NETWORK ROUTING PROTOCOL USING GENETIC ALGORITHM DIJKSTRA ALGORITHM

By

NASRIN ESHRAGHI IVARI

GS31834

July 2013

DEDICATION

This thesis is dedicated to my darling parents for their endless support, encouragement, patience and their great patience.

NETWORK ROUTING PROTOCOL USING GENETIC ALGORITHM AND COMPARE IT WITH DIJKSTRA ALGORITHM

By

NASRIN ESHRAGHI IVARI

July 2013

Chairman: Kweh Yeah Lun, PhD

Faculty: Computer Science and Information Technology

ABSTRACT

The performance and reliability of the Internet depend in large part on the operation of the underlying routing protocols. Today's IP routing protocols compute paths based on the network topology and configuration parameters, without regard to the current traffic load on the routers and links. This research, aims to develop a genetic algorithm to solve a network routing protocol problem. The algorithm has to find the shortest path between the source and destination nodes. Genetic algorithms are stochastic and biology inspired search techniques mostly used in scientific related work to find optimal solutions. They are usually use too much of resources in terms of CPU time and memory size. The proposed genetic algorithm will optimized the routing tables and enhancing the performance of the routers.

The routing problem is solved using search graph techniques to find the shortest path. Dijkstra's algorithm is one of popular techniques to solve this problem. The developed genetic algorithm is compared with Dijkstra's algorithm to solve routing problem. The results affirmed the potential of the proposed genetic algorithm. The obtained performance is similar as Dijkstra's algorithm.

NETWORK ROUTING PROTOKOL MENGGUNAKAN ALGORITMA GENETIK DAN BANDINGKAN DENGAN ALGORITMA DIJKSTRA

Oleh

NASRIN ESHRAGHI IVARI

Julai 2013

Pengerusi: Kweh Yeah Lun, PhD

Fakulti: Sains Komputer dan Teknologi Maklumat

ABSTRAK

Prestasi dan keupayaan sebuah internet adalah bergantung besar kepada operasi protokol routing yang asas. Protokol Internet (IP) pada hari ini mengenal pasti laluan bedasarkan topologi rangkaian and konfigurasi yang dilakukan ke atas beberapa parameter, tanpa mengambil kira beban trafik pada router-router and laluan-laluan. Thesis master ini adalah bertujuan untuk membina sebuah algorithm genetik untuk menyelesaikan masalah protokol routing bagi sebuah rangkaian. Algorithm-algorithm genetik adalah stokastik dan teknik carian berinspirasikan biologi yang kebanyakannya digunakan dalam kerja berkaitan dengan sains untuk mencari penyelesaian optimum. Ia biasanya menggunakan sumber yang banyak dari segi masa CPU dan saiz memori. Kami akan mencadangkan sebuah algorithm genetik yang terperinci bertujuan untuk mengoptimumkan jadual routing dan untuk meningkatkan prestasi router.

Masalah routing ini diselesaikan menggunakan carian melalui teknik graf untuk mengenal pasti laluan terpendek. Algorithm Djikstra adalah salah satu teknik yang popular untuk menyelasaikan masalah ini. Algorithm yang telah dibina dibandingkan dengan algorithm Djikstra untuk menyelesaikan masalah routing. Keputusan membuktikan kebolehan algorithm genetik yang dicadangkan. Prestasi yang dihasilkan adalah seumpama dengan algorithm Djisktra.

ACKNOWLEDGEMENTS

First and foremost, I would like to express my deepest praise to Allah who gave me the strength, faith, confidence, and patience to successfully complete this thesis.

Next, my deepest gratitude and admiration are to Dr. Kweh Yeah Lun, my project supervisor. Without his tremendous help and guidance this project would never have been completed.

Finally, and most importantly, I would like to express my deepest gratitude and admiration to my parents, Mohammad Eshraghi and Zahra Fatemi, for their support and encouragement, which enabled me to complete this project.

____

NASRIN ESHRAGHI IVARI

Date:

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Introduction

Routing is a process of transferring packets from source node to destination node with minimum cost. Hence routing algorithm has to acquire, organize and distribute information about network states. It should generate feasible routes between nodes and send traffic along the selected path and also achieve high performance. Routing in conjunction with congestion control and admission control defines the performance of the network according to another [1].

Genetic algorithm is an optimization and evolutionary algorithm that solves optimization problems. This project is aimed to solve the optimization problem in IP networks using genetic algorithm to find the shortest path between source and destination nodes in the network topology. The solution starts with finding alternative paths to alternate the overloaded path using genetic algorithm. Variable-length chromosomes (strings) and their genes have been used for encoding the problem. The crossover operation exchanges partial chromosomes (partial routes) at appositionally independent crossing sites and the mutation operation maintains the genetic diversity of the population. The proposed algorithm can cure all the infeasible chromosomes with a simple repair function. Crossover and mutation together provide a search capability that results in improved quality of solution and enhanced rate of convergence.

For a given source vertex (node) in the graph, the Dijkstra algorithm finds the path with the lowest cost (i.e. the shortest path) between that vertex and every other vertex. It can also be used for finding costs of shortest paths from a single vertex to a single destination vertex by stopping the algorithm once the shortest path to the destination vertex has been determined. The shortest path first is widely used in network routing protocols, most notably OSPF (Open Shortest Path First). OSPF is a dynamic routing protocol. It is a link state routing protocol and is part of the interior gateway protocols group. OSPF keeps track of the complete network topology and all the nodes and connections within that network.

An OSPF routing protocol is a very important protocol to consider when setting up routing instructions on the network. As OSPF gives the routers the ability to learn the most optimal (shortest) paths it can definitely speed up data transmission from source to destination. Dijkstra's algorithm is often described as a greedy algorithm. The Encyclopedia of Operations Research and Management Science describes it as a "node labeling greedy algorithm" and a greedy algorithm is described as "a heuristic algorithm that at every step selects the best choice available at the step without regard to future consequence" [5].

Cost factors may be the distance of a router (Round-trip-delay), network Throughput of a link or link availability and reliability expressed as simple unit less numbers. Routing should generate feasible routes between nodes and send traffic along the selected path and also achieve high performance [6].

1.2 Problem Statement

if routing becomes overloaded or congestion happened on a link in the path, The packets will not arrive to the desired destination or may be delayed, a lot of queuing and processing on the routers and less service quality.

Traffic in the network cause decreasing the available bandwidth and speed which lead to congestion problem in network and this congestion leads to decreasing total network throughput. The main problem of transferring packets to achieve destination is to find the path with the minimum total length ,as I describe in this project as a cost, between two given nodes

1.3 Objective

This research objective is to achieve high performance in large networks affected by cost factors which includes router network throughput of a link or link availability and reliability. Also in thisproject utilizes Genetic Algorithm To find the path with the minimum total cost between a source node and destination node (shortest path) for improving network performance and find the optimal destination for routing using genetic algorithm.

Also compare the genetic algorithm with Dijkstra to show that result is the same as this famous Dijstra algorithm.

1.4 Scope

We have some kind of networks in World Wide Web, like LAN, WAN, MAN and etc. We can describe routing between wire, wireless, cellular and other kind of networks. My research is on wire networks.

In this area, new routing protocol proposed based on artificial intelligence algorithms for finding the shortest path between nodes in the network.

1.5 Organization of the Thesis

This thesis document is structured as follows:

Chapter 2, include basic concept that should defined for my project such as definition of routing algorithm and protocols, Dijkstra algorithm and genetic algorithm.

Chapter 3 provides a literature review of research in this area, tying together themes from the exploration and analysis.

Chapter 4, in my methodology, proposed a generic algorithm constrained optimized framework for cooperating routing protocols in networks and also Dijkstra algorithm.

Chapter 5, implemented my work with C#.net in visual studio environment and MATLAB and then compare it with Dijkstra algorithm.

Chapter 6, as a conclusion chapter, compared my results and discus about benefits and disadvantages of each algorithm.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

This chapter provides the necessary background for this thesis by describing the basic Internet routing model and its current routing algorithms. Also this chapter describe about GA and Dijkstra algorithm and at the end, I review some papers in this area.

2.2 Routing Algorithm

Routing algorithms select routes for path setup. They use link–state information and candidate routes. In this subsection, we explain three typical routing algorithms, and their advantages and disadvantages [7].

2.2.1 Fixed routing

The fixed routing algorithm uses a predetermined route for all node pairs each time a connection is established [8]. On the arrival of a light path request, the fixed routing algorithm chooses a predetermined route. Route selection does not depend on the actual dynamic link–state change.

A typical fixed routing algorithm is minimum hop routing (for instance, Dijkstra's shortest path algorithm). When light path establishments are congested on certain links, this algorithm cannot reroute the link, and the light path request may be blocked.

2.2.2 Adaptive Routing

With adaptive routing, sender nodes dynamically select a route to the receiver node when a light-path setup request arrives [8], [9].Route selection depends on network–state information, i.e., both the connectivity of each adjacent node and the wavelength utilization of each link. The advantage with this algorithm is that the sender node evaluates all available routes in the network, which is expected to result in less blocking. The disadvantage with this algorithm is that the discrepancy between the current status of wavelength utilization and exchanged link state information greatly affects blocking. With this algorithm, it is necessary to efficiently select the proper route that current link state information can use.

2.2.3 Alternative Routing

There are two types of alternate routing; fixed–alternate and adaptive–alternate routing. With fixed–alternate routing, each node has a route–list for a set of predetermined routes. This list contains an ordered list of routes to each destination node, and the routes are not changed dynamically. When a light path setup request arrives at the sender node, the node selects a route (primary route) according to its order on the list. If the light path cannot be established along the primary route, the sender node then selects the next route. This continues until all the routes in the list have been examined. With adaptive routing, the sender node must calculate another route to avoid failed links.

With adaptive–alternate routing, each node also has a route list, but the order of route changes dynamically according to wavelength utilization in the network.

2.3 Network Routing Protocols

A routing protocol specifies how routers communicate with each other, disseminating information that enables them to select routes between any two nodes on a computer network. Routing algorithms determine the specific choice of route. Each router has a priori knowledge only of networks attached to it directly. A routing protocol shares this information first among immediate neighbors, and then throughout the network. This way, routers gain knowledge of the topology of the network.

Although there are many types of routing protocols, major classes are in widespread use on IP networks:

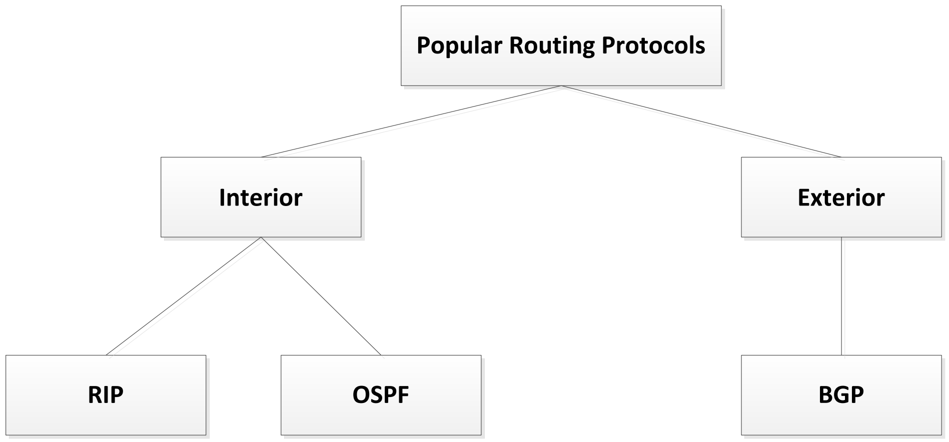


Figure 2.1 Classes of Routing protocol

Packet routing on the Internet is divided into two main groups: interior routing and exterior routing. An interior gateway protocol (IGP) is a routing protocol that is used to exchange routing information within an autonomous system (AS). The interior gateway protocols can be divided into two categories: distance-vector routing protocol and link-state routing protocol. In contrast, an exterior gateway protocol is used to determine network reachability between autonomous systems and makes use of IGPs to resolve routes within an AS.

2.3.1 Link State Routing Protocol

In link-state routing protocols, each router possesses information about the complete network topology. Each router then independently calculates the best next hop from it for every possible destination in the network using local information of the topology. The collection of best-next-hops forms the routing table.

This contrasts with distance-vector routing protocols, which work by having each node share its routing table with its neighbors. In a link-state protocol, the only information passed between the nodes is information used to construct the connectivity maps. Each node independently runs an algorithm over the map to determine the shortest path from itself to every other node in the network; generally some variant of Dijkstra's algorithm is used. This is based around a link cost across each path which includes available bandwidth among other things.

Examples of link-state routing protocols:

- Open Shortest Path First (OSPF)
- Intermediate system to intermediate system (IS-IS)

2.3.1.1 Open Shortest Path First Protocol

Open Shortest Path First (OSPF) is a link-state routing protocol for Internet Protocol (IP) networks. It uses a link state routing algorithm and falls into the group of interior routing protocols, operating within a single autonomous system (AS). OSPF is perhaps the most widely used interior gateway protocol (IGP) in large enterprise networks. The most widely used exterior gateway protocol is the Border Gateway Protocol (BGP), the principal routing protocol between autonomous systems on the Internet. OSPF detects changes in the topology, such as link failures, and converges on a new loop-free routing structure within seconds. It computes the shortest path tree for each route using a method based on Dijkstra's algorithm, a shortest path first algorithm.

2.4 Concept in Genetic algorithm

Charles Darwin stated the theory of natural evolution in the origin of species. Over several generations, biological organisms develop based on the principle of natural selection "survival of the fittest" to reach certain remarkable tasks. Thus, it works so well in nature, as a result it should be interesting to simulate natural evolution and to develop a method, which solves real, and search optimization problems [19].

The genetic algorithm (GA) is an optimization and search technique based on the principles of genetics and natural selection. A GA allows a population composed of many individuals to evolve under specified selection rules to a state that maximizes the "fitness". The method was developed by John Holland [17] over the course of the 1960s and 1970s and finally popularized by one of his students, David Goldberg, who was able to solve a difficult problem involving the control of gas-pipeline transmission for his dissertation [15]. Holland was the first to try to develop a theoretical basis for GAs through his schema theorem. The work of De Jong (1975) showed the usefulness of the GA for function optimization and made the first concerted effort to find optimized GA parameters. Goldberg has probably contributed the most fuel to the GA fire with his successful applications and excellent book (1989). Since then, many versions of evolutionary programming have been tried with varying degrees of success. Some of the advantages of a GA include that it

- Optimizes with continuous or discrete variables,
- Deals with a large number of variables,
- Is well suited for parallel computers,
- Optimizes variables with extremely complex cost surfaces (they can jump out of a local minimum),
- Provides a list of optimum variables, not just a single solution,
- May encode the variables so that the optimization is done with the encoded variables.

These advantages are fascinating and produce stunning results when traditional optimization approaches fail miserably.

The GA begins, like any other optimization algorithm, by defining the optimization variables, the cost function, and the cost. It ends like other optimization algorithms too, by testing for convergence. In between, however, this algorithm is quite different. A path through the components of the GA is shown as a flowchart in Figure 2.4. Each block in this “big picture” overview is discussed.

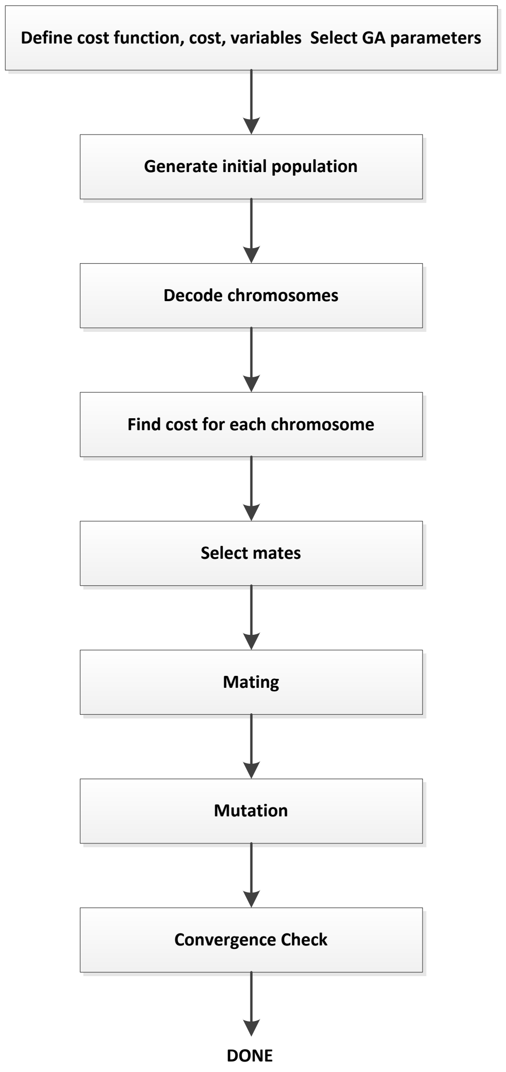

Define cost function, cost, variables Select GA parameters
Generate initial population
Decode chromosomes
Find cost for each chromosome
Select mates
Mating
Mutation
Convergence Check
DONE

Figure 2.2 Genetic process

The flowchart showing the process of GA. Before implementing GAs it is important to understand about each level:

2.4.1 Initialization

At first many individual solutions are randomly generated to form an initial population. The population size depends on the nature of the problem, but typically contains several hundreds or thousands of possible solutions. Traditionally, the population is generated randomly, allowing the entire range of possible solutions (the search space). Occasionally, the solutions may be "seeded" in areas where optimal solutions are likely to be found.

2.4.2 Selection

During each successive generation, a proportion of the existing population is selected to kind a new generation. Individual solutions are selected through a fitness-based process, where fitter solutions (as measured by a fitness function) are typically more likely to be selected. Certain selection methods rate the fitness of each solution and preferentially select the best solutions. Other methods rate only a random sample of the population, as the former process may be very time-consuming.

The fitness function is defined over the genetic representation and measures the quality of the represented solution. The fitness function is always problem dependent. A representation of a solution might be an array of bits, where each bit represents a different object, and the value of the bit (0 or 1) represents whether or not the object is in the knapsack. Not every such representation is valid, as the size of objects may exceed the capacity of the knapsack.

2.4.3 Genetic operators

The next step is to generate a second generation population of solutions from those selected through genetic operators: crossover (also called recombination), and/or mutation.

For each new solution to be produced, a pair of "parent" solutions is selected for breeding from the pool selected previously. By producing a "child" solution using the above methods of crossover and mutation, a new solution is created which typically shares many of the characteristics of its "parents". New parents are selected for each new child, and the process continues until a new population of solutions of appropriate size is generated. Although reproduction methods that are based on the use of two parents are more "biology inspired", some research (Eiben al,1994) suggests that more than two "parents" generate higher quality chromosomes.

These processes ultimately result in the next generation population of chromosomes that is different from the initial generation. Generally the average fitness will have increased by this procedure for the population, since only the best organisms from the first generation are selected for breeding, along with a small proportion of less fit solutions. These less fit solutions ensure genetic diversity within the genetic pool of the parents and therefore ensure the genetic diversity of the subsequent generation of children.

Although crossover and mutation are known as the main genetic operators- that I introduce as follow, it is possible to use other operators such as regrouping, colonization-extinction, or migration in genetic algorithms [21].

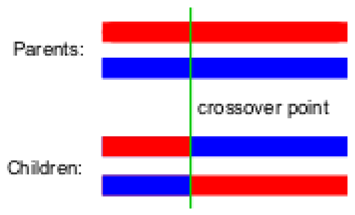
2.4.3.1 Crossover

Crossover is a genetic operator used to vary the programming of a chromosome or chromosomes from one generation to the next. It is analogous to reproduction and biological crossover, upon which genetic algorithms are based. Cross over is a process of taking more than one parent solutions and producing a child solution from them. There are methods for selection of the chromosomes. Those are also given below.

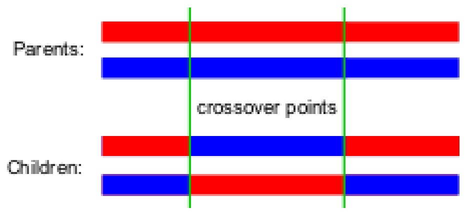
- Roulette wheel selection
- Boltzmann selection
- Tournament selection
- Rank selection
- Steady state selection
- Truncation selection
- Local selection

Many crossover techniques exist for organisms which use different data structures to store themselves.

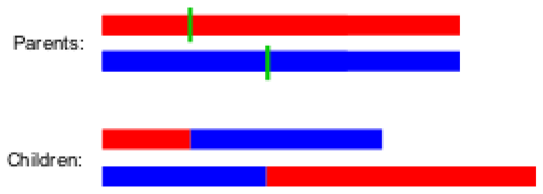
- **One-point crossover**



- **Two-point crossover**



- **Cut and split**



2.4.3.2 Mutation

The purpose of mutation in GAs is preserving and introducing diversity. Mutation should allow the algorithm to avoid local minima by preventing the population of chromosomes from becoming too similar to each other, thus slowing or even stopping evolution. For different genome types, different mutation types are suitable:

- **Bit string mutation**

The mutation of bit strings ensue through bit flips at random positions.

Example:

```
1    0    1    0    0    1    0
↓
1    0    1    0    1    1    0
```

The probability of a mutation of a bit is where the length of the binary vector is.

- **Flip Bit**

This mutation operator takes the chosen genome and inverts the bits. (i.e. if the genome bit is 1,it is changed to 0 and vice versa).

- **Boundary**

This mutation operator replaces the genome with either lower or upper bound randomly. This can be used for integer and float genes.

- **Non-Uniform**

The probability that amount of mutation will go to 0 with the next generation is increased by using non-uniform mutation operator. It keeps the population from stagnating in the early stages of the evolution. It tunes solution in later stages of evolution. This mutation operator can only be used for integer and float genes.

- **Uniform**

This operator replaces the value of the chosen gene with a uniform random value selected between the user-specified upper and lower bounds for that gene. This mutation operator can only be used for integer and float genes.

2.4.4 Termination

This generational process is repeated until a termination condition has been reached. Common terminating conditions are:

- A solution is found that satisfies minimum criteria
- Fixed number of generations reached
- Allocated budget (computation time/money) reached
- The highest ranking solution's fitness is reaching or has reached a plateau such that successive iterations no longer produce better results
- Manual inspection

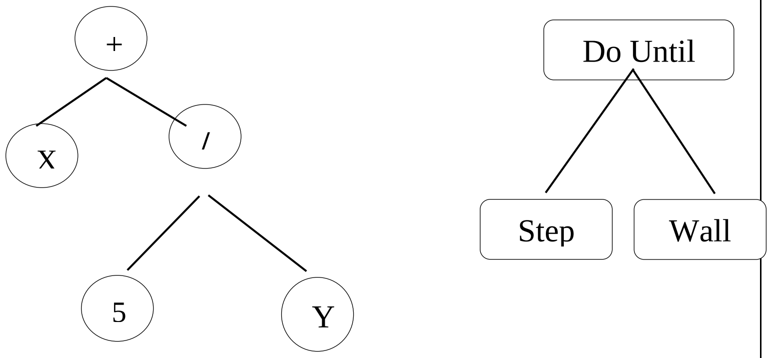
- Combinations of the above

GA has the ability to perform efficiently in the evolution of an optimum solution, but the major difficulty in its implementation is the encoding of a problem solution, as improper encoding may lead to a complete change in the shape of a problem (Obitko [1998], Negnevitsky [2002]).

The first step in the implementation of GA is encoding i.e. the representation of a problem solution/ chromosome. Encoding is mainly dependent upon the problem to be solved. Some of the different encoding techniques are given in Table 3.1 (Noor, 2009). These have been used with some success (Obitko, 1998).

2-1 Encoding technique

S/NO	ENCODING TECHNIQUE	EXAMPLE
1	Binary Encoding	Chromosome 1: 101100101100101011100101 Chromosome 2: 111111100000110000011111
2	Permutation Encoding	Chromosome 1: 7 6 5 4 8 3 9 0 1 Chromosome 2: 2 3 4 5 7 5 3 0 8
3	Value Encoding	Chromosome 1: 3.6556 1.4560 7.4578 3.7654 9.0987 Chromosome 2: ASDFGHJIKOYTGHSIDRUWSDJ

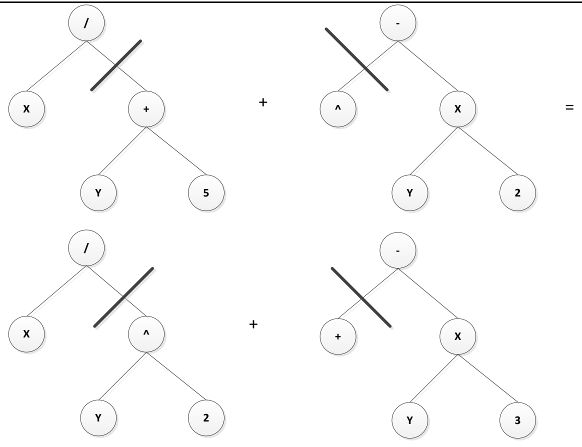
		Chromosome 3: (right), (back), (left), (back), (forward)
4	Tree Encoding	 (+X(/5Y)) (Do until Step Wall)

After encoding and the random generation of an initial population the next step in GA is the selection of chromosomes which would take part as parents in crossover. The main problem is how to carry out this selection. As per Darwin's evolution theory, the fittest chromosomes survive through generations and are most likely to take part in crossover and create offspring. "Some of the well-known techniques of selection are tournament selection, roulette wheel selection, steady state selection and rank selection" (Noor, 2007).

Crossover can be carried out in a number of ways. Some of them are shown in Table 3.2. The type of crossover to be chosen mainly depends upon the type of encoding being used and therefore it can be sometimes quite complicated. A suitable selection of the type of crossover for a particular problem can definitely improve the GA's performance.

2-2 Crossover Technique

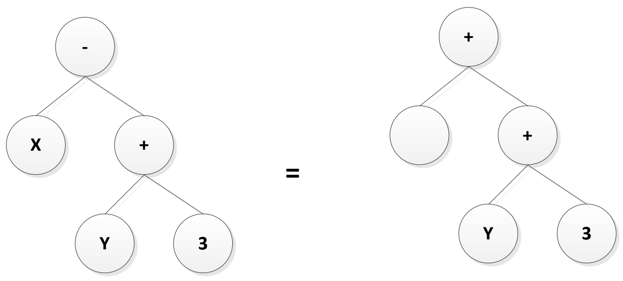
S/NO	Encoding Technique	Crossover Technique	Example
1	Binary	1. single cut point crossover	**1100/1011**+1101/1111= **1100**/1111 + 1101/**1011**
		2. Double cut point	**11/00/1011**+11/01/1111 = **11**01**1011** +**11**00**1111**
		3. Uniform crossover	**11/00/1011**+11/01/1111 = 11011011 +**11**0**0**1**1**1**1**
		4. Arithmetic crossover	0100110 + 0100101 = 1100110 + 0101111 (the first three digits of each parent are added together for child 1 and the remaining digits are added for the child 2)
2	Permutation encoding	Single Cut Point	**(1 2 3 4 5 6 /7 8 9) + (4 5 3 6 8 9/ 7 2 1) = (1 2 3 4 5 6** 8 9 7) +4 5 3 6 8 9 **7 1 2**
3	Real Value Encoding	Same as in Binary value	**(1.29 5.68/ 2.86 4.11 5.55)+**(3.23 4.45/ 6.13 5.67 2.98) = **(1.29 5.68** 6.13 5.67 2.98) +(3.23 4.45 **2**.86 4.11 5.55)**--single point**

4	Tree Encoding	Exchange	

Once the issue of crossover is resolved, the next step is mutation. “The main aim of carrying out mutation is to induce a certain level of diversity into population so that GA can be prevented from getting trapped into a local optimum” (Obitko, 1998). It has been already mentioned, previously, that during mutation a slight change is incorporated in the genetic structure of a chromosome. An offspring, resulted from crossover, is randomly changed by a mutation operator.

Like crossover the decision of how to perform mutation, also, depends upon the type of encoding being used. The different mutation techniques, for different types of encoding, are shown in Table 3.3.

2-3 Mutation Technique

S/NO	Encoding Technique	Mutation Technique	Example
1	Binary	Bit inversion	11001001 => 10001001
2	Permutation	Change in order	(1 2 3 4 5 6 8 9 7) => (1 8 3 4 5 6 2 9 7)
3	Real value	Addition of a small number	(1.29 2.86 5.68 4.11 5.55) => (1.29 2.86 5.68 4.22 5.55)
4	Tree	Change in operator	

“Crossover and mutation rates are the two basic parameters of GA” (Obitko [1998]). The crossover rate determines the number of times crossover of chromosomes will be carried out in one generation. The range for selection of crossover rate is from 0% to 100%. If the crossover rate is 0% it means that chromosomes in the next generation will be the exact copies of chromosomes in the current generation. On the other hand if it is 100%then every chromosome in the population of next generation will be the result of crossover between any two chromosomes of the current generation. Crossover is carried out in the hope that children, created during the process, would contain good parts of their parents and consequently perform better as compared to them. Each selection criteria is so designed that during selection some part of the population in the current generation do get selected in the next generation.

Similarly, mutation rate means how many genes in a population in one generation would get mutated. Here also the range could be from 0% to 100%. If the mutation rate is 0% then it means none of the genes would get selected. But, if it is 100% then it means all the genes in a population of a generation would get mutated. As indicated earlier, mutation is an operator that creates a certain level of diversity in a population and hence GA is prevented from getting trapped into local optimum. Therefore “selection of mutation rate is a delicate decision” (Noor [2007]). Too high a mutation rate would convert GA into a kind of random search and the characteristic of evolution is lost. Also, if the mutation rate is too low then there would be a tendency of GA converging on to a local optimum. In addition to two basic parameters i.e. crossover and mutation, there are some additional parameters of GA. One particularly important additional parameter of GA is the population size. Population size means total number of chromosomes in a population, in one generation. Population size is important because it provides GA the searching space in which search for the optimum solution is carried out. If there are too few chromosomes in a population then it means GA is given a smaller searching space and this would limit the GA's searching ability and there would be every likelihood of GA getting trapped on a local optimum. On the other hand, if there are too many chromosomes, then it means GA is provided with a larger searching space which would definitely slow it down by increasing its computational effort. Therefore, selecting a reasonable population size is again a delicate matter. According to Obitko it is not useful to use a very large population size because it does not solve the problem quickly as compared to a moderate sized population.

2.4.5 Advantages and Limitations of Genetic Algorithm

The advantages of genetic algorithm includes,

1. Parallelism
2. Solution space is wider
3. The fitness landscape is complex
4. Easy to discover global optimum
5. The problem has multi objective function
6. Only uses function evaluations.

7. Easily modified for different problems.

8. Handles noisy functions well.

9. Handles large, poorly understood search spaces easily

10. Good for multi-modal problems Returns a suite of solutions.

11. Very robust to difficulties in the evaluation of the objective function.

12. They are resistant to becoming trapped in local optima

13. They perform very well for large-scale optimization problems

14. Can be employed for a wide variety of optimization problems

The limitation of genetic algorithm includes,

1. The problem of identifying fitness function

2. Definition of representation for the problem

3. Premature convergence occurs

4. Cannot easily incorporate problem specific information

5. Not good at identifying local optima

6. No effective terminator.

7. Needs to be coupled with a local search technique.

2.5 Dijkstra algorithm

Dijkstra's algorithm, conceived by Dutch computer scientist Edsger Dijkstra in 1959 [2] is a graph search algorithm that solves the single-source shortest path problem for a graph with nonnegative edge path costs, producing a shortest path tree [3]. This algorithm is often used in routing. An equivalent algorithm is developed by Edward F [4]. Dijkstra’s algorithm is undoubtedly one of the most famous, celebrated and useful algorithms in Computer Science. The algorithm is an example of a node labeling greedy algorithm. A greedy algorithm is described as "... a heuristic algorithm that at every step selects the best choice available at that step without regard to future consequences” [10]. Which finds minimum length, cost, weight paths from a given start vertex to all possible destination vertices for graphs with positive edge weights. As we know, Dijkstra's algorithm visits or expands vertices in priority order, where the priority for our project is the weight. In this project we will focus on route finding problem by using Dijkstra algorithm that mean find the shortest path between two nodes in weighted and directed network.

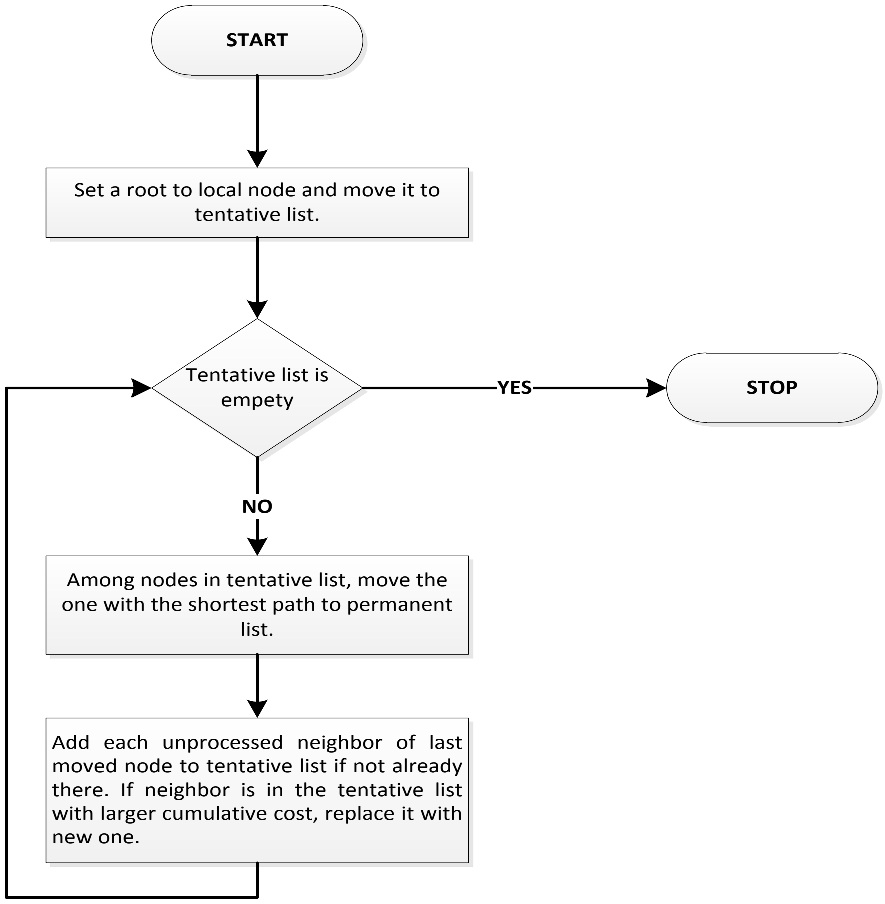

START
Set a root to local node and move it to tentative list.
Tentative list is empety
YES
STOP
NO
Among nodes in tentative list, move the one with the shortest path to permanent list.
Add each unprocessed neighbor of last moved node to tentative list if not already there. If neighbor is in the tentative list with larger cumulative cost, replace it with new one.

Figure 2.3 Process of Dijkstra algorithm

2.6 Reviewed papers

In [22], presents a genetic algorithmic approach to the shortest path (SP) routing problem. Variable-length chromosomes (strings) and their genes (parameters) have been used for encoding the problem. The crossover operation exchanges partial chromosomes (partial routes) at positionally independent crossing sites and the mutation operation maintains the genetic diversity of the population. The proposed algorithm can cure all the infeasible chromosomes with a simple repair function. Crossover and mutation together provide a search capability that results in improved quality of solution and enhanced rate of convergence. This paper also develops a population-sizing equation that facilitates a solution with desired quality. It is based on the gambler's ruin model; the equation has been further enhanced and generalized, however. The equation relates the size of the population, the quality of solution, the cardinality of the alphabet, and other parameters of the proposed algorithm. Computer simulations show that the pro-posed algorithm exhibits a much better quality of solution (route optimality) and a much higher rate of convergence than other algorithms. The results are relatively independent of problem types (network sizes and topologies) for almost all source–destination pairs. Furthermore, simulation studies emphasize the usefulness of the population-sizing equation. The equation scales to larger net-works. It is felt that it can be used for determining an adequate population size (for a desired quality of solution) in the SP routing problem.

This paper [23] demonstrates that a dynamic routing control based on a Genetic Algorithm can provide flexible real-time management of the dynamic traffic changes in broadband networks. The problem with state-dependent method is that finding satisfactory solutions requires a huge amount of calculation power. For each traffic change, we have to find the optimal combination of paths between every pair of nodes, but it is impossible to get the optimal solution within a practical control cycle because there are almost infinitely many possible paths in an actual network. Practical and effective methods for finding quasi-optimal solutions have therefore been investigated. This paper proposes a new dynamic routing method based on a Genetic Algorithm. A new string structure and genetic operations suitable for network problems is proposed, and an optimization method that uses past solutions as the initial data for new searches is also proposed. These techniques dramatically improve the efficiency and convergence speed of the Genetic Algorithm. Finally we demonstrate the capability of the proposed Genetic Algorithm by presenting the results obtained from computer simulation of a path arrangement problem and a dynamic routing problem.

In a dynamic routing problem, the network model is constant and only the traffic demands are changing. All possible routes for each pair of nodes can therefore be thoroughly searched in advance and arranged in a "route table" in order of length. In the proposed Genetic Algorithm, each route for a path is identified by a "route No.", which is its row number in the route table. A gene is a string of route No's for all paths. This gene is called a "configuration string," and its length is M (M-1)/2.

In [24], proposed the idea of using Bionomic Algorithm (BA) for the shortest path finding problem. SPF is a very important issue associated with many other tasks like network routing and graph/map analysis etc. Different techniques have been used for solving the SPF problem, some of them being Dijkstra Algorithm, Floyd Algorithm, Genetic Algorithm, Tabu Search etc. The choice of algorithm for any specific problem depends upon the complex tradeoffs between complexities, performance etc. the aim of this work is to use type 3 heuristics algorithm for SPF. Simulations are carried out for BA algorithm and Dijkstra algorithm.

Dijkstra Algorithm is widely used because of its simplicity. It is based on bellman optimization theory. It finds the shortest path from the source node to all other nodes in the network, for which it requires global topological knowledge of the network (list of all nodes, interconnections, and costs). It computes the shortest path as a function of distance, cost, traffic etc. One major drawback of Dijkstra algorithm is that it consumes more processing time (computation cost) if network is large.

Bionomic algorithm was proposed by N. Christofides in 1994 . BA follows the GA framework, but replaces randomness of GA operators with normative procedure. It also accepts multiple parents ‘combinations and variable cardinality solution sets. Just like GA or other Evolution Strategies (ES), it updates a whole population of solutions at each iteration, where updating means defining a child solution from a set of parents (crossover). GA is limited to two parents, but BA can use multiple parents; an approach which is shared with scattered search. BA formally requires maturation (local optimization of the solution). BA has five major steps; 1) initialization, 2) maturation, 3) propagation, definition of parents sets, 4) propagation, definition of child solutions and 5) termination.

In [25], Genetic Algorithms are applied to select the optimal routes. The proposed QoSRGA protocol used the suggested metric RQoSR to evaluate the value of routes and applied GA operators as an intelligent tool to select a high quality and fast decision routing solution. Also RST is applied to reduce the metrics that are reported by Routing Protocols and give the best result by genetic algorithm. Internet Service Providers (ISPs) try to meet the increasing traffic media and improve utilities of the existing resources. Packets are sent along network paths from source to destination subject to any routing protocols. Open Shortest Path First is the most commonly used Intra-domain Routing Protocol.

They sort two routing approaches: Source routing and hop-by-hop routing. In the hop-by-hop routing, the next hop is decided spatially. While in the front-end routing, the routes are defined by the source. The source routing suits the static network, while the hop routing suits the dynamic networks. QoS requirements meet the complexity of the routing metrics, which may consider :delay jitters, loss ratio, throughput, and admissibility.

Comparing the proposed algorithm with the algorithms of the standard routing protocols like OSPF, EIGRP for the case study. The instance proved that the proposed QoSRGA algorithm is good at selecting the best routes with a higher RQoSR. The proposed routing algorithm is supposed to be used by the network management protocol to decide the optimal route based on the best real time QoS. The current QoS is based on the online characteristics of NE's.

In [26], Genetic algorithm is used for routing in packet switched data networks. They explore solution space in multiple directions at once. Due to huge demand of Internet by various business communities and individuals, ISPs are trying to meet the increasing traffic demand with improved utilization of existing resources and application of new technologies. Routing of data packets can affect network resource utilization. In intra-domain Internet routing protocol Open Shortest Path First (OSPF) is the most commonly used protocol.

In this proposed algorithm, a module is used to generate all possible paths from a given node to all other nodes in the network. Initially 'n' random paths are considered (chromosome). This "n‟ defines the population size. These chromosomes act as population of first generation. This module deals with finding the optimal path using genetic algorithm. The input to this module is the set of paths generated. Each path is called as chromosome. As the source node receives "m ‟ chromosomes.

This paper deals with PMX crossover. In Partially Matched Crossover, two strings are aligned, and two crossover points are selected uniformly at random along the length of the strings. The two crossover points give a matching selection, which is used to affect across through position-by-position exchange operations.

GA has ability to solve problems with no previous knowledge. The performance of GA is based on efficient representation, evaluation of fitness function, population size, crossover rate, mutation probability and the selection method. The proposed algorithm works on initial population created by some other module, access fitness, generate new population using genetic operators and converges after meeting to specified termination condition. This work can be improved by using some intelligent approach for populating routing table and using better crossover, mutation probabilities and enhancing it to support for load balancing.

In [27], he developed a genetic algorithm that finds a shortest path in a limited time. This algorithm is meant to be used in OSPF routing, which is the most commonly used intra-domain Internet routing protocol (IRP).

The results show that GA gets close to optimum very quickly. When using this GA algorithm besides other search algorithms in the USF, such as, multi-start hill-climbing, simulated annealing, Controlled Random Search and RRS (Recursive Random Search), he think that he, can start searching the space with GA first, and then after GA gets close to optimum, then can switch to other search techniques.

In [28], The performance and reliability of the Internet depend in large part on the operation of the underlying routing protocols. This paper discusses routing optimization using Genetic Algorithm Then studied and analyzes the problems of routing optimization in large networks. They proposed a detailed genetic algorithm in order to optimize routing tables and to enhance the performance of the routers.

Before a genetic algorithm can be put to work on any problem, a method is needed to encode potential solutions to that problem in a form that a computer can process. One common approach is to encode solutions as binary strings: sequences of 1's and 0's, where the digit at each position represents the value of some aspect of the solution. Another, similar approach is to encode solutions as arrays of integers or decimal numbers, with each position again representing some particular aspect of the solution. A third approach is to represent individuals in a GA as strings of letters, where each letter again stands for a specific aspect of the solution.

They started by creating a genetic algorithm class (called GA) that contains chromosome objects as instance variables and implements the basic methods for doing crossover, reproduction, mutations, and iterating through the simulated evolution. They algorithm, the first step is to generate a network (graph) with nodes and a cost assigned randomly to the link that connects two sequenced nodes in the path. The location of each network node is given.

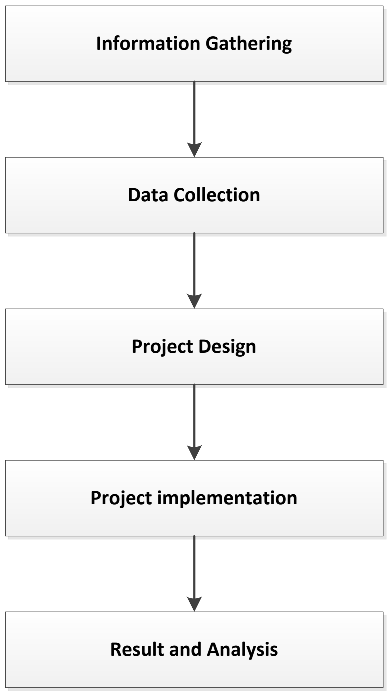
In the next step, choose the source and destination node to generate all the paths between the desired nodes. Their task is to optimize a path in the routing table. In this step we will choose the path that might be congested. Based on the network infrastructure all paths from the source and destination node will be taken and this is the first step of the algorithm. These alternative paths will be listed in a list with a cost for each path.

CHAPTER 3

METHODOLOGY

3.1 Introduction

This chapter explained about the GA that implemented in this project,first decode chromosomes to binary codes and then compared these chromosomes to each other. As to produce an effective research, there are several methodologies which have been used. These methods were undertaken in order to gain necessary information for the research. The overall framework for this project is described in the following flow chart.



3.1.1 Information Gathering

To develop our project we need find out all the data that relevant this topic and to collect all the data we need to know all information is needed. This project will only focus on secondary data sources to get information .Most data is obtained from published materials such as textbooks ,magazines, journals and the Internet .Even though this sources is easily available we need to choose the most relevant data for own project.

3.1.2 Data Collection

Data are collected through the experiments done using Dijkstra's algorithms. The pattern of the algorithm is measured by the distance and difference performance by the algorithm is the result of using different parameters and these parameters are number of nodes.

3.1.3 Project Design

Project design is described about designed that are used to implement program during development phase. In this phase we will design the interface about the system like input and output produced based on requirement needed.

3.1.4 Project Implementation

For the project implementation, there are certain software and hardware needed. For this project, the software and hardware have already been identified in order to develop the project and to make this project more reliable.

3.1.5 Result and Analysis

Data analysis is a process analyzes the collected data. We need to do data analysis because we want to time tune the system according to our results. In this project we use the Dijkstra's Algorithm technique to analyze the data based on given cities.

3.2 Feasibility of the system

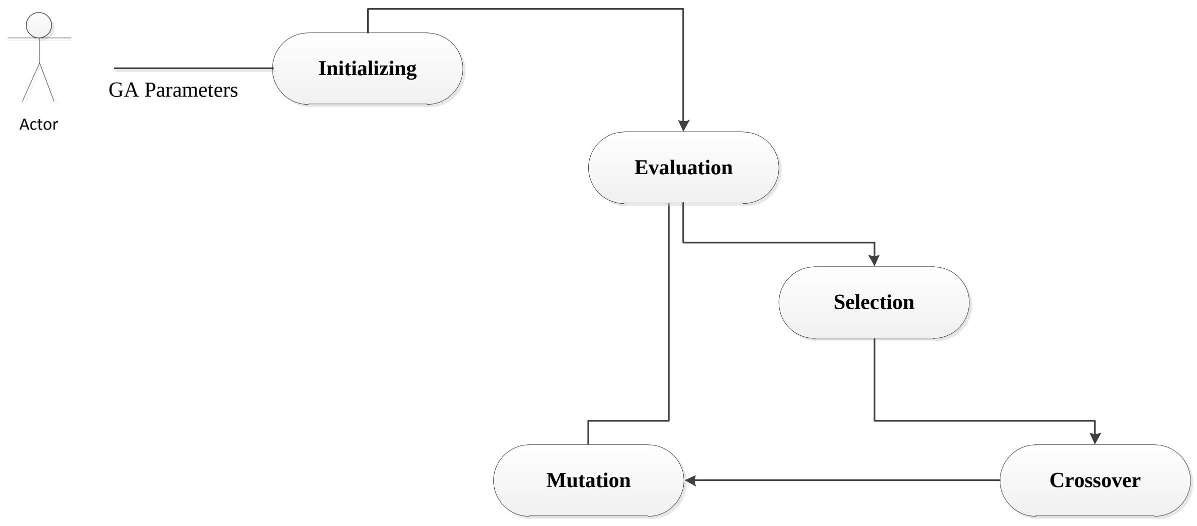


Figure 3.1 Use case Diagram for GA

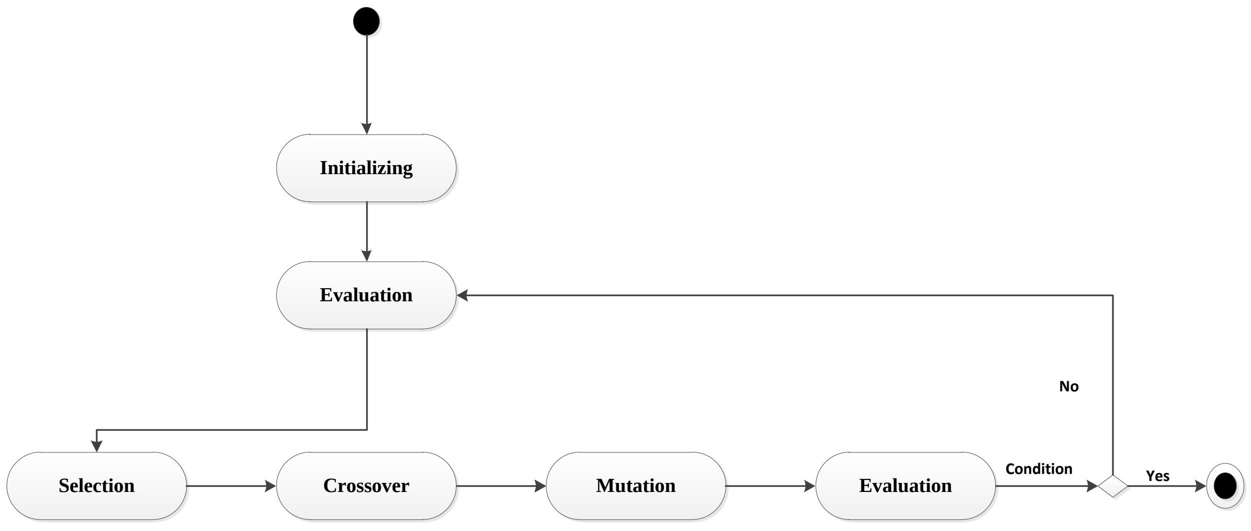


Figure 3.2 Activity Diagram for GA

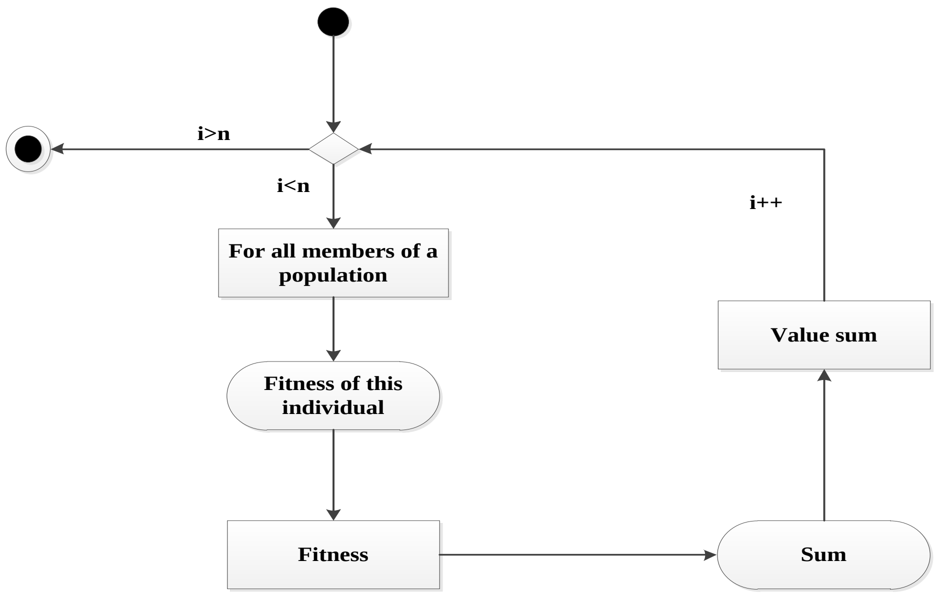


Figure 3.3 Activity Diagram- genetic selection for GA

3.3 Proposed method for Genetic algorithm

As showed the flow chart of genetic algorithm in figure 4.4, I implemented this algorithm as a follow. The network under consideration is represented as a connected graph with N nodes. The metric of optimization is the cost of path between the nodes. The total cost is the sum of the cost of the individual hops. The goal is to find the path with minimum total cost between source node and destination node. This project presents a simple and effective genetic algorithm GA to find the shortest path.

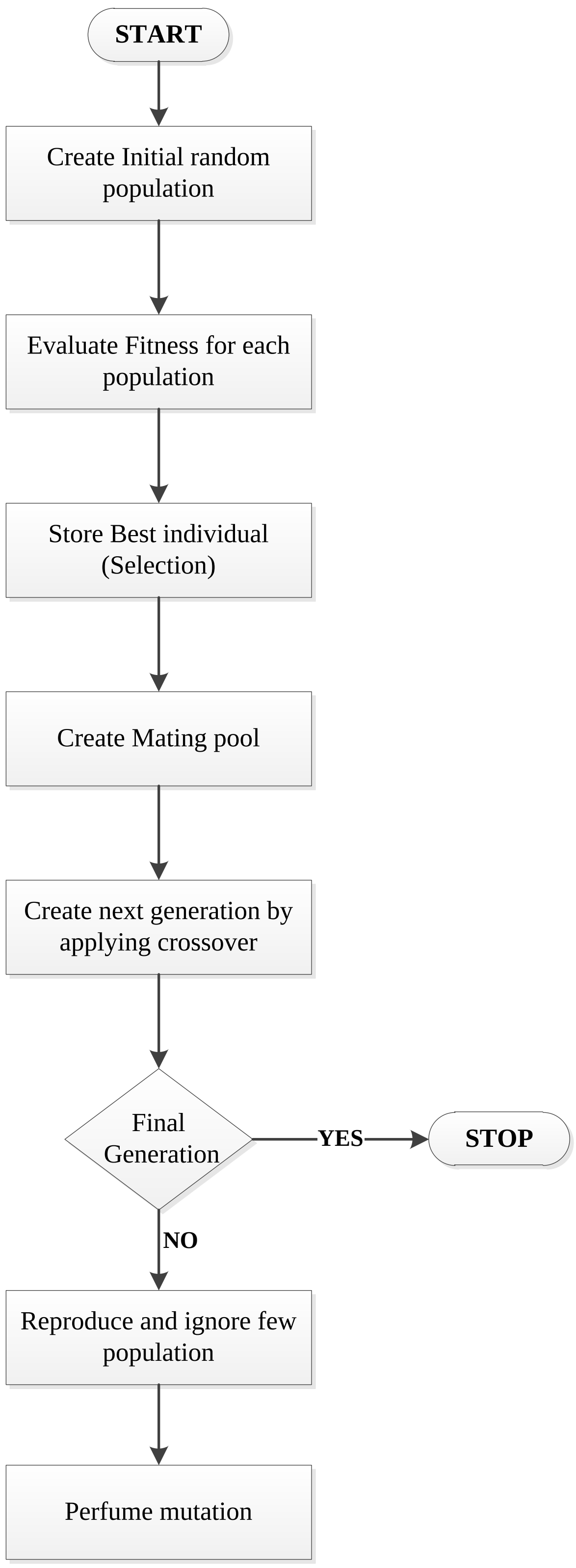

START
Create Initial random population
Evaluate Fitness for each population
Store Best individual (Selection)
Create Mating pool
Create next generation by applying crossover
Final Generation
YES
STOP
NO
Reproduce and ignore few population
Perfume mutation

Figure 3.4 Genetic algorithm overview

In the proposed algorithm, any path from the source node to the destination node is a feasible solution. The optimal solution is the shortest one. At the beginning a random population of strings is generated which represents admissible (feasible) or un-admissible (unfeasible) solutions. Un-admissible solutions are strings that cannot reach the destination. That means, the string solution would lead to a path without link between nodes. Admissible solutions are strings that can reach the target. The un-admissible solution has lowest fitness (zero fitness). Consider the network topology in Figure 4.5 with 10 nodes.

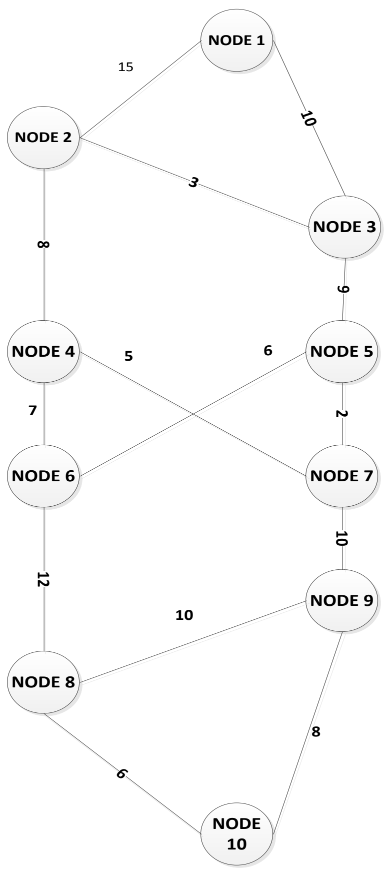


Figure 3.5 Network Topology

Each node has a number in Figure 4.6 and the nodes are used to encode a path as a string expressed by the order on numbers. For example, 1-3-5-6-8-10 is a feasible path with source node 1 and destination node10; it might be optimal or not. Define via nodes as all nodes except the source and destination (2, 3, … 9). The chromosome is represented by a string bits (i.e. natural representation not binary). Hence special crossover and mutation operators must be adopted. During the crossover, a string that has an efficient fitness is randomly selected as a parent. If the second parent contains the common number in the first one; both strings exchange the part of their strings following the common number. If not another string is selected as the second parent and the same procedure is followed. For example using the network in Figure 2:

Parent1: 1-3-5-7-9-10

Parent2: 1-2-4-7-9-8-10

At via node 7; the underlined parts of each string are exchanged, yielding: the two children are:

Child1: 1-3-5-7-9-8-10

Child2: 1-2-4-7-9-10

After crossover has been achieved; children are checked to determine whether each string has repeated number. If so, the part of string between the repeated numbers is cut off. Some correction then required because it might be the case that the child is not admissible solution. This approach makes the algorithm more complex. The important question is how to make the binary coding so possible and easy to be used in a fixed length chromosome definition. One of the more challenging aspects in our proposed method is encoding each string in a binary code with fixed length. The path is encoded using binary numbers where each gene (node) in a chromosome is encoded by 4 bits binary as in the Table 4.1. The number of bits has to be sufficient to encode the network nodes.

The following subsections describe the important components of the proposed genetic algorithm.

Table 3-1 Binary coding of network nodes

Node	Binary Code	Linked nodes
1	0001	2,3
2	0010	3,4
3	0011	5
4	0100	6,7
5	0101	6,7
6	0110	8
7	0111	9
8	1000	10
9	1001	8,10
10	1010	Destination

A. Chromosomes and Initialization A chromosome corresponds to possible solution of the optimization problem. Thus each chromosome represents a path which consists of a set of nodes to complete the feasible solution, as the sequence of nodes with the source node followed by intermediate nodes (via nodes), and the last node indicating the destination, which is the goal. The default maximum chromosome length is equal to the number of nodes times the gene length (4-bit binary code/gene). The network has 10 points where each point is coded in 4 binary bits. That means; the chromosome length is equal to 10X4=40 bits. The chromosome structure is given in Figure 4.6. The first gene represents node 1 (source node) which is coded in 4 binary bits, next gene is node 3, next gene is node 5 and so on. Successive genes in the chromosome are coded similarly. The initial population of chromosomes can be randomly generated such that each chromosome has a random genes (via nodes), while the start and goal nodes are fixed in the population.

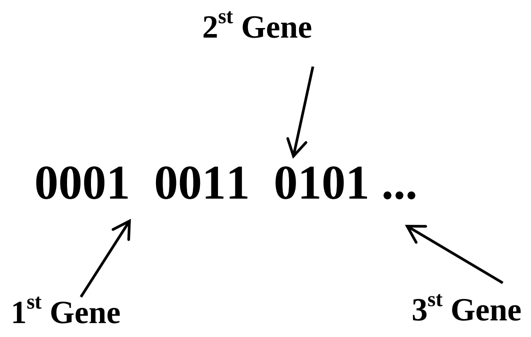


Figure 3.6 Chromosome length

B. Evaluation

The choice of a fitness function is usually very specific to the problem under condition. The evaluation function of a chromosome measures the objective cost function. The cost of a path indicated by the chromosome is used to calculate its fitness. Since the fitness should increase as the cost decreases. Thus, the fitness function (F) of a path is evaluated as defined in equation 1:

$$\mathbf{F} = \begin{cases} \dfrac{\mathbf{1}}{\sum_{i=1}^{N-1} C_i\,(g_i\,,\, g_{i+1})} & ; \quad \textbf{Feasible Path} \\[2em] \mathbf{0} & ; \quad \textbf{Infeasible Path} \end{cases}$$

Equation 1 Fitness Function

Where *Ci (gi, gi+1)* is the cost between gene *gi* and adjacent gene *gi+1* in the chromosome of *N* genes (Nodes). The cost between linked nodes is given in Figure 4.6. If the path is not feasible, its fitness is equal to zero. The proposed algorithm can trace the path points to detect if it is feasible or not using the information in table 4.1. The linked nodes are the admissible next nodes to the current one in the solution.

C. Operators

The algorithm uses the common two genetic operators: crossover and mutation. Crossover recombines two 'parent' paths to produce two 'children' new paths in the next generation. Two points crossover is used. Both parent paths are divided randomly into three parts respectively and recombined. The middle part of the first path between crossover bit positions and the middle part of the second path are exchanged to produce the new children. The crossover bit positions are selected randomly along the chromosome length between bit positions 5 and 36. These limits are chosen in order to keep the start and destination nodes without change during the crossover process. The mutation process is also applied to flip randomly a bit position in the chromosome (between bit position 5 and 36). The pseudo-code of the proposed algorithm is given by:

- **BEGIN**
- **Initialize** the start and destination points
- **Generate** randomly the initial population using via nodes in each chromosome
- **While NOT** (convergence condition) DO
- **Evaluate** the fitness for each chromosome in current population using equation1

- **Rank** the population using the fitness values

- **Eliminate** the lowest fitness chromosome

- **Duplicate** the highest fitness chromosome

- **Apply** randomly crossover process between current parents using the given probability, while keeping the start and end nodes without change in the population

- **Apply** the mutation process with the given probability Generate the new population

- **END** Output the best individual found

- **END**

Crossover and mutation rates are the two basic parameters of GA”. The crossover rate determines the number of times crossover of chromosomes will be carried out in one generation. The range for selection of crossover rate is from 0% to 100%. If the crossover rate is 0% it means that chromosomes in the next generation will be the exact copies of chromosomes in the current generation. On the other hand if it is 100%then every chromosome in the population of next generation will be the result of crossover between any two chromosomes of the current generation.

Similarly, mutation rate means how many genes in a population in one generation would get mutated. Here also the range could be from 0% to 100%. If the mutation rate is 0% then it means none of the genes would get selected. But, if it is 100% then it means all the genes in a population of a generation would get mutated.

3.4 Methodology of Dijkstra algorithm

The Dijkstra's algorithm calculates the shortest path between two points on a network using a graph made up of nodes and edges. It assigns to every node a cost value. Set it to zero four source node and infinity for all other nodes. The algorithm divides the nodes into two sets: tentative and permanent. It chooses nodes, makes them tentative, examines them, and if they pass the criteria, makes them permanent. The algorithm can be defined by the following steps:

1. Start with the source node: the root of the tree.

2. Assign a cost of 0 to this node and make it the first permanent node.

3. Examine each neighbor node of the node that was the last permanent node.

4. Assign a cumulative cost to each node and make it tentative.

5. Among the list of tentative nodes

A) Find the node with the smallest cumulative cost and mark it as permanent. A permanent node will not be checked ever again; its cost recorded now is final.

B) If a node can be reached from more than one direction, select the direction with the shortest cumulative cost.

6. Repeat steps 3 to 5 until every node becomes permanent.

Also in figure 4.6, showed the process of how Dijkstra algorithm works as in flowchart.

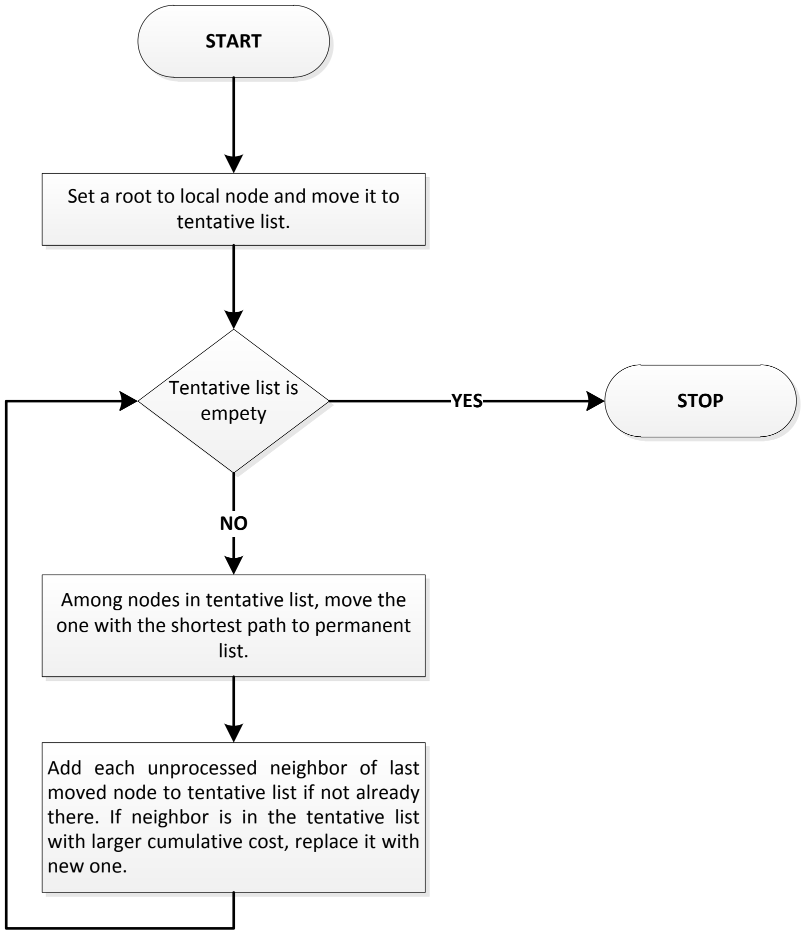

START
Set a root to local node and move it to tentative list.
Tentative list is empety
YES
STOP
NO
Among nodes in tentative list, move the one with the shortest path to permanent list.
Add each unprocessed neighbor of last moved node to tentative list if not already there. If neighbor is in the tentative list with larger cumulative cost, replace it with new one.

Figure 3.7 Overview of Dijkstra algorithm

CHAPTER 5

RESULT AND DISCUSSION

4.1 Implementation overview

My design and implementations are divided into two main parts. The first step is to simulation Dijkstra algorithm and the second one is to implement genetic algorithm in two kind of environment; MATLAB and C#.NET.

This chapter discusses and justifies our software partitioning approach, systems and tools used implementation model, and design and implementation techniques. All results and their behaviors will be explained briefly.

For evaluating my proposed method, I implemented and tested it on my scenario. Before complete discussion about these two steps, we need to have technical background knowledge about the software that I used in my project. In the following sections first discuss MATLAB programming language.

4.2 The MATLAB GA toolbox

The MATLAB Genetic Algorithm Toolbox aims to make GAs access to the control engineer. This allows the retention of existing modeling and simulation tools for building objective functions and allows the user to make direct comparisons between genetic methods and traditional procedures.

4.2.1 Toolbox structure

The GA Toolbox uses MATLAB matrix functions to build a set of multi purpose routines for implementing a wide range of genetic algorithm methods. In this section we outline the major procedures of the GA Toolbox.

- **Population representation and initialization**

The GA Toolbox supports binary, integer and floating-point chromosome representations. Binary and integer populations may be initialized using the Toolbox function to create binary populations, `crtbp`.

- **Fitness assignment**

The fitness function transforms the raw objective function values into non-negative figures of value for each individual. The Toolbox supports the offsetting and scaling method of Goldberg and the linear-ranking algorithm of Baker.

- **Selection functions**

These functions select a given number of individuals from the current population, according to their fitness, and return a column vector to their indices. Currently available routines are roulette wheel selection, `rws`, and stochastic universal sampling, `sus`. A high-level entry function, `select`, is also provided as a convenient interface to the selection routines, particularly where multiple populations are used. In cases where a generation gap is required, i.e. where the entire population is not reproduced in each generation, reins can be used to effect uniform random or fitness-based re-insertion.

- **Crossover operators**

The crossover routines recombine pairs of individuals with given probability to produce offspring. Single-point, double-point and shuffle crossover are implemented in the routines xovsp, xovdpand xovsh respectively. A general multi-point crossover routine, xovmp, that supports uniform crossover is also provided. To support real-valued chromosome representations, discrete, intermediate and line recombination are supplied in the routines, recdis, recintandreclin respectively. A high-level entry function to all the crossover operators supporting multiple subpopulations is provided by the function recombine.

- **Mutation operators**

Binary and integer mutation are performed by the routinemut. Real-value mutation is available using the breeder GA mutation function, mutbga.

4.3 Implementation of Dijkstra algorithm

4.3.1 MALTAB environment

Consider to the network topology, the numbers across each link represent distances or weights. The objective is to find the shortest path for the source node 1 to reach destination node 10.

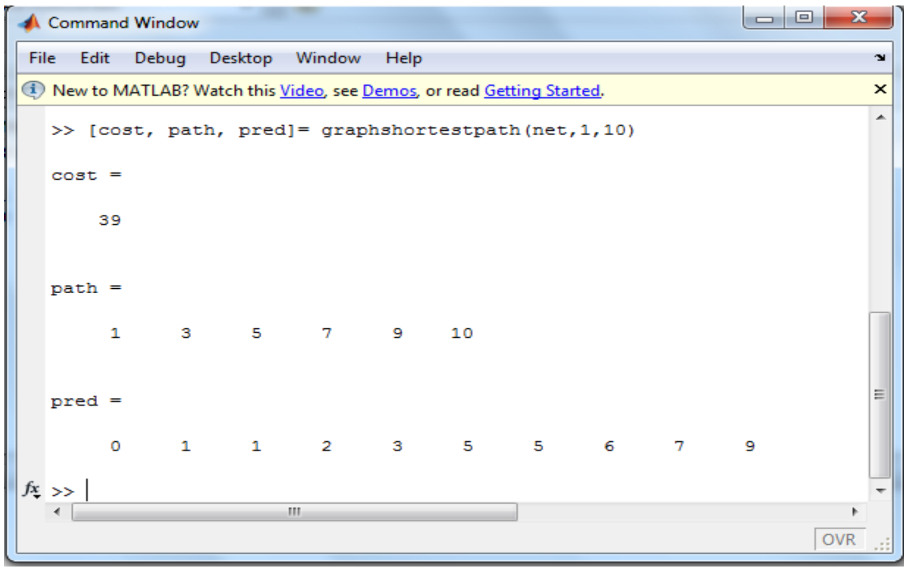
The network is created using the following MATLAB command:

```
net = sparse([1 1 2 2 3 4 4 5 5 6 7 8 9 9],[2 3 3 4 5 6 7 6 7 8 9 10 8 10],[15 10 3 8 9 7 5 6
2 12 10 6 10 8],10,10)
```

The first three row vectors inthe sparse command represent source nodes, destination nodes, and equivalent costs respectively. The MATLAB command graphshortestpath is executed to find the shortest path. This command applies Dijkstra's algorithm as the default one to find the optimal solution.

```
[cost, path, pred]= graphshortestpath(net,1,10)
```

The obtained solution is:


Command Window
File Edit Debug Desktop Window Help
New to MATLAB? Watch this Video, see Demos, or read Getting Started.
OVR

```
>> [cost, path, pred]= graphshortestpath(net,1,10)

cost =

    39


path =

     1     3     5     7     9    10


pred =

     0     1     1     2     3     5     5     6     7     9

>> 
```

Figure 4.1 Results of Dijkstra algorithm in MATLAB

That means; the shortest path from node 1 to node 10 has to pass via nodes 3, 5, 7, 9 wit minimum cost 39. The network is viewed using the following command in MATLAB and showed in software as figure 5.2;

```
View (biograph (net , [] ,' ShowArrows ',' off ',' ShowWeights ' ,' on '))
```

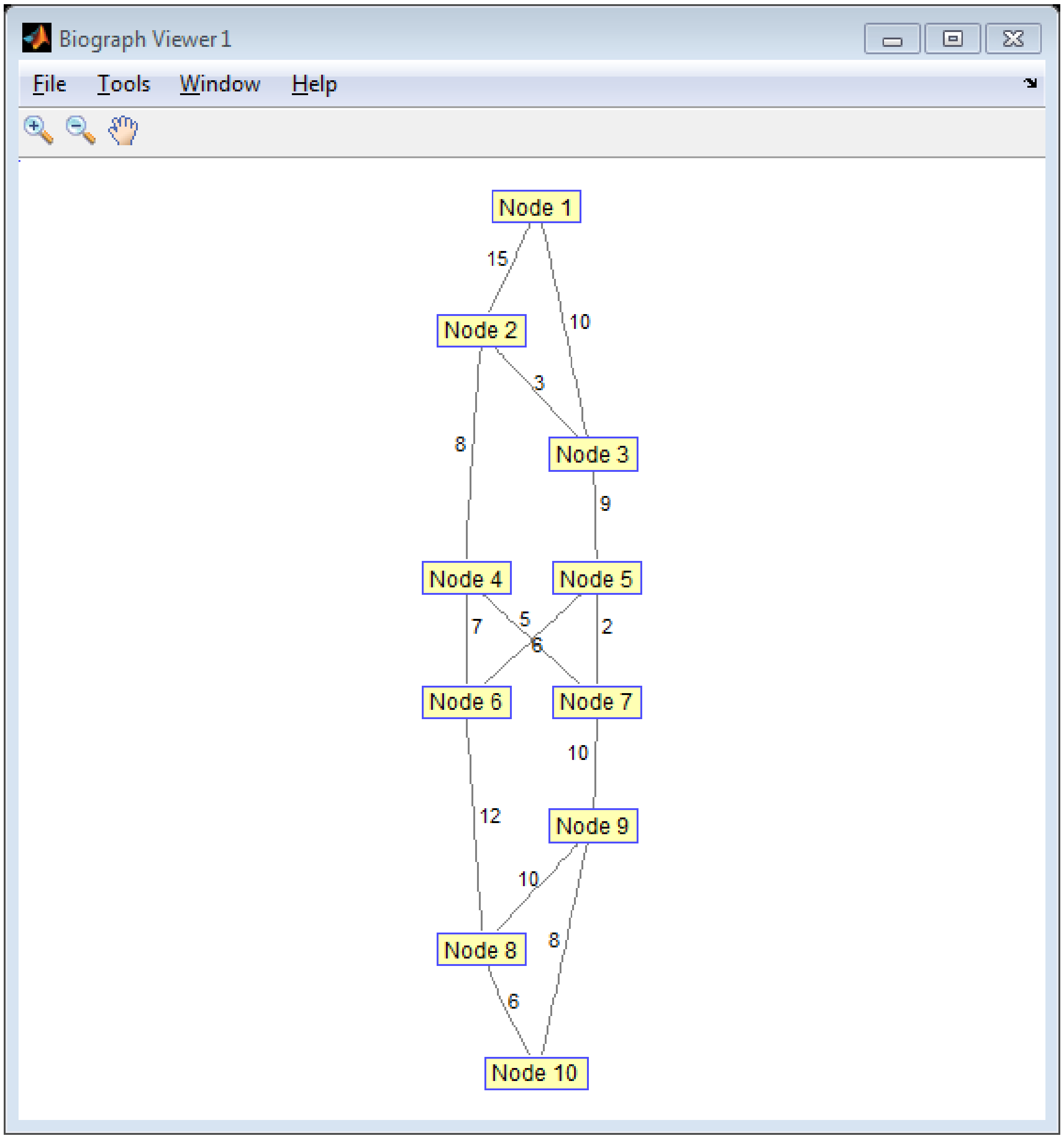
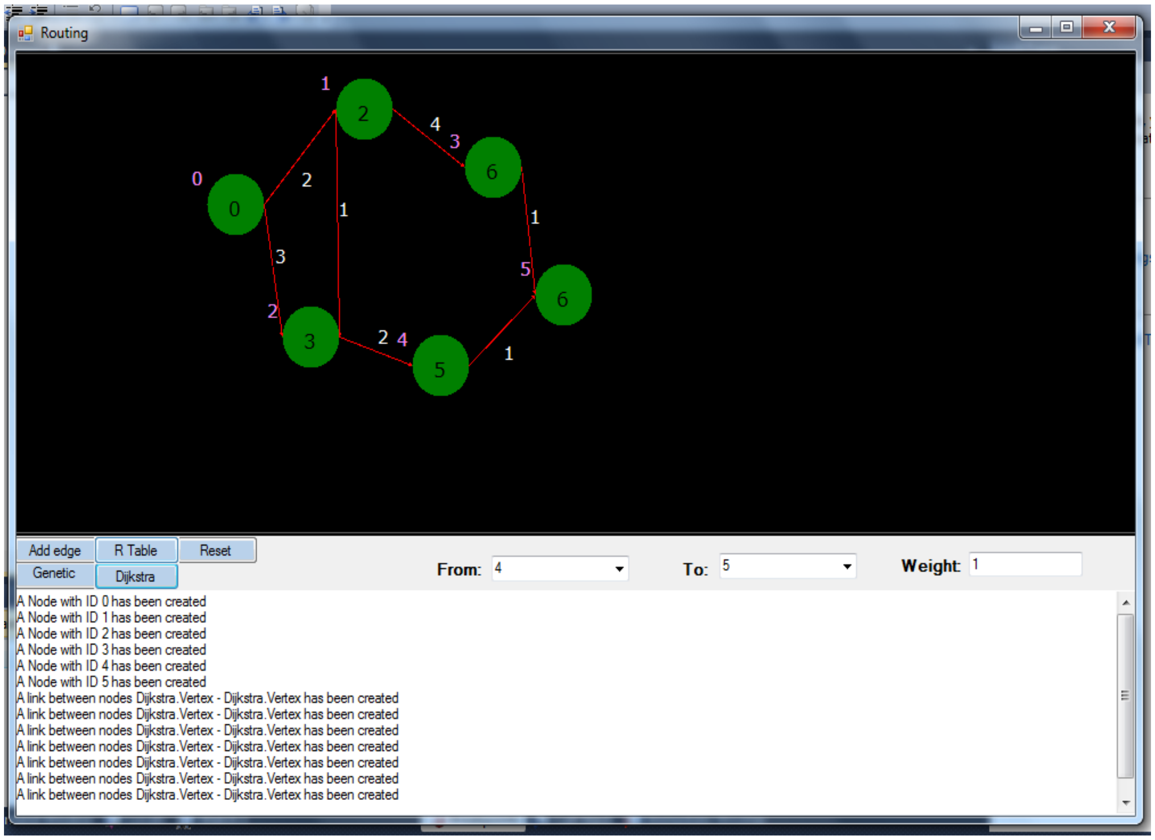


Figure 4.2 Network Topology in MATLAB

4.3.2 C# programming language

I also implemented Dijkstra algorithm in C#.NET programming language as follow:



```
A Node with ID 0 has been created
A Node with ID 1 has been created
A Node with ID 2 has been created
A Node with ID 3 has been created
A Node with ID 4 has been created
A Node with ID 5 has been created
A link between nodes Dijkstra.Vertex - Dijkstra.Vertex has been created
A link between nodes Dijkstra.Vertex - Dijkstra.Vertex has been created
A link between nodes Dijkstra.Vertex - Dijkstra.Vertex has been created
A link between nodes Dijkstra.Vertex - Dijkstra.Vertex has been created
A link between nodes Dijkstra.Vertex - Dijkstra.Vertex has been created
A link between nodes Dijkstra.Vertex - Dijkstra.Vertex has been created
A link between nodes Dijkstra.Vertex - Dijkstra.Vertex has been created
```

Figure 4.3 Dijkstra's algorithm results in C#

In this network, each cycle is one node that the number in each node showed the optimum answer to this node. As it showed the best answer to reach the destination is 6. That means the best cost to achieve a destination is 6.

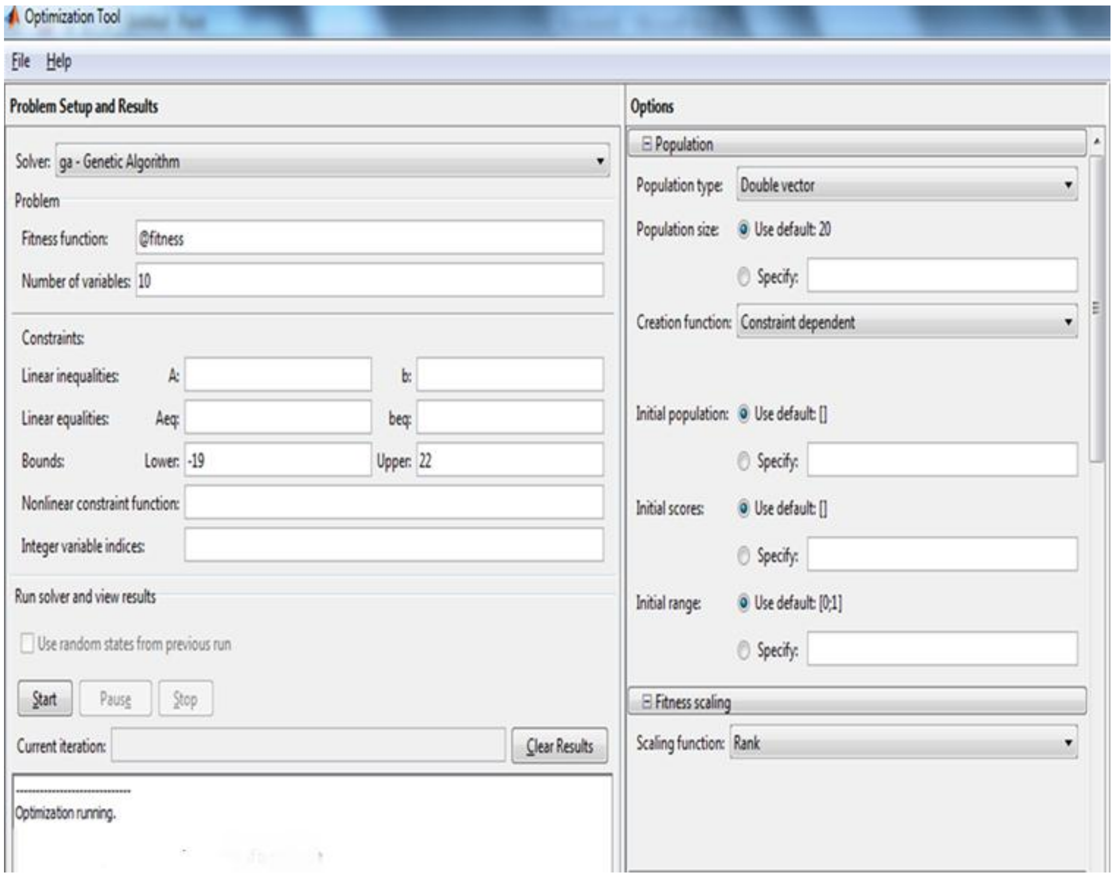
4.4 Implementation of Proposed Genetic algorithm

4.4.1 MATLAB environment

If we want to solve the Network to find the shortest path by using Genetic Algorithm, we have to first formed a fitness function, so that GA calculates the fitness value of each path and then provides the path with low path cost and high Fitness value, since both are inversely proportional to each other.

Now the Fitness function written, is used in GA tool for initialization of this tool enter “GA tool” in the command window. When this tool is opened, write the name of the fitness function as { @”name of the fitness function”}, then enter the number of variables, here it is 10 as there are 10 nodes for static network. When we press the start button on the tool, it starts optimization. After 51 Iterations, it shows the value of objective function=39.

The MATLAB environment has a very powerful string manipulation commands that helps to convert easily the numeric variables into strings and vice versa. Consequently, the bit crossover and mutation were so easily to be implemented in the developed program. Given the source node is 1 and the goal node is 10. The network in Figure 5.2 is simulated to find the shortest path. The initial population consists of 6 chromosomes 40 bits of each. The probability of crossover was chosen as 70% and the mutation rate equal to 2.5%. The best results are obtained using two points of crossover. The simulation result after 51 generations is given by:



Solver: ga - Genetic Algorithm

Problem

Fitness function: @fitness

Number of variables: 10

Constraints:

Linear inequalities: A: b:

Linear equalities: Aeq: beq:

Bounds: Lower: -19 Upper: 22

Nonlinear constraint function:

Integer variable indices:

Run solver and view results

Use random states from previous run

Current iteration:

```
-------------------------
Optimization running.
```

Population type: Double vector

Population size: Use default: 20 / Specify:

Creation function: Constraint dependent

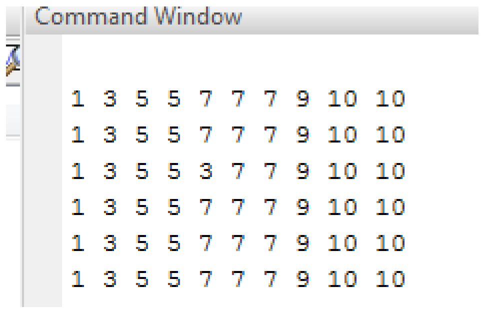
Initial population: Use default: [] / Specify:

Initial scores: Use default: [] / Specify:

Initial range: Use default: [0;1] / Specify:

Scaling function: Rank

Figure 4.4 Genetic algorithm toolbox in MATLAB



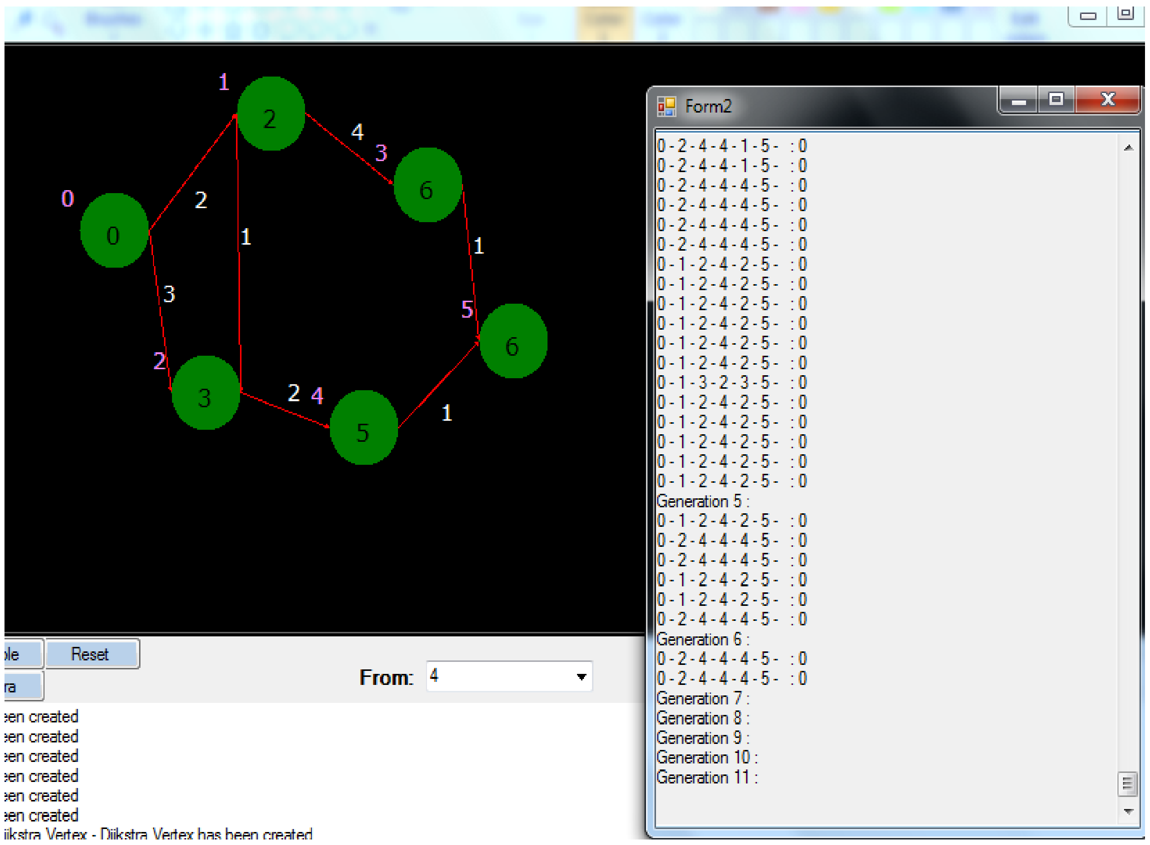
```
1  3  5  5  7  7  7  9  10  10
1  3  5  5  7  7  7  9  10  10
1  3  5  5  3  7  7  9  10  10
1  3  5  5  7  7  7  9  10  10
1  3  5  5  7  7  7  9  10  10
1  3  5  5  7  7  7  9  10  10
```

Figure 4.5 Genetic algorithm results in MATLAB

The third chromosome is slightly different due to mutation process. The part of repeated numbers is cut off and the final result has been converged to the optimal/shortest individual {1-3-5-7-9-10} with the cost of 39; which is the same results as Dijkstra's algorithm. The obtained results affirmed the potential of the proposed algorithm where the convergence was guaranteed to obtain the optimal path in each case.

4.4.2 C# programming language

As showed in follow, it is an implementation of my project in C#.NET. It reaches best answer in 6th of generation. The answer that shows is {0-2-4-4-4-5}, that it means the best way to get destination is 0-2-4-5, starts with source node and pass from node 2 and 4 , after that reaches the destination as node 5. This answer is the same answer for Dijstra algorithm.



```
Form2
0 - 2 - 4 - 4 - 1 - 5 -  : 0
0 - 2 - 4 - 4 - 1 - 5 -  : 0
0 - 2 - 4 - 4 - 4 - 5 -  : 0
0 - 2 - 4 - 4 - 4 - 5 -  : 0
0 - 2 - 4 - 4 - 4 - 5 -  : 0
0 - 2 - 4 - 4 - 4 - 5 -  : 0
0 - 1 - 2 - 4 - 2 - 5 -  : 0
0 - 1 - 2 - 4 - 2 - 5 -  : 0
0 - 1 - 2 - 4 - 2 - 5 -  : 0
0 - 1 - 2 - 4 - 2 - 5 -  : 0
0 - 1 - 2 - 4 - 2 - 5 -  : 0
0 - 1 - 2 - 4 - 2 - 5 -  : 0
0 - 1 - 3 - 2 - 3 - 5 -  : 0
0 - 1 - 2 - 4 - 2 - 5 -  : 0
0 - 1 - 2 - 4 - 2 - 5 -  : 0
0 - 1 - 2 - 4 - 2 - 5 -  : 0
0 - 1 - 2 - 4 - 2 - 5 -  : 0
0 - 1 - 2 - 4 - 2 - 5 -  : 0
Generation 5 :
0 - 1 - 2 - 4 - 2 - 5 -  : 0
0 - 2 - 4 - 4 - 4 - 5 -  : 0
0 - 2 - 4 - 4 - 4 - 5 -  : 0
0 - 1 - 2 - 4 - 2 - 5 -  : 0
0 - 1 - 2 - 4 - 2 - 5 -  : 0
0 - 2 - 4 - 4 - 4 - 5 -  : 0
Generation 6 :
0 - 2 - 4 - 4 - 4 - 5 -  : 0
0 - 2 - 4 - 4 - 4 - 5 -  : 0
Generation 7 :
Generation 8 :
Generation 9 :
Generation 10 :
Generation 11 :
```

```
een created
een created
een created
een created
een created
een created
ijkstra Vertex - Dijkstra Vertex has been created
```

Figure 4.6 Genetic algorithm results in C#.NET

4.4.3 Comparing Dijkstra's Algorithm with Genetic Algorithm

Besides using Dijkstra's Algorithm to find minimum path we will compare the result with another method such as Genetic Algorithm (GA). A GA is a randomized search method modeled on evolution and introduced by John Holland [14]. According to Goldberg [15],GA is being applied to a variety of problems and becoming an important tool in combinatorial optimization problems. Another researcher Bryant [16] says that, GAs are an optimization technique based on natural evolution and include the survival of the fittest idea in to a search algorithm which provides a method of searching which does not need to explore every possible solution in the feasible region to obtain a good result.

[16], GA is an adaptive heuristic search method based on population genetics where the basic concepts are developed by [17] and the practicality of using the GA to solve complex problems is demonstrated in (D eJong,1975) and [15] . GA also is based on the same idea as the theory of evolution where several random sets of parameters are applied to an algorithm and a fitness value is returned for each [18]. Another researcher Vose and Michael (1998), GA use the concept of survival of the fittest by rejecting strings that are not good enough to take part in producing and after creation of a few generations, the best value is selected as the optimal solution.

Another researcher, Vacic and Sobh (August 2002) said, a GA will start with a set of chromosomes called the initial population and it is either randomly generated or generated using some form of heuristics. A selection mechanism will then be used to select the prospective parents based on their fitness. Their offspring will constitute the next generation. The selected parent chromosomes will then be recombined via the crossover operator to create a potential new population. The next step will be to mutate a small number of the newly obtained chromosomes, in order to introduce a level of randomness that will preserve the GA from converging to a local optimum. Finally, new population will then be selected based on the fitness of the candidate chromosomes.

As a conclusion, Dijkstra's Algorithm and GA's are using the same principles to succeed in the minimum path problem that has a globally convex structure of its solution space. Meanwhile, a better understanding of how and why methods based on social insects work and more systematic comparisons with other are needed.

CHAPTER 6

CONCLUSION

5.1 Conclusion

A simple genetic algorithm is developed to find the shortest path routing in a dynamic network. The developed algorithm uses an efficient coding scheme. The chromosome length depends on the number of nodes in the network. The C#.Net and MATLAB environment searches the shortest path using Dijkstra's algorithm as the default one. The algorithm is simulated as an example to solve the network of 10 nodes for the first one as the source node. Also, the developed GA is simulated to find the solution for the same problem. The obtained results affirmed the potential of the proposed algorithm that gave the same results as Dijkstra’s algorithm.

So I suggest using the two algorithms together (GA and Dijkstra). First use the Dijkstra but if the path takes a long time to deliver the packets because of overloading immediately use the genetic algorithm. Then as a conclusion,Dijkstra algorithm is too simple but it is not suitable for huge networks, that’s why it is time consuming for processing data.

5.2 Future works

I visualize the outcomes of this research go beyond local aspects and also become relevant from a distributed point of view because of the efficiency of the genetic algorithm to solve many problems in any field.

- the developed GA will be more investigated to decrease the chromosome length.

- It will be implemented in networks with a large number of nodes.

BIBLIOGRAPHY

[1] A. S. Tannenbaum, *Computer Networks*. india: Prentice-Hall of India, 2008, p. 1123.

[2] E. W. Dijkstra, “A Note on Two Problems in Connexion with Graphs,” *Numerische Mathematik 1*, no. l 959, pp. 269–271, 1959.

[3] and C. S. T. H. Cormen, C. E. Leiserson, R. L. Rivest, “Dijkstra’s algorithm. Introduction to Algorithms,” Second., MIT Press and McGraw-Hill, 2001, pp. 595–601.

[4] E. F. Moore, *The shortest path through a maze, Proceedings of an International Symposium on the Theory of Switching*, Harvard Un., no. March. Cambridge, Massachusetts, 1957, pp. 285–292.

[5] M. Sniedovich, “Dijkstra ’ s algorithm revisited : the dynamic programming connexion by,” vol. 35, no. 3, pp. 87–92, 2006.

[6] C. N. K. and D. K. V Viswanatha, “Routing in Dynamic Network using Ants and Genetic Algorithm,” *Journal, Ijcsns International Science, Computer Security, Network*, vol. 9, no. 3, pp. 194–200, 2009.

[7] M. Murata and T. Toku, “A Study on Routing Algorithms with Delayed Link State Information for Distributed Lightpath Network,” Osaka University, 2002.

[8] H. U. I. Zang, “A Review of Routing and Wavelength Assignment Approaches for Wavelength- Routed Optical WDM Networks,” no. January, pp. 47–60, 2000.

[9] J. Z. and X. Yuan, “A study of dynamic routing and wavelength assignment with imprecise network state information,” *in Proceedings of International Conference on Parallel Processing Workshops (ICPPW’02)*, pp. 202–207, 2002.

[10] C. M. Gass, S.I. and Harris, *Encyclopedia of Operations Research and management Science*, vol. 79490888. 1996, p. 223.

[11] R. E. Fredman, Michael Lawrence; Tarjan, “Fibonacci heaps and their uses in improved network optimization algorithms,” *25th Annual Symposium on Foundations of Computer Science (IEEE)*, pp. 338–346, 1984.

[12] P. Biswas, P. K. Mishra, and N. C. Mahanti, "COMPUTATIONAL EFFICIENCY OF OPTIMIZED SHORTEST PATH ALGORITHMS," *International Journal of Computer Science & Applications*, vol. II, no. Ii, pp. 22–37, 2005.

[13] B. A. Forouzan, *DATA COMMUNICATIONS AND NETWORKING*. Tehran: McGraw-Hill, 2003.

[14] J. H. Holland, *Adaptation in Natural and Artificial Systems*. MI, The University of Michigan Press, 1975.

[15] D. E. Goldberg, *Genetic Algorithms in Search,Optimization, and Machine Learning*. Addison-Wesley, 1989, p. 412.

[16] K. Bryant, "Genetic Algorithms and the Traveling Salesman Problem by Kylie Bryant," 2000.

[17] J. H. Holland, *Adaptation in Natural and Artificial Systems*. University of Michigan Press, 1975, p. 434.

[18] C.-D. R, "World Survey of Genomics Research Funding, Interim Report," *Switzerland*, no. 2, 2000.

[19] S. N. D. S.N.Sivanandam, *Introduction to Genetic Algorithms*. Springer Berlin Heidelberg New York, 2008, p. 453.

[20] K. A. De Jong, "Analysis of the behavior of a class of genetic adaptive systems," University of Michigan, Ann Arbor, 1975.

[21] R. Akbari and K. Ziarati, "A multilevel evolutionary algorithm for optimizing numerical functions," *International Journal of Industrial Engineering Computations*, vol. 2, no. 2, pp. 419–430, Apr. 2011.

[22] C. W. Ahn, S. Member, R. S. Ramakrishna, and S. Member, "A Genetic Algorithm for Shortest Path Routing Problem and the Sizing of Populations," vol. 6, no. 6, pp. 566–579, 2002.

[23] N. Shimamoto, a. Hiramatsu, and K. Yamasaki, "A dynamic routing control based on a genetic algorithm," *IEEE International Conference on Neural Networks*, pp. 1123–1128.

[24] A. Khan, S. Beg, F. Ahsan, and S. Mohsin, "BIONOMIC ALGORITHM FOR SHORTEST PATH FIRST," vol. 34, no. 1, 2011.

[25] F. Xiang, L. Junzhou, W. Jieyi, and G. Guanqun, “QoS routing based on genetic algorithm,” *Computer communications*, vol. 22, no. 15, pp. 1392–1399, 1999.

[26] R. Kumar and M. Kumar, “Exploring Genetic Algorithm for Shortest Path Optimization in Data Networks,” vol. 10, no. 11, pp. 8–12, 2010.

[27] B. Gonen, “Genetic Algorithm Finding the Shortest Path in Networks,” no. USA, pp. 1–4, 2008.

[28] D. Journal, “ROUTING USING GENETIC ALGORITHM FOR LARGE NETWORKS,” *Diyala Journal of Engineering Sciences*, vol. 03, no. 02, pp. 53–70, 2010.