2009-01-01

Electric Power Distribution Optimization Using Evolutionary Algorithms

Sowmya Parimi
University of Texas at El Paso, sparimi@miners.utep.edu

ELECTRIC POWER DISTRIBUTION OPTIMIZATION USING EVOLUTIONARY ALGORITHMS

SOWMYA PARIMI

Department of Industrial Engineering

APPROVED:

Jose F. Espiritu, Ph.D., Chair.

Raphael S. Gutierrez, Ph.D.

Virgilio Gonzalez, Ph.D.

Patricia D. Witherspoon, Ph.D.
Dean of the Graduate School

ELECTRIC POWER DISTRIBUTION OPTIMIZATION USING EVOLUTIONARY ALGORITHMS

by

SOWMYA PARIMI, (B.E)

THESIS

Presented to the Faculty of the Graduate School of

The University of Texas at El Paso

in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE

Department of Industrial Engineering

THE UNIVERSITY OF TEXAS AT EL PASO

December 2009

Acknowledgements

I would like to thank my graduate advisor, Dr. Jose F. Espiritu, Assisstant Professor of Industrial Engineering Department at University of Texas at El Paso for giving me an opportunity to work with him and also for his continuous support, patience and advice in completing my thesis. It was great experience working under him and learnt many things from him.

I would also like to thank Dr. Heidi Taboada for her encouragement and support in completing my thesis. She has always given me a word of support. I would also like to thank Dr. Rafael S. Guiterrez for his time to time valuable advice and also for attending my thesis defense and assessing my research work. I would also like to thank Dr. Virgilio Gonzalez for giving his valuable time towards attending my thesis defense and assessing my research work.

And last but not the least I am grateful to God, my parents and friends, specially my father Mr. Parimi. Sambasiva Rao and my mother Mrs. Parimi Lakshmi for their continuous love, affection, support and inspiration. I would like to dedicate this work to my parents.

Abstract

In the present research, a new evolutionary algorithm is developed to solve the component allocation problem in electricity distribution systems. The problem addresses the upgrade/design of an electricity distribution system with the objective of minimizing expected system downtime subject to cost and repair time constraints. The algorithm is tested on the Dual Element Spot Network (DESN) configuration which is one of the most commonly used configurations by the power industry. This algorithm is demonstrated with two examples.

Table of Contents

List of Tables

List of Figures

INTRODUCTION

The present research is related to power distribution systems. In the present research, a new evolutionary algorithm to solve power systems design optimization problems is developed. The thesis paper has been divided into five chapters each explaining different segments of the research. Chapter 1 will focus on what power systems are and different segments of power systems: Generation, Transmission and Distribution. Each segment is explained clearly and the working methodology will be explained to understand the power system better.

In the chapter 2 different optimization techniques like evolutionary techniques will be presented. Some of the evolutionary techniques like genetic algorithms simulated annealing, tabu search and ant colony optimization are discussed. The purpose of the chapter 2 is to get to know some of the available techniques, their mechanism and their fitness to our objective function. And since the problem taken up in the research is an optimization problem, it has some constraints. Sometimes it is difficult to handle these constraints. For handling such kind of constraints, constraint handling techniques are introduced in chapter 3.

Chapter 3 purely focuses on the different constraint handling techniques like penalty functions, static function, death penalty functions and some other constraint handling techniques which can be implemented to solve the constraint optimization problem. In the present thesis an optimization problem involved with some constraints is solved. To solve optimization problem with constraints it is necessary to use some constraint handling techniques should be used or implemented to get promising results. Hence in this chapter some of the constraint handling techniques will be introduced.

In Chapter 4, electricity distribution system evaluation and optimization is developed. Firstly, some of the system parameters will be evaluated for the distribution system. The evaluated system parameters are expected system downtime, outage rate, repair time and cost. For calculation of these particular factors different techniques are available among which approximation method being the suitable one is used. As in the previous chapters' optimization techniques (chapter 2) and constraint

handling techniques (chapter 3) are learnt, they have been applied to an electric distribution problem after evaluating the parameters. An algorithm is developed in this chapter 4 which is run on MATLAB for specified number of generations. At the end of the program, a distribution system has been developed with required components and the factors for the system are evaluated. And finally in the chapter 5, the final conclusions of the thesis are produced and some of the future work will be suggested.

Chapter 1: Introduction to Power Systems

1.1. Introduction:

The primary objective of the power industry is to provide electricity to the customers and satisfy the needs of the customers as economical as possible with a reasonable assurance of continuity and quality Billinton & Allan (1996, 1998) and Diaz J. F *et al* (2005) explained broadly about the power system and its objective in their work. The power system is a large complex network and it is generally divided into three main segments namely generation, transmission and distribution.

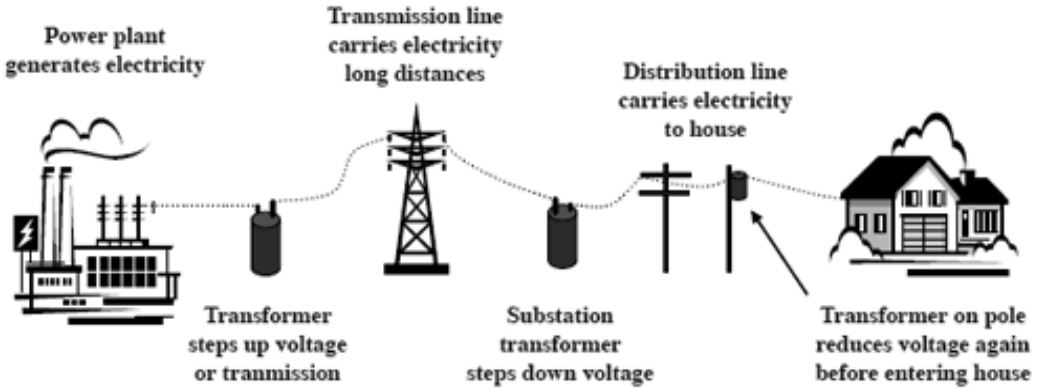

Figure 1: Electric Power System

Each segment has its own importance and own function. A shown in the figure.1, power is generated in the power generation plant, transmitted through transmission lines to distribution lines from where the electricity is distributed among residential areas, commercial areas, hospitals, etc. This being a short introduction about the power system, in the upcoming sections an elaborated explanation about each segment is explained.

1.2 Description of the Electric Power System:

In this section, a brief explanation of each segment is given to understand them more better and to know the functions of each segment which constitute to form a power system.

1.2.1. Generation: Electricity generation is the process of creating electricity from other means of sources. Michael Faraday was the first to discover the fundamentals of electricity generation which are still in use. Electric generation is the first process in the delivery of electricity to the customers. Electricity is generated at the power station by electrochemical generators, primarily driven by heat engines fueled by chemical combustion or nuclear fission. There are other means of producing electric power such as kinetic energy of flowing water and wind, geothermal power and solar photovoltaic. Today the most reliable electricity generators are coal, natural gas, nuclear, hydroelectric and petroleum also including with a small amounts of solar energy, wind generators and solar generators. In thermal power plant, the chemical energy of fossil fuels such as coal, natural gas is transformed into thermal, mechanical and finally into electrical energy. The main advantage of a thermal power plant is their availability; they can operate continuously for a year between shutdown and maintenance. Hydro power is a renewable source of energy and the power is produced by the gravitational force of falling or flowing water. It's a safe power source in terms of pollution and it comprises 20% of the world's electricity. Wind power plant produces electricity by converting wind energy to electricity by using the wind turbines. It has been stated that world 1.5% electricity is produced by windmills and the production rate is increasing every year. And also wind power is again a renewable source and eco-friendly. Using controlled nuclear reactions the energy is being produced from a atomic nuclei in a nuclear power plant. It is been noted that 14% of world's electricity production comes from here.

1.2.2 Transmission: Transmission is the second segment of a power system. A transmission line is the material medium or a structure that forms all or part of the path from one place to another to transport energy. Electric transmission lines carry energy or electricity from the power generation plant to the local substations, where they are connected to distribution lines. But sometimes the electric power transmission directly connects energy sources to the consumers in populated areas. This may allow

exploitation of low grades energy sources such as coal. Usually transmission lines uses three phases alternated current (AC). To reduce the energy or electricity loss during the transmission, the electricity is transmitted at high voltages. Multiple lines are provided in the network, which can be routed from any power plant to any load center through a variety of routes, based on the economics of transmission path and cost of power. Much analysis is done by the transmission companies to determine the maximum reliable capacity of each line, which, due to system stability consideration, may be less than the physical and chemical limit of the line. The transmission system usually carries components such as power lines, circuit breakers, switches and transformers. Usually the electricity is generated at a very medium voltage between about 2300 volts and 30,000 volts which is being stepped up by the power station transformer to a higher voltage for transmission over long distances. The transmission lines are usually categorized into **a)** overhead transmission and **b)** underground transmission lines.

a. Overhead transmission: The overhead transmission lines suspend from the poles or towers. The overhead transmission lines are the lowest cost methods for transmission of large quantities of electric power. The poles or towers can be either made of wood, steel, concrete, aluminum and sometimes reinforced plastics. The bare wires are mostly made with aluminum and also with copper for medium voltage and low voltage connections. Improved conductor materials and shapes are made in overhead power transmission are again sub classified into low voltage (<1000 volts), medium voltage (between 1000 V and 33 kV), high voltage (33 kV and 230 kV), extra high voltage (over 230 kV to 800 kV) and ultra high voltage (>800kV). The main purpose of the overhead transmission lines is to maintain required clearance between the conductors and the ground to maintain safety. This factor is dependent on the voltage of the line.

b. Underground transmission: The buried transmission lines are referred to as underground transmission lines. The underground transmission lines are very infrequent because the materials such as cables and insulating fluids are quite expensive. In the past reviews or studies it has been proved that the underground lines are nearly four to ten times more expensive than the overhead transmission lines.

Hence the underground transmission lines are rarely supported due to its cost and reliability. There some more disadvantages with the underground lines like repairs would be difficult to spot and is a time consuming process including difficulty of operation. Besides they have some advantages such as less damage due to weather conditions, reduced electromagnetic field emission and the narrow surrounding strip for the underground cables (1-10 meters) is considerably less than the overhead line (20-200 meters). But the advantages in some cases outweigh the disadvantages as discussed before in terms of cost and maintenance. The underground transmission lines are used to transmit power across in densely populated areas, places where land is unavailable for overhead lines and rivers.

1.2.3. Distribution: The distribution system is the third segment of the power system and plays a vital role in the total electric supply system, as it provides link between a utility's bulk transmission system and its consumers. It has been reported that 80% of all the customer interruptions are due to failure in the distribution system [Chowdhary A and Koval D.O (1998)]. A typical distribution system would mainly include power lines, transformers, buses, breakers and sometimes electric meters. Modern distribution system behaves as a primary circuit, leaving the sub-station and ends at the socket of the customer meter. A wide range of materials and methods are being used by the companies provided the same result. The energy leaves the sub- station in a primary circuit as three phases. The distribution network configurations are broadly divided into two types, radial and interconnected configurations (as shown in Fig: 2). A radial configuration leaves the station and passes through the network area with no normal connection to any other supply while an interconnected configuration has multiple connections to other points of supply.

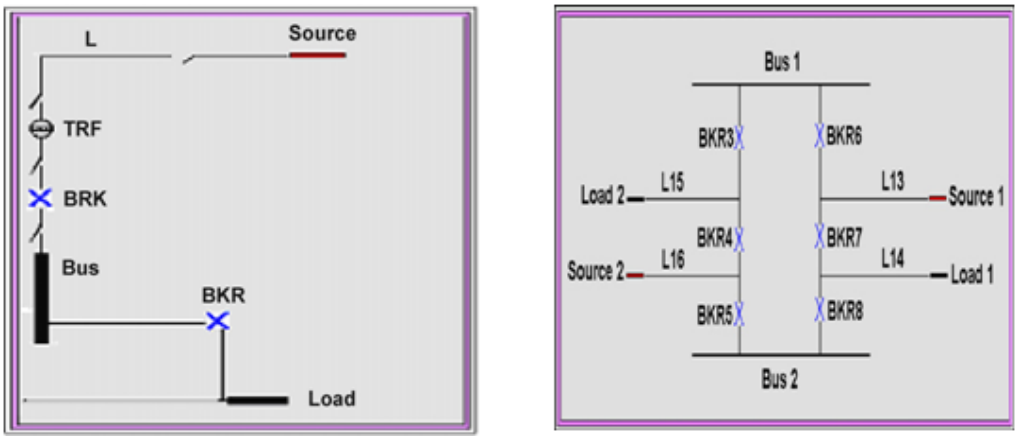

Figure 2: Radial and Interconnected distribution configurations

In general, the radial configurations are found in rural areas while the interconnected configurations are found in the urban areas. The advantage of interconnected configuration is that in the event of a fault or required maintenance a small area of network can be isolated and the remainder kept on supply. Breaker-and-a-half, breaker-and-a- third and Dual Element Spot Network (DESN) configurations are some of the examples of the interconnected configurations.

1.3. Conclusions:

In the present chapter a brief view of power system is presented. The present research focuses much on electric distribution system, explained in a detailed manner in the present chapter. In the real world, the distribution system is encountering lot of optimization and allocation problem. In the present work similar kind of optimization problem is taken and solved for system allocation. An objective function is defined in the later chapters for which to solve some of the methods are used. In the next chapter of this thesis some of the evolutionary techniques will be introduced. But since the objective function is an optimization it has some constraints, to handle these constraints some of the constraint handling methods will be introduced in the chapter 3.

Chapter 2: Heuristic Optimization

2.1. Introduction:

In the previous chapter an introduction to the area of power systems was presented. In the present chapter, a brief introduction of what a heuristic optimization is and some of the heuristic techniques will be discussed in this chapter in the later sections. Heuristic search or optimization is a new approach for solving complex problems. These techniques are very flexible and can be applied to many types of optimization problems with diverse objective functions and constraints. Quit often the heuristic optimization mimics successful strategies found in nature: like genetic algorithms mimics the behavior of species generation, simulated annealing is based on how crystals emerge when materials in cooling and particles find a structure that minimizes the energy balance and many more; which will be discussed in an elaborated manner in the upcoming sections.

2.2. Purpose and Application:

Several factors can make optimization problems fairly complex and difficult to solve. Multiple decision variables, complex nature of the relationships between decision variables and the outcome, presence of risk or uncertainty and existence of one or more complex constraints on the decision variable are some of the factors in optimization problems. The purpose of these heuristic optimization problems is to solve such kind of optimization problems filled with complex constraints. In the present world, these techniques are being adapted so quickly due to their flexibility and efficiency. Some of the application where these methods can be applied or can be said have already been applied are in combinatorial optimization like travelling salesman problem, set covering, in aircraft design, resource allocation, facility scheduling, trajectory planning, fiber design.,. etc.

Hence by now knowing how important these heuristic optimization techniques are and some of their application, in this chapter some of the heuristic techniques will be introduced. Some of the

techniques which will be focused in this chapter are Genetic algorithms, Ant Colony Optimization, Simulated Annealing and Tabu Search. The problems such as NP- hard can be solved using these techniques. These techniques have two advantages with them: 1) less time consuming and 2) being insensitive the system is very robust. Now in the later section a brief introduction to each of the techniques is provided to understand them and their working clearly.

2.3. Genetic Algorithms:

John Holland from the University of Michigan introduced the concept in 1975. According to Goldberg, Genetic Algorithms (GA's) are stochastic search techniques for approximating optimal solution with complex search space. A Genetic Algorithm is an Evolutionary Technique to solve problems such as numerical and combinatorial optimization problems. Homaifar *et al*., [1994], in his work presented application of genetic algorithms to non- linear constrained optimization problems. These genetic algorithms can be applied to various optimization problems where finding a exact solution can be difficult and impossible sometimes and also when exact algorithm is unknown. Michalewicz Z and Janikow C. Z. [1990]; presented some novel ways to handle such numerical optimization problems in their work. As discussed earlier GA's are applied in various fields of engineering and science, industrial engineering applications being one of them. In Industrial Engineering problems such as travelling salesman problem, scheduling facility layout and computer aided design and many other are quite hard and complex due to high complexity of the objective function and a significant number of constraints. Michalewicz Z *et al*., [1996] in his paper presented some constraint handling heuristics for evolutionary computation technique used in industrial engineering examples with some examples. Also many researchers have combined different evolutionary techniques to produce more prominent results. Youssef L. Abdel- Magid *et al*., [1999]; in their piece of work presented a new algorithm, combination of genetic algorithm, tabu search and simulated annealing to solve unite commitment problem. In From decades many heuristic methods have existed for solving complex optimization problems. However there is no

such method which is fully efficient in solving all the optimization problems in the real world and engineering field. Genetic Algorithms are based on the Darwin's concept of natural selection and evolution. GA's are intelligent search techniques used for solving optimization problems. Under the category of heuristic they are sub characterized as global search heuristics. Genetic algorithms have been applied in various fields such as computational science, engineering, manufacturing, chemistry, etc. It is useful to formally introduce some of the keywords or biological terminology that will be used throughout this research. The two most important words in GA's are genes and chromosomes. The genes are the operations and chromosomes are the sequence of entire solution. A typical GA requires: 1) a genetic representation of the solution domain and 2) a fitness function to evaluate the solution domain. Once the genetic representation and fitness function are defined, GA proceeds to initialize a population randomly and then improve it through repetitive application of mutation, crossover and new population. The basic operators of GA are explained below:

1. **Reproduction**: A new generation is being created. Including from the starting generation, strings are reproduced with a probability respective to their fitness value. Strings with good properties have more chance of survival than the strings with bad properties. The principle followed is "survival to fittest"
2. **Crossover**: Crossover operator combines two parent or chromosomes to produce a new offspring or chromosome. The basic idea behind crossover is that offspring would be better than the parents if it takes the best features/characteristics from each of the parents. The crossover probability is set by the user and is usually low. There are different types of crossover available and among them an appropriate one can be chosen which best works for the solution. Some of the popular crossover are listed and explained below:
    - **One Point Crossover**: A random crossover point is selected within a chromosome and the genes are interchanges between the two parents beyond this crossover point. Hence two new offsprings are produced. Below is the example explaining the one point crossover.

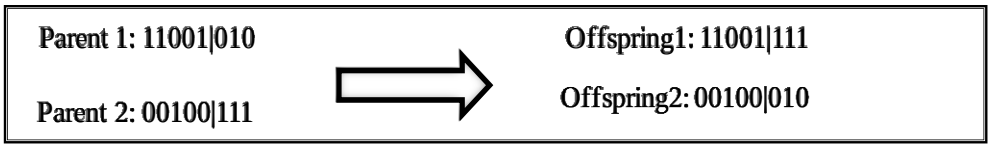

Figure 3: One Point Crossover

- **Two Point Crossove**r: A crossover operators randomly selects two points in the chromosome. The genes are interchanges between these two points of the two parents as shown below, hence producing two offsprings.

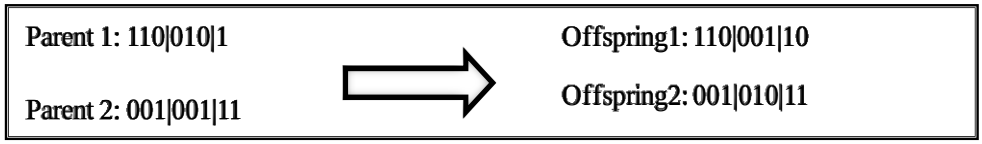

Figure 4: Two Point Crossover

- **Uniform Crossover**: With some probability the crossover operator mixes the two parents to form two offsprings. This is explained clearly with the help of an example. For instance if the probability is chosen as 0.5, then half of the genes between the two parents are exchanged between the two parents. As shown below, there are two parents before, after a uniform crossover is operated on these two parents half of the genes are exchanged between the parents forming two new children.

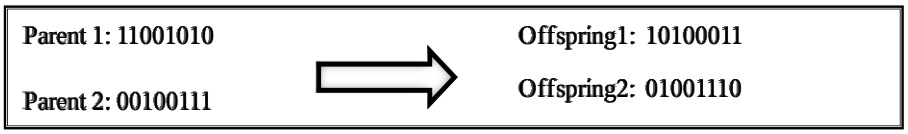

Figure 5: Uniform Crossover

3. **Mutation**: In the mutation operator, one of the gene is selected randomly based on some probability and is replaced with some other gene. This is done for having best out of better. Mutation is important part of genetic algorithm as it prevents the population from stagnating at any

local optima. Mutation rate depend on the problem being solved. In general the probability of mutation should be very low around 1%. If this probability is kept high, the search will turn into a primitive random search. Genetic algorithms usually have different kinds of mutation among which some are listed and explained below:

- Flip Bit: This mutation operator can only be used for binary genes. As the name indicates, a mutation operator just flips the value of the selected genes. For example: 0 goes to 1 and 1 goes to 0.
- Boundary: Boundary mutation operator can be used only for integer and float genes. A mutation operator replaces the chosen gene with upper or lower bounds, process being random selection.
- Uniform: This type of operator is also used for only integer and float genes. The mutation operator replaces the value of the chosen gene with a uniform random value selected between the user- specified upper and lower bounds of the genes.

These are some of the different types of available mutation operators, depending upon the criteria of the problem, a suitable operator can be chosen.

A problem should have the following components to apply genetic algorithms [2]:

1. Chromosomal representation of solution for the problem
2. Initial population creation of solutions
3. An evaluation function that plays the role of the environment, rating solutions in terms of their "fitness"
4. Genetic operators
5. Parameter values such as population size, probability of mutation, **** probability of applying genetic operators, etc

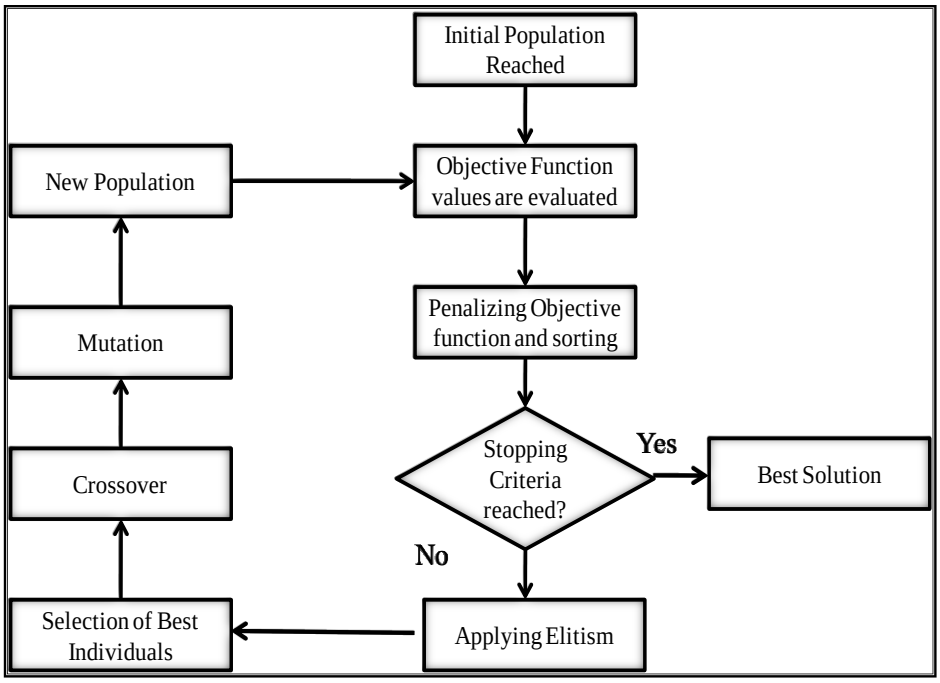

Figure 6: Genetic Algorithm Flowchart

The above flow chart gives a brief description of the genetic algorithms how they work. To understand them even better, here is a small introduction what is done in each step.

Start – A population of n chromosomes in generated randomly.

1. **Fitness** – The fitness function f(**x**) of each chromosome **x** is evaluated.
2. **New population** – a new population is created by following the steps below:
    a. **Selection –** two parents from the population are selected according to their fitness.
    b. **Crossover –** a new offspring is created from the two parents, if no new offspring is created it means the two parents are similar.
    c. **Mutation –** with mutation operation the new off springs are positioned at each point.
    d. **Accepting –** the new offsprings are now placed in the new population.
3. **Replace –** the new generated population is used in here for further algorithm calculation.
4. **Test –** if the program reaches end point, return the best solution at current position.
5. **Loop –** go to step 2.

2.4. Ant Colony Optimization:

The Ant Colony Optimization (ACO) was initially introduced by Marco Dorigo in his PhD thesis [Marco Dorigo, 1992]. The developed algorithm was aiming to search for an optimal path in the graph, based on the ant behavior of seeking food to their homes in the shortest path. The Ant Colony Optimization algorithm is a probabilistic technique for solving computational problems which can be used to find the good paths through graph. Though it was developed to solve small optimization problems in the beginning, but now the ACO is being used in wider class of Numerical problems. Marco Dorigo *et al* [1996] introduced a new search methodology based on a distributed autocatalytic process and its application to the solution of a classical optimization problem. Dorigo and Gamberdella [1997] in their paper introduced ant colony optimization where it was applied to the Travelling Salesman Problem. The ant colony optimization is a heuristic developed with following characteristics:

1. It is *versatile*, means ACO can be applied to similar kind of problems. For instance, this heuristic has been applied to Travelling Salesman Problem (TSP), hence there is an extension of this algorithm straightly that can also be applied to Asymmetric Travelling Salesman Problem (ATSP). The asymmetric travelling salesman problem is a almost a travelling salesman problem in which the distance between the nodes is not symmetric. The ATSP is more difficult than TSP.
2. Because of its *robustness* ACO can be applied to different combinatorial optimization problems with small changes in the algorithm such as Quadratic Assignment Problem (QAP) and Job-Shop Scheduling (JSP).
3. It is a *population based* approach. It allows exploitation of positive feedback as a search mechanism. Parallel to implementations the system is made amenable.

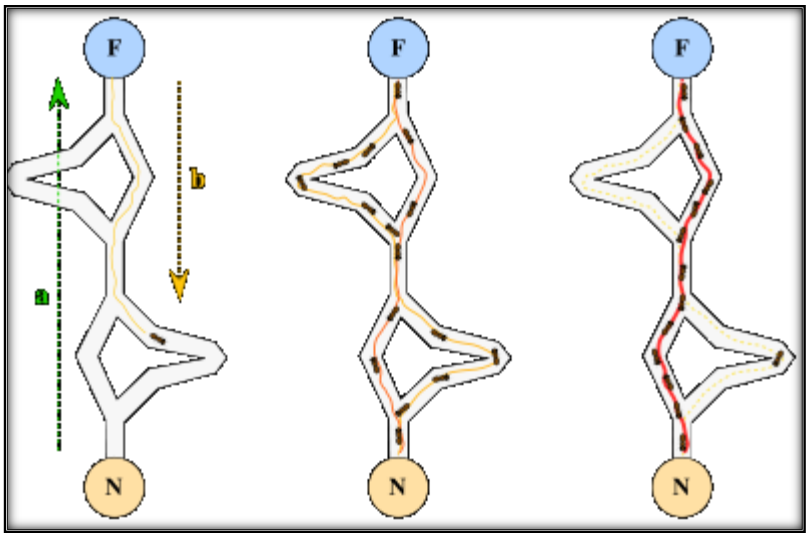

Figure 7: Ant Colony Optimization

To understand better how the ACO or algorithm works in solving optimization problems, a summarized methodology is written below.

1. As shown in Figure 7, the ant finds the food source (F), travels in path 'a' and then travels to colony (N) leaving behind a trail of pheromones for the later ants to follow the path.
2. Ants travel randomly in different ways but the path which is travelled most is strengthened and is followed by the ants. Thus that route becomes the shortest route.
3. Hence, ants takes the shortest route and the paths where the pheromones where laid gets evaporated.

When the ants find their best route (shortest path) between the colony and food source all the ants follow the same path, hence leaving behind the other paths where the pheromones get evaporated. In the same way while solving combinatorial optimization problems the local optimum solution reaches the global optimum, thus being the best solution. In the past some of the research was done on power systems using Ant Colony optimization. J.F. Gomez (2004) *et al.*, used ant colony optimization technique to optimize a radial distribution network providing minimum investment and loss cost solution. Ant Colony optimization also helps in solving NP-hard problems. Ruey – Maw Chen *et al*

(2008) presented a modified scheme of ant colony optimization algorithm for solving flow- shop scheduling problems, it being an NP- hard sequencing scheduling problem. Ant colony optimization is also implemented in the medical field. Myun- Eun Lee (2009) *et al*., described a segmentation method for brain MR (Magnetic Resonance) images using an Ant colony optimization (ACO). Hence, by seeing so many researches in different area and field it can be said that ACO is a new meta- heuristic algorithm and a successful paradigm of all the algorithms which take advantage of insects' behavior.

2.5. Simulated Annealing:

Simulated Annealing (SA) is a random search technique which exploits an analogy between the way the metals cools and freezes into a minimum energy crystalline structure and the search for a minimum in a more general systems; it forms the basis of an optimization technique for combinatorial and other problems. Simulated Annealing (SA) was first introduced by S. Kirkpatric *et al* (1983) [13, 14].

SA was mainly developed to deal with non- linear problems. SA approaches the global maximization problem similar to using a bouncing ball that can bounce over mountain from valley to valley. It begins to bounce at high temperatures (peak) and as the temperatures (inclination) decreases the bounce cannot bounce as before and slowly settle to become trapped between small valleys. A generating distribution generates possible valleys or states to be explored. An acceptance distribution depends on the difference between the function values of present generated valley to be explored and the last saved lowest valley. It probabilistically decides to stay in a lower valley or to bounce out of it. All the distributions depend on the temperature. Its been proved from various studies that with controlled cooling temperatures SA can find global optimum. The drawback is the time, takes long time. Fast annealing and very fast simulated annealing or adaptive simulated annealing are each in turn exponentially faster and overcome the time problem. SA's major and main advantage is an ability to avoid becoming trapped in local minima. Following is the Figure. 8 explaining SA in the form of a flow chart.

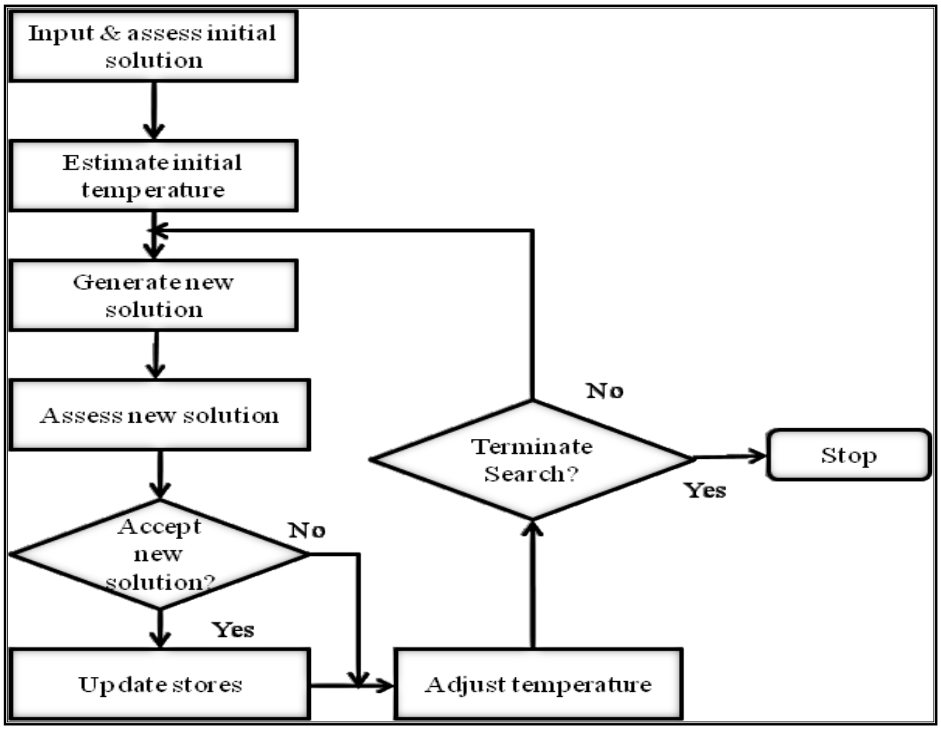

Figure 8: Simulated Annealing Flowchart

For the algorithm to run the following basic elements must be provided:

- Representation of possible solutions.
- Generator of random changes in solution.
- Mean of evaluating the problem function.
- Annealing schedule – initial temperature and rules for lowering it as the search progresses.

SA has been applied in various fields of power system optimization problems. Some of the researchers have combined SA with other meta- heuristic techniques like genetic algorithm to give better solutions. Deeb N (1992) e*t al.,* (1992) explained the theory of SA and its application in the unit commitment, maintenance scheduling and reactive power sources planning. He also showed the practical advantages and disadvantages of SA. Abido M. A. (2000) proposed a simulated annealing approach for the robust design of multi machine power system stabilizers. System decomposition is a discrete number

combinatorial problem. Hiroyuki Mori *et al* (1994) proposed a parallel simulated annealing technique to decompose a power system for voltage control; this is was in turn tested on an IEEE 30, 57 and 118 node systems.

SA is flexible, versatile and its ability to approach global optimality makes the algorithm advantageous. They can deal with highly non- linear models, chaotic and noisy data and many constraints. Besides SA also has some weakness. Lot of choices are required to turn SA into an actual algorithm. Time taken comparative with algorithms is pretty high.

2.6. Tabu Search:

Difficulty in optimization problems encountered in practical settings such as telecommunications, logistics, financial planning, transportation and production has motivated in development of optimization techniques. Tabu search (TS) is a higher level heuristic algorithm for solving combinatorial optimization problems. It is an iterative improvement procedure that starts form an initial solution and attempts to determine a better solution. Tabu Search basic idea was described by Fred Glover in 1986. It has now become an established optimization approach that is rapidly spreading to many new fields. In the past decades Tabu search has been proved to be effective heuristic algorithm for handling combinatorial optimization problems [J. A Bland (1991 and F. Glover (1989, 1990, 1993)]. Tabu search algorithm uses flexible memory search history preventing entrapment in local optima. It is an interactive search technique, starts with an initial solution and moves towards or can be said progresses towards neighborhood by applying a series of modification. F. Glover (1990) has proved that in some cases under certain conditions, tabu search algorithm can yield a global optima with probability one. Simulated annealing is based on physical process of metallurgy, Genetic algorithms imitate the phenomenon of evolutionary reproduction, ant colony optimization mimic the behavior of ants while tabu search derives and exploit a collection of principles of intelligent problem solving. Hence it can be stated that Tabu search is based on both the concepts of artificial intelligence and optimization [F. Glover and M. Laguna,

2000]. Generally TS is unique because of its ability to entrapment in local optimal solution and prevent repeated solution by using memory of search history. To learn TS to get familiar with basic elements of TS is necessary. Hence below are some of the basic elements and their definition:

- Current Solution: It is set of optimized parameter values at any iteration. And generates the neighbor trial solution.
- Moves: Characterize the process of generating trial solutions that are related to current solution.
- Set of candidate moves: It is the set of all possible moves or trial solution in the neighbor of current solution.
- Tabu restrictions: There are certain voidable rules that are applied to tabu search. The moves are forbidden to a certain size and are called as tabu list. This is to avoid cycling and prevent returning to the local optimum. Tabu list plays a vital role in the search of high quality solutions. A simple choice of the tabu list size in a range centered at 7 seems to be quite effective. Generally, the size of should grow with the size of the problem.
- Aspiration criteria: This rules over rides the rules of tabu restriction. If a particular move is forbidden by tabu restriction, the aspiration criteria, when satisfied can make that move allowable. Different forms of aspiration criterions are available according to the problem.
- Stopping Criteria: These are the certain conditions under which the tabu search will be terminated.

Tabu search being so popular till now, many researchers have used tabu search for many optimization problems. The redundancy allocation problem (RAP) is a complex combinatorial optimization problem, which plays a vital role in the industrial application. For this reason, Mohmed Ouzineb *et al* 2006 , have presented a new tabu search meta- heuristic optimization method to solve the redundancy allocation problem for multi- state series- parallel system. Sara Carcangiu *et al* 2008, presented scalarization of vector problem is performed by introducing fitness functions that controls both

pareto and optimality of the solutions, and the uniformity in the Pareto front sampling using tabu search meta- heauristic approach for electromagnetic devices. A capacitor placement problem is kind of tough to solve in terms of global optimization due to high non- linear and mixed integer problem. The complication of any problems increases with number of variable. A transmission network optimal planning is one such problem and is a non- linear , large scale combinatorial optimization problem. Wen and Chang (1997) presented a new method in their paper to solve the single stage optimal planning problem for a transmission network given future generation and load demands, subject to overloads and right of way constraints. Mori H and Ogita Y 2000, presented a new method to determine capacitor placement is distribution systems using parallel tabu search technique to evaluate better solution in terms of computation efforts and accuracy. But not only Tabu search but sometimes tabu search is being coupled with other evolutionary algorithms to give accurate solutions. Mantawy A.H. (1996) *et al*., integrated genetic algorithm, simulated annealing and tabu search to solve a unit commitment problem. In their work, tabu search generated new population in the reproduction phase of genetic algorithm and simulated annealing accelerated the convergence of genetic algorithm. A new algorithm is developed by fussing certain characteristics of the tabu search method into standard simulated annealing technique. The performance of the newly developed algorithm was tested on various examples by Vasconcelos J. A. (1996) *et al*., and insulator geometry is optimized using the new algorithm in order to test it in electromagnetic.

2.7. Conclusion:

In the present chapter a review of different heuristic optimization techniques was introduced. As explained in the chapter these Evolutionary Algorithms (EA's) are very effective in solving optimization problem. The exact solution to the optimization problems can be obtained through these EA's.

Chapter 3: Constraint Handling Techniques

3.1. Introduction:

In the previous chapter a brief introduction of evolutionary algorithms was given. Evolutionary algorithms including the heuristics discussed in the chapter 2 are used to solve numerical optimization problems. Evolutionary algorithms have been successfully applied to solve optimization problems in the stream of science and engineering [Goldberg D. E, 1989]. EA's were specifically designed for solving unconstrained problems, but in real world problems are NP-hard or NP-complete and have constraints. Hence researchers in the past decades have tailored constraint handling techniques into EA's. The optimization problem can be single objective or multi objective problem. The single objective optimization problem aims to find a single solution reflects best objective function solution; whereas multi- objective problems aims to find a set of non- dominated solutions close to Pareto Optimal Set. In an optimization problem it is always necessary to satisfy the constraint. With the increase of number of constraints the complexity of the problem increases. The optimal solution of the optimization problems must satisfy all the constraints in the problem and thus are posed as constrained optimization problems. The constraints can be classified based upon criticality, number, metric and difficulty.

To handle the constraints in an optimization problem, some of the constraint handling techniques were introduced and are designed to solve constrained optimization problems. The necessity of approaching the boundary between the feasible and infeasible search space for many constrained optimization problems is a paramount challenge for every constraint handling technique [G. Leguizamon and C. C. Carlos, 2009]. One of the popular methods to handle these constraints is by using penalty functions when mainly using evolutionary algorithms. A detailed overview of the penalty function is explained in the coming sections. An efficient and adequate constraint handling technique is a key element in the design of competitive EA's to solve complex optimization problems [K. Janusz, 2009].

Usually the solutions obtained in an evolutionary algorithm are always located in the feasible region [Y. Nakagawa & K. Nakashima (1977) and D. H. Shi (1987)]. In the present research an evolutionary algorithm (genetic algorithm) is used solve an optimization problem and a constraint handling technique is also utilized. As discussed previously with a dedicated work towards these unconstrained problems some of the handling techniques have been evolved such as: special encodings and operators, decoders, etc but amongst them most of them are based on penalty functions. May be a constraint is violated continuously/ frequently but still the solution can easily be made feasible. Conversely, a resolving a constraint violation may be difficult, but occur rarely in the search [A. E. Smith and D. W. Coit, 1995]. In this chapter in the coming section, different constraint handling techniques and how they work will be explained further.

3.1 Methods of Handling Constraints:

The constraints can be linear and non-linear with both equality and inequality constraints. As discussed earlier the solution can be feasible and infeasible. The infeasible solutions are not ignored because of large fraction and waste efforts, infeasible spaces provide knowledge about fitness landscape, optimal solutions are close to the feasibility boundary and lastly the infeasible regions build bridges between feasible regions, and importantly the individuals to move between them. Basically efforts have always been put to get the best solution from the evolutionary algorithms. In this chapter, some of the constraint handling techniques will be discussed and from among them one of the techniques is used to solve the electricity distribution systems optimization problem. Some of the most commonly used constraint handling techniques are penalty functions, repair approach, purist approach, separatist approach and hybrid approach. A brief description of each of these methods in presented in the next sections.

The first approach is the elimination process i.e. infeasible chromosomes are eliminated from the group so that they do not participate in further reproduction. This strategy is classified as the death penalty function. This method is good when the search space contains more of feasible solution. The

second approach dealing with the infeasible solutions is the repair approach that corrects the infeasible solutions generated. Since it repairs the infeasible solutions it is termed as repair approach. The drawback of this approach is that it can be limited to certain applications. A special repair algorithm is designed when there is no standard approach. The third approach or strategy is the decoder strategy. This technique uses a special mapping representation which generates feasible chromosomes only. The advantage with this approach is that it never generates infeasible chromosomes. But there is also a disadvantage with this approach; it does not consider any points outside the feasible region. The process is expensive and is tailored according to the problem. The fourth strategy in handling the infeasible solutions is the penalty strategy. The solutions are generated without considering the constraints and the penalized by decreasing the goodness of the evaluation function. The constraints are handled by transforming constrained problem into unconstrained problem by injecting some penalty into infeasible chromosomes. Penalty is calculated through a penalty function [L. Guillerno and C. C. Carlos, 2009].

3.2 Penalty Functions:

Penalty functions have been used to solve constrained optimization problems and have proved to be efficient methods. Penalty functions converts a constrained problem into an unconstrained problem by introducing a penalty into the objective function. Schwefel (1995) [H. P. Schwefel, 1995] proposed that there are three degrees of penalty functions: barrier methods, partial penalty functions and global penalty functions. Barrier methods do not consider any infeasible solutions, in partial penalty function a penalty is applied if and only if it is near the feasibility boundary and in the global penalty functions a penalty is applied throughout the whole infeasible region. Considering combinatorial optimization problems, they are solved using Lagrangian Relaxation Method; the method works like penalty with a slight variation. The toughest constraints of the problems are temporarily relaxed to avoid the distance from the feasible region. The penalty function method is explained in the present section. First, an optimization problem as below is considered.

Min $f(x)$

Subject to: $\boldsymbol{x}\ \varepsilon\ A$

$\boldsymbol{x}\ \varepsilon\ B$

Where $\boldsymbol{x}$ is a decision variable vector, $\boldsymbol{x}\ \varepsilon\ A$ are constraints comparatively easy to solve and $\boldsymbol{x}\ \varepsilon\ B$ are difficult constraints to satisfy. When some penalty is applied to the above problem then it can be reformulated as below:

Min $f(x) + p(d(\boldsymbol{x}, B))$

Subject to: $\boldsymbol{x}\ \varepsilon\ A$

Where $d(\boldsymbol{x}, B)$ is the distance of the solution vector $\boldsymbol{x}$ from B and $p(.)$ is a monotonically non-decreasing penalty function such that $p(0) = 0$ [3].

In practice, the constraints such as $\boldsymbol{x}\ \varepsilon\ B$ are expressed as equality and inequality constraints in the form of:

$g_i(\boldsymbol{x}) \leq 0$ for $i = 1, \ldots, q$

$h_i(\boldsymbol{x}) = 0$ for $i = q+1, \ldots., m$

Where q = number of inequality constraints.

m-q = number of equality constraints.

Sometimes, it is hard to decide which constraint technique is effective and efficient to handle the constraints. It was been noted by Siedlecki and Sklansky (1989) that many of the optimal solutions lie on the boundary of the feasible solutions arises problems. The solutions similar to the genotype of the optimum solutions are infeasible. Hence Smith & Tate (1993), Anderson & Ferris (1994), Coit *et al* (1995) and Michalewicz (1995) proved in their research that if the search space is restricted to only feasible region or if very rigorous penalties are used then making a way for the population to the optimum would become difficult. On a contrary side, if the penalty is not really too tough to handle, then

lot of time is spent searching the space around the feasible region. Hence the search is set outside the feasible region. Michalewicz (1995) has given a very good comparison of six penalty strategies applied to continuous optimization problems includes static and dynamic approaches. There are some sequential constraint handling [Schoenauer M and Xanthakis, 1993] and the feasible solutions dominate the infeasible solutions [Powell D. and Skolnick M. M., 1993].

3.3 Static Penalty Functions:

The static penalty functions are widely used for handling constraints. They are pre-defined and fixed during evolution. The static penalty penalizes infeasible solution by constantly applying (addition or subtraction) to the solutions violating feasibility. If a minimization problem is considered then a penalized function will be equal to unpenalized function plus a penalty and vice versa. When there are multiple constraints (considering a minimization problem), the penalty function would then be a metric added depending on number of constraints violated. The penalized objective function with *m* constraints can then be given as below:

$$f_p(x) = f(x) + \sum_{i=1}^{m} c_i \delta_i$$

$$\text{Where } \begin{cases} \delta_i = 1, if\ constraint\ i\ is\ violated \\ \delta_i = 0, if\ constraint\ is\ satsfied \end{cases}$$

Where $f_p(x)$ is the penalized objective function; *f(x)* is the unpenalized objective function, and c_i is a constant imposed for violation of constraint *i*. Goldberg and Richardson *et al* (1989) stated that the static penalty function is purely based on violated number of constraints and is also based on distance metrics from the feasible region.

Richardson *et al* (1989) termed "cost of completion" which means penalizing an objective function would be more effective if distance to feasibility is considered. This was also done with an assumption that distance can be solely by number of constraints violated. Sometimes distance metrics can be

continuous or discrete and can also be linear or non- linear. To explain it further consider the following formulation for a minimization problem:

$$f_p(x) = f(x) + \sum_{i=1}^{m} c_i d_i^k$$

$$\text{Where } d_i = \begin{cases} \delta_i g_i, & for\ i = 1, \ldots., q \\ |h_i(x)|, & for\ i = q+1, .., m \end{cases}$$

Where d_i is the distance metric of constraint *i* applied to solution **x** and *k* is user defined exponent, with values of k 1 or 2. According to Richardson *et al* (1989) c_i should be based on expected cost to repair the solution. Using this rule, it is possible to find the c_i value for most of the problems. Richardson *et al* 1989, Baeck and Khuri 1994, Huang *et al* 1994 and Olsen 1994 researchers in evolutionary computation have explored the static penalty functions with various examples with various distance metrics.

In the present research, a static penalty function is applied to the objective function with two constraints. A constant distance metric is being used.

3.4 Dynamic Penalty Function:

As discussed in previous section, the primary inefficiency with the static penalty function was the difficulty in deciding the value for c_i coefficients. In Dynamic penalty, as the generation grows the penalty increases. A dynamic penalty function has the ability of allowing highly infeasible solutions early in the search, while continually increasing the penalty imposed to eventually move the final solution to the feasible region. In general a dynamic penalty is based on length of search *t*. Hence for a minimization problem, a general form of dynamic penalty function can be given as below:

$$f_p(x) = f(x) + \sum_{i=1}^{m} S_i(t) d_i^K$$

Where $S_i(t)$ is a monotonically non-decreasing in value with *t*. The t value gives the number of generations or can be said as number of solutions searched.

In this particular penalty method, the penalty parameter is dependent on the current generation number. Joines and Houck (1994) proposed a dynamic function to evaluate individuals at generation *t* as follows:

$$eval(x') = f(x') + (Ct)^{\alpha} p(\beta, x')$$

Where C, α and β are constants determined by users. Joines and Houck used C= 0.5, α= 1 or 2 and β= 1 or 2 and *p* can be described as:

$$p(\beta, x') = \sum_{i=1}^{q} D_i^{\beta}(x') + \sum_{j=q+1}^{m} D_j(x')$$

$$\text{Where } D_i(x) = \begin{cases} 0 & g_i(x') \le 0, 1 \le i \le q \\ |g_i(x')|, & otherwise \end{cases}$$

$$D_j(x') = \begin{cases} 0, & -\varepsilon \le h_j(x') \le \varepsilon,\ q+1 \le j \le m \\ |h_j(x')|, & otherwise \end{cases}$$

The solution is sensitive to changes made to α and β. But none has given any explanation about the sensitivity of the method. Joines and Houck also used C= 0.5 and α=β= 2. Michalewicz (1995) stated that parameter values cause premature convergence with some supportive examples. And also proved that the dynamic penalty method converges to an infeasible solution or a solution that is far from optimal solution.

Olsen (1994) and Joines & Houck (1994) considered distance and evolution time and compared several penalty functions for optimization problems. At the end of the evolution they result in feasible solutions.

3.5 Adaptive Penalty Function:

The adaptive penalty function was originally proposed for multi- choice integer programs. But later it was generalized for other problems. The method tries to avoid infeasible solutions by adjusting the penalty coefficient. A new parameter *h* i.e. number of iterations whose best individuals is examined. If all the best individuals of the past *h* iterations are feasible, the penalty coefficient is decreased by dividing it with a parameter c_1>1. Otherwise if all the best individuals of the past *h* iterations are infeasible, the penalty coefficient is increased by multiplying it with parameter c_2>1. And suppose if some are feasible and some are infeasible then, current penalty coefficient is continued. Hence, the penalty coefficient is updated if and only if there is a possibility that the boundary of the feasible region is not covered.

As said before the penalty parameters are updated for every iteration accordingly to the gathered information. Hadj- Alouane and Bean [1997]; evaluated an individual by using the following formula.

$$eval(x') = f(x') + \lambda(t)\left[\sum_{i=1}^{q} g_i^2(x') + \sum_{j=q+1}^{m} |h_j(x')|\right]$$

The penalty parameter is updated at every generation *k*:

$$\lambda(t+1) = \begin{cases} \frac{\lambda(t)}{\beta_1}, & if\ previous\ generations\ have\ feasible\ best\ solution. \\ \beta_2\lambda(t), & if\ previous\ generations\ have\ infeasible\ solution. \\ \lambda(t) & Otherwise \end{cases}$$

The main problem with this method lies in the difficulty of choosing a proper values for *k*, β_1 and β_2. The purpose of the adaptive penalty method is to protect the population diversity and to protect unfeasible solutions, which form the big part of the population in the early generations, from high penalties.

Apart from penalty function, there are other constraint handling techniques which are introduced in this section. The Repair Approach is one among them, the infeasible solution are repaired making them feasible solutions. The Repair approach is found to be similar to penalty function at times. The other is the Purist Approach; all the infeasible solutions evaluated by solving the optimization problems

are ignored or rejected. Lastly is Hybrid Approaches, two or more different constraint handling techniques are fussed to form a better approach. Hence now we are very much familiar with the constraint handling techniques.

However some of the researchers have introduced some other constraint handling methods. Michalewicz and Attia (1994) developed a GENECOP II based on the idea of simulated annealing. GENECOP II distinguishes between linear and non linear constraints. It considers only the active constraints at every iteration with decrease in temperature pressure on the infeasible solutions increases. The main feature of this method is that there is no diversity of the initial generation that consists of multiple copies of a solution satisfied all linear constraints. Temperature decrease at each iteration and the best solution from previous iteration is taken as the starting point for the next iteration. GENECOP II is sensitive to the parametric values and also there is no specific considerations to take assume these parameters. Le Riche *et al,* (1995) proposed another method called Segregated GA which uses two penalty parameters in two different populations. It turned up to be a good method to avoid extreme high and low penalties. And this method was successfully applied to laminated design and yielded excellent results. But again there is a disadvantage with this method is selection of two penalties for two populations is again difficult. Artificial Immune Systems (AIS) are kind of computational intelligent systems inspired from biological immune systems. Nareli Cruz- Cortes *et al,* (2005) have successfully used AIS to solve complex problems in machine learning, global optimization and information security. Hajela P. *et al,* (1996) proposed AIS to handle constraints into a GA. AIS imitate the process in which an antigen (feasible individuals) is detected by the immune system, the antibodies (Infeasible individuals) have to learn which are the correct antibodies able to neutralize that antigen (Feasible individual). The main aim is to evolve from infeasible to feasible solutions. Later Coello *et al,* (2002) with some advancement in Hajela's algorithm solved a problem which did yield promising results.

3.6. Conclusion:

In the real- world, the optimization problems have constraints in the problem which need to be satisfied. A variety of constraint handling techniques are introduced and explained in brief in this chapter. Each technique has different features and its own abilities and disabilities. Depending on type of the optimization problem, a proper technique has to be picked. In particular it is difficult to say that one technique is good. The fitness of the technique depends on the type of the problem solved. In the previous chapter some of the EA's were introduced to solve optimization problem. In the next chapter, with a combination of EA's and constraint handling technique, a component allocation optimization problem is solved. This will be explained with the help of two examples.

Chapter 4: Electricity Distribution System Optimization

4.1. Introduction:

In chapter 2 different evolutionary algorithms (EA's) which are being used in the fields of science and engineering to solve real world optimization problems were introduced. And it has been shown that these EA's have given promising and effective results. But optimization problems posses some constraints which are hard to solve just with EA's, hence for this purpose some of the constraint handling techniques are addressed and explained in the chapter 3. The main focus of the present research is on the electricity distribution system. Considered optimization problem in the electricity distribution is solved using genetic algorithms combined with constraint handling techniques. In electricity distribution systems there are number of optimization problems such as transmission planning, constrained security dispatch, optimal power flow, emergency control, etc [Ponnambalam K. *et al*, 1992].

Su and Lii (2002), developed an evolutionary algorithm to determine the optimal reliability of the indices for the different components of a distribution system, they applied to the secondary station at the Taiwan Power Company. Carvelho *et al*., (2006), presented different algorithms involved in the recombination-based evolutionary software developed for planning electric distribution networks and on specificities in the search space. Bouri *et al*., (2005), proposed an Ant Colony Optimization algorithm to solve the shunt capacitor placement problem in radial distribution systems under capacitor switching constrains. They applied their shunt capacitor allocation model in the distribution system of Ouest Algerian Network. Haghifam M. R, (2008), proposed an approach for optimal determination of the number and location of tie switches in radial distribution system using genetic algorithms. Ignacio J. Ramirez Rosado and Jose L. Bernal- Agustine (1998) developed a genetic algorithm to solve the sizing and locating problems of feeders and substations using its corresponding fixed costs as well as non-linear variable costs. In the present work, a genetic algorithm is being developed to solve the component

allocation problem in electricity distribution system; the developed model is applied to the DESN configuration. In the problem presented, the objective is to minimize the total expected system downtime subject to repair time and cost constraints. In the upcoming chapters, a brief description will be explained to approach the problem solution.

4.2. Power Systems Reliability Evaluation:

Different methods can be used to obtain the system reliability for electricity distribution configuration such as Marchov Chains, Monte Carlo simulation [B. Roy and P. Wang, 1999] and Approximation techniques. Many literatures in the area of power system reliability evaluation have shown that it has been a standard practice to use approximation evaluation techniques. Hence in the present research an approximation method is being used to evaluate the required system parameters. Minimal path sets and minimal cut sets are two methods categorized under approximation method. Billinton and Allan (1998), Billinton and Li (1994) and Billinton and Zhang (2000) presented reliability evaluation of power systems in their previous works. They proposed approximation techniques for the reliability analysis of series-parallel configurations composed with two or more components connected in parallel. They also noted that for a complex system, a series-parallel transformation based on the system minimal cut sets can provide a good approximation to the actual system reliability metric.

In the present research, the system reliability is estimated using minimal cut sets [T. F. Tsao and C. Hong- Chang (2003) and J. F. Espiritu *et al*, (2005)], the electricity distribution configuration was transformed into an equivalent series- parallel configuration and by using the system minimal cut sets the system outage rate, repair time, cost and expected system downtime were obtained. Figure 9 represents a series- parallel system with *n* subsystems connected in series. Each subsystem has m_j components connected in parallel.

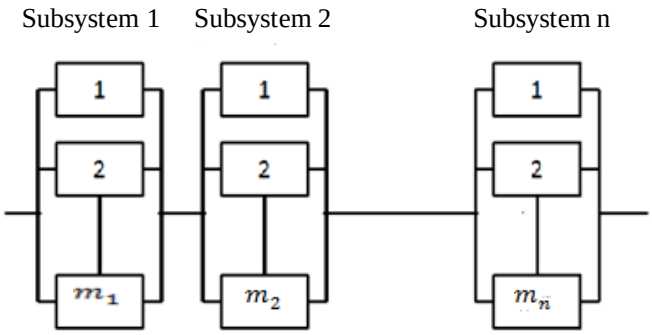

Figure 9: Series-Parallel System

As previously mentioned, in the present research a series- parallel approximation will be used to estimate the power system reliability measures based on minimal cut sets, which is a lower bound approximation to the system reliability. In the approach, each cut set is considered as a sub- system, as shown in fig. 10.

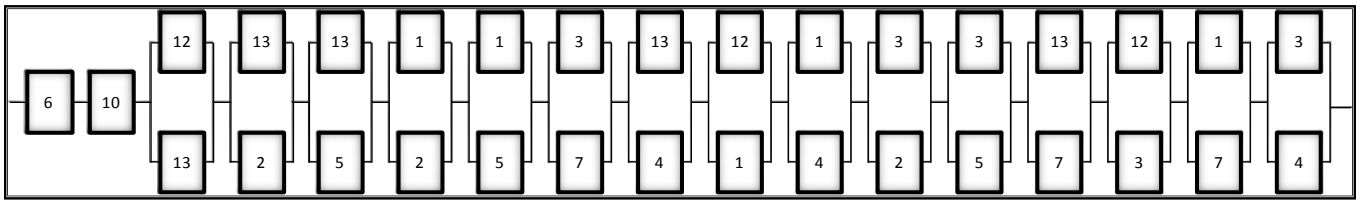

Figure 10: DESN transformation for failure at load 8

Once a series- parallel transformation has been obtained for a specific ETD configuration, Equations 1-2 can be applied to each of the sub-system as in [Billinton & Allan (1996) and Billinton & Li (1994)] to obtain each subsystem outage rate and average repair time and Equations 3,4,5 can be applied to obtain total system outage rate, average repair time and expected system downtime. The cost can be obtained by just summing up the all the subsystem cost involved to form a system.

$$\lambda_{s_i}^{m_i} = \lambda_{s_i}^{m_i-1}\lambda_{im_i}\left(r_{s_i}^{m_i-1} + \lambda_{im_i}\right) \qquad (1)$$

$$r_{s_i}^{m_i} = \frac{r_{s_i}^{m_i-1} r_{im_i}}{r_{s_i}^{m_i-1} + r_{im_i}} \qquad (2)$$

$$\lambda_{s-p} = \sum_{i=1}^{n} \lambda_{s_i}^{m_i} \quad (3)$$

$$r_{s-p} = \frac{\sum_{i=1}^{n} \lambda_{s_i}^{m_i} \, r_{s_i}^{m_i}}{\lambda_{s-p}} \quad (4)$$

$$U_{s-p} = \lambda_{s-p} r_{s-p} = \sum_{i=1}^{n} U_{s_i}^{m_i} \quad (5)$$

Where:

$\lambda_{s_i}^{m_i}$ = Sustained outage rate for subsystem *i* with m_i components.

λ_{s-p} = Sustained outage rate for series parallel system.

$r_{s_i}^{m_i}$= Average repair time for subsystem *i* with m_i components.

r_{s-p} =Average repair time for series parallel system.

U_{s-p} = Expected downtime for series parallel system.

$U_{s_i}^{m_i}$ = Expected downtime for subsystem *i*.

4.3. Problem Description:

Many types of electricity distribution systems are used by the customer [T. Gonen, 1986]. But usually two major kinds of distribution systems are radial and interconnected (complex) systems. For the present research problem an interconnected configuration i.e. the Dual Element Spot Network (DESN) configuration is used as an example. The DESN configuration is composed of total 11 components, two lines, two transformers, six breakers of which one breaker is open and two buses as shown in figure. 8.

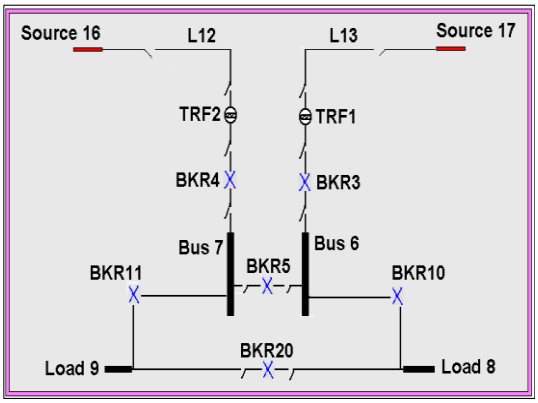

Figure 11: Dual Element Spot Network Configuration

In the particular problem, to determine the reliability of the ETD's at a specific load points, the minimal cut sets technique is used. From decades it has been a complex task for the researchers to have a best solution for a reconfiguration or a allocation problem. The reason is when a allocation/ reconfiguration problem is solved, number of solutions come out of which choosing the best one depends on our objective function. Evolutionary techniques such as genetic algorithms, particle swarm, ant colony and simulated annealing optimization techniques have been applied to the distribution system to solve various optimization problems.

In the present research, new evolutionary algorithm to solve the component allocation problem in electricity distribution systems is developed. The problem addresses the upgrade/design of an electricity distribution system with the objective of minimize total system downtime subject to cost and repair time constraint and this algorithm is tested on the DESN configuration, the most commonly used by the power industry.

4.4 Algorithm Developed:

A flow chart for the problem is shown in figure. 12.

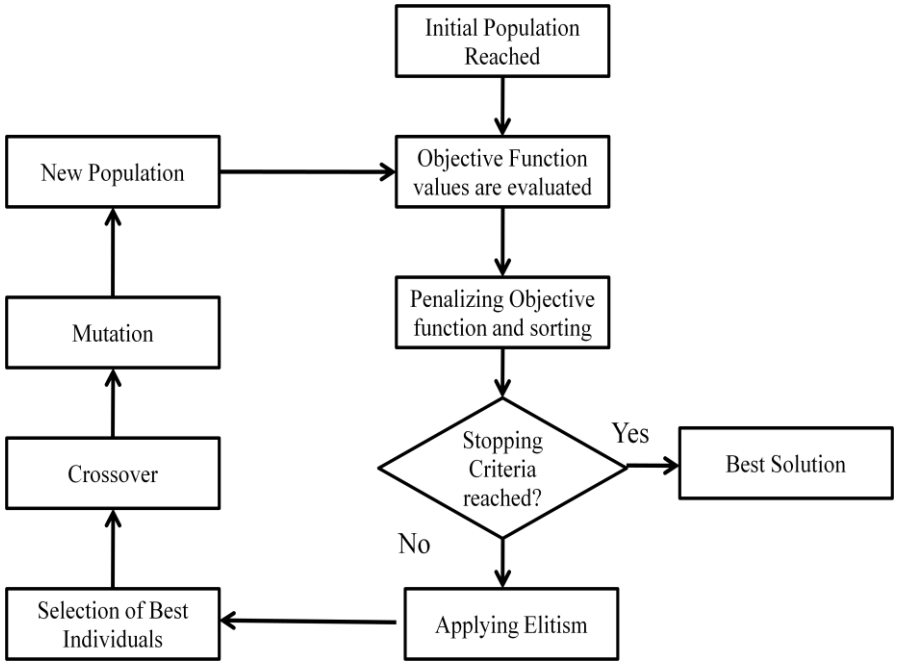

Figure 12: Genetic Algorithm Flow Chart

A brief description of the step by step procedure is given bellow which explains the flow chart clearly. This will give a clear idea of the algorithm developed and how it works.

1. [*Chromosome representation*]: In the present example there are 11 components in the DESN configuration. Therefore, the chromosome length contains 11 cells, one for each component in the DESN configuration. As shown in Figure 13.

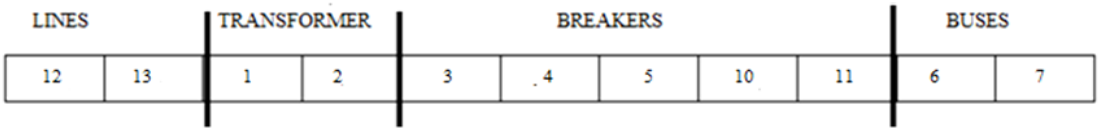

Figure 83: Chromosome representation of DESN

As seen the figure. 11, DESN configuration is built with 2 lines, 2 transformers, 5 breakers and 2

buses.

2. [*Evaluation*]: For each chromosome generated in step 1, the different parameters of the system are evaluated i.e. system outage rate, average repair time, expected system downtime and cost. If we generate 100 chromosomes we obtain 100 sets of outage rate, average repair time and expected system downtime and cost
3. [*Penalizing*]: The objective function is penalized by using the static penalty function method. Once the objective function is penalized they are sorted from the best to the worst.
4. [*Elitism*]: The chromosomes are differentiated with elitism function in the ratio of 30:70. Elitism is the process of selecting the better individuals, or more to the point, selecting individual with a bias towards the better ones. Elitism is important since allows the solution to get better over time.
5. [*Crossover*]: The best chromosomes (solutions generated so far) are selected and they are used to create new solutions to be used in the next generation.
6. [*Mutation*]: Some of the new elements in the population undergo mutation.
7. [*New Population*]: A new population is formed and the problem goes to step 2 and several iterations are performed until a specified stopping criterion is satisfied, such as the number of generations.

4.5 Examples:

In the present problem the load 8 is considered. The objective is to minimize the total system downtime at load 8. The minimal cut sets for failure at load 8 are obtained and are shown below in the table. The series- parallel transformation for the minimal cut sets are framed in Figure 2.

Table 1: Minimal cut sets for DESN configuration for load 8.

Line 12, Line 13	Bus 6	BRK 10
Line 13, TRF 2	Line 13, BKR 4	Line 13, Bus 7
Line 13, BKR 5	Line 12, TRF 1	Line 12, BKR 3
TRF 1, TRF 2	TRF 1, BKR 4	TRF 1, BUS 7
TRF 1, BKR 5	BKR 3, TRF 2	BKR 3, BKR 4
BKR 3, Bus 7	BKR 3, BKR 5	

As in the market different choices are available for each component from different manufacturers. Table.2 shows different component choices to build a DESN configuration. Each component has its own outage rate, repair time and cost. The objective is to find appropriate configuration of the components which yields our objective function.

Table 2: Component choices for DESN

Lines				Transformers				Breakers				Buses			
#	λ	r	c	#	λ	r	c	#	λ	r	c	#	λ	r	c
1	0.098	20	12	5	0.0095	116	19	9	0.0055	35	15	13	0.250	5.9	3
2	0.084	28	7	6	0.0250	145	22	10	0.0067	15	20	14	0.018	3.2	8
3	0.055	12	18	7	0.0019	100	25	11	0.0099	17	8	15	0.035	3,0	7
4	0.091	21	10	8	0.0400	130	20	12	0.0876	80	4	16	0.130	4.8	5

4.4 Results:

The above algorithm is coded in MATLAB® 2007 with the data given in Table: 2. Accordingly the above steps, initially a random population (P) is generated. Next for each of the chromosome the objective function and the rest of the parameters are evaluated using the power system reliability evaluation equations. After evaluation some of the solutions may fall outside the feasible region, then penalty function is applied to objective function to pull them towards the feasible region. After penalizing the objective function value, genetic algorithm is implemented for the rest of the process to get promising results. Since this is a single objective function, a single solution is found for the problem.

Example 1: The above methodology is applied to DESN configuration for load 8 to obtain minimum expected system downtime. The code was run for 100 generations to yield optimal design values of the DESN configuration for load 8. The following results were obtained. Figure 14 shows a new DESN configuration which provides minimum total system downtime with respect to reduced cost and less

repair time. Table 4 shows the values of outage rate, repair time and cost of the reformed distribution system.

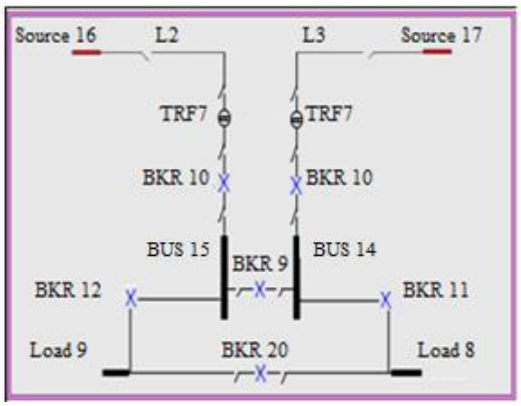

Figure 94: Recommended DESN configuration for load 8

Table 3: Optimal System Design Values for the DESN developed for load 8.

System Outage Rate (λ)	0.23 o/y
Repair Time (r)	211.99 h/o
Expected System Downtime (U)	48.59 h/y
Total System Cost (c)	$157

Example 2: In the above example, the total expected system downtime (U) is calculated at load 8. The same MATLAB code with some modifications in the evaluation can be used to calculate the system parameters of any system having cut sets. Hence using the same code, the system parameters can be calculated for load 8 & 9. However as said before there has been some changes made to the program evaluation. Table 4 represent the minimal cut sets of the DESN configuration for loads 8 & 9.

Table 4: Minimal cut sets for load 8 & 9

Bus 7, Bus 6	TRF 2, BKR 3	Bus 6, BKR 4
Bus 7, BKR 10	TRF 2, TRF 1	Bus 6, TRF 2
BKR 11, Bus 6	TRF 2, Line 13	Bus 6, Line 12
BKR 11, BKR 10	Line 12, BKR 3	Bus 7, BKR 3
BKR 4, BKR 3	Line 12, TRF 1	Bus 7, TRF 1
BKR 4, TRF 1	Line 12, Line 13	Bus 7, Line 13
BKR 4, Line 13		

As done for the before example, a series-parallel configuration is sketched from the above minimal cut sets and then similarly the system parameters for the system for load 8 & 9 are calculated. The MATLAB ® code was also run for 100 generations and Figure 15 shows recommended DESN configuration which provides minimum total system downtime for load 8 & 9 with respect to reduced cost and less repair time. And Table 5 shows the values of system outage rate, repair time, expected system downtime and total cost of the system.

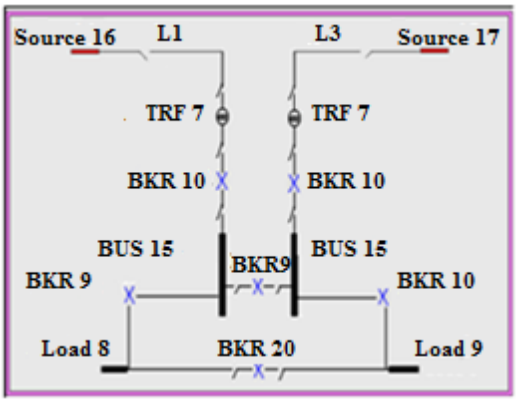

Figure 15: Recommended DESN configuration for load 8 & 9

Table 5: Optimal system design values for the DESN developed for load 8 & 9

System Outage Rate (λ)	0.40 o/y
Repair Time (r)	166.80 h/o
Expected System Downtime (U)	65.90 h/y
Total System Cost (c)	$184

The MATLAB code is included in appendix I and II. As can be seen in the appendix the programs carry the same coding but they differ in their evaluation segment. Hence it can be said that the code can be applied to any of the electric distribution configurations.

4.5 Conclusions:

In the present research, a new genetic algorithm has been developed to solve the component allocation problem in electricity distribution systems with the objective to minimize expected system down time subjected to repair time and budget cost. The algorithm is tested on the Dual Element Spot Network configuration.

4.6 Future Work:

The future work for the present research is some of more factors (constraints) can be considered. The model can be tested in different test system such as the IEEE 14- bus, 30- bus etc IEEE systems.

References:

1. Billinton R. and Allan R. N. (1996). "Reliability Evaluation of Power Systems", Plenum Press, New York, 1994
2. Billiton R. and Allan R. N. (1998). "Reliability Assessment of Large Electric Power Systems". Kluwer Academic Publishers, Boston, 1998.
3. Diaz J. F., Guiterrez G, Olarte C. A and Rueda C (2005), "Using Constraint Programming for Reconfiguration of Electric Power Distribution Networks", *Springer- Berlin Heidelberg*, Pgs: 263- 276.
4. Chowdhary A and Koval D.O. (1998). "Value-based distribution system reliability planning". *IEEE Transaction on Industry Application*, Vol. 34, Issue: 1, Pgs: 23- 29, 1998
5. Billinton R. and Li W. (1994). "Reliability Assessment of Electric Power Systems using Monte Carlo methods", Plenum press, New York, 1994
6. Billinton R. and Zhang W. (2000). "State extension for adequacy evaluation of composite power systems- applications", *IEEE Transactions on Power Systems*, Vol. 15, Issue: 1, Pgs: 427- 432, 2000.
7. Espiritu J.F., Coit D. W. and Prakash U. A. (2007). "Component critical importance measures of power industry", *Electric Power Systems Research*, Vol. 77, Issue: 5- 6, Pgs: 407- 420, 2007.
8. Homaifar A., Lai S. and Qi. X. (1994). "Constrained Optimization via Genetic Algorithms". *SAGE Full Text Collection, Simulation*, Vol. 62, No. 4, Pgs: 242- 253, 1994.
9. Michalewicz Z. and Janikow Cezarry Z. (1990). "Genetic algorithms for numerical optimization", Journal of Statistics and Computing, Springer Netherland, Vol. 1, No. 2, 1990.
10. Michalewicz Z., Dasgupta D., Lerichie R. and Schoenauer M. (1996). "Evolutionary algorithms for constrained engineering problems". *Computers and Industrial Engineering Journal*, Vol. 30, Issue. 4, Pgs: 851- 870, April 1996.

11. Youssef L. Abdel- Magid, Mantawy A.H. and Shokri Z. Selim. (1999). "Integrated Genetic Algorithms, Tabu Search and Simulated Annealing For the Unit Commitment Problem", *IEEE Transaction on Power Systems*, Vol. 14, Issue: 3, Pgs: 829- 836, 1999.
12. Marco Dorigo, Vittorio Maniezzo and Alberto Colorni. (1996). "The Ant System: Optimization by a colony of cooperating agents", *IEEE Transaction on Systems, Man and Cybernetics- Part B*, Vol. 26, No. 1, 1996, pp 1- 13, 1996.
13. M. Dorigo. (1992). "Optimization, Learning and Natural Algorithms", PhD Thesis. Politecnico do Milano, Milano 1992.
14. Marco Dorigo and Luca Maria Gambardella. (1997). "Ant Colony System: A Cooperative Learning Approach to the Travelling Salesman Problem", *IEEE Transaction on Evolutionary Computation*, Vol. 1, No. 1, Pgs: 53- 66, 1997.
15. Gomez J. F., Khodr H. M., De Oliveira P. M., Ocque L., Yusta J. M., Villasana R. R., Urdaneta A. J. (2004). "Ant Colony System Algorithm for the planning of primary distribution circuits", *IEEE Transaction of Power Systems*, Vol. 19, No.2, Pgs: 996-1004., May 2004.
16. Kirkpatrick S., Gelatt S. D. and Vecchi M. P. (1983). "Optimization by Simulated Annealing" Science, Vol. 220, No. 4598, Pgs: 671- 680, 1983.
17. Cerny V. (1985). "A Thermodynamic Approach to the Travelling Salesman Problem: An Efficient Simulation Algorithm", *Journal of Optimization Theory and Applications, Springerlink*, Vol. 45, No. 1, Pgs: 41-51, 1985.
18. Deeb N, Sargent and Lundy (1992). "Simulated Annealing in Power Systems", *IEEE Transaction of Systems, Man and Cybernetics*, Vol. 2, Pgs: 1086- 1089, 1992.
19. Abido M. A. (2000). "Robust design of multi machine power system stabilizers using simulated annealing", *IEEE Transaction on energy conversion*, Vol. 15, Issue. 3, September 2000.
20. Hiroyuki Mori and Koichi Takeda (1994). "Parallel Simulated Annealing for power system decomposition", *IEEE Transaction on Power Systems*, Vol. 9, No. 2, May 1994.

21. Bland J. A. and Dawson G. (1991). “Tabu Search and Design Optimization” Computer Aided Design, Vol. 23, Issue.3, 1991, Pgs: 195-201, 1991.
22. Glover F. (1993) “A User’s Guide to Tabu Search”, Annals of Operation Research, 41, Pgs: 3.28, 1993.
23. Glover F. (1989). Tabu Search Part I”, ORSA Journal on Computing. Vol. 1, No. 3, Pgs: 3-28, 1989.
24. Glover F. (1990). “Tabu Search Part II”, ORSA Journal on Computing. Vol. 2, No.1, Pgs: 4-32, 1990.
25. Mohamed Ouzineb, Mustapha Nourelfath, and Michel Gendreau, (2006). “Tabu Search for the redundancy allocation problem of homogenous series- parallel mutli- state systems”, Science Direct on Reliability & Systems Safety, Vol. 93, Issue 8, Pgs: 1257- 1272, August 2006
26. Sara Carcangiu, Alessandra Fanni and Augusto Montisci (2008). “Multiobjective Tabu Search Algorithm for optimal design of electromagnetic devices”, *IEEE Transaction on Magnetics*, Vol. 44. Issue. 6, June 2008.
27. Hiroyuki Mori and Yoshihiro Ogita (2000). “Parallel Tabu search for capacitor placement in radial distribution system”, *IEEE Power Engineering Society Winter Meeting 2000*, Vol. 4, Pgs: 2334- 2339, 2000.
28. Mantawy A. H., Youssef L. Abdel – Magid and Shokri Z. Selim (1999). “Integrating genetic algorithms, tabu search, and simulated annealing for the unit commitment problem”, *IEEE Transaction on Power Systems*, Vol. 14, Issue. 3, Pgs: 829- 836, August 1999.
29. Vasconcelos J. A., Krahenbuhl L. and Nicolas A. (1996). “Simulated Annealing coupled with the tabu search method for continuum optimization is electromagnetic”, *IEEE Transaction on Magnetics*, Vol. 32, Issue: 3, Part 1, Pgs: 1206- 1209, May 1996.

30. Ruey- Maw Chen, Shih – Tang Lo, Chung- Lun Wu and Tsung- Hung Lin (2008). "An effective ant colony optimization- based algorithm for flow shop scheduling", IEEE Transaction on soft computing in Industrial Applications, Pgs: 101- 106, Muroran, Japan, 2008.
31. Myun- Eun Lee, Soo- Hyung Kim, Wan- Hyun Cho, Soon- Young Park and Jun- Sik Lim (2009). "Segmentation of Brain MR Images Using an Ant Colony Optimization Algorithm", *2009 9th IEEE International Conference on Bioinformatics and Bioengineering*, Pgs: 366- 369, 2009
32. Glover F. and Manuel Laguna (2000). "Tabu Search", Kluwer Academic Publishers, sixth printing 2000.
33. Fushuan Wen and Chang C. S. (1997). "Transmission network optimal planning using the tabu search method", *Science Direct of electric power systems research*, Vol. 42, Issue 2, Pgs: 153- 163, August 1997.
34. Nakagawa Y. and Nakashima K. (1997). "A heuristic method for determining optimal reliability allocation", *IEEE Transaction for Reliability*, Vol. R-26, Issue. 3, Pgs. 156- 161, 1997.
35. Shi D. H. (1987). "A new heuristic algorithm for constraint redundancy optimization in complex system", IEEE Transaction for Reliability, Vol. R-36, Issue. 5, Pgs: 621- 623, 1987.
36. Smith A. E. and Coit D. W. (1995). "Handbook of Evolutionary Computation" A joint publication of Oxford University Press and Institute of Physics Publishing, September 1995.
37. Schwefel H. P. (1995). "Evolution and optimum seeking" John Wiley & Sons.
38. Siedlecki W. and Slansky J. (1989). "Constrained genetic optimization via dynamic reward penalty balancing and its use in pattern recognition", *Proceedings of the Third International Conference on Genetic Algorithm*s, Pgs: 141- 150, 1989.
39. Smith A. E. and Tate D. M. (1993). "Genetic Optimization Using Penalty Functions", *Proceedings of Fifth International Conference on Genetic Algorithm*, Pgs: 499- 505, 1993.

40. Anderson E. J. and Ferris M. C. (1994), "Genetic algorithms for combinatorial optimization: the assembly line balancing problem", *ORSA Journal on Computing*, Vol. 6 No. 2, Pp: 161- 173, 1994.
41. Coit D. W, Smith A. E. and Tate D. M. (1995), "Adaptive penalty methods for genetic optimization of constrained combinatorial problems", *INFORMS Journal on Computing*. Vol. 8, No.2, Pp: 173- 182, 1995.
42. Michalewicz Z. (1995). "Genetic algorithms, numerical optimization and constraints", Proceedings of sixth international conference of genetic algorithms, Morgan Kaufmann, Pp: 151- 158, 1995.
43. Schoenauer M. and Xanthakis S. (1993). "Constrained GA Optimization", *Proceeding of fifth international conference on genetic algorithms* pg: 573- 580, 1993.
44. Powell D. and Skolnick M. M. (1993), "Using genetic algorithms in engineering design optimization with non-linear constraints", *Proceeding of the fifth international conference on genetic algorithms*, Pgs: 424- 430, 1993.
45. Goldberg D. E. (1989). "Genetic Algorithms in Search Optimization and Machine Learning", Addison Wesley Reading MA, 1989.
46. Richardson J. T., Palmer M. R., Liepins G. and Hilliard M. (1989), "Some guidelines for genetic algorithms with penalty functions", *Proceedings of the third International conference on genetic algorithms.* Pgs: 191- 197, 1989.
47. Baeck K. and Khuri S. (1994). "An evolutionary heuristic for the maximum independent set problem", *Proceedings of first IEEE Conference on Evolutionary Computation, Vol. 2*, Pgs: 531- 535, 1994.
48. Huang W. C., Kao C. Y. and Horng, J. T. (1994). "A genetic algorithm approach for set covering problem", *Proceedings of the first IEEE conference on Evolutionary computation,* Vol. 2, Pgs: 569- 574, 1994.

49. Olsen A. L. (1994). “Penalty functions and the knapsack problem”, *Proceedings of the first IEEE conference of evolutionary computation,* Vol. 2, Pgs: 554- 558, 1994.
50. Joines J. A. and Houck C. R. (1994). “On the use of non- stationary penalty functions to solve nonlinear constrained optimization problems with GA’s”, *Proceedings of the first IEEE conference on evolutionary computation*, Pgs: 579- 584, 1994.
51. Hadj- Alouane, A. B. and Bean J. C. (1997). “A genetic algorithm for the multiple choice integer program”, Operation Research, Vol. 45, No. 1, pp. 92- 101, February 1997.
52. Janusz Kacprzyk (2009). “Constraint- Handling in Evolutionary Optimization”, Studies in computational intelligence, *Springer Berlin* (Publisher), Vol. 198, 2009.
53. Elmer P. Dadios and Jamshaid Ashraf (2006). “Genetic Algorithms with adaptive and dynamic penalty functions for the selection of cleaner production measures: a constrained optimization problem”, *Springer Berlin*, Vol. 8, No. 2, Pgs: 85- 95, 2006.
54. Guillerno Leguizamon and Carlos Coello Coello (2009). “Boundary Search for Constraint Numerical Optimization Problem”, *Studies in Computational Intelligence, Springer Berlin*, Vol. 198, 2009, Pgs 25-49, 2009.
55. Michalewicz Z. and Attia N. (1994). “Evolutionary optimization of constrained problems”, Proceedings of Third Annual Conference on Evolutionary Programming, World Scientific, 98-108, 1994.
56. Le Riche R., Knopf- Lenior C. and Haftka R. T. (1995). “A segregated genetic algorithm for constrained structural optimization”, *Proceedings of the Sixth International Conference on Genetic Algorithms*, Morgan Kaufmann, Pgs: 558- 565, 1995.
57. Nareli Cruz- Cortes, Daniel Trejo- Perez and Carlos A. Coello Coello. (2005). “Handling Constraints in Global Optimization Using an Artificial Immune Systems: A survey”, *Studies in Computational Intelligence, Springer Berlin Publisher*, Vol. 198/2009, Pgs: 234- 247, 2005.

58. Hajela P. and Lee J. (1996). "Constrained genetic search via schema adaptation: An immune network solution", *Structural and Multidisciplinary Optimization, Springer Berlin*, Vol. 12, No. 1, Pgs: 11- 15, 1996.

59. Hajela. P and Yoo, J. S. (1999). "Immune network simulations in multicriteria design", *Structural and Multidisciplinary Optimization, Springer Berlin*, Vol. 18, No. 2- 3, Pgs: 85- 94, 1999.

60. Coello, C. A. C., Cruz Cortes, N. (2002). "A parallel implementation of an artificial immune system to handle constraints in genetic algorithms: preliminary results", Proceedings of the 2002 Congress on Evolutionary Computation, Vol. 1, Pgs: 819- 824, 2002.

61. Ponnambalam K., Qunitana V.H., and Vannelli A. (1992). "A Fast Algorithm For Power System Optimization Problems Using an Interior Point Method", *IEEE Transaction on Power System* Vol. 7(2), 1992.

62. SU C- T and Lii G-R. (2002). "Reliability design of distribution systems using modified genetic algorithms", Electric Power Systems Research, Vol. 60, Issue. 3, Pgs: 201- 206, January 2002.

63. Carvalho P. M. S., Ferreira L. A. F. M and Barruncho L. M. F. (2001). "On spanning-tree recombination in evolutionary large-scale network problems – application to electrical distribution planning", *IEEE Transactions on Evolutionary Computation*. Vol. 5, Issue. 6, Pgs: 623-630, 2001.

64. Bouri S., Zeblah A., Ghoraf A. and Hamdaoui H.S. (2005). "Ant Colony optimization to shunt capacitor allocation in radial distribution systems" *Acta Electrotechnica*; Vol. 5, No. 4, 2005.

65. Haghifam M. R., (2004) "Optimal allocation of tie points in radial distribution systems using a genetic algorithm", *European Transaction On Electrical Power*, Vol.14, Issue. 2, Pgs: 85-96, 2004.

66. Ramirez-Rosado I. J. and Bernal-Agustin J. L. (1998). “Genetic Algorithms Applied To The Design Of Large Power Distribution Systems”, *IEEE Transaction on Power Systems*, Vol. 13, Issue. 2, Pgs: 696-703, 1998.

67. Roy Billiton and Peng Wang. (1999). “Teaching Distribution System Reliability Evaluation Using Monte Carlo Simulation”, *IEEE Transactions on Power Systems*, Vol. 14, No. 2, May 1999.

68. Billinton R. and Li W. (1994). “Reliability Assessment of Electric Power Systems using Monte Carlo methods”, Plenum press, New York, 1994

69. Teng Fa Tsao and Hong- Chan Chang. (2003). “Composite reliability evaluation model for different types of distribution systems”, *IEEE Transaction on Power Systems*, Vol. 18, No. 2, May 2003.

70. Espiritu J. F., Coit D. W., Prakash U. and Ramirez-Marquez J. E. (2005). "Reliability Modeling of Electricity Transmission Systems: An Extension of Traditional Reliability Methods”, *Proceedings of the Industrial Engineering Research Conference* (IERC), Atlanta, GA, May 2005.

71. T. Gonen. (1986). “Electric Power Distribution System Engineering”. New York: Mc Graw- Hill, pp 174- 187, 1986.

APPENDIX

Appendix I

MATLAB code for example:1.

```
%%%%%%%STEP:1- CREATING CHROMOSOMES%%%%%%%%
crom = zeros(100,11);
for i=1:100
    for j = 1:2;  % no of lines
        crom(i,j)=ceil(4.*rand());
    end
    for j=3:4;
        crom(i,j)=ceil(8.*rand());
        while  crom(i,j)<5
        crom(i,j)=ceil(8.*rand());
        end
    end
    for j=5:9;
        crom(i,j)=ceil(12.*rand());
        while crom(i,j)<9
        crom(i,j)=ceil(12.*rand());
        end
    end
    for j=10:11;
        crom(i,j)=ceil(16.*rand());
        while crom(i,j)<13
        crom(i,j)=ceil(16.*rand());
        end
    end
end
crom


%%%%%%%step:2- evaluation%%%%%
%%%%evaluating the A, r and c values for chromosomes%%%%
x=[0.098 20 12; 0.084 28 7; 0.055 12 18; 0.091 21 10; 0.0095 116 19; 0.0250 145 22;
0.0019 100 25; 0.0400 130 20; 0.0055 35 15; 0.0067 15 20; 0.0099 17 8; 0.0876 80 4;
0.250 5.9 3; 0.018 3.2 8; 0.035 3.0 7; 0.130 4.8 5];
for i=1:100

%%part 1%%
l(i,1)=x(crom(i,1),1)*x(crom(i,2),1)*(x(crom(i,1),2)+x(crom(i,2),2));
r(i,1)=x(crom(i,1),2)*x(crom(i,2),2)/(x(crom(i,1),2)+x(crom(i,2),2));
%c(i,1)=x(crom(i,1),3);
u(i,1)=l(i,1)*r(i,1);

%%%part 2%%%
l(i,2)=x(crom(i,2),1)*x(crom(i,3),1)*(x(crom(i,2),2)+x(crom(i,3),2));
r(i,2)=x(crom(i,2),2)*x(crom(i,3),2)/(x(crom(i,2),2)+x(crom(i,3),2));
%c(i,2)=x(crom(i,1),3);
u(i,2)=l(i,2)*r(i,2);

%%%part 3%%%
l(i,3)=x(crom(i,2),1)*x(crom(i,7),1)*(x(crom(i,2),2)+x(crom(i,7),2));
r(i,3)=x(crom(i,2),2)*x(crom(i,7),2)/(x(crom(i,2),2)+x(crom(i,7),2));
```

```
%c(i,3)=x(crom(i,1),3);
u(i,3)=l(i,3)*r(i,3);


%%%part 4%%%
l(i,4)=x(crom(i,3),1)*x(crom(i,4),1)*(x(crom(i,3),2)+x(crom(i,4),2));
r(i,4)=x(crom(i,3),2)*x(crom(i,4),2)/(x(crom(i,3),2)+x(crom(i,4),2));
%c(i,4)=x(crom(i,4),3);
u(i,4)=l(i,4)*r(i,4);

%%%part 5%%%
l(i,5)=x(crom(i,3),1)*x(crom(i,7),1)*(x(crom(i,3),2)+x(crom(i,7),2));
r(i,5)=x(crom(i,3),2)*x(crom(i,7),2)/(x(crom(i,3),2)+x(crom(i,7),2));
%c(i,5)=x(crom(i,4),3);
u(i,5)=l(i,5)*r(i,5);

%%%parts 6%%%
l(i,6)=x(crom(i,5),1)*x(crom(i,11),1)*(x(crom(i,5),2)+x(crom(i,11),2));
r(i,6)=x(crom(i,5),2)*x(crom(i,11),2)/(x(crom(i,5),2)+x(crom(i,11),2));
%c(i,6)=x(crom(i,5),3));
u(i,6)=l(i,6)*r(i,6);

%%%parts 7%%%
l(i,7)=x(crom(i,10),1);
r(i,7)=x(crom(i,10),2);
%c(i,7)=x(crom(i,10),3);
u(i,7)=l(i,7)*r(i,7);

%%%part 8%%%
l(i,8)=x(crom(i,2),1)*x(crom(i,6),1)*(x(crom(i,2),2)+x(crom(i,6),2));
r(i,8)=x(crom(i,2),2)*x(crom(i,6),2)/(x(crom(i,2),2)+x(crom(i,6),2));
%c(i,8)=x(crom(i,2),3)+x(crom(i,6),3);
u(i,8)=l(i,8)*r(i,8);

%%%part 9%%%
l(i,9)=x(crom(i,1),1)*x(crom(i,3),1)*(x(crom(i,1),2)+x(crom(i,3),2));
r(i,9)=x(crom(i,1),2)*x(crom(i,3),2)/(x(crom(i,1),2)+x(crom(i,3),2));
%c(i,9)=x(crom(i,1),3)+x(crom(i,4),3);
u(i,9)=l(i,9)*r(i,9);

%%%part 10%%%
l(i,10)=x(crom(i,3),1)*x(crom(i,6),1)*(x(crom(i,3),2)+x(crom(i,6),2));
r(i,10)=x(crom(i,3),2)*x(crom(i,6),2)/(x(crom(i,3),2)+x(crom(i,6),2));
%c(i,10)=x(crom(i,4),3)+x(crom(i,6),3);
u(i,10)=l(i,10)*r(i,10);

%%%part 11%%%
l(i,11)=x(crom(i,5),1)*x(crom(i,4),1)*(x(crom(i,5),2)+x(crom(i,4),2));
r(i,11)=x(crom(i,5),2)*x(crom(i,4),2)/(x(crom(i,5),2)+x(crom(i,4),2));
%c(i,11)=x(crom(i,5),3)+x(crom(i,3),3);
u(i,11)=l(i,11)*r(i,11);

%%%part 12%%%
l(i,12)=x(crom(i,5),1)*x(crom(i,7),1)*(x(crom(i,5),2)+x(crom(i,7),2));
r(i,12)=x(crom(i,5),2)*x(crom(i,7),2)/(x(crom(i,5),2)+x(crom(i,7),2));
%c(i,12)=x(crom(i,5),3)+x(crom(i,7),3);
u(i,12)=l(i,12)*r(i,12);
```

```
%%%part 13%%%
l(i,13)=x(crom(i,8),1);
r(i,13)=x(crom(i,8),2);
%c(i,13)=x(crom(i,8),3);
u(i,13)=l(i,13)*r(i,13);

%%%part 14%%%
l(i,14)=x(crom(i,2),1)*x(crom(i,11),1)*(x(crom(i,2),2)+x(crom(i,11),2));
r(i,14)=x(crom(i,2),2)*x(crom(i,11),2)/(x(crom(i,2),2)+x(crom(i,11),2));
%c(i,14)=x(crom(i,2),3)+x(crom(i,11),3);
u(i,14)=l(i,14)*r(i,14);

%%%part 15%%%
l(i,15)=x(crom(i,1),1)*x(crom(i,5),1)*(x(crom(i,1),2)+x(crom(i,5),2));
r(i,15)=x(crom(i,1),2)*x(crom(i,5),2)/(x(crom(i,1),2)+x(crom(i,5),2));
%c(i,15)=x(crom(i,1),3)+x(crom(i,5),3);
u(i,15)=l(i,15)*r(i,15);

%%%part 16%%%
l(i,16)=x(crom(i,3),1)*x(crom(i,11),1)*(x(crom(i,3),2)+x(crom(i,11),2));
r(i,16)=x(crom(i,3),2)*x(crom(i,11),2)/(x(crom(i,3),2)+x(crom(i,11),2));
%c(i,16)=x(crom(i,4),3)+x(crom(i,11),3);
u(i,16)=l(i,16)*r(i,16);

%%%part 17%%%
l(i,17)=x(crom(i,5),1)*x(crom(i,6),1)*(x(crom(i,5),2)+x(crom(i,6),2));
r(i,17)=x(crom(i,5),2)*x(crom(i,6),2)/(x(crom(i,5),2)+x(crom(i,6),2));
%c(i,17)=x(crom(i,5),3)+x(crom(i,5),3);
u(i,17)=l(i,17)*r(i,17);



%%%%%cost%%%%
c1=x(crom(i,1),3);
c2=x(crom(i,2),3);
c3=x(crom(i,3),3);
c4=x(crom(i,4),3);
c5=x(crom(i,5),3);
c6=x(crom(i,6),3);
c7=x(crom(i,7),3);
c8=x(crom(i,8),3);
c9=x(crom(i,9),3);
c10=x(crom(i,10),3);
c11=x(crom(i,11),3);

L(i,1)=l(i,1)+l(i,2)+l(i,3)+l(i,4)+l(i,5)+l(i,6)+l(i,7)+l(i,8)+l(i,9)+l(i,10)+l(i,11
)+l(i,12)+l(i,13)+l(i,14)+l(i,15)+l(i,16)+l(i,17);
R(i,1)=r(i,1)+r(i,2)+r(i,3)+r(i,4)+r(i,5)+r(i,6)+r(i,7)+r(i,8)+r(i,9)+r(i,10)+r(i,11
)+r(i,12)+r(i,13)+r(i,14)+r(i,15)+r(i,16)+r(i,17);
C(i,1)=c1+c2+c3+c4+c5+c6+c7+c8+c9+c10+c11;
U(i,1)=L(i,1)*R(i,1);
end
matrix=[U(:,1),R(:,1),C(:,1),L(:,1)];


%%%%%%step:3- penalizing%%%%%
```

```
%%%%penalizing the constraints %%%%%

%getting needed children
[m,b]=size(crom);
cneeded=m-round(100*.30);
selectchildren=zeros(1,1);
ctest=cmatrix2;
for i=1:cneeded
    [a,b]=size(ctest);
    b=gna2(1,a);
    selectchildren(i,1)=b;
    ctest(b,:)=[];
end




%%%%%%% Step:7- new children%%%%%

%getting new population from children%%%
[a,b]=size(crom);
for i=1:(round(100*.30))
    newpop(i,:)=crom(Nelite(i,1),:);
end

for i=1:(100*.70)
       newpop(i+30,:)=cmatrix2(selectchildren(i,1),:);
end




%%%%% repeating the aboove for 9 iteration to get the best solution%%%%%%



for i=1:25
%%%%%%%step:2- evaluation%%%%%
%%%%evaluating the A, r and c values for chromosomes%%%%
x=[0.098 20 12; 0.084 28 7; 0.055 12 18; 0.091 21 10; 0.0095 116 19; 0.0250 145 22;
0.0019 100 25; 0.0400 130 20; 0.0055 35 15; 0.0067 15 20; 0.0099 17 8; 0.0876 80 4;
0.250 5.9 3; 0.018 3.2 8; 0.035 3.0 7; 0.130 4.8 5];
for i=1:100

%%part 1%%
l(i,1)=x(crom(i,1),1)*x(crom(i,2),1)*(x(crom(i,1),2)+x(crom(i,2),2));
r(i,1)=x(crom(i,1),2)*x(crom(i,2),2)/(x(crom(i,1),2)+x(crom(i,2),2));
%c(i,1)=x(newpop(i,1),3);
u(i,1)=l(i,1)*r(i,1);

%%%part 2%%%
l(i,2)=x(newpop(i,2),1)*x(newpop(i,3),1)*(x(newpop(i,2),2)+x(newpop(i,3),2));
r(i,2)=x(newpop(i,2),2)*x(newpop(i,3),2)/(x(newpop(i,2),2)+x(newpop(i,3),2));
%c(i,2)=x(newpop(i,1),3);
u(i,2)=l(i,2)*r(i,2);
```

```
%%%part 3%%%
l(i,3)=x(newpop(i,2),1)*x(newpop(i,7),1)*(x(newpop(i,2),2)+x(newpop(i,7),2));
r(i,3)=x(newpop(i,2),2)*x(newpop(i,7),2)/(x(newpop(i,2),2)+x(newpop(i,7),2));
%c(i,3)=x(newpop(i,1),3);
u(i,3)=l(i,3)*r(i,3);


%%%part 4%%%
l(i,4)=x(newpop(i,3),1)*x(newpop(i,4),1)*(x(newpop(i,3),2)+x(newpop(i,4),2));
r(i,4)=x(newpop(i,3),2)*x(newpop(i,4),2)/(x(newpop(i,3),2)+x(newpop(i,4),2));
%c(i,4)=x(newpop(i,4),3);
u(i,4)=l(i,4)*r(i,4);

%%%part 5%%%
l(i,5)=x(newpop(i,3),1)*x(newpop(i,7),1)*(x(newpop(i,3),2)+x(newpop(i,7),2));
r(i,5)=x(newpop(i,3),2)*x(newpop(i,7),2)/(x(newpop(i,3),2)+x(newpop(i,7),2));
%c(i,5)=x(newpop(i,4),3);
u(i,5)=l(i,5)*r(i,5);

%%%parts 6%%%
l(i,6)=x(newpop(i,5),1)*x(newpop(i,11),1)*(x(newpop(i,5),2)+x(newpop(i,11),2));
r(i,6)=x(newpop(i,5),2)*x(newpop(i,11),2)/(x(newpop(i,5),2)+x(newpop(i,11),2));
%c(i,6)=x(newpop(i,5),3));
u(i,6)=l(i,6)*r(i,6);

%%%parts 7%%%
l(i,7)=x(newpop(i,10),1);
r(i,7)=x(newpop(i,10),2);
%c(i,7)=x(newpop(i,10),3);
u(i,7)=l(i,7)*r(i,7);

%%%part 8%%%
l(i,8)=x(newpop(i,2),1)*x(newpop(i,6),1)*(x(newpop(i,2),2)+x(newpop(i,6),2));
r(i,8)=x(newpop(i,2),2)*x(newpop(i,6),2)/(x(newpop(i,2),2)+x(newpop(i,6),2));
%c(i,8)=x(newpop(i,2),3)+x(newpop(i,6),3);
u(i,8)=l(i,8)*r(i,8);

%%%part 9%%%
l(i,9)=x(newpop(i,1),1)*x(newpop(i,3),1)*(x(newpop(i,1),2)+x(newpop(i,3),2));
r(i,9)=x(newpop(i,1),2)*x(newpop(i,3),2)/(x(newpop(i,1),2)+x(newpop(i,3),2));
%c(i,9)=x(newpop(i,1),3)+x(newpop(i,4),3);
u(i,9)=l(i,9)*r(i,9);

%%%part 10%%%
l(i,10)=x(newpop(i,3),1)*x(newpop(i,6),1)*(x(newpop(i,3),2)+x(newpop(i,6),2));
r(i,10)=x(newpop(i,3),2)*x(newpop(i,6),2)/(x(newpop(i,3),2)+x(newpop(i,6),2));
%c(i,10)=x(newpop(i,4),3)+x(newpop(i,6),3);
u(i,10)=l(i,10)*r(i,10);

%%%part 11%%%
l(i,11)=x(newpop(i,5),1)*x(newpop(i,4),1)*(x(newpop(i,5),2)+x(newpop(i,4),2));
r(i,11)=x(newpop(i,5),2)*x(newpop(i,4),2)/(x(newpop(i,5),2)+x(newpop(i,4),2));
%c(i,11)=x(newpop(i,5),3)+x(newpop(i,3),3);
u(i,11)=l(i,11)*r(i,11);

%%%part 12%%%
```

```
l(i,12)=x(newpop(i,5),1)*x(newpop(i,7),1)*(x(newpop(i,5),2)+x(newpop(i,7),2));
r(i,12)=x(newpop(i,5),2)*x(newpop(i,7),2)/(x(newpop(i,5),2)+x(newpop(i,7),2));
%c(i,12)=x(newpop(i,5),3)+x(newpop(i,7),3);
u(i,12)=l(i,12)*r(i,12);

%%%part 13%%%
l(i,13)=x(newpop(i,8),1);
r(i,13)=x(newpop(i,8),2);
%c(i,13)=x(newpop(i,8),3);
u(i,13)=l(i,13)*r(i,13);

%%%part 14%%%
l(i,14)=x(newpop(i,2),1)*x(newpop(i,11),1)*(x(newpop(i,2),2)+x(newpop(i,11),2));
r(i,14)=x(newpop(i,2),2)*x(newpop(i,11),2)/(x(newpop(i,2),2)+x(newpop(i,11),2));
%c(i,14)=x(newpop(i,2),3)+x(newpop(i,11),3);
u(i,14)=l(i,14)*r(i,14);

%%%part 15%%%
l(i,15)=x(newpop(i,1),1)*x(newpop(i,5),1)*(x(newpop(i,1),2)+x(newpop(i,5),2));
r(i,15)=x(newpop(i,1),2)*x(newpop(i,5),2)/(x(newpop(i,1),2)+x(newpop(i,5),2));
%c(i,15)=x(newpop(i,1),3)+x(newpop(i,5),3);
u(i,15)=l(i,15)*r(i,15);

%%%part 16%%%
l(i,16)=x(newpop(i,3),1)*x(newpop(i,11),1)*(x(newpop(i,3),2)+x(newpop(i,11),2));
r(i,16)=x(newpop(i,3),2)*x(newpop(i,11),2)/(x(newpop(i,3),2)+x(newpop(i,11),2));
%c(i,16)=x(newpop(i,4),3)+x(newpop(i,11),3);
u(i,16)=l(i,16)*r(i,16);

%%%part 17%%%
l(i,17)=x(newpop(i,5),1)*x(newpop(i,6),1)*(x(newpop(i,5),2)+x(newpop(i,6),2));
r(i,17)=x(newpop(i,5),2)*x(newpop(i,6),2)/(x(newpop(i,5),2)+x(newpop(i,6),2));
%c(i,17)=x(newpop(i,5),3)+x(newpop(i,5),3);
u(i,17)=l(i,17)*r(i,17);



%%%%%cost%%%%
c1=x(newpop(i,1),3);
c2=x(newpop(i,2),3);
c3=x(newpop(i,3),3);
c4=x(newpop(i,4),3);
c5=x(newpop(i,5),3);
c6=x(newpop(i,6),3);
c7=x(newpop(i,7),3);
c8=x(newpop(i,8),3);
c9=x(newpop(i,9),3);
c10=x(newpop(i,10),3);
c11=x(newpop(i,11),3);

L(i,1)=l(i,1)+l(i,2)+l(i,3)+l(i,4)+l(i,5)+l(i,6)+l(i,7)+l(i,8)+l(i,9)+l(i,10)+l(i,11
)+l(i,12)+l(i,13)+l(i,14)+l(i,15)+l(i,16)+l(i,17);
R(i,1)=r(i,1)+r(i,2)+r(i,3)+r(i,4)+r(i,5)+r(i,6)+r(i,7)+r(i,8)+r(i,9)+r(i,10)+r(i,11
)+r(i,12)+r(i,13)+r(i,14)+r(i,15)+r(i,16)+r(i,17);
C(i,1)=c1+c2+c3+c4+c5+c6+c7+c8+c9+c10+c11;
U(i,1)=L(i,1)*R(i,1);
end
matrix=[U(:,1),R(:,1),C(:,1),L(:,1)];
```

```
%%%%%%step:3- penalizing%%%%%
%%%%penalizing the constraints %%%%%
%%%%indexing%%%%%
f=zeros(100,1);
for i=1:100;
    f(i,1)=matrix(i,1);
if matrix(i,2)>=700
    f(i,1)=f(i,1)+3;
end
    if matrix(i,3)>=110;
       f(i,1)=f(i,1)+5;
     end
 end
for i=1:100
f(i,2)=i;
end
pf=sortrows(f);



%%%%step:4- Elitism%%%%
% selecting Nelite %elitism=30%
Nelite=zeros(1,1);
test=pf;
for i=1:round(100*.30),Nelite(i,1)=test(i,2);
end
%crossover Selection (ranking selection)
% selecting parents Pcross=70%
parents=zeros(1,1);
for i=1:round(100*.70),parents(i,1)=test(i,2);end
% create parent matrix
[psize,b]=size(parents);
for i=1:psize
pmatrix(i,:)=(newpop(parents(i,1),:));
end

index=[1,2;3, 4;5, 9;10, 11;0,0]
start=1;
fin=2;


%%%%step:5 crossover%%%%
%%crossover for system%%
for i=1:4
    clear blk1
    clear blk2
    clear tchild
    children=pmatrix;
    blk2(1,:)=pmatrix(psize,start:fin);
    for k=1:psize-1
        for j=1:psize
            blk1(1,:)=children(j,start:fin)
            tchild(j,:)=blk2;
            blk2=blk1;
        end
       children(1:psize,start:fin)=tchild;
        if i==1,
```

```
            cmatrix=children;
        else
            cmatrix((k-1)*psize+1:(k-1)*psize+psize,:)=children;
        end
    end
    start=index(i,1);
    fin=index(i,2);
    if i==1,
        cmatrix2=cmatrix;
    else
        cmatrix2=[cmatrix2;cmatrix];
    end
end


 %erasing all similar children
[csize,b]=size(cmatrix2);
index1=1;
while index1<=csize
    index2=index1+1;
    while index2<=csize
        if cmatrix2(index1,:)==cmatrix2(index2,:)
            cmatrix2(index2,:)=[];
            csize=csize-1;
        else
            index2=index2+1;
        end
    end
    index1=index1+1;
end




%%%%%step:6- Mutation%%%
%%% aplying mutation on selected children%%%

for i=1:csize
    mut=rand();
    if mut<.01
        a=gna2(1,11);
        if a>0 && a<3
            b=gna2(1,4);
            cmatrix2(i,a)=b;
        end
        if a>2 && a<=4;
            b=gna2(5,8);
            cmatrix2(i,a)=b;
        end
        if a>4 && a<=9;
            b=gna2(9,12);

            cmatrix2(i,a)=b;
        end
        if a>9 && a<=11;
            b=gna2(13,16);
            cmatrix2(i,a)=b;
        end
```

```
    end

end



%getting needed children
[m,b]=size(newpop);
cneeded=m-round(100*.30);
selectchildren=zeros(1,1);
ctest=cmatrix2;
for i=1:cneeded
    [a,b]=size(ctest);
    b=gna2(1,a);
    selectchildren(i,1)=b;
    ctest(b,:)=[];
end




%%%%%%% Step:7- new children%%%%%

%getting new population from children%%%
[a,b]=size(newpop);
for i=1:(round(100*.30))
    newpop(i,:)=newpop(Nelite(i,1),:);
end

for i=1:(100*.70)
       newpop(i+30,:)=cmatrix2(selectchildren(i,1),:)
end

end

%%%%for best solution%%%%%
%%%%%%%step:2- evaluation%%%%%
%%%%evaluating the A, r and c values for chromosomes%%%%
x=[0.098 20 12; 0.084 28 7; 0.055 12 18; 0.091 21 10; 0.0095 116 19; 0.0250 145 22;
0.0019 100 25; 0.0400 130 20; 0.0055 35 15; 0.0067 15 20; 0.0099 17 8; 0.0876 80 4;
0.250 5.9 3; 0.018 3.2 8; 0.035 3.0 7; 0.130 4.8 5];
for i=1:100

%%part 1%%
l(i,1)=x(crom(i,1),1)*x(crom(i,2),1)*(x(crom(i,1),2)+x(crom(i,2),2));
r(i,1)=x(crom(i,1),2)*x(crom(i,2),2)/(x(crom(i,1),2)+x(crom(i,2),2));
%c(i,1)=x(newpop(i,1),3);
u(i,1)=l(i,1)*r(i,1);

%%%part 2%%%
l(i,2)=x(newpop(i,2),1)*x(newpop(i,3),1)*(x(newpop(i,2),2)+x(newpop(i,3),2));
r(i,2)=x(newpop(i,2),2)*x(newpop(i,3),2)/(x(newpop(i,2),2)+x(newpop(i,3),2));
%c(i,2)=x(newpop(i,1),3);
u(i,2)=l(i,2)*r(i,2);

%%%part 3%%%
```

```
l(i,3)=x(newpop(i,2),1)*x(newpop(i,7),1)*(x(newpop(i,2),2)+x(newpop(i,7),2));
r(i,3)=x(newpop(i,2),2)*x(newpop(i,7),2)/(x(newpop(i,2),2)+x(newpop(i,7),2));
%c(i,3)=x(newpop(i,1),3);
u(i,3)=l(i,3)*r(i,3);



%%%part 4%%%
l(i,4)=x(newpop(i,3),1)*x(newpop(i,4),1)*(x(newpop(i,3),2)+x(newpop(i,4),2));
r(i,4)=x(newpop(i,3),2)*x(newpop(i,4),2)/(x(newpop(i,3),2)+x(newpop(i,4),2));
%c(i,4)=x(newpop(i,4),3);
u(i,4)=l(i,4)*r(i,4);

%%%part 5%%%
l(i,5)=x(newpop(i,3),1)*x(newpop(i,7),1)*(x(newpop(i,3),2)+x(newpop(i,7),2));
r(i,5)=x(newpop(i,3),2)*x(newpop(i,7),2)/(x(newpop(i,3),2)+x(newpop(i,7),2));
%c(i,5)=x(newpop(i,4),3);
u(i,5)=l(i,5)*r(i,5);

%%%parts 6%%%
l(i,6)=x(newpop(i,5),1)*x(newpop(i,11),1)*(x(newpop(i,5),2)+x(newpop(i,11),2));
r(i,6)=x(newpop(i,5),2)*x(newpop(i,11),2)/(x(newpop(i,5),2)+x(newpop(i,11),2));
%c(i,6)=x(newpop(i,5),3));
u(i,6)=l(i,6)*r(i,6);

%%%parts 7%%%
l(i,7)=x(newpop(i,10),1);
r(i,7)=x(newpop(i,10),2);
%c(i,7)=x(newpop(i,10),3);
u(i,7)=l(i,7)*r(i,7);

%%%part 8%%%
l(i,8)=x(newpop(i,2),1)*x(newpop(i,6),1)*(x(newpop(i,2),2)+x(newpop(i,6),2));
r(i,8)=x(newpop(i,2),2)*x(newpop(i,6),2)/(x(newpop(i,2),2)+x(newpop(i,6),2));
%c(i,8)=x(newpop(i,2),3)+x(newpop(i,6),3);
u(i,8)=l(i,8)*r(i,8);

%%%part 9%%%
l(i,9)=x(newpop(i,1),1)*x(newpop(i,3),1)*(x(newpop(i,1),2)+x(newpop(i,3),2));
r(i,9)=x(newpop(i,1),2)*x(newpop(i,3),2)/(x(newpop(i,1),2)+x(newpop(i,3),2));
%c(i,9)=x(newpop(i,1),3)+x(newpop(i,4),3);
u(i,9)=l(i,9)*r(i,9);

%%%part 10%%%
l(i,10)=x(newpop(i,3),1)*x(newpop(i,6),1)*(x(newpop(i,3),2)+x(newpop(i,6),2));
r(i,10)=x(newpop(i,3),2)*x(newpop(i,6),2)/(x(newpop(i,3),2)+x(newpop(i,6),2));
%c(i,10)=x(newpop(i,4),3)+x(newpop(i,6),3);
u(i,10)=l(i,10)*r(i,10);

%%%part 11%%%
l(i,11)=x(newpop(i,5),1)*x(newpop(i,4),1)*(x(newpop(i,5),2)+x(newpop(i,4),2));
r(i,11)=x(newpop(i,5),2)*x(newpop(i,4),2)/(x(newpop(i,5),2)+x(newpop(i,4),2));
%c(i,11)=x(newpop(i,5),3)+x(newpop(i,3),3);
u(i,11)=l(i,11)*r(i,11);

%%%part 12%%%
l(i,12)=x(newpop(i,5),1)*x(newpop(i,7),1)*(x(newpop(i,5),2)+x(newpop(i,7),2));
r(i,12)=x(newpop(i,5),2)*x(newpop(i,7),2)/(x(newpop(i,5),2)+x(newpop(i,7),2));
```

```
%c(i,12)=x(newpop(i,5),3)+x(newpop(i,7),3);
u(i,12)=l(i,12)*r(i,12);

%%%part 13%%%
l(i,13)=x(newpop(i,8),1);
r(i,13)=x(newpop(i,8),2);
%c(i,13)=x(newpop(i,8),3);
u(i,13)=l(i,13)*r(i,13);

%%%part 14%%%
l(i,14)=x(newpop(i,2),1)*x(newpop(i,11),1)*(x(newpop(i,2),2)+x(newpop(i,11),2));
r(i,14)=x(newpop(i,2),2)*x(newpop(i,11),2)/(x(newpop(i,2),2)+x(newpop(i,11),2));
%c(i,14)=x(newpop(i,2),3)+x(newpop(i,11),3);
u(i,14)=l(i,14)*r(i,14);

%%%part 15%%%
l(i,15)=x(newpop(i,1),1)*x(newpop(i,5),1)*(x(newpop(i,1),2)+x(newpop(i,5),2));
r(i,15)=x(newpop(i,1),2)*x(newpop(i,5),2)/(x(newpop(i,1),2)+x(newpop(i,5),2));
%c(i,15)=x(newpop(i,1),3)+x(newpop(i,5),3);
u(i,15)=l(i,15)*r(i,15);

%%%part 16%%%
l(i,16)=x(newpop(i,3),1)*x(newpop(i,11),1)*(x(newpop(i,3),2)+x(newpop(i,11),2));
r(i,16)=x(newpop(i,3),2)*x(newpop(i,11),2)/(x(newpop(i,3),2)+x(newpop(i,11),2));
%c(i,16)=x(newpop(i,4),3)+x(newpop(i,11),3);
u(i,16)=l(i,16)*r(i,16);

%%%part 17%%%
l(i,17)=x(newpop(i,5),1)*x(newpop(i,6),1)*(x(newpop(i,5),2)+x(newpop(i,6),2));
r(i,17)=x(newpop(i,5),2)*x(newpop(i,6),2)/(x(newpop(i,5),2)+x(newpop(i,6),2));
%c(i,17)=x(newpop(i,5),3)+x(newpop(i,5),3);
u(i,17)=l(i,17)*r(i,17);




%%%%%cost%%%%
c1=x(newpop(i,1),3);
c2=x(newpop(i,2),3);
c3=x(newpop(i,3),3);
c4=x(newpop(i,4),3);
c5=x(newpop(i,5),3);
c6=x(newpop(i,6),3);
c7=x(newpop(i,7),3);
c8=x(newpop(i,8),3);
c9=x(newpop(i,9),3);
c10=x(newpop(i,10),3);
c11=x(newpop(i,11),3);

L(i,1)=l(i,1)+l(i,2)+l(i,3)+l(i,4)+l(i,5)+l(i,6)+l(i,7)+l(i,8)+l(i,9)+l(i,10)+l(i,11
)+l(i,12)+l(i,13)+l(i,14)+l(i,15)+l(i,16)+l(i,17);
R(i,1)=r(i,1)+r(i,2)+r(i,3)+r(i,4)+r(i,5)+r(i,6)+r(i,7)+r(i,8)+r(i,9)+r(i,10)+r(i,11
)+r(i,12)+r(i,13)+r(i,14)+r(i,15)+r(i,16)+r(i,17);
C(i,1)=c1+c2+c3+c4+c5+c6+c7+c8+c9+c10+c11;
U(i,1)=L(i,1)*R(i,1);
end
matrix=[U(:,1),R(:,1),C(:,1),L(:,1)];
```

```
%%%%%%step:3- penalizing%%%%%
%%%%penalizing the constraints %%%%%
%%%%indexing%%%%%
f=zeros(100,1);
for i=1:100;
    f(i,1)=matrix(i,1);
if matrix(i,2)>=700
    f(i,1)=f(i,1)+3;
end
    if matrix(i,3)>=110;
       f(i,1)=f(i,1)+5;
     end
 end
for i=1:100
f(i,2)=i;
end
pf=sortrows(f)
```

Appendix II: MATLAB code for example: 2.

```
%%%%%%%STEP:1- CREATING CHROMOSOMES%%%%%%%%
crom = zeros(100,11);
for i=1:100
    for j = 1:2;  % no of lines
        crom(i,j)=ceil(4.*rand());
    end
    for j=3:4;
        crom(i,j)=ceil(8.*rand());
        while  crom(i,j)<5
        crom(i,j)=ceil(8.*rand());
        end
    end
    for j=5:9;
        crom(i,j)=ceil(12.*rand());
        while crom(i,j)<9
        crom(i,j)=ceil(12.*rand());
        end
    end
    for j=10:11;
        crom(i,j)=ceil(16.*rand());
        while crom(i,j)<13
        crom(i,j)=ceil(16.*rand());
        end
    end
end
crom

%%%%%%%step:2- evaluation%%%%%
%%%%evaluating the A, r and c values for chromosomes%%%%
x=[0.098 20 12; 0.084 28 7; 0.055 12 18; 0.091 21 10; 0.0095 116 19; 0.0250 145 22;
0.0019 100 25; 0.0400 130 20; 0.0055 35 15; 0.0067 15 20; 0.0099 17 8; 0.0876 80 4;
0.250 5.9 3; 0.018 3.2 8; 0.035 3.0 7; 0.130 4.8 5];
for i=1:100

%%part 1%%
```

```
l(i,1)=x(crom(i,11),1)*x(crom(i,10),1)*(x(crom(i,11),2)+x(crom(i,10),2));
r(i,1)=x(crom(i,11),2)*x(crom(i,10),2)/(x(crom(i,11),2)+x(crom(i,10),2));
u(i,1)=l(i,1)*r(i,1);

%%%part 2%%%
l(i,2)=x(crom(i,11),1)*x(crom(i,8),1)*(x(crom(i,11),2)+x(crom(i,8),2));
r(i,2)=x(crom(i,11),2)*x(crom(i,8),2)/(x(crom(i,11),2)+x(crom(i,8),2));
u(i,2)=l(i,2)*r(i,2);

%%%part 3%%%
l(i,3)=x(crom(i,9),1)*x(crom(i,10),1)*(x(crom(i,9),2)+x(crom(i,10),2));
r(i,3)=x(crom(i,9),2)*x(crom(i,10),2)/(x(crom(i,9),2)+x(crom(i,10),2));
u(i,3)=l(i,3)*r(i,3);



%%%part 4%%%
l(i,4)=x(crom(i,9),1)*x(crom(i,8),1)*(x(crom(i,9),2)+x(crom(i,8),2));
r(i,4)=x(crom(i,9),2)*x(crom(i,8),2)/(x(crom(i,9),2)+x(crom(i,8),2));
u(i,4)=l(i,4)*r(i,4);

%%%part 5%%%
l(i,5)=x(crom(i,6),1)*x(crom(i,5),1)*(x(crom(i,6),2)+x(crom(i,5),2));
r(i,5)=x(crom(i,6),2)*x(crom(i,5),2)/(x(crom(i,6),2)+x(crom(i,5),2));
u(i,5)=l(i,5)*r(i,5);

%%%parts 6%%%
l(i,6)=x(crom(i,6),1)*x(crom(i,3),1)*(x(crom(i,6),2)+x(crom(i,3),2));
r(i,6)=x(crom(i,6),2)*x(crom(i,3),2)/(x(crom(i,6),2)+x(crom(i,3),2));
u(i,6)=l(i,6)*r(i,6);

%%%parts 7%%%
l(i,7)=x(crom(i,6),1)*x(crom(i,2),1)*(x(crom(i,6),2)+x(crom(i,2),2));
r(i,7)=x(crom(i,6),2)*x(crom(i,2),2)/(x(crom(i,6),2)+x(crom(i,2),2));
u(i,7)=l(i,7)*r(i,7);

%%%part 8%%%
l(i,8)=x(crom(i,4),1)*x(crom(i,5),1)*(x(crom(i,4),2)+x(crom(i,5),2));
r(i,8)=x(crom(i,4),2)*x(crom(i,5),2)/(x(crom(i,4),2)+x(crom(i,5),2));
u(i,8)=l(i,8)*r(i,8);

%%%part 9%%%
l(i,9)=x(crom(i,4),1)*x(crom(i,3),1)*(x(crom(i,4),2)+x(crom(i,3),2));
r(i,9)=x(crom(i,4),2)*x(crom(i,3),2)/(x(crom(i,4),2)+x(crom(i,3),2));
u(i,9)=l(i,9)*r(i,9);

%%%part 10%%%
l(i,10)=x(crom(i,4),1)*x(crom(i,2),1)*(x(crom(i,4),2)+x(crom(i,2),2));
r(i,10)=x(crom(i,4),2)*x(crom(i,2),2)/(x(crom(i,4),2)+x(crom(i,2),2));
u(i,10)=l(i,10)*r(i,10);

%%%part 11%%%
l(i,11)=x(crom(i,1),1)*x(crom(i,5),1)*(x(crom(i,1),2)+x(crom(i,5),2));
r(i,11)=x(crom(i,1),2)*x(crom(i,5),2)/(x(crom(i,1),2)+x(crom(i,5),2));
u(i,11)=l(i,11)*r(i,11);

%%%part 12%%%
l(i,12)=x(crom(i,1),1)*x(crom(i,3),1)*(x(crom(i,1),2)+x(crom(i,3),2));
```

```
r(i,12)=x(crom(i,1),2)*x(crom(i,3),2)/(x(crom(i,1),2)+x(crom(i,3),2));
u(i,12)=l(i,12)*r(i,12);

%%%part 13%%%
l(i,13)=x(crom(i,1),1)*x(crom(i,2),1)*(x(crom(i,1),2)+x(crom(i,2),2));
r(i,13)=x(crom(i,1),2)*x(crom(i,2),2)/(x(crom(i,1),2)+x(crom(i,2),2));
u(i,13)=l(i,13)*r(i,13);

%%%part 14%%%
l(i,14)=x(crom(i,10),1)*x(crom(i,6),1)*(x(crom(i,10),2)+x(crom(i,6),2));
r(i,14)=x(crom(i,10),2)*x(crom(i,6),2)/(x(crom(i,10),2)+x(crom(i,6),2));
u(i,14)=l(i,14)*r(i,14);

%%%part 15%%%
l(i,15)=x(crom(i,10),1)*x(crom(i,4),1)*(x(crom(i,10),2)+x(crom(i,4),2));
r(i,15)=x(crom(i,10),2)*x(crom(i,4),2)/(x(crom(i,10),2)+x(crom(i,4),2));
u(i,15)=l(i,15)*r(i,15);

%%%part 16%%%
l(i,16)=x(crom(i,10),1)*x(crom(i,1),1)*(x(crom(i,10),2)+x(crom(i,1),2));
r(i,16)=x(crom(i,10),2)*x(crom(i,1),2)/(x(crom(i,10),2)+x(crom(i,1),2));
u(i,16)=l(i,16)*r(i,16);

%%%part 17%%%
l(i,17)=x(crom(i,11),1)*x(crom(i,5),1)*(x(crom(i,11),2)+x(crom(i,5),2));
r(i,17)=x(crom(i,11),2)*x(crom(i,5),2)/(x(crom(i,11),2)+x(crom(i,5),2));
u(i,17)=l(i,17)*r(i,17);

%%%%%%Part 18%%%%%
l(i,18)=x(crom(i,11),1)*x(crom(i,3),1)*(x(crom(i,11),2)+x(crom(i,3),2));
r(i,18)=x(crom(i,11),2)*x(crom(i,3),2)/(x(crom(i,11),2)+x(crom(i,3),2));
u(i,18)=l(i,18)*r(i,18);



%%%%%%Part 19%%%%%%
l(i,19)=x(crom(i,11),1)*x(crom(i,2),1)*(x(crom(i,11),2)+x(crom(i,2),2));
r(i,19)=x(crom(i,11),2)*x(crom(i,2),2)/(x(crom(i,11),2)+x(crom(i,2),2));
u(i,19)=l(i,19)*r(i,19);



%%%%%cost%%%%
c1=x(crom(i,1),3);
c2=x(crom(i,2),3);
c3=x(crom(i,3),3);
c4=x(crom(i,4),3);
c5=x(crom(i,5),3);
c6=x(crom(i,6),3);
c7=x(crom(i,7),3);
c8=x(crom(i,8),3);
c9=x(crom(i,9),3);
c10=x(crom(i,10),3);
c11=x(crom(i,11),3);

L(i,1)=l(i,1)+l(i,2)+l(i,3)+l(i,4)+l(i,5)+l(i,6)+l(i,7)+l(i,8)+l(i,9)+l(i,10)+l(i,11
)+l(i,12)+l(i,13)+l(i,14)+l(i,15)+l(i,16)+l(i,17)+l(i,18)+l(i,19);
R(i,1)=r(i,1)+r(i,2)+r(i,3)+r(i,4)+r(i,5)+r(i,6)+r(i,7)+r(i,8)+r(i,9)+r(i,10)+r(i,11
)+r(i,12)+r(i,13)+r(i,14)+r(i,15)+r(i,16)+r(i,17)+r(i,18)+r(i,19);
```

```
C(i,1)=c1+c2+c3+c4+c5+c6+c7+c8+c9+c10+c11;
U(i,1)=L(i,1)*R(i,1);
end
matrix=[U(:,1),R(:,1),C(:,1),L(:,1)];


%%%%%%step:3- penalizing%%%%%
%%%%penalizing the constraints %%%%%
%%%%indexing%%%%%
f=zeros(100,1);
for i=1:100;
    f(i,1)=matrix(i,1);
if matrix(i,2)>=225
    f(i,1)=f(i,1)+3;
end
    if matrix(i,3)>=140;
       f(i,1)=f(i,1)+5;
     end
 end
for i=1:100
f(i,2)=i;
end
pf=sortrows(f);



%%%%step:4- Elitism%%%%
% selecting Nelite %elitism=30%
Nelite=zeros(1,1);
test=pf;
for i=1:round(100*.30),Nelite(i,1)=test(i,2);end
%crossover Selection (ranking selection)
% selecting parents Pcross=70%
parents=zeros(1,1);
for i=1:round(100*.70),parents(i,1)=test(i,2);end
% create parent matrix
[psize,b]=size(parents);
for i=1:psize
pmatrix(i,:)=(crom(parents(i,1),:));
end

index=[1,2;3, 4;5, 9;10, 11;0,0]
start=1;
fin=2;


%%%%step:5 crossover%%%%
%%crossover for system%%
for i=1:4
    clear blk1
    clear blk2
    clear tchild
    children=pmatrix;
    blk2(1,:)=pmatrix(psize,start:fin);
    for k=1:psize-1
        for j=1:psize
            blk1(1,:)=children(j,start:fin)
            tchild(j,:)=blk2;
```

```
            blk2=blk1;
        end
       children(1:psize,start:fin)=tchild;
        if i==1,
            cmatrix=children;
        else
            cmatrix((k-1)*psize+1:(k-1)*psize+psize,:)=children;
        end
    end
    start=index(i,1);
    fin=index(i,2);
    if i==1,
        cmatrix2=cmatrix;
    else
        cmatrix2=[cmatrix2;cmatrix];
    end
end

 %erasing all similar children
[csize,b]=size(cmatrix2);
index1=1;
while index1<=csize
    index2=index1+1;
    while index2<=csize
        if cmatrix2(index1,:)==cmatrix2(index2,:)
            cmatrix2(index2,:)=[];
            csize=csize-1;
        else
            index2=index2+1;
        end
    end
    index1=index1+1;
end




%%%%%step:6- Mutation%%%
%%% aplying mutation on selected children%%%

for i=1:csize
    mut=rand();
    if mut<.01
        a=gna2(1,11);
        if a>0 && a<3
            b=gna2(1,4);
            cmatrix2(i,a)=b;
        end
        if a>2 && a<=4;
            b=gna2(5,8);
            cmatrix2(i,a)=b;
        end
        if a>4 && a<=9;
            b=gna2(9,12);

            cmatrix2(i,a)=b;
        end
        if a>9 && a<=11;
```

```
            b=gna2(13,16);
            cmatrix2(i,a)=b;
        end
    end
end

%getting needed children
[m,b]=size(crom);
cneeded=m-round(100*.30);
selectchildren=zeros(1,1);
ctest=cmatrix2;
for i=1:cneeded
    [a,b]=size(ctest);
    b=gna2(1,a);
    selectchildren(i,1)=b;
    ctest(b,:)=[];
end




%%%%%%% Step:7- new children%%%%%

%getting new population from children%%%
[a,b]=size(crom);
for i=1:(round(100*.30))
    newpop(i,:)=crom(Nelite(i,1),:);
end

for i=1:(100*.70)
       newpop(i+30,:)=cmatrix2(selectchildren(i,1),:);
end

for i=1:25

%%%%%%%step:2- evaluation%%%%%
%%%%evaluating the A, r and c values for chromosomes%%%%
x=[0.098 20 12; 0.084 28 7; 0.055 12 18; 0.091 21 10; 0.0095 116 19; 0.0250 145 22;
0.0019 100 25; 0.0400 130 20; 0.0055 35 15; 0.0067 15 20; 0.0099 17 8; 0.0876 80 4;
0.250 5.9 3; 0.018 3.2 8; 0.035 3.0 7; 0.130 4.8 5];
for i=1:100

%%part 1%%
l(i,1)=x(newpop(i,11),1)*x(newpop(i,10),1)*(x(newpop(i,11),2)+x(newpop(i,10),2));
r(i,1)=x(newpop(i,11),2)*x(newpop(i,10),2)/(x(newpop(i,11),2)+x(newpop(i,10),2));
u(i,1)=l(i,1)*r(i,1);

%%%part 2%%%
l(i,2)=x(newpop(i,11),1)*x(newpop(i,8),1)*(x(newpop(i,11),2)+x(newpop(i,8),2));
r(i,2)=x(newpop(i,11),2)*x(newpop(i,8),2)/(x(newpop(i,11),2)+x(newpop(i,8),2));
u(i,2)=l(i,2)*r(i,2);

%%%part 3%%%
l(i,3)=x(newpop(i,9),1)*x(newpop(i,10),1)*(x(newpop(i,9),2)+x(newpop(i,10),2));
r(i,3)=x(newpop(i,9),2)*x(newpop(i,10),2)/(x(newpop(i,9),2)+x(newpop(i,10),2));
u(i,3)=l(i,3)*r(i,3);
```

```
%%%part 4%%%
l(i,4)=x(newpop(i,9),1)*x(newpop(i,8),1)*(x(newpop(i,9),2)+x(newpop(i,8),2));
r(i,4)=x(newpop(i,9),2)*x(newpop(i,8),2)/(x(newpop(i,9),2)+x(newpop(i,8),2));
u(i,4)=l(i,4)*r(i,4);

%%%part 5%%%
l(i,5)=x(newpop(i,6),1)*x(newpop(i,5),1)*(x(newpop(i,6),2)+x(newpop(i,5),2));
r(i,5)=x(newpop(i,6),2)*x(newpop(i,5),2)/(x(newpop(i,6),2)+x(newpop(i,5),2));
u(i,5)=l(i,5)*r(i,5);

%%%parts 6%%%
l(i,6)=x(newpop(i,6),1)*x(newpop(i,3),1)*(x(newpop(i,6),2)+x(newpop(i,3),2));
r(i,6)=x(newpop(i,6),2)*x(newpop(i,3),2)/(x(newpop(i,6),2)+x(newpop(i,3),2));
u(i,6)=l(i,6)*r(i,6);

%%%parts 7%%%
l(i,7)=x(newpop(i,6),1)*x(newpop(i,2),1)*(x(newpop(i,6),2)+x(newpop(i,2),2));
r(i,7)=x(newpop(i,6),2)*x(newpop(i,2),2)/(x(newpop(i,6),2)+x(newpop(i,2),2));
u(i,7)=l(i,7)*r(i,7);

%%%part 8%%%
l(i,8)=x(newpop(i,4),1)*x(newpop(i,5),1)*(x(newpop(i,4),2)+x(newpop(i,5),2));
r(i,8)=x(newpop(i,4),2)*x(newpop(i,5),2)/(x(newpop(i,4),2)+x(newpop(i,5),2));
u(i,8)=l(i,8)*r(i,8);

%%%part 9%%%
l(i,9)=x(newpop(i,4),1)*x(newpop(i,3),1)*(x(newpop(i,4),2)+x(newpop(i,3),2));
r(i,9)=x(newpop(i,4),2)*x(newpop(i,3),2)/(x(newpop(i,4),2)+x(newpop(i,3),2));
u(i,9)=l(i,9)*r(i,9);

%%%part 10%%%
l(i,10)=x(newpop(i,4),1)*x(newpop(i,2),1)*(x(newpop(i,4),2)+x(newpop(i,2),2));
r(i,10)=x(newpop(i,4),2)*x(newpop(i,2),2)/(x(newpop(i,4),2)+x(newpop(i,2),2));
u(i,10)=l(i,10)*r(i,10);

%%%part 11%%%
l(i,11)=x(newpop(i,1),1)*x(newpop(i,5),1)*(x(newpop(i,1),2)+x(newpop(i,5),2));
r(i,11)=x(newpop(i,1),2)*x(newpop(i,5),2)/(x(newpop(i,1),2)+x(newpop(i,5),2));
u(i,11)=l(i,11)*r(i,11);

%%%part 12%%%
l(i,12)=x(newpop(i,1),1)*x(newpop(i,3),1)*(x(newpop(i,1),2)+x(newpop(i,3),2));
r(i,12)=x(newpop(i,1),2)*x(newpop(i,3),2)/(x(newpop(i,1),2)+x(newpop(i,3),2));
u(i,12)=l(i,12)*r(i,12);

%%%part 13%%%
l(i,13)=x(newpop(i,1),1)*x(newpop(i,2),1)*(x(newpop(i,1),2)+x(newpop(i,2),2));
r(i,13)=x(newpop(i,1),2)*x(newpop(i,2),2)/(x(newpop(i,1),2)+x(newpop(i,2),2));
u(i,13)=l(i,13)*r(i,13);

%%%part 14%%%
l(i,14)=x(newpop(i,10),1)*x(newpop(i,6),1)*(x(newpop(i,10),2)+x(newpop(i,6),2));
r(i,14)=x(newpop(i,10),2)*x(newpop(i,6),2)/(x(newpop(i,10),2)+x(newpop(i,6),2));
u(i,14)=l(i,14)*r(i,14);

%%%part 15%%%
```

```
l(i,15)=x(newpop(i,10),1)*x(newpop(i,4),1)*(x(newpop(i,10),2)+x(newpop(i,4),2));
r(i,15)=x(newpop(i,10),2)*x(newpop(i,4),2)/(x(newpop(i,10),2)+x(newpop(i,4),2));
u(i,15)=l(i,15)*r(i,15);

%%%part 16%%%
l(i,16)=x(newpop(i,10),1)*x(newpop(i,1),1)*(x(newpop(i,10),2)+x(newpop(i,1),2));
r(i,16)=x(newpop(i,10),2)*x(newpop(i,1),2)/(x(newpop(i,10),2)+x(newpop(i,1),2));
u(i,16)=l(i,16)*r(i,16);

%%%part 17%%%
l(i,17)=x(newpop(i,11),1)*x(newpop(i,5),1)*(x(newpop(i,11),2)+x(newpop(i,5),2));
r(i,17)=x(newpop(i,11),2)*x(newpop(i,5),2)/(x(newpop(i,11),2)+x(newpop(i,5),2));
u(i,17)=l(i,17)*r(i,17);

%%%%%%Part 18%%%%%
l(i,18)=x(newpop(i,11),1)*x(newpop(i,3),1)*(x(newpop(i,11),2)+x(newpop(i,3),2));
r(i,18)=x(newpop(i,11),2)*x(newpop(i,3),2)/(x(newpop(i,11),2)+x(newpop(i,3),2));
u(i,18)=l(i,18)*r(i,18);



%%%%%%Part 19%%%%%%
l(i,19)=x(newpop(i,11),1)*x(newpop(i,2),1)*(x(newpop(i,11),2)+x(newpop(i,2),2));
r(i,19)=x(newpop(i,11),2)*x(newpop(i,2),2)/(x(newpop(i,11),2)+x(newpop(i,2),2));
u(i,19)=l(i,19)*r(i,19);



%%%%%cost%%%%
c1=x(newpop(i,1),3);
c2=x(newpop(i,2),3);
c3=x(newpop(i,3),3);
c4=x(newpop(i,4),3);
c5=x(newpop(i,5),3);
c6=x(newpop(i,6),3);
c7=x(newpop(i,7),3);
c8=x(newpop(i,8),3);
c9=x(newpop(i,9),3);
c10=x(newpop(i,10),3);
c11=x(newpop(i,11),3);

L(i,1)=l(i,1)+l(i,2)+l(i,3)+l(i,4)+l(i,5)+l(i,6)+l(i,7)+l(i,8)+l(i,9)+l(i,10)+l(i,11
)+l(i,12)+l(i,13)+l(i,14)+l(i,15)+l(i,16)+l(i,17)+l(i,18)+l(i,19);
R(i,1)=r(i,1)+r(i,2)+r(i,3)+r(i,4)+r(i,5)+r(i,6)+r(i,7)+r(i,8)+r(i,9)+r(i,10)+r(i,11
)+r(i,12)+r(i,13)+r(i,14)+r(i,15)+r(i,16)+r(i,17)+r(i,18)+r(i,19);
C(i,1)=c1+c2+c3+c4+c5+c6+c7+c8+c9+c10+c11;
U(i,1)=L(i,1)*R(i,1);
end
matrix=[U(:,1),R(:,1),C(:,1),L(:,1)];



%%%%%%step:3- penalizing%%%%%
%%%%penalizing the constraints %%%%%
%%%%indexing%%%%%
f=zeros(100,1);
for i=1:100;
    f(i,1)=matrix(i,1);
if matrix(i,2)>=225
    f(i,1)=f(i,1)+3;
```

```
end
    if matrix(i,3)>=140;
       f(i,1)=f(i,1)+5;
     end
 end
for i=1:100
f(i,2)=i;
end
pf=sortrows(f);



%%%%step:4- Elitism%%%%
% selecting Nelite %elitism=30%
Nelite=zeros(1,1);
test=pf;
for i=1:round(100*.30),Nelite(i,1)=test(i,2);end
%crossover Selection (ranking selection)
% selecting parents Pcross=70%
parents=zeros(1,1);
for i=1:round(100*.70),parents(i,1)=test(i,2);end
% create parent matrix
[psize,b]=size(parents);
for i=1:psize
pmatrix(i,:)=(crom(parents(i,1),:));
end

index=[1,2;3, 4;5, 9;10, 11;0,0]
start=1;
fin=2;



%%%%step:5 crossover%%%%
%%crossover for system%%
for i=1:4
    clear blk1
    clear blk2
    clear tchild
    children=pmatrix;
    blk2(1,:)=pmatrix(psize,start:fin);
    for k=1:psize-1
        for j=1:psize
            blk1(1,:)=children(j,start:fin)
            tchild(j,:)=blk2;
            blk2=blk1;
        end
       children(1:psize,start:fin)=tchild;
        if i==1,
            cmatrix=children;
        else
            cmatrix((k-1)*psize+1:(k-1)*psize+psize,:)=children;
        end
    end
    start=index(i,1);
    fin=index(i,2);
    if i==1,
        cmatrix2=cmatrix;
    else
```

```
        cmatrix2=[cmatrix2;cmatrix];
    end
end

 %erasing all similar children
[csize,b]=size(cmatrix2);
index1=1;
while index1<=csize
    index2=index1+1;
    while index2<=csize
        if cmatrix2(index1,:)==cmatrix2(index2,:)
            cmatrix2(index2,:)=[];
            csize=csize-1;
        else
            index2=index2+1;
        end
    end
    index1=index1+1;
end




%%%%%step:6- Mutation%%%
%%% aplying mutation on selected children%%%

for i=1:csize
    mut=rand();
    if mut<.01
        a=gna2(1,11);
        if a>0 && a<3
            b=gna2(1,4);
            cmatrix2(i,a)=b;
        end
        if a>2 && a<=4;
            b=gna2(5,8);
            cmatrix2(i,a)=b;
        end
        if a>4 && a<=9;
            b=gna2(9,12);

            cmatrix2(i,a)=b;
        end
        if a>9 && a<=11;
            b=gna2(13,16);
            cmatrix2(i,a)=b;
        end
    end
end

%getting needed children
[m,b]=size(crom);
cneeded=m-round(100*.30);
selectchildren=zeros(1,1);
ctest=cmatrix2;
for i=1:cneeded
    [a,b]=size(ctest);
    b=gna2(1,a);
```

```
    selectchildren(i,1)=b;
    ctest(b,:)=[];
end




%%%%%%% Step:7- new children%%%%%

%getting new population from children%%%
[a,b]=size(crom);
for i=1:(round(100*.30))
    newpop(i,:)=crom(Nelite(i,1),:);
end

for i=1:(100*.70)
       newpop(i+30,:)=cmatrix2(selectchildren(i,1),:);
end



%%%%%%%step:2- evaluation%%%%%
%%%%evaluating the A, r and c values for chromosomes%%%%
x=[0.098 20 12; 0.084 28 7; 0.055 12 18; 0.091 21 10; 0.0095 116 19; 0.0250 145 22;
0.0019 100 25; 0.0400 130 20; 0.0055 35 15; 0.0067 15 20; 0.0099 17 8; 0.0876 80 4;
0.250 5.9 3; 0.018 3.2 8; 0.035 3.0 7; 0.130 4.8 5];
for i=1:100

%%part 1%%
l(i,1)=x(newpop(i,11),1)*x(newpop(i,10),1)*(x(newpop(i,11),2)+x(newpop(i,10),2));
r(i,1)=x(newpop(i,11),2)*x(newpop(i,10),2)/(x(newpop(i,11),2)+x(newpop(i,10),2));
u(i,1)=l(i,1)*r(i,1);

%%%part 2%%%
l(i,2)=x(newpop(i,11),1)*x(newpop(i,8),1)*(x(newpop(i,11),2)+x(newpop(i,8),2));
r(i,2)=x(newpop(i,11),2)*x(newpop(i,8),2)/(x(newpop(i,11),2)+x(newpop(i,8),2));
u(i,2)=l(i,2)*r(i,2);

%%%part 3%%%
l(i,3)=x(newpop(i,9),1)*x(newpop(i,10),1)*(x(newpop(i,9),2)+x(newpop(i,10),2));
r(i,3)=x(newpop(i,9),2)*x(newpop(i,10),2)/(x(newpop(i,9),2)+x(newpop(i,10),2));
u(i,3)=l(i,3)*r(i,3);



%%%part 4%%%
l(i,4)=x(newpop(i,9),1)*x(newpop(i,8),1)*(x(newpop(i,9),2)+x(newpop(i,8),2));
r(i,4)=x(newpop(i,9),2)*x(newpop(i,8),2)/(x(newpop(i,9),2)+x(newpop(i,8),2));
u(i,4)=l(i,4)*r(i,4);

%%%part 5%%%
l(i,5)=x(newpop(i,6),1)*x(newpop(i,5),1)*(x(newpop(i,6),2)+x(newpop(i,5),2));
r(i,5)=x(newpop(i,6),2)*x(newpop(i,5),2)/(x(newpop(i,6),2)+x(newpop(i,5),2));
u(i,5)=l(i,5)*r(i,5);

%%%parts 6%%%
l(i,6)=x(newpop(i,6),1)*x(newpop(i,3),1)*(x(newpop(i,6),2)+x(newpop(i,3),2));
r(i,6)=x(newpop(i,6),2)*x(newpop(i,3),2)/(x(newpop(i,6),2)+x(newpop(i,3),2));
```

```
u(i,6)=l(i,6)*r(i,6);

%%%parts 7%%%
l(i,7)=x(newpop(i,6),1)*x(newpop(i,2),1)*(x(newpop(i,6),2)+x(newpop(i,2),2));
r(i,7)=x(newpop(i,6),2)*x(newpop(i,2),2)/(x(newpop(i,6),2)+x(newpop(i,2),2));
u(i,7)=l(i,7)*r(i,7);

%%%part 8%%%
l(i,8)=x(newpop(i,4),1)*x(newpop(i,5),1)*(x(newpop(i,4),2)+x(newpop(i,5),2));
r(i,8)=x(newpop(i,4),2)*x(newpop(i,5),2)/(x(newpop(i,4),2)+x(newpop(i,5),2));
u(i,8)=l(i,8)*r(i,8);

%%%part 9%%%
l(i,9)=x(newpop(i,4),1)*x(newpop(i,3),1)*(x(newpop(i,4),2)+x(newpop(i,3),2));
r(i,9)=x(newpop(i,4),2)*x(newpop(i,3),2)/(x(newpop(i,4),2)+x(newpop(i,3),2));
u(i,9)=l(i,9)*r(i,9);

%%%part 10%%%
l(i,10)=x(newpop(i,4),1)*x(newpop(i,2),1)*(x(newpop(i,4),2)+x(newpop(i,2),2));
r(i,10)=x(newpop(i,4),2)*x(newpop(i,2),2)/(x(newpop(i,4),2)+x(newpop(i,2),2));
u(i,10)=l(i,10)*r(i,10);

%%%part 11%%%
l(i,11)=x(newpop(i,1),1)*x(newpop(i,5),1)*(x(newpop(i,1),2)+x(newpop(i,5),2));
r(i,11)=x(newpop(i,1),2)*x(newpop(i,5),2)/(x(newpop(i,1),2)+x(newpop(i,5),2));
u(i,11)=l(i,11)*r(i,11);

%%%part 12%%%
l(i,12)=x(newpop(i,1),1)*x(newpop(i,3),1)*(x(newpop(i,1),2)+x(newpop(i,3),2));
r(i,12)=x(newpop(i,1),2)*x(newpop(i,3),2)/(x(newpop(i,1),2)+x(newpop(i,3),2));
u(i,12)=l(i,12)*r(i,12);

%%%part 13%%%
l(i,13)=x(newpop(i,1),1)*x(newpop(i,2),1)*(x(newpop(i,1),2)+x(newpop(i,2),2));
r(i,13)=x(newpop(i,1),2)*x(newpop(i,2),2)/(x(newpop(i,1),2)+x(newpop(i,2),2));
u(i,13)=l(i,13)*r(i,13);

%%%part 14%%%
l(i,14)=x(newpop(i,10),1)*x(newpop(i,6),1)*(x(newpop(i,10),2)+x(newpop(i,6),2));
r(i,14)=x(newpop(i,10),2)*x(newpop(i,6),2)/(x(newpop(i,10),2)+x(newpop(i,6),2));
u(i,14)=l(i,14)*r(i,14);

%%%part 15%%%
l(i,15)=x(newpop(i,10),1)*x(newpop(i,4),1)*(x(newpop(i,10),2)+x(newpop(i,4),2));
r(i,15)=x(newpop(i,10),2)*x(newpop(i,4),2)/(x(newpop(i,10),2)+x(newpop(i,4),2));
u(i,15)=l(i,15)*r(i,15);

%%%part 16%%%
l(i,16)=x(newpop(i,10),1)*x(newpop(i,1),1)*(x(newpop(i,10),2)+x(newpop(i,1),2));
r(i,16)=x(newpop(i,10),2)*x(newpop(i,1),2)/(x(newpop(i,10),2)+x(newpop(i,1),2));
u(i,16)=l(i,16)*r(i,16);

%%%part 17%%%
l(i,17)=x(newpop(i,11),1)*x(newpop(i,5),1)*(x(newpop(i,11),2)+x(newpop(i,5),2));
r(i,17)=x(newpop(i,11),2)*x(newpop(i,5),2)/(x(newpop(i,11),2)+x(newpop(i,5),2));
u(i,17)=l(i,17)*r(i,17);
```

```
%%%%%%Part 18%%%%%
l(i,18)=x(newpop(i,11),1)*x(newpop(i,3),1)*(x(newpop(i,11),2)+x(newpop(i,3),2));
r(i,18)=x(newpop(i,11),2)*x(newpop(i,3),2)/(x(newpop(i,11),2)+x(newpop(i,3),2));
u(i,18)=l(i,18)*r(i,18);


%%%%%%Part 19%%%%%%
l(i,19)=x(newpop(i,11),1)*x(newpop(i,2),1)*(x(newpop(i,11),2)+x(newpop(i,2),2));
r(i,19)=x(newpop(i,11),2)*x(newpop(i,2),2)/(x(newpop(i,11),2)+x(newpop(i,2),2));
u(i,19)=l(i,19)*r(i,19);


%%%%%cost%%%%
c1=x(newpop(i,1),3);
c2=x(newpop(i,2),3);
c3=x(newpop(i,3),3);
c4=x(newpop(i,4),3);
c5=x(newpop(i,5),3);
c6=x(newpop(i,6),3);
c7=x(newpop(i,7),3);
c8=x(newpop(i,8),3);
c9=x(newpop(i,9),3);
c10=x(newpop(i,10),3);
c11=x(newpop(i,11),3);

L(i,1)=l(i,1)+l(i,2)+l(i,3)+l(i,4)+l(i,5)+l(i,6)+l(i,7)+l(i,8)+l(i,9)+l(i,10)+l(i,11
)+l(i,12)+l(i,13)+l(i,14)+l(i,15)+l(i,16)+l(i,17)+l(i,18)+l(i,19);
R(i,1)=r(i,1)+r(i,2)+r(i,3)+r(i,4)+r(i,5)+r(i,6)+r(i,7)+r(i,8)+r(i,9)+r(i,10)+r(i,11
)+r(i,12)+r(i,13)+r(i,14)+r(i,15)+r(i,16)+r(i,17)+r(i,18)+r(i,19);
C(i,1)=c1+c2+c3+c4+c5+c6+c7+c8+c9+c10+c11;
U(i,1)=L(i,1)*R(i,1);
end
matrix=[U(:,1),R(:,1),C(:,1),L(:,1)];

%%%%%%step:3- penalizing%%%%%
%%%%penalizing the constraints %%%%%
%%%%indexing%%%%%
f=zeros(100,1);
for i=1:100;
    f(i,1)=matrix(i,1);
if matrix(i,2)>=225
    f(i,1)=f(i,1)+3;
end
    if matrix(i,3)>=140;
       f(i,1)=f(i,1)+5;
     end
 end
for i=1:100
f(i,2)=i;
end
end
pf=sortrows(f)
```

Curriculum Vita

Sowmya Parimi was born in Guntur, India. Sowmya Parimi is the only child of Mr. Sambasiva Rao. Parimi and Lakshmi. Parimi. She graduated in Production Engineering from Acharya Nagarjuna University, India. And after graduation she applied for Masters in University of Texas at El Paso for spring 2007. During her Masters program she had a wonderful chance to work with Dr. Espiritu and Dr. Taboada and also had a chance to do her thesis under Dr. Espiritu in the area of Power System Optimization. In the Masters, she had a chance to present her work at INFORMS and IIE conference. And now she is a Master Graduate student from University of Texas at El Paso (Fall 2009) in Industrial Engineering.