over each subset, and using them to discard certain subsets from further consideration. The bounds are obtained by replacing the problem over a given subset with an easier problem, such that the solution value of the latter bounds that of the former. The procedure ends when each subset has either produced a feasible solution, or was shown to contain no better solution than the one already in hand. The best solution found during the procedure is a global optimum. [7]

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

i. a relaxation of P, i.e. a problem R of the form $min\{g(x)|x \in T\}$, such that $S \subseteq T$ and for every $x, y \in S$, $f(x) < f(y)$ implies $g(x) < g(y)$.
ii. a branching or separating rule, i.e. a rule for breaking up the feasible set S_i of the current subproblem P_i into subsets $S_{i1}, \dots, S_{iq}$, such that $\bigcup_{j=1}^{q} S_{ij} = S_i$;
iii. a lower bounding procedure, i.e. a procedure for finding (or approximating from below) $v(R_i)$ for the relaxation R_i of each sub-problem P_i; and
iv. a sub-problem selection rule, i.e. a rule for choosing the next sub-problem to be processed.

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

v. an upper bounding procedure, i.e. a heuristic[7] for finding feasible solutions to P; and

vi. a testing procedure, i.e. a procedure for using the logical implications[8] of the constraints and bounds to fix the values of some variables (reduction, variable fixing) or to discard an entire sub-problem (dominance tests[9]).

[7]

After the procedure of the branch and bound algorithm, we point out the formulation of integer programming with details in the following discussion.

The integer programming formulation of the TSP that we will refer to when discussing the various solution methods is defined on a complete directed graph $G = (V, A)$ on n nodes, with node set $V = \{1, \dots, n\}$, arc set $A = \{(i, j) | i, j = 1, \dots, n\}$, and nonnegative costs c_{ij} associated with the arcs. The fact that G is complete involves no restriction,

---

[7] Heuristics are used to compute upper bounds.

[8] Whenever we tighten variable bounds either by branching on them or by using reduced cost criteria, we try to tighten more bounds by searching for logical implications of the improved bounds.

[9] The dominance test is known as another test for sub-problem. While the bounding test compares a lower bound of sub-problem with an incumbent value, the dominance test compares a sub-problem with an another sub-problem.

since arcs that one wishes to ignore can be assigned the cost $c_{ij} = \infty$. In all cases $c_{ij} = \infty, \forall\, i, j \in V$. The TSP can be formulated, following Dantzig, Fulkerson and Johnson (1954), as the problem

$$min \sum_{i \in V} \sum_{j \in V} c_{ij} x_{ij} \qquad (3.1)$$

subject to

$$\sum_{j \in V} x_{ij} = 1 \qquad (i \in V) \qquad (3.2)$$

$$\sum_{i \in V} x_{ij} = 1 \qquad (j \in V) \qquad (3.3)$$

$$\sum_{i \in S} \sum_{j \in S} x_{ij} \leq |S| - 1 \qquad \forall\, S \subset V, S \neq \emptyset \qquad (3.4)$$

$$x_{ij} = 0 \; or \; 1 \qquad (i, j) \in V \qquad (3.5)$$

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

The sub-tour elimination inequalities (3.3) can also be written as

$$\sum_{i \in S} \sum_{j \in V \setminus S} x_{ij} \geq 1 \qquad \forall\, S \subset V, S \neq \emptyset.$$

[7]

Let us discuss the branching rules which help to understand the idea of the branch and bound algorithms.

3.2 Branching Rules

During this section, we talk about branching rules and look at how can we apply it on the related example.

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

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

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

In addition to (3.1), (3.2), (3.3), (3.4). [7]

3.2.1 Branching Rule 1: (Little, Murty, Sweeney and Karel [1963])

Given the current relaxed sub-problem AP_k and its reduced costs $\bar{c}_{ij} = c_{ij} - u_i - v_j$, where u_i and v_j are optimal dual variables, for every (i, j) such that $\bar{c}_{ij} = 0$ define the penalty

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

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

$$P_{rs} = max\{P_{ij}: \bar{c}_{ih} = 0\}.$$

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

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

and

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

[7]

Let us now consider a simple application that enables one to understand the case of Little's Rule more clearly.

3.2.1.1 Example: Little's Branch and Bound Algorithm

Consider the assignment problem with matrix,

	x_1	x_2	x_3	x_4	x_5
x_1	∞	10	8	9	7
x_2	10	∞	10	5	6
x_3	8	10	∞	8	9
x_4	9	5	8	∞	6
x_5	7	6	9	6	∞

Step 1: At first, we calculate the row minima as

	x_1	x_2	x_3	x_4	x_5	Row Min
x_1	∞	10	8	9	7	7
x_2	10	∞	10	5	6	5
x_3	8	10	∞	8	9	8
x_4	9	5	8	∞	6	5
x_5	7	6	9	6	∞	6
Σ						31

Then subtracting 7 from Row 1, 5 from Row 2, 8 from Row 3, 5 from Row 4, and 6 from Row 5, we get

	x_1	x_2	x_3	x_4	x_5
x_1	∞	3	1	2	0
x_2	5	∞	5	0	1
x_3	0	2	∞	0	1
x_4	4	0	3	∞	1
x_5	1	0	3	0	∞

Step 2: We calculate the column minima as

	x_1	x_2	x_3	x_4	x_5	Σ
x_1	∞	3	1	2	0	
x_2	5	∞	5	0	1	
x_3	0	2	∞	0	1	
x_4	4	0	3	∞	1	
x_5	1	0	3	0	∞	
Column Min	0	0	1	0	0	1

Then, subtracting 0 from Column 1, 0 from Column 2, 1 from Column 3, 0 from Column 4, and 0 from Column 5, we get

*	x_1	x_2	x_3	x_4	x_5
x_1	∞	3	0	2	0
x_2	5	∞	4	0	1
x_3	0	2	∞	0	1
x_4	4	0	2	∞	1
x_5	1	0	2	0	∞

Step 3: We will get the lower bound as

$$L.B = \sum row\ min + \sum column\ min = 31 + 1 = 32.$$

We look at matrix (*) and as in the assignment problem; we set a group of assignable zeros for each row. We begin this process by fixing assignable zeros in Row 1.

Step 4: We proceed to get the new matrix by choosing the zeros, to calculate the row penalties and the column penalties. To get this number we are going to add the row minimum and the column minimum of the chosen zeros.

	x_1	x_2	x_3	x_4	x_5
x_1	∞	3	$0^{(0+2)}$	2	$0^{(0+1)}$
x_2	5	∞	4	$0^{(1+0)}$	1
x_3	$0^{(0+1)}$	2	∞	$0^{(0+0)}$	1
x_4	4	$0^{(1+0)}$	2	∞	1
x_5	1	$0^{(0+0)}$	2	$0^{(0+0)}$	∞

	x_1	x_2	x_3	x_4	x_5
x_1	∞	3	$0^{(2)}$	2	$0^{(1)}$
x_2	5	∞	4	$0^{(1)}$	1
x_3	$0^{(1)}$	2	∞	$0^{(0)}$	1
x_4	4	$0^{(1)}$	2	∞	1
x_5	1	$0^{(0)}$	2	$0^{(0)}$	∞

Row penalty belongs to the zero with the highest sum for the row and the column minimum. If the next highest element happens to be a zero, then the penalty is zero, otherwise there is a positive penalty.

Step 5: We look at a zero which has the highest penalty and in this example it turns out to be x_{13}. After picking this zero, we start our assignment problem. We insert the maximum total penalty of 2. We choose this and desire to branch on this particular variable. Therefore we branch on the variable x_{13} and create two nodes. These are $x_{13} = 0$ and $x_{13} = 1$. Since the $x_{13} = 0$ solution has already been computed we do not make an assignment here. For $x_{13} = 1$ we will have an increase cost of 2, so that the lower bound for this becomes 33 plus 2, which is 34.

Step 6: In the next case, we have to allocate x_{13}, so we will leave row of x_1 and column of x_3 and create a 4×4 matrix as

	x_1	x_2	x_4	x_5
x_2	5	∞	0	1
x_3	∞	2	0	1
x_4	4	0	∞	1
x_5	1	0	0	∞

If we have allocated x_{13} to 1 we should also not have x_{31} in the solution, otherwise this will create a sub-tour, so we will not have x_{31} in the solution.

Step 7: In this reduced matrix, we now have to check whether every row and every column has an assignable zero. We realize that all rows and some columns have the zeros, but the columns for x_1 and x_5 do not have zeros. If we subtract the column minimum, from these particular columns, the matrix will change as follows.

	x_1	x_2	x_4	x_5
x_2	4	∞	0	0
x_3	∞	2	0	0
x_4	3	0	∞	0
x_5	0	0	0	∞

We subtract 1 from Column x_1 and 1 from Column x_5, so effectively we subtract 2. Therefore, the lower bound increases from 32 to 34.

Step 8: Now again we will find the row penalty and the column penalty from the last reduced matrix. We get

	x_1	x_2	x_4	x_5
x_2	4	∞	$0^{(0)}$	$0^{(0)}$
x_3	∞	2	$0^{(0)}$	$0^{(0)}$
x_4	3	$0^{(0)}$	∞	$0^{(0)}$
x_5	$0^{(3)}$	$0^{(0)}$	$0^{(0)}$	∞

Here maximum penalty is 3, so this is the next candidate for branch. Thus x_{51} is the next candidate to branch. Again we create two branches here; one of them is $x_{51} = 0$ and the other one is $x_{51} = 1$.

Step 9: As the same condition For $x_{15} = 0$ solution, lower bound will be 37. For $x_{15} = 1$ condition we leave row of x_5 and column of x_1 and then we create a 3×3 matrix as

	x_2	x_4	x_5
x_2	∞	0	0
x_3	2	0	∞
x_4	0	∞	0

We should not have x_{35} because of sub-tour ($x_5 \to x_1 \to x_3$). Now after this, every row and every column has zeros so we don't need to subtract the row minimum or the column minimum. Therefore the lower bound is $34 + 0 = 34$.

Step 10: Now again computing the penalties, we get

	x_2	x_4	x_5
x_2	∞	$0^{(0)}$	$0^{(0)}$
x_3	2	$0^{(2)}$	∞
x_4	$0^{(2)}$	∞	$0^{(0)}$

Thus we can choose either x_{34} or x_{42}, because each of them has the same maximum penalty. Therefore let us choose x_{34} to begin branch.

Step 11: There are two branches. The first one is $x_{34} = 0$, which has a lower bound $34 + 2 = 36$. For the second one, we leave row of x_3 and column of x_4, to get the 2×2 matrix

	x_2	x_5
x_2	∞	$\boxed{0}$
x_4	$\boxed{0}$	∞

There is a sub-tour for x_{45}, so we replace the 0 by ∞ ($x_5 \to x_1 \to x_3 \to x_4$). After this every row and every column has assignable zero. Therefore the lower bound does not increase. Thus the lower bound stays as 34.

The feasible solution to the TSP is

$$x_{13} = x_{34} = x_{42} = x_{25} = x_{51} = 1.$$

Thus the optimal solution is

$$z = 8 + 8 + 5 + 6 + 7 = 34.$$

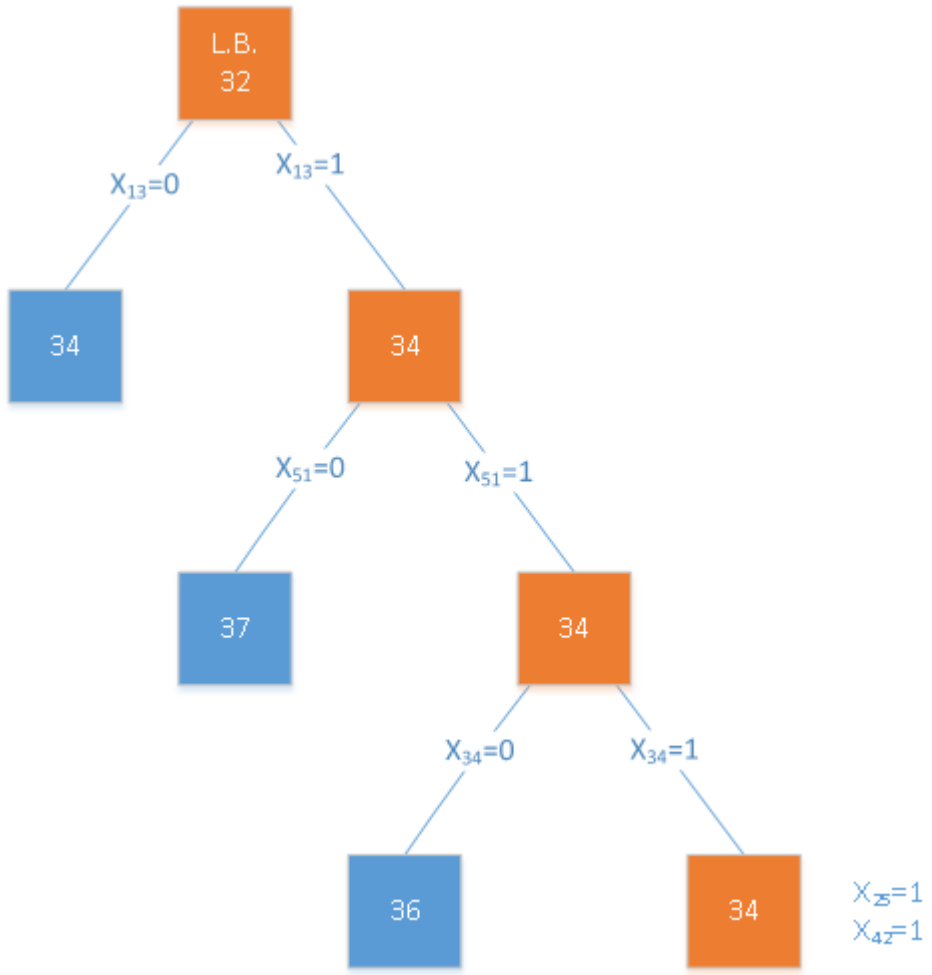


Figure 3.1: The branch and bound diagram with optimal solution at node 6

Another important rule is known as Eastman's Rule. We use the below inequality for solving results on the example. Before we proceed with an example, we give some basic inequality.

3.2.2 Branching Rule 2: (Eastman [1958], Shapiro [1966])

Let x^k be the optimal solution to the current relaxed sub-problem AP_k, and let $A_s = \{(i_1, i_2), \dots, (i_t, i_1)\}$ be the arc set of a minimum cardinality sub-tour of x^k involving the node set $S = \{i_1, \dots, i_t\}$. Constraint (3.3) for S implies the inequality

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

which in turn implies the disjunction

$$x_{i_1 i_2} = 0, \ldots, x_{i_t i_1} = 0. \qquad (3.6)$$

Generate t successors of node k, defined by

$$E_{k+r} = E_k \cup \{(i_r, i_{r+1})\}, \qquad I_{k+1} = I_k, \qquad r = 1, \ldots, t$$

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

Now x^k is clearly infeasible for all $AP_{k+r}, r = 1, \ldots, t$, and the choice of a shortest sub-tour for branching keeps the number of successor nodes small. However, the disjunction (3.6) does not define a partition of the feasible set of AP_k, and thus different successors of AP_k may have common solutions. [7]

Let us now apply the above inequality in the following example.

3.2.1.2 Example: Eastman's Branch and Bound Algorithm

Consider the assignment problem with the matrix below,

	x_1	x_2	x_3	x_4	x_5
x_1	∞	10	8	9	7
x_2	10	∞	10	5	6
x_3	8	10	∞	8	9
x_4	9	5	8	∞	6
x_5	7	6	9	6	∞

Step 1: First let us solve the given assignment problem by using the Hungarian Algorithm[10]. We calculate the row minima as

[10] The Hungarian Algorithm is an algorithm for solving a matching problem or more generally an assignment linear programming problem.

	x_1	x_2	x_3	x_4	x_5	Row Min
x_1	∞	10	8	9	7	7
x_2	10	∞	10	5	6	5
x_3	8	10	∞	8	9	8
x_4	9	5	8	∞	6	5
x_5	7	6	9	6	∞	6

Then, subtracting 7 from Row 1, 5 from Row 2, 8 from Row 3, 5 from Row 4, and 6 from Row 5, we get,

	x_1	x_2	x_3	x_4	x_5
x_1	∞	3	1	2	0
x_2	5	∞	5	0	1
x_3	0	2	∞	0	1
x_4	4	0	3	∞	1
x_5	1	0	3	0	∞

Step 2: We calculate the column minima as

4	x_1	x_2	x_3	x_4	x_5
x_1	∞	3	1	2	0
x_2	5	∞	5	0	1
x_3	0	2	∞	0	1
x_4	4	0	3	∞	1
x_5	1	0	3	0	∞
Column Min	0	0	1	0	0

Then, subtracting 0 from Column 1, 0 from Column 2, 1 from Column 3, 0 from Column 4, and 0 from Column 5, we get,

	x_1	x_2	x_3	x_4	x_5
x_1	∞	3	0	2	0
x_2	5	∞	4	0	1
x_3	0	2	∞	0	1
x_4	4	0	2	∞	1
x_5	1	0	2	0	∞

Step 3: We start to make assignments. Since we already have two assignable zeros in the first row, we don't make any other assignment here. In the second row there is only one assignable zero so after fixing it, the other zeros in Column 4 disappears. In this way, only one assignable zero remains in Row3. In the next Row, again we have only one assignable zero. Therefore we cross out the zero in column 2. Hence we cannot make any assignment in Row 4.

Then we look at the columns', first two columns already have assignments but the third column doesn't. Therefore, we can make an assignment in the third column and in this way the other zero in Row 1 goes. Now we cannot make any more assignments here.

	x_1	x_2	x_3	x_4	x_5
x_1	∞	3	[0]	2	0
x_2	5	∞	4	[0]	1
x_3	[0]	2	∞	0	1
x_4	4	[0]	2	∞	1
x_5	1	0	2	0	∞

Step 4: We go to the ticking process. The first tick goes to the non assignee Row 5. Then we tick the columns containing the non-assignable zeros. Next, we tick the rows containing the assignable zeros of the particular columns. Therefore we choose Row 4 and Row 2. If there are other zeros in these rows, we will tick them too. However, we don't have this condition in our example.

		x_1	x_2	x_3	x_4	x_5
	x_1	∞	3	[0]	2	0
→	x_2	5	∞	4	[0]	1
	x_3	[0]	2	∞	0	1
→	x_4	4	[0]	2	∞	1
→	x_5	1	0	2	0	∞
			↑		↑	

Step 5: Now, we cover the non-ticked rows and the tick columns. These are rows 1, 3 and columns 2 and 4. This means covering all the zeros of the matrix with the minimum number of horizontal or vertical lines.

	x_1	x_2	x_3	x_4	x_5
x_1	∞	3	[0]	2	0
x_2	5	∞	4	[0]	1
x_3	[0]	2	∞	0	1
x_4	4	[0]	2	∞	1
x_5	1	0	2	0	∞

Step 6: Now find the minimum numbers of the rows not covered with lines. Let θ be the minimum of these picked numbers. Now, subtract θ from the non-picked entries, and add it to the picked ones.

	x_1	x_2	x_3	x_4	x_5
x_1	∞	$3+\theta$	$\boxed{0}$	$2+\theta$	0
x_2	$5-\theta$	∞	$4-\theta$	$\boxed{0}$	$1-\theta$
x_3	$\boxed{0}$	$2+\theta$	∞	$0+\theta$	$1+\theta$
x_4	$4-\theta$	$\boxed{0}$	$2-\theta$	∞	$1-\theta$
x_5	$1-\theta$	0	$2-\theta$	0	∞

	x_1	x_2	x_3	x_4	x_5
x_1	∞	4	0	3	0
x_2	4	∞	3	0	0
x_3	0	3	∞	1	1
x_4	3	0	1	∞	0
x_5	0	0	1	0	∞

where $\theta = \min(x_{25},\ x_{45},\ x_{51}) = 1$.

Step 7: Now for this reduced matrix, we start making the allocation again. This means that we turn back to Step 3 to get a new matrix.

	x_1	x_2	x_3	x_4	x_5
x_1	∞	4	[0]	3	0
x_2	4	∞	3	0	0
x_3	[0]	3	∞	1	1
x_4	3	0	1	∞	0
x_5	0	0	1	0	∞

We get into a situation where we randomly choose our assignable zero. Therefore, we start with x_{24} which is an arbitrary allocation. Then in the same way, finally, we get the five allocations as

	x_1	x_2	x_3	x_4	x_5
x_1	∞	4	[0]	3	0
x_2	4	∞	3	[0]	0
x_3	[0]	3	∞	1	1
x_4	3	0	1	∞	[0]
x_5	0	[0]	1	0	∞

Thus the assignment solution is

$$x_{13} = x_{24} = x_{31} = x_{45} = x_{52} = 1.$$

The corresponding value

$$z = 8 + 5 + 8 + 6 + 6 = 33.$$

Step 8: We need to study this assignment solution carefully and verify whether it is feasible to TSP. If the solution is feasible to the TSP, then it is optimal, but we should check the answer carefully. We have a sub-tour which is

$$x_{13} = x_{31} = 1 \text{ and } x_{24} = x_{45} = x_{52} = 1.$$

Therefore, there are two sub-tours in this solution and these two sub-tours are shown below;

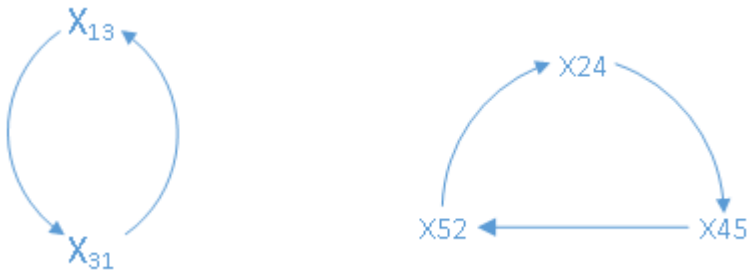


Figure 3.2: Sub-tour diagram of branch and bound

This $z = 33$ becomes the lower bound to the optimum value of the TSP.

Step 9: There are two sub-tours here and in order to get that tour, we have to move one link of figure A along with one link of figure B. these broken links are later replaced by ones that go from A to B and B to A. We go back to the last reduced matrix and create a sub-tour to sub-tour matrix which is

$$min\left(\begin{bmatrix} x_{131} \times x_{131} & x_{131} \times x_{2452} \\ x_{2452} \times x_{131} & x_{2452} \times x_{2452} \end{bmatrix}\right) = \begin{bmatrix} \infty & 0 \\ 0 & \infty \end{bmatrix}.$$

Step 10: We can start a Branch-and-Bound Algorithm with the L.B 33 of TSP increased by zero units. There are two sub-tours which are $x_{13} \rightarrow x_{31}$ and $x_{24} \rightarrow x_{45} \rightarrow x_{52}$. Therefore we have to eliminate these subtours. We take the smaller sub-tour $x_{13} \rightarrow x_{31}$, of length 2, and then create two branches here.

The values $x_{13} = 0$ or 1 imply the existence or nonexistence of the line x_{13} in the solution.

Step 11: When $x_{13} = 0$, we will go back to the matrix 1?, replace the entry there by ∞ and solve the assignment problem again. We get a final matrix like the one (matrix 11?) so whatever we have here will be a lower bound and we will check if it is a feasible solution of the TSP. Thus if we get an increase in the L.B. from 33 to 34, we can't get a feasible solution. We still get two sub-tours as $x_{15} \rightarrow x_{53} \rightarrow x_{31}$ and $x_{24} \rightarrow x_{42}$. The optimal solution to the assignment problem has objective function value of 34 which is a lower bound and it has two subtours.

Step 12: When we solve the other case, we put $x_{13} = 1$. Since we force $x_{13} = 1$ to be 1, we remove the first row and third column and solve the resulting 4×4 problem. Then add this 8 to the new L.B. and get the new optimal solution. In this case we also get the lower bound of 34, but we still get sub-tours $x_{13} \rightarrow x_{35} \rightarrow x_{51}$ and $x_{24} \rightarrow x_{42}$. One can easily observe that these two solutions are actually the same with respect to the TSP, because $x_{15} \rightarrow x_{53} \rightarrow x_{31}$ is same as $x_{13} \rightarrow x_{35} \rightarrow x_{51}$. Both of the results are still a feasible solution to the TSP and have two sub-tours. Therefore, we have to eliminate the sub-tours again.

Step 13: We can separate two branches here as $x_{24} = 0$ and $x_{24} = 1$. Before we solve this particular assignment problem, we need to study the existing ones. We have to force $x_{13} = 0$ and $x_{24} = 0$ which means ∞ will replace x_{13} and x_{24} in matrix 1. Then the lower bound becomes 36, but we still have a solution

$$x_{15} = x_{54} = x_{42} = x_{23} = x_{31}$$

which is a feasible solution to the TSP. Therefore this 36 is an upper bound. We have a feasible solution with the value 36. We are no longer interested in the solutions with the value 36 or above.

Step 14: We will go back and evaluate the $x_{24} = 1$ branch which means we can remove the second row and the forth column, solve the remaining 4×4, and add 5 to the solution. Therefore this gives us a lower bound of 34 and we get the solution

$$x_{15} = x_{52} = x_{24} = x_{43} = x_{31} = 1.$$

Therefore the lower bound becomes 34, which is the final feasible solution to the TSP.

Thus the optimal solution is

$$z = 7 + 6 + 5 + 8 + 8 = 34.$$

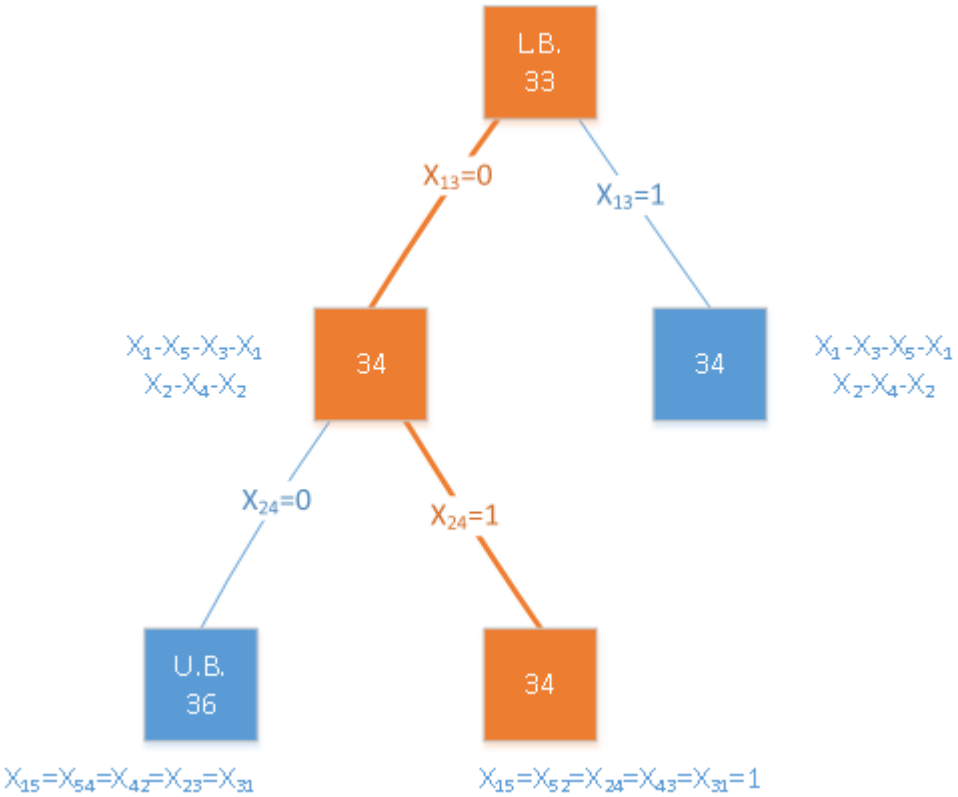


Figure 3.3: The branch and bound diagram with optimal solution at node 4

Chapter 4

GENETIC ALGORITHMS

4.1 Introduction

We start this section with the following explanation and then we mention the genetic operator.

Genetic Algorithms are essentially evolutionary optimization techniques where we try to mimic to process of evolution. Evolution is taken as an optimizing process. GA is used as evolutionary mechanism to create a solution for the random choosing. The basic logic of this method is provided for the extinction of the bad solution and good solution is achieved gradually over year. In GA, the chromosomes represent possible solutions for the problem. Population is a set which consists of chromosomes. Sufficiency rate is determining the value of solution and is calculated with using sufficiency function. New population can be obtained by genetic operator like evaluation, crossover and mutation.

4.1.1 Coding of Chromosomes

Coding of chromosome is the first step of problem solving with genetic algorithm. Coding may differ from according to the type of the problem. The most useful methods are binary coding and permutation coding. In binary coding every (each) chromosomes is represented by a series of characters (involving) 0 and 1. In permutation encoding,

each chromosomes is represented by a series. Permutation coding is used very often in TSP.

4.1.2 Selection

Individuals must be selected from the group to create a new population by crossover and mutation. By theory, good individuals should continue their life and create a new individual. Therefore, the probability of selection is important for the fitness value of the individual. The best known selection methods are Roulette-wheel selection, Tournament selection, and Rank selection.

4.1.2.1 Roulette-Wheel Selection

Firstly the fitness value is collected for each individual in the group and the probability of selection comes from the collection of fitnesses. However, if there are big differences between the fitnesses, it will continue to give the same solution.

4.1.2.2 Tournament Selection

k individuals are chosen randomly from the population and the ones with the best fitnesses are picked. In here k might change according to the size of the population.

4.1.2.3 Rank Selection

Chromosome which has bad fitness is given a value of 1, and the value increases to 2 or 3 as the fitnesses of the chromosome improves. The aim of in this selection is give a chance to enable the use of bad fitness chromosomes in other places. Only disadvantage is that this process may take a long time.

4.1.3 Evolution

The quit valuable role of genetic algorithm is the evolution function. We always use the fittnest function to choose a good chromosome. Besides, if we have a large number of initial populations, this method helps us to choose the best chromosome for the next cross over step. [8]

4.1.4 Cross over

After the completion the evaluation process, the chromosomes chosen are recombined as a new chromosome to determine the parent for the next generation. Briefly, the most effective step for the procedure is called cross over. [8]

4.1.5 Mutation

The operation of mutation permit new individual to be created. It starts by selecting an individual from the population based on its fitness. A point along the string is randomly selected and the character at that point is changed at random, the alternate individual is then copied in to the next generation of the population.

After cross over, mutation is performed. During this process, a chromosome in the new generation is randomly chosen to mutate and switch that point. There are different kinds of mutation operators exist. For a binary chromosomes representation, we always use the bit flip method which alters a particular point on the chromosomes to its opposite. [8]

4.2 Matrix Representation

These studies are related to the symmetric and asymmetric cases in the binary matrix.

In this work, the tours are represented by binary matrix. In figure, x_1, x_2, x_3, x_4, x_5 represents binary bits and the 1′s in the set represent an edge from x_i to x_j where $i, j = 1:5$.

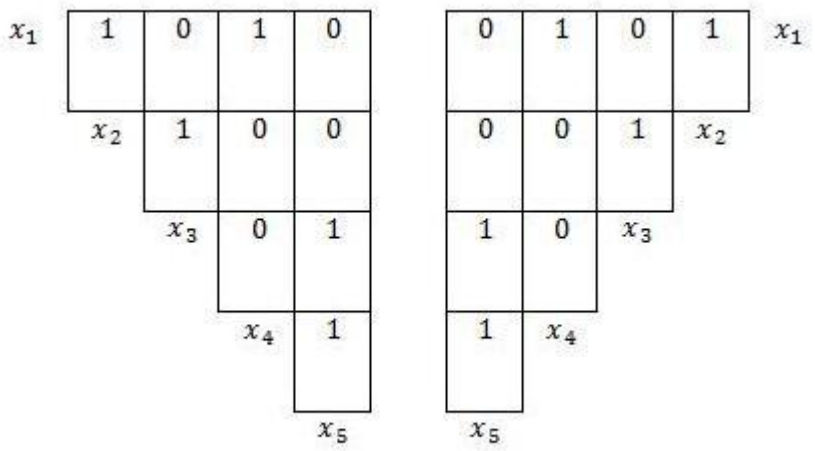


Figure 4.1: The binary matrix for the symmetric case

	x_1	x_2	x_3	x_4	x_5
x_1	0	1	0	1	0
x_2	1	0	1	0	0
x_3	0	1	0	0	1
x_4	1	0	0	0	1
x_5	0	0	1	1	0

Figure 4.2: The matrix for the symmetric case

For symmetric case, the LUTM shows the distance between cities i and j, if $d_{ij} = d_{ji}$ in Figure1.a. As in Figure2.b below, if $d_{ij} \neq d_{ji}$, the given matrix for TSP is asymmetric.

For the symmetric case, the LUTM shows the flow from left to right. It works as $x_1 \rightarrow x_2$, $x_2 \rightarrow x_3$, $x_3 \rightarrow x_5$, then the backward motion starts from the lower parts of the RUTM to move up like $x_5 \rightarrow x_4$, $x_4 \rightarrow x_1$. In that case the closed path is $x_1 \rightarrow x_2 \rightarrow x_3 \rightarrow x_5 \rightarrow x_4 \rightarrow x_1$.

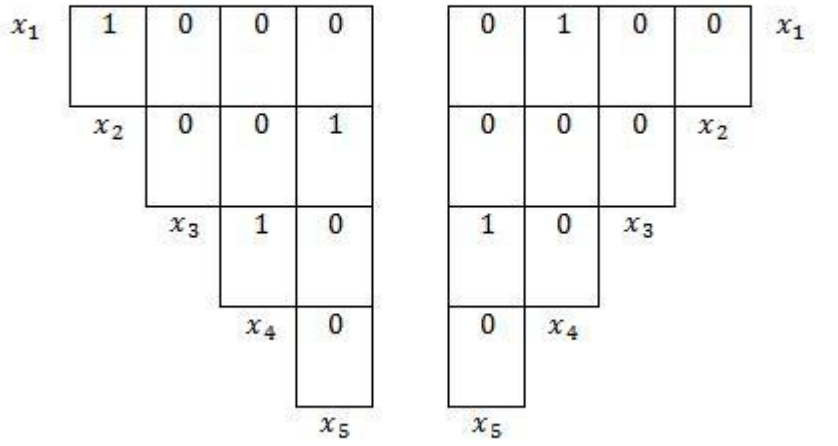


Figure 4.3: The binary matrix for asymmetric case

	x_1	x_2	x_3	x_4	x_5
x_1	0	1	0	1	0
x_2	1	0	0	0	1
x_3	0	0	0	1	1
x_4	1	0	1	0	0
x_5	0	1	1	0	0

Figure 4.4: The matrix for asymmetric case

There are two valid conditions for the TSP tour to be **symmetric**.

1. The number of x_i must be equal to number of $1'$s in the matrix.
2. Each x_i must have only 2 edges running from one to other.

Also there are two **asymmetric** situations;

1. The sum of x_i must be equal to number of edges in both matrices.
2. Each x_i must have only 2 edges running from one to other.

4.2.1 Cross over Operation

The binary OR operation is useful for the Crossover operation. In here we have two parent matrices as input argument and one single parent matrix as output argument. When we apply the OR operation are outputs apart from the 0 and 0 case give 1.

4.2.2 Mutation Operation

If the final answer does not satisfy our symmetric or asymmetric cases, it must be restored until the number of edges between any pair of vertices is 2. Otherwise, we either delete the largest edge or adding an appropriate edge by greedy algorithm to reach the required number. We can apply greedy algorithm to construct optimal solution step by step. The main idea of the greedy algorithm is to construct the missing parts of the optimal solution and extend it by identifying the next part.

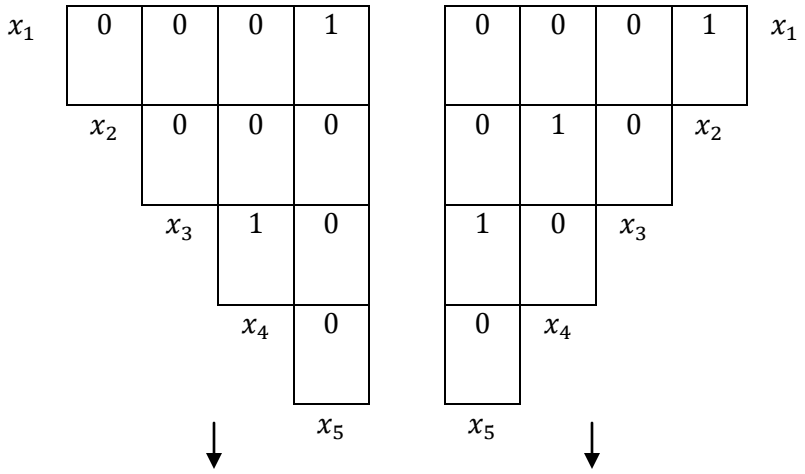
Considering two tours

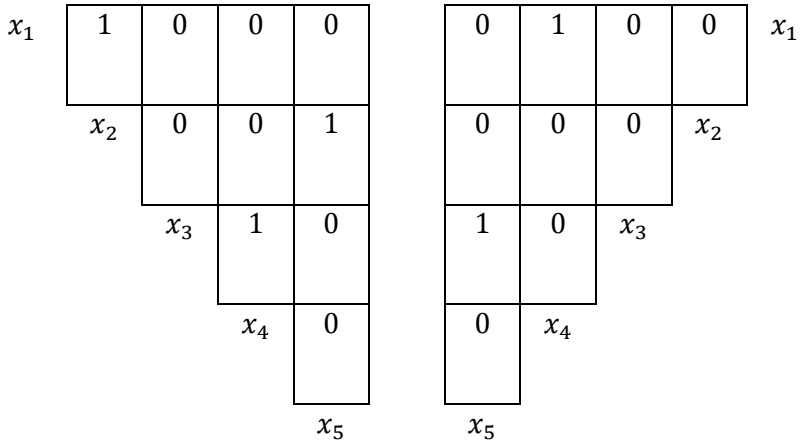
$$T_1: x_1 \to x_5 \to x_3 \to x_4 \to x_2 \to x_1 = 17$$

$$T_2: x_1 \rightarrow x_2 \rightarrow x_5 \rightarrow x_3 \rightarrow x_4 \rightarrow x_1 = 22$$

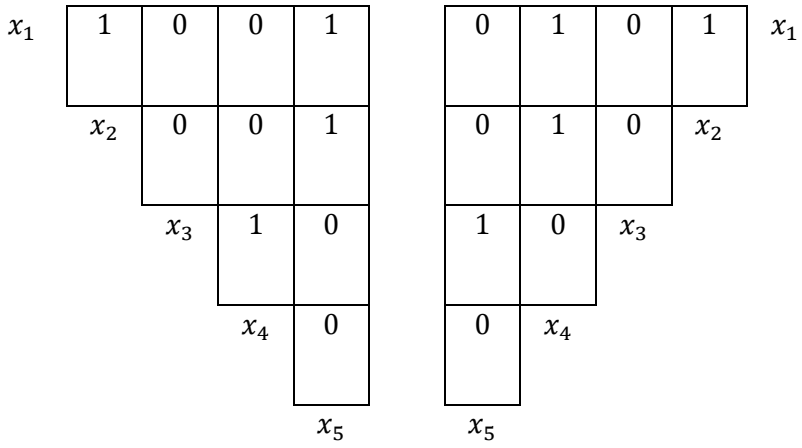
then cross over and mutation of these two tours will be as in steps. [8].

Step 1: Apply OR operation for LUTM and RUTM separately in T_1 and T_2.

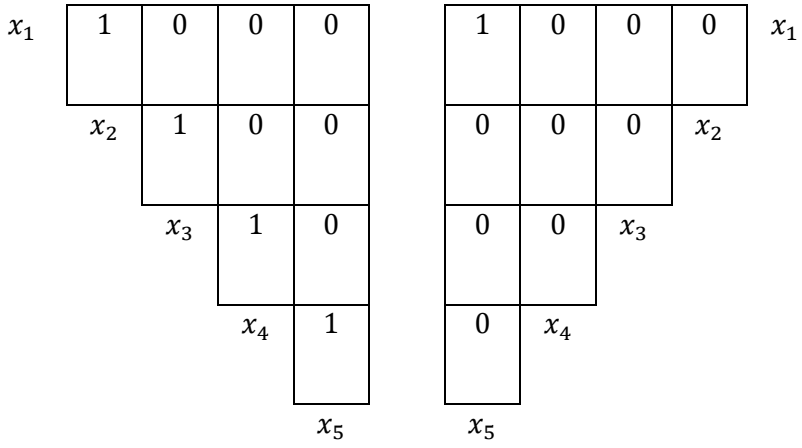




Step 2: All 0 and 1 case give 1, then we get



Step 3: Delete the largest edge or adding an appropriate edge.



4.3 Steps of Algorithms

The procedure of getting solution for GA is:

1. [Start] Randomly complete the graph which is including N initial population.
2. [Sufficiency] Calculate the fitness for each possibility, using selection mechanism.
3. [New population] Create new population which is repeated in the steps below.

a. [Selection] Choose two populations which have highest fitnesses.

b. [Crossover] Create new population by applying crossover operation.

c. [Mutation] Mutate the population according to any possibility.

d. [Addition] Add the new one in the new population.

4. [Changing] Change the old population for the new population.

5. [Test] Find the best solution in new population if it exists, or return back to step 2.

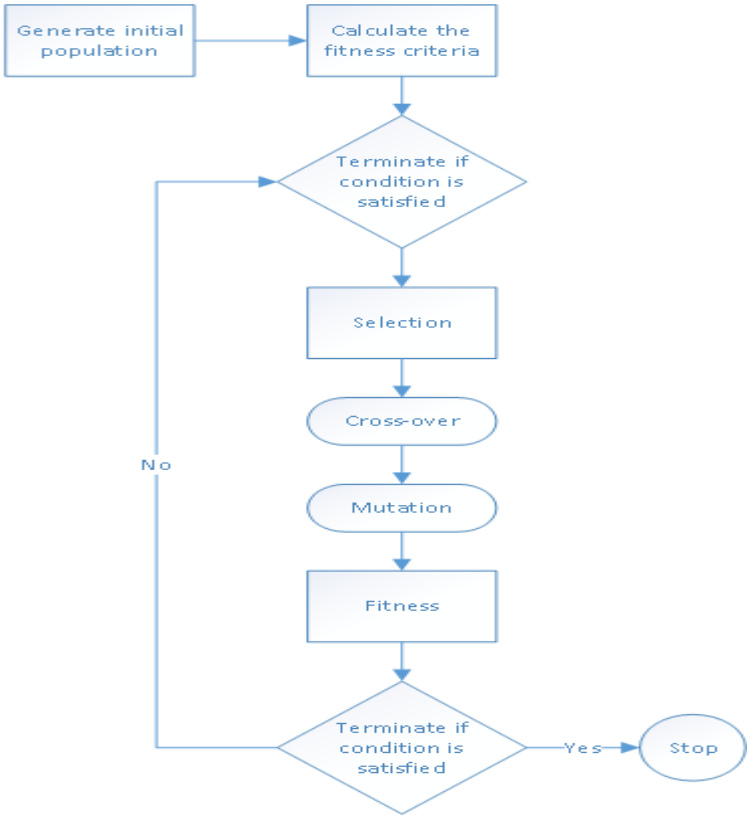


Figure 4.5: Genetic Algorithm process diagram

4.4 Example

The following two examples will help us to understand the result that is concerned with the solution of genetic algorithm.

4.4.1 The Numerical Example for ATSP in Genetic Algorithm

Consider the weighted matrix,

	x_1	x_2	x_3	x_4	x_5
x_1	∞	2	5	7	1
x_2	6	∞	3	8	2
x_3	8	7	∞	4	7
x_4	12	4	6	∞	5
x_5	1	3	2	8	∞

The value of the assignment of the above problem is

$$\sum_{i=1}^{5} \min(d(x_i)) = 1 + 3 + 4 + 4 + 1 = 13.$$

Initial population:

$$T_1: x_1 \to x_5 \to x_3 \to x_4 \to x_2 \to x_1 = 1 + 2 + 4 + 4 + 6 = 17$$

$$T_2: x_1 \to x_2 \to x_5 \to x_3 \to x_4 \to x_1 = 2 + 2 + 2 + 4 + 12 = 22$$

$$T_3: x_1 \to x_3 \to x_4 \to x_5 \to x_2 \to x_1 = 5 + 4 + 5 + 3 + 6 = 23$$

$$T_4: x_1 \to x_2 \to x_4 \to x_3 \to x_5 \to x_1 = 2 + 8 + 6 + 7 + 1 = 24$$

$$T_5: x_1 \to x_5 \to x_2 \to x_3 \to x_4 \to x_1 = 1 + 3 + 3 + 4 + 12 = 25$$

$$T_6: x_1 \to x_2 \to x_3 \to x_5 \to x_4 \to x_1 = 2 + 3 + 7 + 8 + 12 = 26$$

$$T_7: x_1 \to x_5 \to x_4 \to x_3 \to x_2 \to x_1 = 1 + 8 + 6 + 7 + 6 = 28$$

$$T_8: x_1 \to x_4 \to x_3 \to x_5 \to x_2 \to x_1 = 7 + 6 + 7 + 3 + 6 = 29$$

$$T_9: x_1 \to x_5 \to x_3 \to x_2 \to x_4 \to x_1 = 1 + 2 + 7 + 8 + 12 = 30$$

The fitness functions are

$$F(T_1) = \frac{13}{17} = 0{,}76$$

$$F(T_2) = \frac{13}{22} = 0{,}59$$

$$F(T_3) = \frac{13}{23} = 0{,}56$$

$$F(T_4) = \frac{13}{24} = 0{,}54$$

$$F(T_5) = \frac{13}{25} = 0{,}52$$

$$F(T_6) = \frac{13}{26} = 0{,}5$$

$$F(T_7) = \frac{13}{28} = 0{,}46$$

$$F(T_8) = \frac{13}{29} = 0{,}44$$

$$F(T_9) = \frac{13}{30} = 0{,}43$$

According to fitness criteria we are selecting the values which is close to 1 are T_1, T_2, T_3 and T_4.

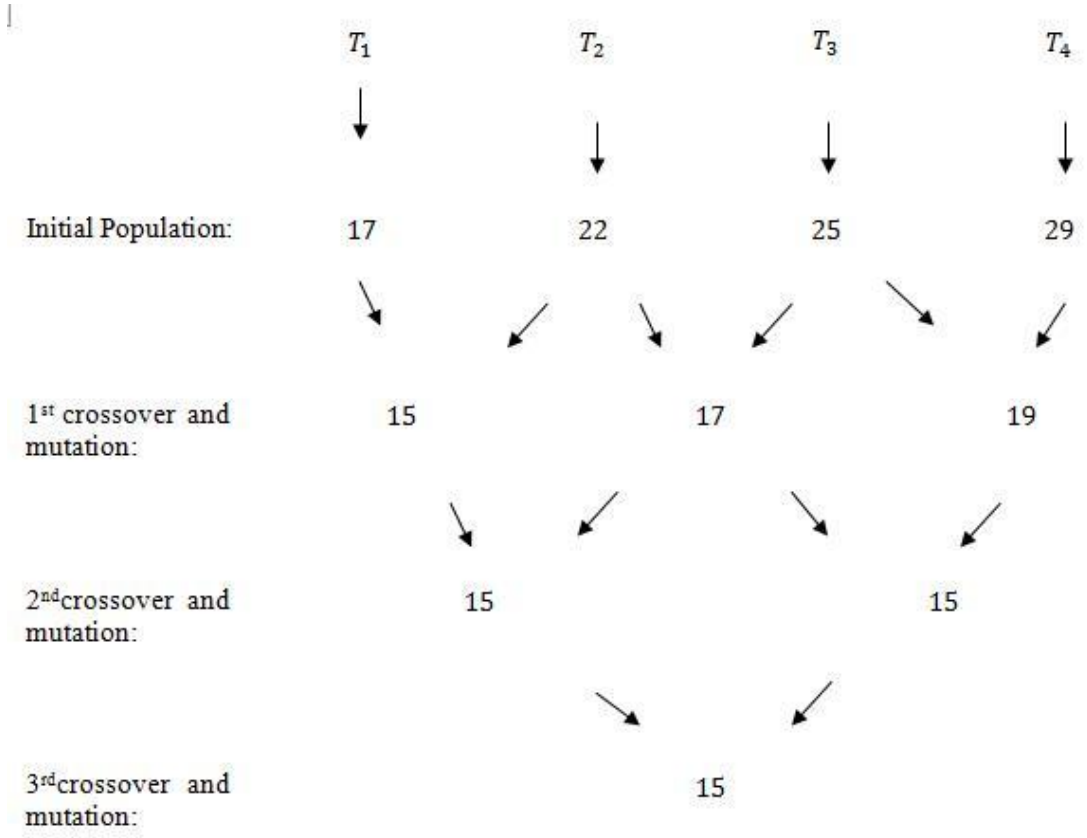


Figure 4.6: Solution set for a given example

The resultant tour will be,

$$x_1 \to x_2 \to x_3 \to x_4 \to x_5 \to x_1 = 2 + 3 + 4 + 5 + 1 = 15.$$

[8].

4.4.2 The Numerical Example for STSP in Genetic Algorithm

Consider the weighted matrix,

	x_1	x_2	x_3	x_4	x_5
x_1	∞	10	8	9	7
x_2	10	∞	10	5	6
x_3	8	10	∞	8	9
x_4	9	5	8	∞	6
x_5	7	6	9	6	∞

The value of the assignment of the above problem is

$$\sum_{i=1}^{5} \min(d(x_i)) = 7 + 5 + 8 + 5 + 6 = 31.$$

Initial population:

$$T_1: x_1 \to x_5 \to x_3 \to x_4 \to x_2 \to x_1 = 7 + 9 + 8 + 5 + 10 = 39$$

$$T_2: x_1 \to x_2 \to x_5 \to x_3 \to x_4 \to x_1 = 10 + 6 + 9 + 8 + 9 = 42$$

$$T_3: x_1 \to x_3 \to x_4 \to x_5 \to x_2 \to x_1 = 8 + 8 + 6 + 6 + 10 = 38$$

$$T_4: x_1 \to x_2 \to x_4 \to x_3 \to x_5 \to x_1 = 10 + 5 + 8 + 9 + 7 = 39$$

$$T_5: x_1 \rightarrow x_5 \rightarrow x_2 \rightarrow x_3 \rightarrow x_4 \rightarrow x_1 = 7 + 6 + 10 + 8 + 9 = 40$$

$$T_6: x_1 \rightarrow x_2 \rightarrow x_3 \rightarrow x_5 \rightarrow x_4 \rightarrow x_1 = 10 + 10 + 9 + 6 + 9 = 44$$

$$T_7: x_1 \rightarrow x_5 \rightarrow x_4 \rightarrow x_3 \rightarrow x_2 \rightarrow x_1 = 7 + 6 + 8 + 10 + 10 = 41$$

$$T_8: x_1 \rightarrow x_4 \rightarrow x_3 \rightarrow x_5 \rightarrow x_2 \rightarrow x_1 = 9 + 8 + 9 + 6 + 10 = 42$$

$$T_9: x_1 \rightarrow x_5 \rightarrow x_3 \rightarrow x_2 \rightarrow x_4 \rightarrow x_1 = 7 + 9 + 10 + 5 + 9 = 40$$

The fitness functions are

$$F(T_1) = \frac{31}{39} = 0{,}794$$

$$F(T_2) = \frac{31}{42} = 0{,}738$$

$$F(T_3) = \frac{31}{38} = 0{,}815$$

$$F(T_4) = \frac{31}{39} = 0{,}794$$

$$F(T_5) = \frac{31}{40} = 0{,}775$$

$$F(T_6) = \frac{31}{44} = 0{,}704$$

$$F(T_7) = \frac{31}{41} = 0{,}756$$

$$F(T_8) = \frac{31}{42} = 0{,}738$$

$$F(T_9) = \frac{31}{40} = 0{,}775$$

According to fitness criteria we are selecting the values which is close to 1 are T_1, T_3, T_4 and T_5.

The resultant tour will be,

$$x_1 \to x_5 \to x_2 \to x_4 \to x_3 \to x_1 = 7 + 6 + 5 + 8 + 8 = 34.$$

Chapter 5

CONCLUSION

In this thesis we investigate solution methods of TSP. We concentrated on Genetic algorithm branch and bound algorithm.

The nodes and CPU Time results for Genetic Algorithm are shown in Table 1.

Table 1: The nodes and CPU Time results for Genetic Algorithm

	CPU Time	**CPU Time**	**CPU Time**	**CPU Time**	**CPU Time**	**AVG.CPU Time**
N=5	12,85	7,13	7,36	6,18	9,23	8,552
N=10	8,06	15,4	9,91	12,66	5,96	10,398
N=50	4,81	11,43	14,15	15,38	6,35	10,424
N=60	5,68	19,3	14,8	12,06	18,26	14,02
N=100	24,1	17,31	17,96	23,18	49,03	26,316

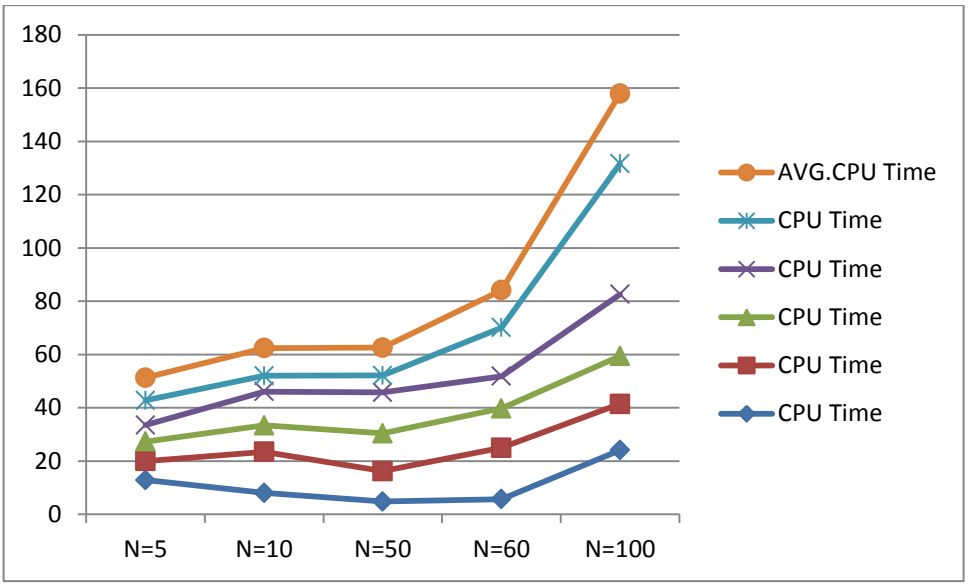


Figure 5.1: The graph of given data for Genetic Algorithm

In Table 1 given above, we provide the computer outputs of the solution of TSP via Genetic Algorithm. We obtain the average speed per second (CPU Time) of the program over the nodes. As we can see in the table above, the CPU Time speed increases in proportion with the number of nodes.

As we investigate the same problem by Branch and Bound Algorithm, we see that it takes longer time for the latter to produce the same output.

Genetic algorithms appear to find good solutions for the traveling salesman problem, however it depends very much on the way the problem is encoded and on which crossover and mutation methods are used. It seems that the methods that use heuristic information or encoding the edges of the tour (such as the matrix representation and crossover) perform the best and give good indications for future work in this area.

Generally genetic algorithms have proved useful for solving the traveling salesman problem. As yet, genetic algorithms have not found a better solution to the traveling salesman problem than the one already known, but many of the known best solutions have also been found by some genetic algorithm method.

The main problem with the genetic algorithms devised for the traveling salesman problem is that it is difficult to maintain structure from the parent chromosomes and still end up with a legal tour in the child chromosomes. Maybe a better crossover or mutation routine that retains structure from the parent chromosomes would give a better solution than the one already found for some traveling salesman problems.

Branch and bound is a well known, all-purpose optimization strategy. It requires a search tree evaluation function and provide an upper bound cost as a reference for the pruning. It is an effective technique for solving constraint optimization problems (COP's). However, its search space expands very rapidly as the domain sizes of the problem variables grow. Branch-and-bound may also be a base of various heuristics. For example, one may wish to stop branching when the gap between the upper and lower bounds becomes smaller than a certain threshold. This is used when the solution is "good enough for practical purposes" and can greatly reduce the computations required. This type of solution is particularly applicable when the cost function used is noisy or is the result of statistical estimates and so is not known precisely but rather only known to lie within a range of values with a specific probability.

REFERENCES

[1] Matai, R., Singh, S. P. and Mittal, M. L. (2010). Traveling Salesman Problem: An Overview of Applications, Formulations, and Solution Approaches, readings on the Traveling Salesman Problem, Theory and Applications, edited by Davendra, D.

[2] Laporte, G. (2006). A Short History of the Traveling Salesman Problem. *Canada Research Chair in Distribution Management, Centre for Research on Transportation and GERAD HEC Montreal*, Canada.

[3] Zambito, L., D. (2006). The Travelling Salesman Problem: A Comprehensive Survey. *Collecting Primary Data Using Semi-Structured and In-Depth Interviews in Sounders, M., Lewis, P. and Thornhill, (2003), Research Methods for Business Students.* Halow, New-York: Prentice Hall.

[4] http://en.wikipedia.org/wiki/Travelling_salesman_problem.

[5] Zelinka, R., Singh, S. P. and Mittal, M. L. (2010). Traveling Salesman Problem: An Overview of Applications, Formulations, and Solution Approaches, readings on the Traveling Salesman Problem, Theory and Applications, Edited by Donald Davendra.

[6] Fritzche, P., Rexachs, D. and Luque, E. (2010). Predicting Parallel TSP Performance: A Computational Approach. DACSO, University Autonoma of Barcelone, Spain.

[7] Balas, E. and Toth, P. (1983). Branch and Bound Methods for the Traveling Salesman Problem. *Management Science Research Report*.

[8] Khan, F. H., Khan, N., Inayatullah, S. and Nizami, S. T. Solving TSP Problem by Using Genetic Algorithm. *International Journal of Basic and Applied Sciences, Vol:9, No:10, pp.79-88.*