



**AN APPROACH FOR FINDING
A BEST PATH ON ROUTERS**

**By
Mohammad Hafez Abdul-Rahim Mustafa**

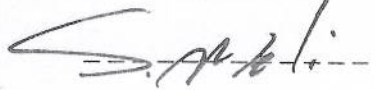
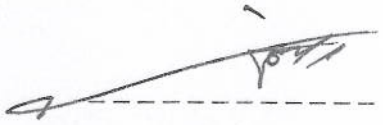
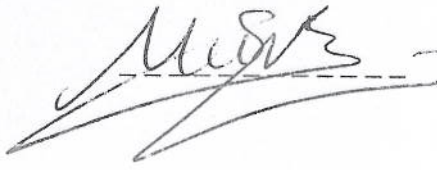

**Supervisor
Dr. Saad Al-Mahdawy**

**Co-Supervisor
Dr. Ramzi Sunaa**

**This Thesis was Submitted in Partial Fulfillment of the
Requirements for the Master's Degree in Computer Science**

**Deanship of Academic Research and Graduate Studies
Philadelphia University
February 2008**

Successfully defended and approved on 2-4-2008

Examination Committee	Signature
Dr. <u>Saad A. Al Mahdawy</u> , Chairman. Academic Rank: <u>Associate Professor</u>	
Dr. <u>Rashid Al-Zubaidy</u> , Member. Academic Rank: <u>Associative prof.</u>	
Dr. <u>Issa Shehabat</u> , Member. Academic Rank: <u>Assistant prof.</u>	
Dr. <u>Ahmad Dalalah</u> , External Member. Academic Rank: <u>Asst. Prof</u> (Jordan University of Science & Technology)	

جامعة فيلادلفيا
نموذج تفويض

انا محمد حافظ عبدالرحيم مصطفى، أفوض جامعة فيلادلفيا بتزويد نسخ من رسالتي للمكتبات او المؤسسات او الهيئات او الاشخاص عند طلبها.

التوقيع:
التاريخ:

Philadelphia University
Authorization Form

I, Mohammad Hafiz Abdul-Rahim Mustafa, authorize Philadelphia University to supply copies of my Thesis to libraries or establishments or individuals upon request.

Signature:
Date:

AN APPROACH FOR FINDING A BEST PATH ON ROUTERS

By
Mohammad Hafiz Abdul-Rahim Mustafa

Supervisor
Dr. Saad Al-Mahdawy

Co-Supervisor
Dr. Ramzi Sunaa

**This Thesis was Submitted in Partial Fulfillment of the Requirements for
the Master's Degree in Computer Science**

Deanship of Academic Research and Graduate Studies

Philadelphia University

February 2008

Successfully defended and approved on _____

Examination Committee

Signature

Dr, _____, Chairman.

Academic Rank: _____

Dr, _____, Member.

Academic Rank: _____

Dr, _____, Member.

Academic Rank: _____

Dr, _____, External Member.

Academic Rank: _____

(_____)

Dedication

I fully dedicate this thesis to my whole family, especially my mother whose love and support kept me going. This thesis is also dedication to my special father whose soul will always be loved and cherished.

Also, I will also extend this dedication to my brothers and friends.

Acknowledgment

I would like to express my regards and appreciation to the committee members including Dr. Saad Al-Mahdawy and Dr. Ramzi Sunaa whom has evaluated my work from the beginning, and to all of those who have helped and encouraged me.

Mohammad Mustafa

Table of Contents

Subject	Page
Committee Decision	i
Title	ii
Dedication	iv
Acknowledgement	v
Table of Contents	vi
List of Tables	ix
List of Figures	x
List of Abbreviations	xi
Abstract	xii
Chapter one	1
1.1 Introduction	2
1.2 Etymology	2
1.3 Why algorithms are necessary: an informal definition	2
1.4 Formalization of Algorithms	3
1.4.1 Termination	4
1.4.2 Expressing Algorithms	4
1.4.3 Implementation	5
1.5 Application of Genetic Algorithm	6
1.6 Routing	6
1.6.1 Delivery Semantics	8
1.6.2 Topology Distribution	8
1.6.2.1 Distance Vector Algorithms	8
1.6.2.2 Link-state Algorithms	9
1.6.2.3 Path Vector Protocol	10
1.6.2.4 Comparison of Routing Algorithms	10
1.6.3 Path Selection	11
1.6.4 Routing Algorithms	11
1.6.5 Routing Algorithm Types	12
1.6.6 Network Protocols	12
1.7 Routing Security	13
Chapter Two	14
2.1 Introduction	15
2.2 Using GA to solve routing problem	15
2.3 Using Random Key-Based GA	18
2.4 Using Adaptive Fitness Function	19
2.5 Using GA to Finding Shortest Path	21
2.6 Conclusions of Related Works	23
Chapter Three	24
3.1 Introduction	25

3.2 Motivations	25
3.3 Genetic Algorithms and Adaptive Genetic Algorithms	25
3.3.1 Why Using Adaptive GA?	26
3.4 Manchester Encoding	27
	27
3.5 MATLAB	
3.5.1 MATLAB System	28
3.6 The Method	29
3.6.1 Initializing path searching mechanism	29
3.6.2 Finding the fitness value	32
3.6.3 Reproduction and data selection	32
3.6.4 Crossover and mutation	33
3.6.5 Suggested Adaptive Fitness Function	34
3.7 Flow Charts	34
3.8 Flow Charts Description	42
3.9 Program User Interface	43
Chapter Four	45
4.1 Introduction	46
4.2 Limitation	46
4.3 Results	47
Chapter Five	53
5.1 Conclusions	54
5.2 Future Works	54

List of Tables

Table Number	Table Title	Page
Table (3-1)	Cost Matrix	30
Table (3-2)	Speed Matrix	30
Table (3-3)	the GA initial population	32

List of Figures

Figure Number	Figure Title	Page
Figure (2-1)	A simple undirected network with 7 nodes and 12 edges	18
Figure (2-2)	Example of generated chromosome and its decoded path	19
Figure (2-3)	Crossover Operator	22
Figure (3-1)	diagram of all possible paths with Their cost and speed	33
Figure (3-2)	crossover operation	34
Figure (3-3)	a flow chart explaining the stage of initializing	35

	the parameters	
Figure (3-4)	a flow chart explaining the stage of Pre-GA stage	36
Figure (3-5)	a flow chart explaining the stage Reproduction	37
Figure (3-6)	flow chart explaining the reproduction stage (continue)	38
Figure (3-7)	flow chart explaining the crossover stage	39
Figure(3-8)	a flow chart explaining the crossover stage	40
Figure(3-9)	a flow chart explaining the adaptive fitness stage	41
Figure(3-10)	user interface	43
Figure(3-11)	user interface in action	44
Figure (4-1)	no. of total network nodes versus best path no. of nodes (GA)	48
Figure (4-2)	no. of total network nodes versus best path no. of nodes (AGA)	48
Figure (4-3)	no. of total network nodes versus best path fitness values (GA)	49
Figure (4-4)	no. of total network nodes versus best path fitness values (AGA)	49
Figure (4-5)	No. of total Network Nodes versus Best Path No. of Generation (GA)	50
Figure (4-6)	No. of total network nodes versus best path No. of generation (AGA)	50
Figure (4-7)	no. of total network nodes versus time to find best path (GA)	51
Figure (4-8)	no. of total network nodes versus time to find best path (AGA)	52

List of Abbreviations

AGA	Adaptive Genetic Algorithm
BGP	Border Gateway Protocol
COP	Constrained Optimization Problem
CSP	Constraint Satisfaction Problem
EGP	Exterior Gateway Protocol
EON	European Optical Network
EAs	Evolutionary Algorithms
FOP	Free Optimization Problem
GA	Genetic Algorithm
GBML	Genetic Based Machine Learning
IGRP	Interior Gateway Routing Protocol
IP	Internet Protocol
IRP	Intra-domain Internet Routing Protocol
ISP	Internet Service Providers
IS-IS	Intermediate System-to-Intermediate System
MATLAB	Matrix Library
MOEA	Multiobjective Evolutionary Algorithm
NLP	Nonlinear Programming
OSI	Open System Interconnection
OSPF	Open Shortest Path First
PSTN	Public Switch Telephone Network

RIP	Routing Information Protocol
RRS	Recursive Random Search
WAN	Wide Area Network
WDM	Wavelength Division Multiplexing
XNS	Xerox Network System

Abstract

One of the prevailing tendencies of the modern stage of development of information technology is the telecommunication technologies integration based on computer networks which become more complex and the traffic load increases. There is a need to determine the routing traffic within a network so as to minimize the number of communication channels used. To reduce the risk of being unable to handle traffic required to find the best and optimal path from the source to the destination and to minimize the total cost of the [system](#) operation.

This thesis will focus on the methodology that implements hybrid dynamic routing protocol that can solve congestion problem and hacking problem using Adaptive Genetic Algorithm. The new suggested structure combines different solutions to select the optimal path. Such structure will take into consideration different circumstances related to high load and utilization on advanced Wide Area Networks due to Security gaps and probable attacks and network activities.

Chapter One

Introduction

1.1 Introduction

Routing problems can occur when either the host's or routers routing tables contain information that does not reflect the correct topology of the internet work. The aim of this thesis is to define a new approach that solves the problem of finding best path on routing. This approach will be designed by using Adaptive Genetic Algorithm (AGA). In mathematics, computing, linguistics, and related disciplines, an **algorithm** is a finite list of well-defined instructions for accomplishing some task that, given an initial state, will proceed through a well-defined series of successive states, possibly eventually terminating in an end-state. The concept of an algorithm originated as a means of recording procedures for solving mathematical problems such as finding the common divisor of two numbers or multiplying two numbers.

1.2 Etymology

Al-Khwārizmī, Persian astronomer and mathematician, wrote a treatise in Arabic in 825 AD, on calculation with Hindu Numerals. It was translated into Latin in the 12th century as *Algoritmi de numero Indorum*. That title was likely intended to mean "Algoritmi on the numbers of the Indians", where "Algoritmi" was the translator's rendition of the author's name; but people misunderstood the title and treated "Algoritmi" as a Latin plural and this led to the word "algorithm" (Latin *algorithmus*) which coming to mean "calculation method". The intrusive "th" is most likely due to a false cognate with the Greek "ἀριθμός" (*arithmos*) meaning "number"[1].

1.3 Why algorithms are necessary: an informal definition

Generally, no accepted formal definition of "algorithm" exists yet. We can, however, derive clues to the issues involved and an informal meaning of the word from the following quotation from Boolos and Jeffrey, "No human being can write fast enough, or long enough, or small enough to list all members of an enumerable infinite set by writing out their names, one after another, in some notation". But humans can do something equally

useful, in the case of certain enumerable infinite sets: They can give explicit instructions for determining the n^{th} member of the set, for arbitrary finite n . Such instructions are to be given quite explicitly, in a form in which they could be followed by a computing machine or by a human who is capable of carrying out only very elementary operations on symbols. The words "enumerable infinite" mean "countable using integers perhaps extending to infinity". Thus Boolos and Jeffrey are saying that an algorithm implies instructions for a process that "creates" output integers from an arbitrary "input" integer or integers that, in theory, can be chosen from 0 to infinity. Thus we might expect an algorithm to be an algebraic equation such as $y = m + n$ two arbitrary "input variables" m and n that produce an output y . Precise instructions (in language understood by "the computer") for a "fast, efficient, good" process that specifies the "moves" of "the computer" (machine or human, equipped with the necessary internally-contained information and capabilities) to find, decode, and then much arbitrary input integers/symbols m and n , symbols $+$ and $=$... and (reliably, correctly, "effectively") produce, in a "reasonable" time, output-integer y at a specified place and in a specified format. The concept of algorithm is also used to define the notion of decidability (logic). That notion is central for explaining how formal systems come into being starting from a small set of axioms and rules. In logic, the time that an algorithm requires to complete cannot be measured, as it is not apparently related with our customary physical dimension. From such uncertainties, that characterize ongoing work, stems the unavailability of a definition of algorithm that suits both concrete (in some sense) and abstract usage of the term[2].

1.4 Formalization of Algorithms

Algorithms are essential to the way computers process information. This is because a computer program is essentially an algorithm that tells the computer what specific steps to perform (in what specific order) in order to carry out a specified task, such as calculating employees' paychecks or printing students' report cards. Thus, an algorithm can be considered to be any sequence of operations that can be performed by a Turing-complete system. Typically, when an algorithm is associated with processing information, data are read from an input source or device, written to an output sink or device, and/or stored for further processing. Stored data are regarded as part of the internal state of the entity

performing the algorithm. In practice, the state is stored in a data structure, but an algorithm requires the internal data only for specific operation sets called abstract data types. For any such computational process, the algorithm must be rigorously defined: specified in the way it applies in all possible circumstances that could arise. That is, any conditional steps must be systematically dealt with, case-by-case; the criteria for each case must be clear (and computable). Because an algorithm is a precise list of precise steps, the order of computation will almost always be critical to the functioning of the algorithm. Instructions are usually assumed to be listed explicitly, and are described as starting 'from the top' and going 'down to the bottom', an idea that is described more formally by flow of control. So far, this discussion of the formalization of an algorithm has assumed the premises of imperative programming. This is the most common conception, and it attempts to describe a task in discrete, 'mechanical' means. Unique to this conception of formalized algorithms is the assignment operation, setting the value of a variable. It derives from the intuition of 'memory' as a scratchpad. There is an example below of such an assignment.

1.4.1 Termination

Some writers restrict the definition of *algorithm* to procedures that eventually finish. In such a category Kleene places the "*decision procedure* or *decision method* or *algorithm*". Others, including Kleene, include procedures that could run forever without stopping; such a procedure has been called a "computational method or "*calculation procedure* or *algorithm*"; however, Kleene notes that such a method must eventually exhibit "some object"[3].

1.4.2 Expressing algorithms

Algorithms can be expressed in many kinds of notation, including natural languages, pseudo-code, flowcharts, and programming languages. Natural language expressions of algorithms tend to be verbose and uncertain, and are rarely used for complex or technical algorithms. Pseudo-code and flowcharts are structured ways to express algorithms that avoid many of the uncertainty common in natural language statements, while remaining independent of a particular implementation language. Programming languages are primarily intended for expressing algorithms in a form that can be executed

by a computer, but are often used as a way to define or document algorithms. There is a wide variety of representations possible and one can express a given Turing machine program as a sequence of machine tables, as flowcharts, or as a form of rudimentary machine code or assembly code called "sets of quadruples". Sometimes it is helpful in the description of an algorithm to supplement small "flow charts" (state diagrams) with natural-language and/or arithmetic expressions written inside "block diagrams" to summarize what the "flow charts" are accomplishing [3].

Representations of algorithms are generally classed into three accepted levels of Turing machine description:

High-level description:

Prose to describe an algorithm, ignoring the implementation details. At this level we do not need to mention how the machine manages its tape or head.

Implementation description:

Prose used to define the way the Turing machine uses its head and the way that it stores data on its tape. At this level we do not give details of states or transition function.

Formal description:

Most detailed, "lowest level", gives the Turing machine's "state table".

1.4.3 Implementation

Most algorithms are intended to be implemented as computer programs. However, algorithms are also implemented by other means, such as in a biological neural network (for example, the human brain implementing arithmetic or an insect looking for food), in an electrical circuit, or in a mechanical device.

1.5 Application of Genetic Algorithm

Genetic Algorithms (GA) are a very effective way of quickly finding a reasonable solution to a complex problem. Granted they aren't instantaneous, or even close, but they do an excellent job of searching through a large and complex search space. Genetic algorithms are most effective in a search space for which little is known. We may know exactly what we want a solution to do but have no idea how we want it to go about doing it. This is where genetic algorithms thrive. They produce solutions that solve the problem in ways we may never have even considered. Then again, they can also produce solutions that only work within the test environment and flounder once we try to use them in the real world [4].

In the mean time GA are used in different spheres:

1. Signal processing
2. Speech processing
3. Time delay estimation
4. Active noise control
5. Image processing
6. Neural networks
7. Computer networks

1.6 Routing

Routing is the process of selecting paths in a network along which to send data or physical traffic. Routing is performed for many kinds of networks, including the telephone network, the Internet, and transport networks.

Routing directs forwarding, the passing of logically addressed packets from their source toward their ultimate destination through intermediary nodes; typically hardware devices called routers, bridges, gateways, firewalls, or switches. Ordinary computers with multiple network cards can also forward packets and perform routing, though with more limited performance. The routing process usually directs forwarding on the basis of routing tables which maintain a record of the routes to various network destinations. Thus constructing routing tables, which are held in the routers' memory, becomes very important for efficient routing. Routing, in a more narrow sense of the term, is often contrasted with

bridging in its assumption that network addresses are structured and that similar addresses imply proximity within the network. Because structured addresses allow a single routing table entry to represent the route to a group of devices, structured addressing (routing, in the narrow sense) outperforms unstructured addressing (bridging) in large networks, and has become the dominant form of addressing on the Internet, though bridging is still widely used, albeit within localized environments[5]. There are two types of routing as follow:

Static routing: Static routing is not really a protocol, simply the process of manually entering routes into the routing table via a configuration file that is loaded when the routing device starts up. As an alternative, these routes can be entered by a network administrator who configures the routes. Since these routes don't change after they are configured (unless a human changes them) they are called 'static' routes. Static routing is the simplest form of routing, but it is a manual process and does not work well when the routing information has to be changed frequently or needs to be configured on a large number of routing devices (routers). Static routing also does not handle outages or down connections well because any route that is configured manually must be reconfigured manually to fix or repair any lost connectivity.

Dynamic routing: Dynamic routing protocols are software applications that dynamically discover network destinations and how to get to them. A router will 'learn' routes to all directly connected networks first. It will then learn routes from other routers that run the same routing protocol. The router will then sort through its list of routes and select one or more 'best' routes for each network destination it knows or has learned. Dynamic protocols will then distribute this 'best route' information to other routers running the same routing protocol, thereby extending the information on what networks exist and can be reached. This gives dynamic routing protocols the ability to adapt to logical network topology changes, equipment failures or network outages 'on the fly'.

1.6.1 Delivery Semantics

Routing schemes differ in their delivery semantics:

Uni-cast delivers a message to a single specified node;

broadcast delivers a message to all nodes in the network;

multicast delivers a message to a group of nodes that have expressed interest in receiving the message;

Any-cast delivers a message to any one out of a group of nodes, typically the one nearest to the source;

Uni-cast is the dominant form of message delivery on the Internet.

1.6.2 Topology Distribution

Small networks may involve manually configured routing tables, while larger networks involve complex topologies and may change rapidly, making the manual construction of routing tables infeasible. Nevertheless, most of the Public Switched Telephone Network (PSTN) uses pre-computed routing tables, with fallback routes if the most direct route becomes blocked. *Dynamic routing* attempts to solve this problem by constructing routing tables automatically, based on information carried by routing protocols, and allowing the network to act nearly autonomously in avoiding network failures and blockages. Dynamic routing dominates the Internet. However, the configuration of the routing protocols often requires a skilled touch; one should not suppose that networking technology has developed to the point of the complete automation of routing.

1.6.2.1 Distance Vector Algorithms

Distance vector algorithms use the Bellman-Ford algorithm. This approach assigns a number, the cost, to each of the links between each node in the network. Nodes will send information from point A to point B via the path that results in the lowest total cost (i.e. the sum of the costs of the links between the nodes used). The algorithm operates in a very simple manner. When a node first starts, it only knows of its immediate neighbors, and the direct cost involved in reaching them. (This information, the list of destinations, the total cost to each, and the next hop to send data to get there, makes up the routing table, or distance table.) Each node, on a regular basis, sends to each neighbor its own current idea of the total cost to get to all the destinations it knows of. The neighboring node(s) examine this information, and compare it to what they already ‘know’; anything which represents an

improvement on what they already have, they insert in their own routing table(s). Over time, all the nodes in the network will discover the best next hop for all destinations, and the best total cost. When one of the nodes involved goes down, those nodes which used it as their next hop for certain destinations discard those entries, and create new routing-table information. They then pass this information to all adjacent nodes, which then repeat the process. Eventually all the nodes in the network receive the updated information, and will then discover new paths to all the destinations which they can still "reach"[6].

1.6.2.2 Link-state Algorithms

When applying link-state algorithms, each node uses as its fundamental data a map of the network in the form of a graph. To produce this, each node floods the entire network with information about what other nodes it can connect to, and each node then independently assembles this information into a map. Using this map, each router then independently determines the least-cost path from itself to every other node using a standard shortest paths algorithm such as Dijkstra's algorithm. The result is a tree rooted at the current node such that the path through the tree from the root to any other node is the least-cost path to that node. This tree then serves to construct the routing table, which specifies the best next hop to get from the current node to any other node [7]. The link state algorithm used to solve the routing problems, in our methodology the link state constraints will be used to find the best path.

1.6.2.3 Path Vector Protocol

Distance vector and link state routing are both intra-domain routing protocols. They are used inside an autonomous system, but not between autonomous systems. Both of these routing protocols become intractable in large networks and cannot be used in Inter-domain routing. Distance vector routing is subject to instability if there are more than few hops in the domain. Link state routing needs huge amount of resources to calculate routing tables. It

also creates heavy traffic because of flooding. Path vector routing is used for inter-domain routing. It is similar to Distance vector routing. In path vector routing we assume there is one node (there can be many) in each autonomous system which acts on behalf of the entire autonomous system. This node is called the speaker node. The speaker node creates a routing table and advertises it to neighboring speaker nodes in neighboring autonomous systems. The idea is the same as Distance vector routing except that only speaker nodes in each autonomous system can communicate with each other. The speaker node advertises the path, not the metric of the nodes, in its autonomous system or other autonomous systems [8].

In our methodology a hybrid routing protocol will be used to find the best path, this hybrid routing protocol contains the link state and path vector dynamic routing protocol.

1.6.2.4 Comparison of Routing Algorithms

Distance-vector routing protocols are simple and efficient in small networks, and require little, if any management. However, naïve distance-vector algorithms do not scale well (due to the count-to-infinity problem), and have poor convergence properties, which has led to the development of more complex but more scalable algorithms for use in large networks, such as link-state routing protocols and loop-free distance-vector protocols. Loop-free distance-vector protocols are as robust and manageable as distance-vector protocols, while avoiding counting to infinity and hence having good worst-case convergence times. The primary advantage of link-state routing is that it reacts more quickly, and in a bounded amount of time, to connectivity changes. Also, the link-state packets that are sent over the network are smaller than the packets used in distance-vector routing. Distance-vector routing requires a node's entire routing table to be transmitted, while in link-state routing only information about the node's immediate neighbors are transmitted. Therefore, these packets are small enough that they do not use network resources to any significant degree. The primary disadvantage of link-state routing is that it requires more storage and more computing to run than distance-vector routing, in the last years GA was used intensively to modify Routing protocols to enhance the performance and the efficacy of routing operations, because GA has the ability to collect the principles of different routing protocols[1].

1.6.3 Path Selection

A **routing metric** is a value used by a routing algorithm to determine whether one route should perform better than another. Metrics can cover such information as bandwidth, delay, hop count, path cost, load, reliability, and communication cost. The routing table stores only the best possible routes, while link-state or topological databases may store all other information as well. As a routing metric is specific to a given routing protocol, multi-protocol routers must use some external heuristic in order to select between routes learned from different routing protocols. Cisco's routers, for example, attribute a value known as the administrative distance to each route, where smaller administrative distances indicate routes learned from a supposedly more reliable protocol.

1.6.4 Routing Algorithms

Routing algorithms can be differentiated based on several key characteristics. First, the particular goals of the algorithm designer affect the operation of the resulting routing protocol. Second, various types of routing algorithms exist, and each algorithm has a different impact on network and router resources. Finally, routing algorithms use a variety of metrics that affect calculation of optimal routes. The following sections analyze these routing algorithm attributes.

Design Goals

Routing algorithms often have one or more of the following design goals:

- Optimality
- Simplicity and low overhead
- Robustness and stability
- Rapid convergence
- Flexibility

1.6.5 Routing Algorithm Types

Routing algorithms can be classified by type. Key differentiators include these:

- Static versus dynamic
- Single-path versus multi-path
- Flat versus hierarchical
- Host-intelligent versus router-intelligent
- Intra-domain versus inter-domain
- Link-state versus distance vector

1.6.6 Network Protocols

Routed protocols are transported by routing protocols across an inter-network. In general, routed protocols in this context also are referred to as network protocols. These network protocols perform a variety of functions required for communication between user applications in source and destination devices, and these functions can differ widely among protocol suites. Network protocols occur at the upper five layers of the OSI reference model: the network layer, the transport layer, the session layer, the presentation layer, and the application layer. Confusion about the terms *routed protocol* and *routing protocol* is common. Routed protocols are protocols that are routed over an inter-network. Examples of such protocols are the Internet Protocol (IP), DECnet, AppleTalk, Novell NetWare, OSI, Banyan VINES, and Xerox Network System (XNS). Routing protocols, on the other hand, are protocols that implement routing algorithms. Put simply, routing protocols are used by intermediate systems to build tables used in determining path selection of routed protocols. Examples of these protocols include Interior Gateway Routing Protocol (IGRP), Enhanced Interior Gateway Routing Protocol (Enhanced IGRP), Open Shortest Path First (OSPF), Exterior Gateway Protocol (EGP), Border Gateway Protocol (BGP), Intermediate System-to-Intermediate System (IS-IS), and Routing Information Protocol (RIP) [9].

1.7 Routing Security

In mean time there is a high security concerns related to different routing operation that we should focus on to secure advanced computer networks. One of the important

mechanisms is to integrate GA with routing protocols to decrease the security level by securing the optimal path from hijacking sessions attacks and different link attacks. For example if there is a computer networks witch use OSPF as a routing protocol the hacker can determine the best path by doing simple calculations to attacks important data on computer networks in our approach by using GA we decrease the risk possibility based on the most important feature of GA that calculate in random way based on fitness function. In n other words no hackers can know the best path unless the hackers determine the fitness function and the used random generated numbers.

Chapter Two

Literature Study

2.1 Introduction

In this chapter a surveying of some related work will be explored. Due to the importance of those related work, we will the necessary aspects and try to comments on that. Also we will indicate the importance and the relation of such works.

2.2 Using GA to solve routing problem

One of the most important and related work is “Nilanjan Banerjee et al [10]”. In this work, they tried to solve the Routing and Wavelength Assignment problem in Wavelength Division Multiplexing (WDM) network using GA. The problem definition in this work is by given a set of source – destination pairs and the aim is to minimize the number of wavelength needed to support the given set of light paths for the SD pairs.

The design parameters of this work are:

1. A physical topology of an optical network which is a graph $G=(V, E)$ where V is the number of vertices of the graph and E refers to the number of edges in the network.

2. A set $S = \left\{ \begin{pmatrix} i \\ j \end{pmatrix} \mid i \text{ is a source and } j \text{ is a destination} \right\}$.

Which problem is subject to the following constraints

1. Wavelength Continuity Constraint.
2. Wavelength Conflict Constraint.

A hybrid approach is used for the initialization of the population. For every source-destination pair the k -shortest paths connecting them are evaluated using Yen’s algorithm. Each gene in a chromosome represents one of the shortest paths selected randomly. Every gene in the chromosome has a pointer to an entry in a look up table which contains the actual path. Thus a single chromosome contains a set of plausible paths for all the source-destination pairs.

In the single objective formalization of this problem they associate a cost with every chromosome, a simple single objective fitness based genetic algorithm is used for the optimization.

The GA in this work has the following salient features:

1. Crossover.
2. Mutation.

The GA finds the individuals with the least cost function, which is found to be proportional to the number of the wavelengths used in the network. The wavelength assignment to the fittest individuals is done using Brelaz heuristic.

They compare the results with the well-known First-Fit heuristic which has been used for solving the static RWA problem. Here, they underline the salient features of the algorithm:

1. Initially a lower bound on the number of wavelength is calculated.
2. W Copies of the network graph are made.
3. The first graph is taken and a path is searched for the first SD pair in the graph. Consequently the edges in the graph corresponding to the path are removed.
4. A path is searched for the next SD pair in the first graph. If it cannot be found then the next graph is considered and Step 3 is repeated.
5. Step 3 and 4 are repeated till no path can be found in any of the set of graphs.
6. W Is incremented by 1 and the above steps are repeated till paths have been found for all the SD pairs.
7. Output.

The design inputs are:

1. Maximum number of wavelengths per fiber.
2. Physical topology.
3. Distance Matrix.
4. Number of transmitters at node.
5. Traffic matrix.
6. Additional assumption is: the packet interarrival durations at node and the packet lengths are exponentially distributed.
7. Capacity of each channel.

And the design variables are:

1. Virtual Topology.
2. Traffic Routing.
3. Physical Topology Route.
4. Wavelength color.

The two objective functions that they need to minimize simultaneously for a given set of source-destination pairs are:

1. Delay Minimization.
2. Total number of wavelengths for the network.

In the results they evaluate the effectiveness of the proposed algorithm by extensive simulation. The simulation networks considered are real life existing networks like the 20 node ARPA network, 18 node European optical networks (EON), 22 node UK network and 14 node NSF network.

The simulation results are presented for both the single objective and the multiobjective formalization of the problem for the above networks.

For the multi-objective case they tested MOEA on the same networks as in the single objective case. Synthetic and matrices were generated considering Poisson's distribution of arrival of packets for the traffic. The distance matrices were formulated using the real distances between the cities in the network. Both the matrices are symmetric matrices. The above research paper discussed a single and multiobjective formalization of the Static RWA problem in WDM networks and solved it using Evolutionary algorithms. The results in the single objective show that they are comparable to solutions obtained by existing heuristics like the first-fit algorithm. In fact, the solutions obtained are superior when the size of the input is increased. The multi-objective formalization is new where the Average Delay and the number of wavelengths are simultaneously minimized to obtain optimal paths between various SD pairs. The two objectives to be optimized have not been combined into one and hence the general nature of the solution is maintained. In most multiobjective optimization problems it is crucial that the obtained solution is diverse. In this work we find that the GA obtains a good diversity. Thus a virtual topology designer having a range of wavelengths and delays in mind can examine several topologies and can

choose the one that best matches his requirement and other engineering considerations. This is the main advantage of using EAs for such a NP-hard problem. In the above work genetic algorithm is used to solve the Routing and Wavelength Assignment problem in WDM network. In our work genetic algorithm and adaptive genetic algorithm will be used to find the best optimal solution that can solve the congestion problem and the hacking problem.

2.3 Using Random Key-based GA

Mitsuo Gen et al [11] used another approach for finding the shortest path routing problem; this research paper considers a SPR problem with a negative cycle that is an NP-complete problem. Observe that for any network containing a negative cycle (W), the linear programming formulation has an unbounded solution because we can send an infinite amount of flow along (W).

Random key encoding is a powerful method to represent permutations, particularly, because there is no infeasibility problem to deal with - traditional crossover operators produces only feasible offspring. Moreover, relative and absolute ordering information can be preserved after recombination. An example of generated chromosome and its decoded path is shown in Figure (2-1), for the undirected network shown in Figure (2-2). At the beginning, we try to find a node for the position next to source node 1. Nodes 2, 3 and 4 are eligible for the position, which can be easily fixed according to adjacent relation among nodes. The priorities of them are 1.24, 6.68 and 4.74, respectively. Node 3 has the highest priority and is put into the path. The possible nodes next to node 3 are nodes 4 and 6. Because node 4 has the largest priority value, it is put into the path. Then we form the set of nodes available for next position and select the one with the highest priority among them. Repeat these steps until a complete path is obtained.

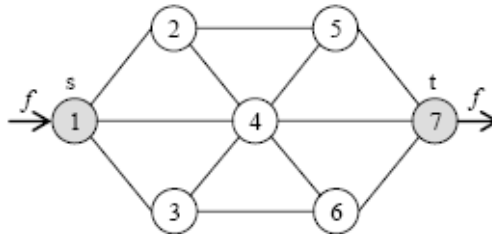


Figure (2-1): A simple undirected network with 7 nodes and 12 edges

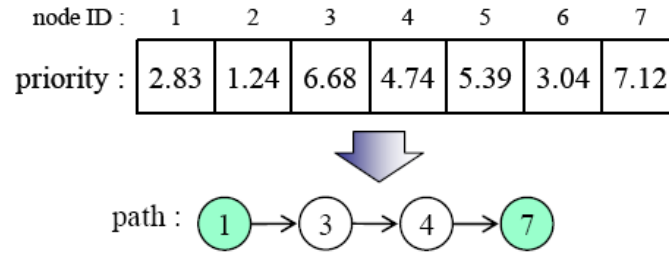


Figure (2-2): Example of generated chromosome and its decoded path

The basic concept of this kind of operator is borrowed from the convex set theory. Selects two positions at random and then swaps the gene on these positions. In this paper, we adopt swap mutation for generating various offspring.

The algorithm is modified to:

1. include immigration routine, in each generation,
2. generate,
3. evaluate $popSize \cdot \mu$ random members,
4. replace the $popSize \cdot \mu$ worst members of the population with the $popSize \cdot \mu$ random members (μ , called the immigration probability)

Each test problem was run by 20 times for each GA approach. Each test problem is divided into several numerical experiments to investigate the effects with different GA parameter setting and maximum generation $maxGen=1000$; Immigration rate, $\mu = 0.15$ was employed.

This approach has a higher search capability that enhanced rate of reaching optimal solutions and improve computation time than other GA approaches using different genetic representation methods.

In our work Hybrid Routing Protocol (link state and distance vector) will be used to find the best path Using GA and AGA.

2.4 Using Adaptive Fitness Function

A third approach for solving routing problem is using fitness function. Eiben et al [12] describe a problem independent method for treating constraints in an evolutionary algorithm. Technically, this method amounts to changing the definition of the fitness function during a run of an Evolutionary Algorithms (EAs), based on feedback from the

search process. They illustrate the power of the method on different constraint satisfaction problems and point out other application areas of this technique.

The common opinion about (EAs) is that they are good optimizers, but cannot handle constraints well. The opinion in this work is based on the observation that the variation operators, mutation and recombination, are "blind" to constraints. In other words, if the parents satisfy certain constraints the offspring Obtained by mutation and/or recombination might violate them. In the constrain problem. A natural classification of problems can be found in. This classification distinguishes free optimization problem, where no constraints are present, and constraint satisfaction and constrained optimization problems that do have constraints to be satisfied.

A Free Optimization Problem (FOP) is a pair $\langle S, F \rangle$, where S is a free search space and F is a (real valued) objective function on S , which has to be minimized. A solution of a free optimization problem is a $(s \in S)$ with an optimal (minimal) f - value. A Constrained Optimization Problem (COP) is a triple $\langle S, F, \Phi \rangle$, where S is a free search space, F is a (real valued) objective function on S and Φ is a formula (Boolean function on S). A solution of a constrained optimization problem is an $s \in S$ with $\Phi(s) = \text{true}$ and optimal F -value. A Constraint Satisfaction Problem (CSP) is a pair $\langle S, \Phi \rangle$, where S is a free search space and (Φ) is a formula (Boolean function on S). A solution of a constraint satisfaction problem is an $s \in S$ with $(\Phi) = \text{true}$. Usually Φ is called the feasibility condition, and it is defined by a number of constraints (relations) C_1, \dots, C_m on the domain, that is the formula Φ is the conjunction of the given constraints. Satisfying the constraints means finding an instantiation of variables v_1, \dots, v_n within the domains D_1, \dots, D_n such that the relations C_1, \dots, C_m hold.

Solving a CSP means finding one feasible element of the search space, solving a COP means finding a feasible and optimal element.

For both case the commonly listed options for treating this problem are the following

- 1. Eliminating** infeasible individuals / chromosomes.
- 2. Penalizing** infeasible individuals / chromosomes.
- 3. Repairing** infeasible individuals / chromosomes.

4. Special variation operators preserving the feasibility of the parents.

5. Special representation / decoding such that chromosomes always stand for feasible individuals.

The SAW-ing mechanism has been applied to various constraint satisfaction problems

In this research the following findings as most important.

1. A small population size, counterintuitive as it may seem, happens to work very well on the problems that have been tested.
2. Second is the insensitivity that SAW-ing has to its parameters T_p and $\Delta\omega$. This insensitivity has been found in experiments on graph-coloring and satisfy-ability.

In our work adaptive genetic algorithm will be used to find the best path so adaptive fitness function will be used to solve this problem.

2.5 Using GA to Finding Shortest Path

Bilal Gonen [13] try to find a better approach for finding the shortest path in a network. He uses GA to solve this problem.

Routing is a fundamental engineering task on the Internet. It consists in finding a path from a source to a destination host. Routing is complex in large networks because of the many potential intermediate destinations a packet might traverse before reaching its destination.

The steps of the GA are explained below:

1. When initializing the population, the algorithm starts from the SOURCE.
SOURCE is a constant in the program, so the user may want to pick another node as the starting point. The algorithm selects one of the neighbors provided that it has not been picked before. It keeps doing this operation until it reaches to DESTINATION. Like SOURCE, DESTINATION is also a constant that user may change as they wish. In this operation. The program got stuck several times in some nodes which has no unvisited neighbor. In that case, ignore that path, and start from the source again.
2. The evaluation function takes a path in the population. It gets the distance between each node pair in the path, by calling a function to read from the distance array. Adds them together and returns the sum as the cost of the path.

3. The algorithm selects two individuals from the population with the lowest costs.
4. With some probability, the program mates the two individuals. The crossover function is shown in Figure (2-3), takes two parents to mate. It looks for the common points in the parents. The common nodes are where these two paths intersect. Among the common points, the program selects one of them randomly. It makes the crossover from that point.

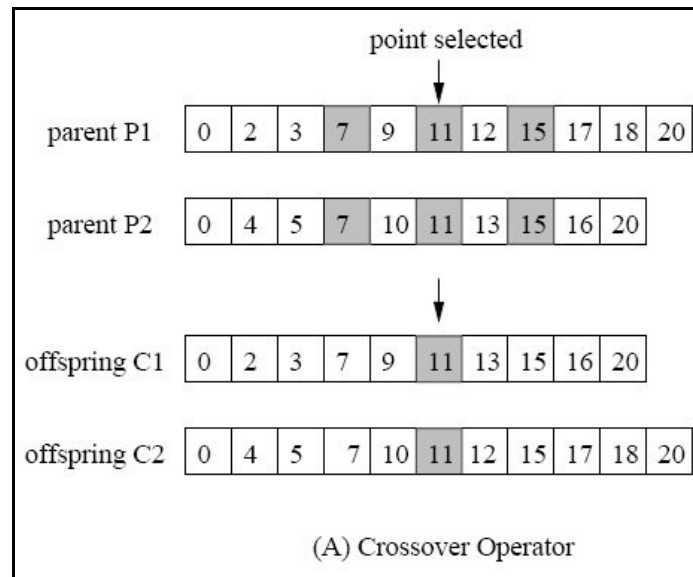


Figure (2-3): Crossover Operator

These offspring will be sent to the evaluation function to get their fitnesses. If the offsprings' fitnesses are less than the nodes with maximum fitnesses in the population, they replace them with the nodes with the maximum fitness's. The terminating condition is a predefined number of iterations. There reason is that in the network topology, the goal is not to find the global optimum, but to find a path with a reasonable cost in a limited time. The results of this work can be illustrated below:

He generated a network topology with 20 nodes and 62 links to test my Genetic Algorithm. Each link has a cost associated with them. He set two nodes as source and destination. The goal of his GA application is to find a path between source and destination with the lowest cost. They set several parameters for the experiment. They are as follows;

Population size = 50
 Number of runs = 30
 Number of generations = 50
 Crossover probability = 0.99
 Mutation probability = 0.1

They run the steps selection, crossover, and replace part 50 times (number of generations). The numbers below shows the average of maximum numbers of 30 runs, the average of minimum numbers of 30 runs, and the average of average numbers of 30 runs.

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

In the above work he try to find a better approach for finding the shortest path in a network using genetic algorithm, in our work genetic algorithm and adaptive genetic algorithm will be used not to find the shortest path but to find the best path or to find the best optimal solution that can solve the congestion problems and avoid the hacking problems.

2.6 Conclusions of Related Works

From the previous related works, GA is used to find the best path in advanced wide area computer networks, in our work we will try to find best path by using AGA and comparing the archived results with GA in the computer network, by suggesting using hybrid dynamic routing protocol. To find the best path, here we take the principles of distance vector dynamic protocol and link state dynamic protocol.

Chapter Three

Methodology

3.1 Introduction

Telecommunications networks technologies are becoming faster and more complex to handle different types of traffic and there is a varieties of problems related to traffic load utilization, the main objective of the research to find best path using hybrid dynamic routing protocol that combines between the principles of link state routing protocol (cost) and distances vector routing protocol (hope count), and the transmission speed between routers in advanced WAN topologies based on GA and AGA.

In this chapter we focus on the methodology. Here we will explain in details the package that helps us in solving the problem. MATLAB is a powerful package that helps us solving many scientific problems. Then we will explain in details the code that we used by writing a simple pseudo-code and a flowchart explaining the procedure.

3.2 Motivations

To find the shortest path in routing, many algorithms are used like Open Shortest Path First (OSPF) also Djakstra Algorithm and other algorithms. All of the previous algorithms find the shortest path depending on low cost path. Here the problem of congestion (high traffic) on the shortest path occurred because all packets transfer using this shortest path , and also the Hackers may be able to compute what is the shortest path, then easily they can hacking the packets so we must find an alternative solution to find another path to transfer the packets. This path called the best path not the shortest path, the best path can be found by using:

- GA and AGA to find the best solution.
- Hybrid Routing Protocol (link state and distance vector).

3.3 Genetic Algorithms and Adaptive Genetic Algorithms

Genetic Algorithms have been used in science and engineering as adaptive algorithms for solving practical problems and as computational models of natural evolutionary systems. This brief accessible introduction describes some of the most

interesting research in the field and also enables readers to implement and experiment with genetic algorithms on their own. It focuses in depth on a small set of important and interesting topics particularly in machine learning, scientific modeling, and artificial life and reviews a broad span of research, including the work of Mitchell and her colleagues. The descriptions of applications and modeling projects stretch beyond the strict boundaries of computer science to include dynamical systems theory, game theory, molecular biology, evolutionary biology, and population genetics. Adaptive Genetic Algorithms (AGA) is to automatically and dynamically perform an auto-conguration of GA-parameters which are considered to have the highest impact on solution quality: crossover, mutation and selection operator. Within AGA not only information on the solution itself is represented in the chromosomes but also information on the parameterization, the so-called environment, which was applied in the generation of this chromosome, is coded and submitted to the competition process [14].

3.3.1 Why using AGA?

The suggested AGA let the crossover and mutation increase rate and optimize GA, it's greatly decreases the workload for iterative debugging the corresponding parameters, AGA has the following characters:

1. Solve nonlinear programming (NLP) problems with equality and inequality constraints.
2. Entropy-based searching technique with narrowing down space is taken to speed up the convergence.
3. A specific strategy of reserving the most fitness member with evolutionary historic information is effectively used to approximate the solution of the nonlinear programming problems to the global optimization.
4. A new adaptive strategy is employed to overcome the difficulty in confirming the genetic parameters.
5. A new iteration scheme is used in conjunction with multi-population genetic strategy to terminate the evolution procedure appropriately.

Redefining the fitness function happens by adding a value $\Delta\omega$ to the weights of those constraints that are violated by the best individual at the end of each period of fitness evaluations.

Set initial weights (thus fitness function f)

While not termination do

For the next fitness evaluations do

Let GA go with this f

End **for**

Redefine f and recalculate fitness of individuals

End **while**

3.4 Manchester Encoding

Manchester encoding is a form of digital encoding in which data bits is represented by transitions from one logical state to the other. When the Manchester code is used, the length of each data bit is set by default. This makes the signal self-clocking. The state of a bit is determined according to the direction of the transition. In some systems, the transition from low to high represents logic 1, and the transition from high to low represents logic 0. In other systems, the transition from low to high represents logic 0, and the transition from high to low represents logic 1 [15].

3.5 MATLAB

MATLAB is a high-performance language for technical computing. It integrates computation, visualization, and programming in an easy-to-use environment where problems and solutions are expressed in familiar mathematical notation. Typical uses include Math and computation Algorithm development Data acquisition Modeling, simulation, and prototyping, Data analysis, exploration, and visualization Scientific and engineering graphics Application development, including graphical user interface Building MATLAB is an interactive system whose basic data element is an array that does not require dimensioning. This allows us to solve many technical computing problems, especially those with matrix and vector formulations, in a fraction of the time it would take

to write a program in a scalar no interactive language such as C or FORTRAN. The name MATLAB stands for matrix laboratory [16].

3.5.1 MATLAB System

The MATLAB system consists of several parts: Desktop Tools and Development Environment. This is the set of tools and facilities that help us to use MATLAB functions and files. Many of these tools are graphical user interfaces. It includes the MATLAB desktop and Command Window, a command history, an editor and debugger, a code analyzer and other reports, and browsers for viewing help, the workspace, files, and the search path. The MATLAB Mathematical Function Library, this is a vast collection of computational algorithms ranging from elementary Functions, like sum, sine, cosine, and complex arithmetic, to more sophisticated functions like matrix inverse, matrix Eigen values, Bessel functions, and fast Fourier transforms. This is a high-level matrix/array language with control flow statements, functions, data structures, input/output, and object-oriented programming features. It allows both "programming in the small" to rapidly create quick and dirty throw-away programs, and "programming in the large" to create large and complex application programs. Graphics, MATLAB has extensive facilities for displaying vectors and matrices as graphs, as well as annotating and printing these graphs. It includes high-level functions for two-dimensional and three-dimensional data visualization, image Processing, animation, and presentation graphics. It also includes low-level functions that allow us to fully customize the appearance of graphics as well as to build complete graphical user interfaces on your MATLAB applications. The MATLAB External Interfaces/API, this is a library that allows us to write C and FORTRAN programs that interact with MATLAB. It includes facilities for calling routines from MATLAB (dynamic linking), calling MATLAB as a computational engine, and for reading and writing MAT-files [16].

3.6 The Method

In this work we tried to find best path based on least cost and least hope counts and least decreasing transmission based on GA and AGA.

Presented below the suggested steps to find best path:

1. Path initializing and searching.
2. Finding the fitness value for each of the paths to find the best that its fitness value approximately equal to zero.
3. Reproduction and data selection then crossover and mutation.
4. Adaptive fitness to calculate the fitness values for each of the new individual in the generation. But according to the bellow equations (3-1) and (3-2).
5. We repeat the above two steps agent until we accede the no. of generations or the average fitness is greater than 0.9.

3.6.1 Initializing path searching mechanism

For our first step we will generate two matrices, the first is based formed from a seed random generated number with a zero diagonal and an other matrix that will be formed form assigning each cell within matrix one of the three values [64,128 or 256] also the diagonal will be zero to avoid assign cost and speed to node it self.

Now to represent the Speed S , if the value change from higher to lower speed or no change occur it gives zero value else it gives 1 value [14] , then we calculate the number of ones to represent the Speed value S because the summation of ones indicate the path speed acceleration.

Next step is to apply the path searching mechanism to find all possible paths, based on finding the next node in path and the cost that connect two nodes, if the cost is small so this path will be taken as in the following representation.

Table (3-1): cost matrix

	1	2	3	4	5	6
1	0	5	9	4	1	0
2	2	0	7	9	2	7
3	6	8	0	1	2	4
4	5	4	4	0	6	9
5	9	6	9	8	0	5
6	8	8	9	0	2	0

Table (3-2): Speed matrix

	1	2	3	4	5	6
1	0	256	128	256	128	128
2	64	0	64	64	128	256
3	256	256	0	128	64	64
4	128	64	256	0	128	64
5	256	64	64	256	0	64
6	256	128	64	256	64	0

We can represent the above condition with the following finite state diagram, as shown in figure (3-1).

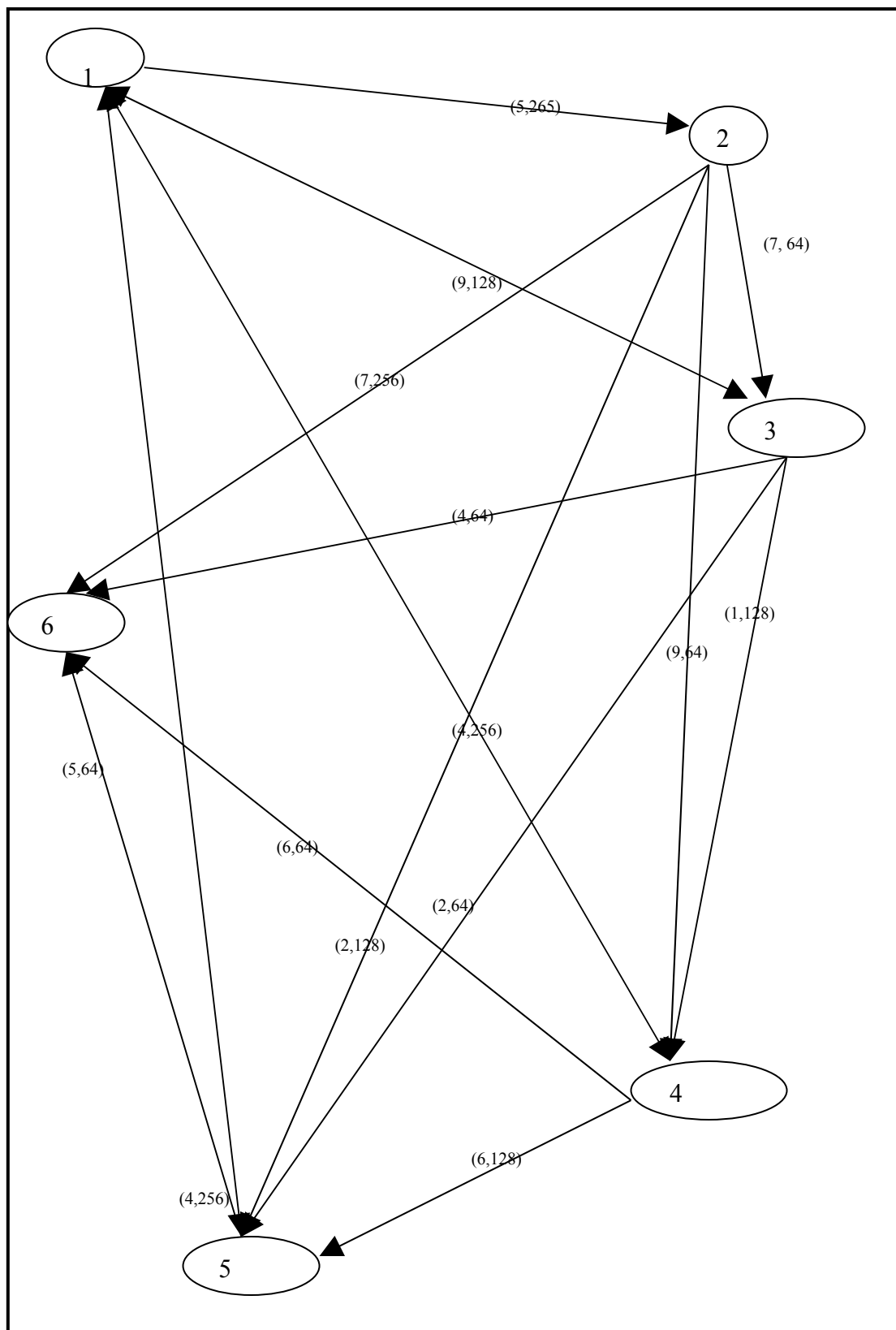


Figure (3-1): diagram of all possible paths with their cost and speed

By searching for all the possible paths we will have a list for all the paths with fitness function as in figure (3-11). The algorithm starts by searching the array row that is equal to the starting node.

Suppose that the source is node 0 and the destination is node 4. Our method can represent the result as in table (3-3).

Table (3-3): shows the GA initial population

X	Paths	No. of nodes (n)	Path cost	Path speed	F(x)	Fitness
0	0 – 1 – 2 – 4	4	C1	S1	F(1)	F1
1	0 – 4	2	C2	S2	F(2)	F2
2	0 – 3 – 2 – 4	4	C3	S3	F(3)	F3

3.6.2 Finding the fitness value

The fitness value is a number that represents the fitness of that individual to the problem, i.e. if the value is high this means it is suitable for the solution else it is not, we find the fitness value by dividing each category over the total summation of that category then summing all things together as the following suggested equations

$$Fx = \frac{n}{\sum n} + \frac{\text{cost}}{\sum \text{cost}} + \frac{\text{speed}}{\sum \text{speed}} \dots\dots\dots (3.1)$$

$$\text{Fitness} = \frac{f(x)}{\max f(x)} \dots\dots\dots (3.2)$$

3.6.3 Reproduction and data selection

Reproduction can be achieved by applying total population. this process will eliminate the weakest individuals and preserve the good individuals based on comparison with seed random numbers, In this stage we will compare the fitness value with average fitness, if less it will be taken otherwise it will be discarded. The reproduction part starts by generating a seed random number, then we will compare the result number with each

individual fitness value fitness (F_x) if it is less than greater number it will be selected otherwise it will be discard for the next run.

3.6.4 Crossover and mutation

After we select the best individual we will apply crossover and mutation technique to each of the chilled individual. But first we must convert the numerical representation to binary in order to apply mutation and crossover.

First the data will be applied to crossover, then convert the information from decimal form to binary form in order to apply crossover properly, after the crossover where applied it will converted back to decimal.

Really here we are taking the path no. as a criterion for our vision so the mutation and crossover will reflect the selected path is the best by knowing the F_x value of the crossover individuals by choosing the individual with the maximum F_x .

After that the same reproduced individual will be mutated as follow:

1. Generate seed random number, if $>$ from different values (0.3, 0.5, 0.7)

Go to step 2 else no mutate

2. First we convert the path no. from decimal form to binary form.

3. We apply the following suggestion conditions.

4. For the first two paths we will mutate the first bit.

5. Between path no. 3 and path no. 8 we will mutate the second bit.

6. Between path no. 7 and path no. 16 we will mutate the third bit.

7. Between the path no. 15 and path no 32 we will mutate the fourth part.

8. Larger than 64 we will mutate the fifth path.

After we mutate the individuals we will search for the maximum fitness as the best paths. By comparing the results between the reproduction results and the cross over results a see which path was repeated in at lest twice in algorithm. The running will continue for a number of generations until it reaches the generation limits or the average fitness is greater than 0.9. Figure (3-2) shows the crossover operation.

Parent 1	0 0	0 0	0 0 1 0	Chiled 1
Parent 2	0 0	0 1	0 0 0 0	Chiled 3

Figure (3-2): crossover operation

3.6.5 Suggested Adaptive Fitness Function

After forming the new generation we will calculate fitness value according to the following suggestion equation as follow:

$$Fx = r0 \frac{n}{\sum n} + r1 \frac{\cos t}{\sum \cos t} + r2 \frac{speed}{\sum speed} \dots\dots\dots (3.3)$$

Here the three R values represent s a three seed random numbers less than 1.

3.7 Flow Charts

The flow charts of the methods are shown in figures below:

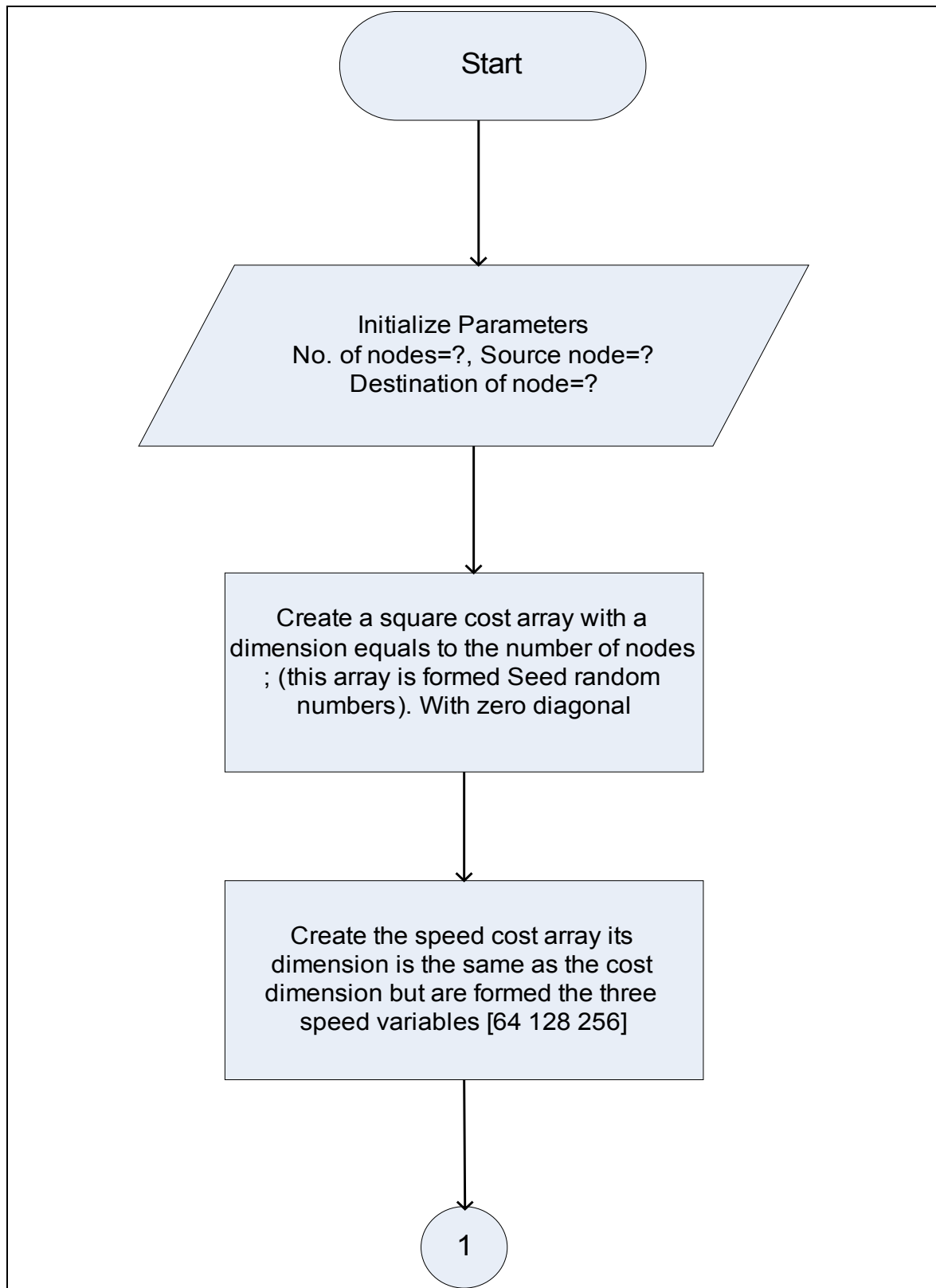


Fig (3- 3) is a flow chart explaining the stage of initializing the parameters

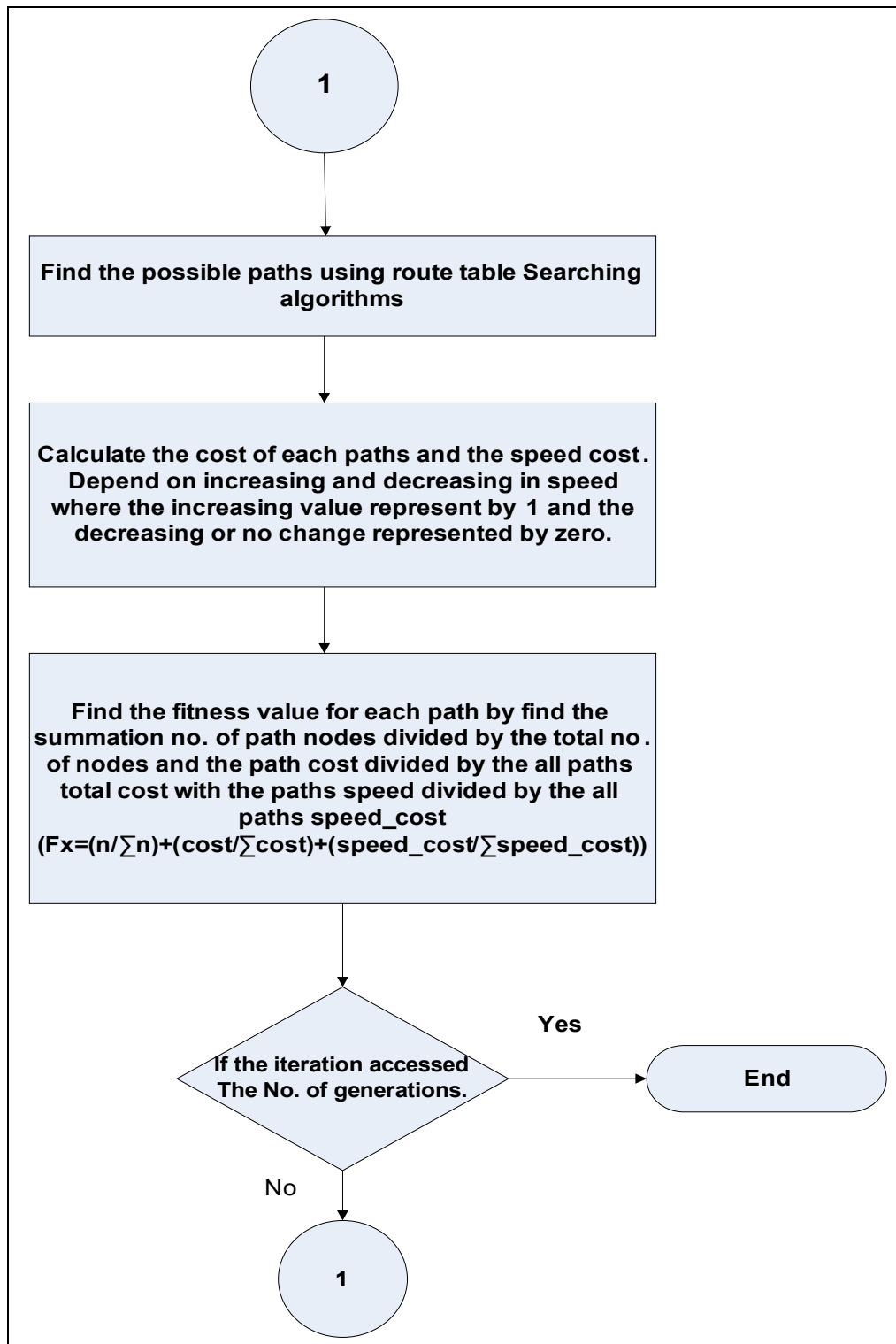


Fig (3- 4) a flow chart explaining the stage of Pre-GA stage

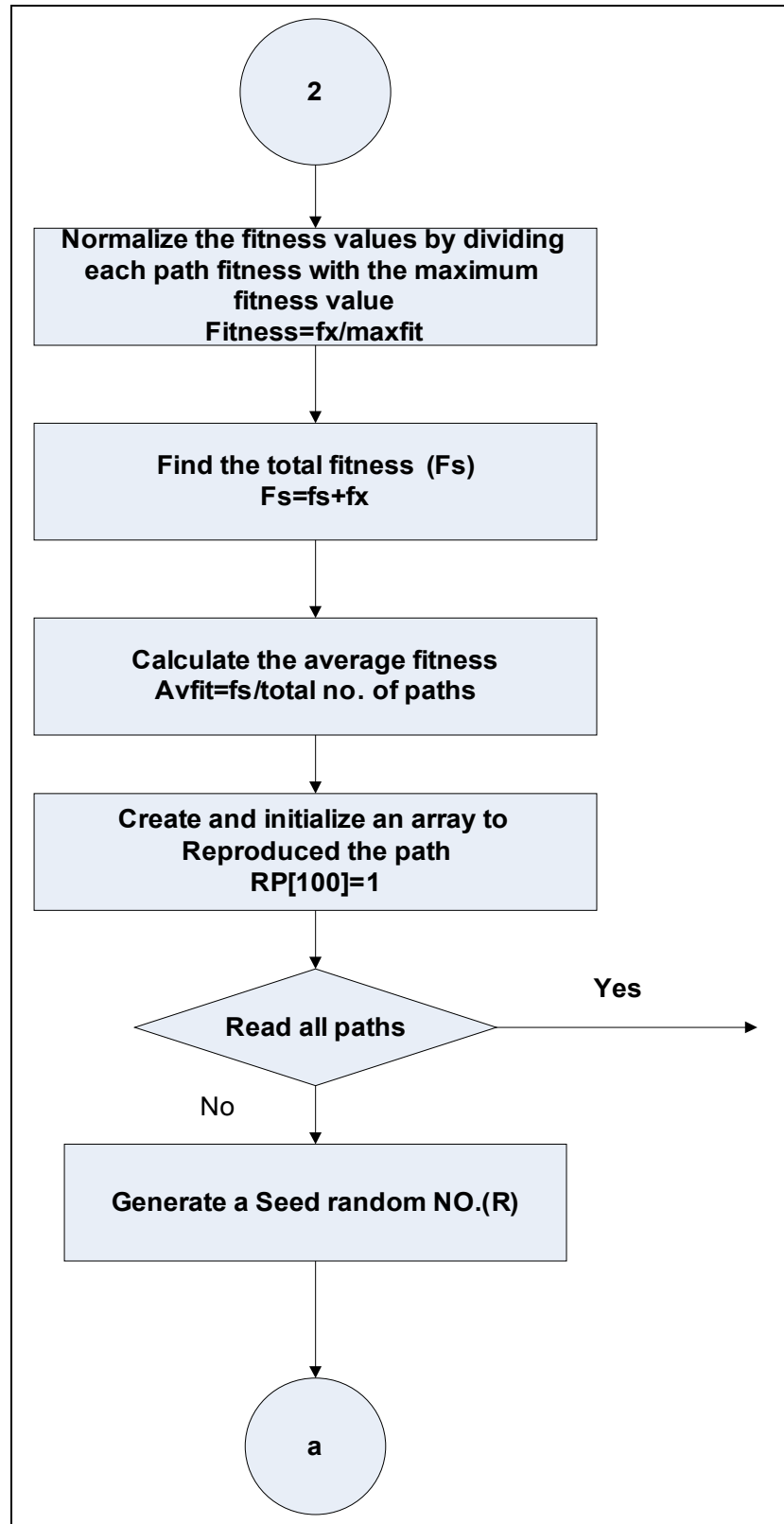


Fig (3- 5) is a flow chart explaining the stage Reproduction.

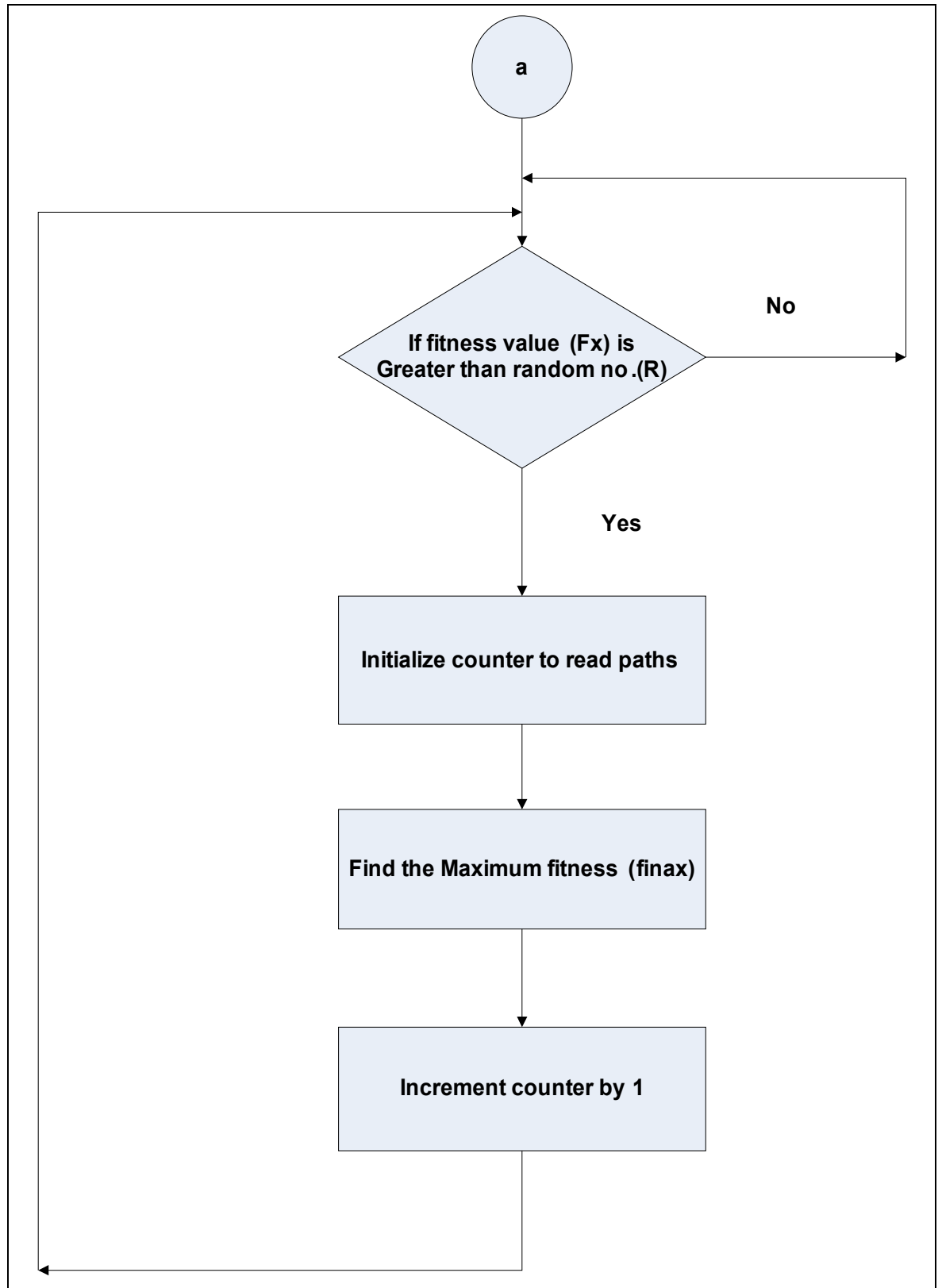


Fig.(3- 6) is a flow chart explaining the reproduction stage (continue).

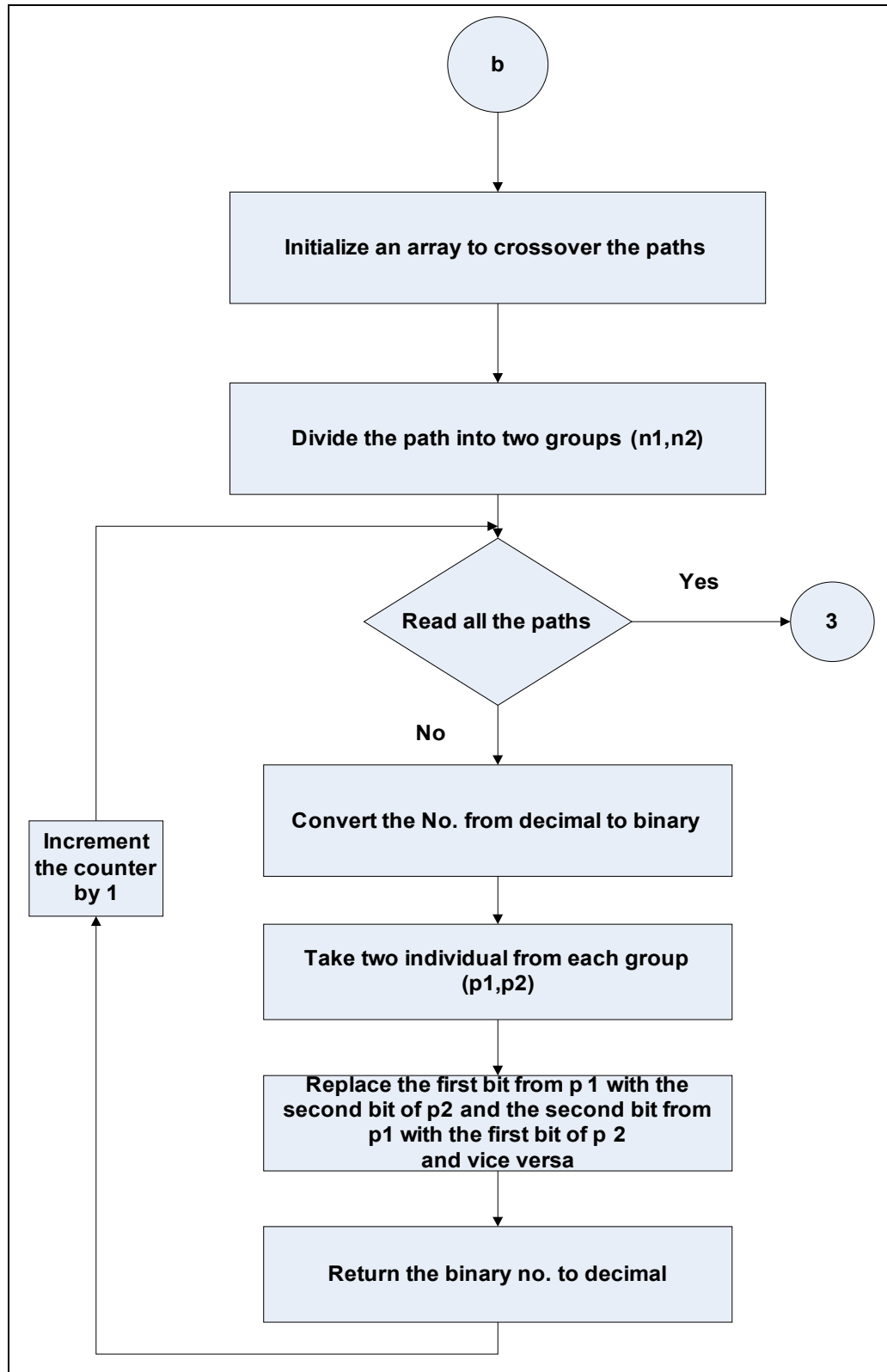


Fig (3- 7) is a flow chart explaining the crossover stage.

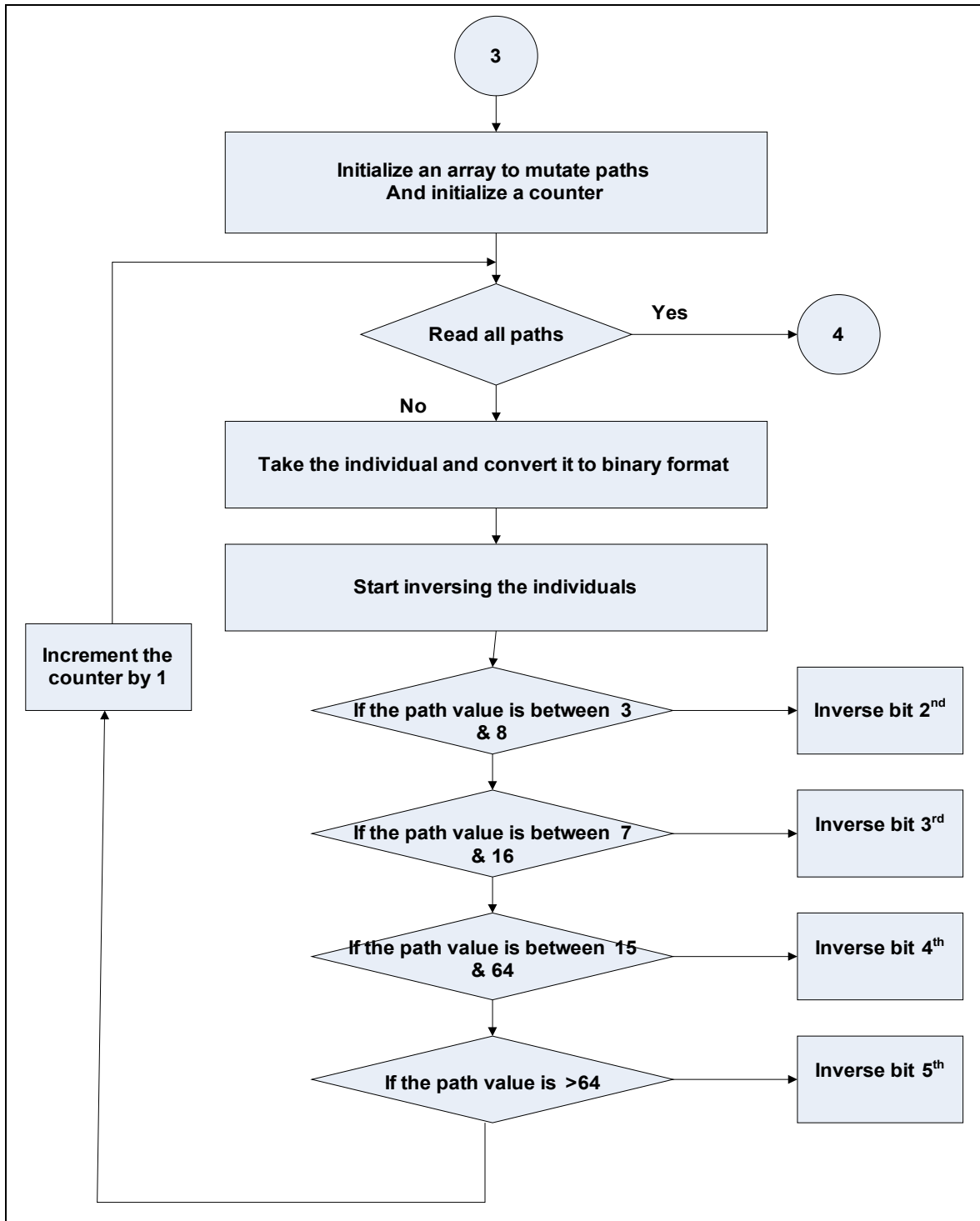


Fig (3- 8) is a flow chart explaining the crossover stage.

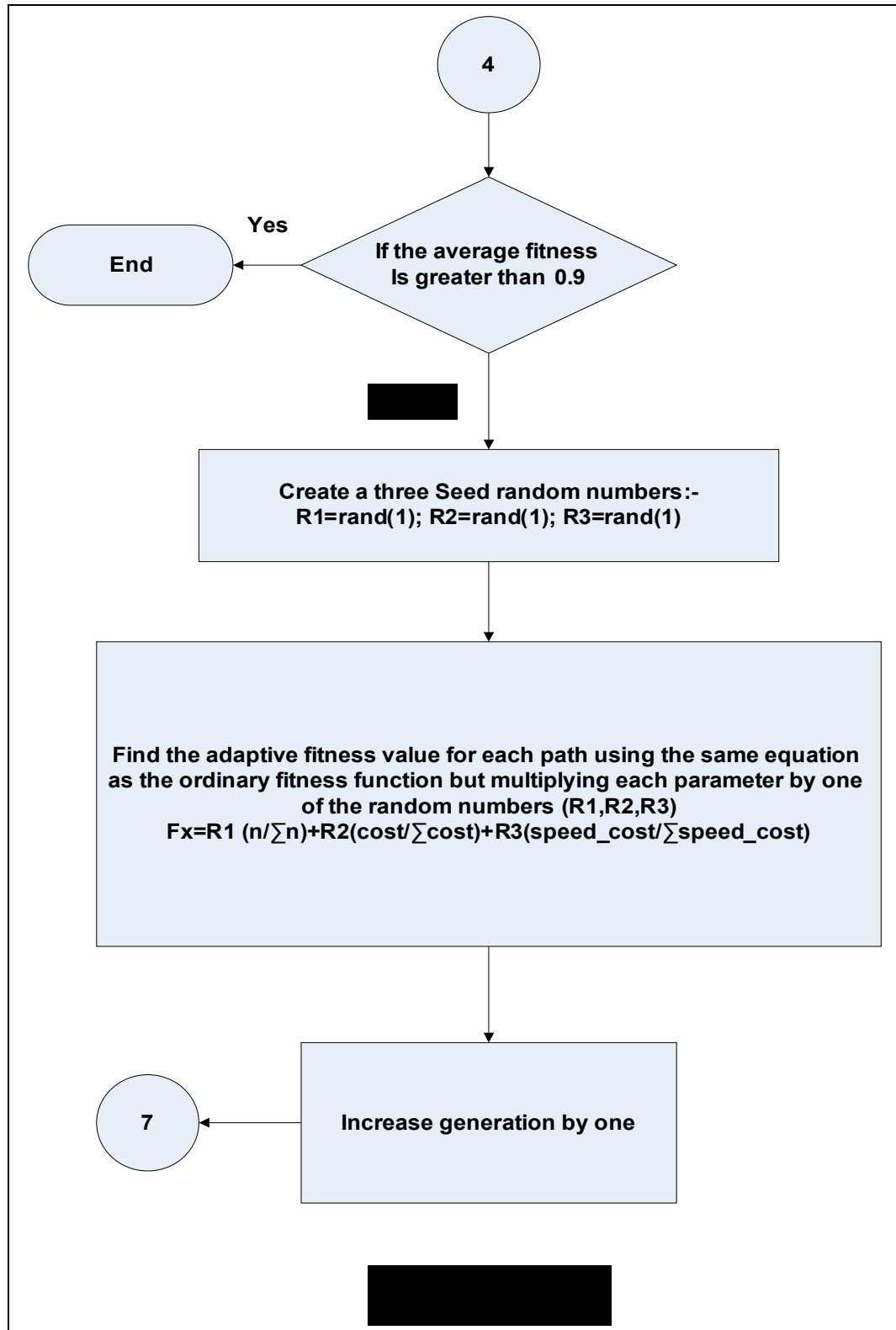


Fig (3- 9) is a flow chart explaining the adaptive fitness stage.

3.8 Flow Charts Description

In the above flow chart the data program are assigned to be used, number of node are assigned then source node and destination node, the program will create the cost matrix [node][node]; (this array are formed from seed random numbers with zero diagonal. Then the speed_cost matrix is created its dimension is the same as the cost matrix dimension but are formed from three speeds [64,128,256].

After that, the path routing techniques will start by search the all possible paths. This is done by starting with source node and jumping until reach destination node. For all possible paths the speed_cost is calculated by counting the amount in change is speed for each path. Then the fitness values are calculate for each path to begin GA part

The program will reproduce the paths, this is done by generating seed random numbers and comparing it with each fitness for each path. If the random is less than fitness, the path will selected and will be taken to the next stage.

Then we applied crossover on reproduce paths. The number of path must be converted to binary, then do crossover, after that the path number will converted back to decimal.

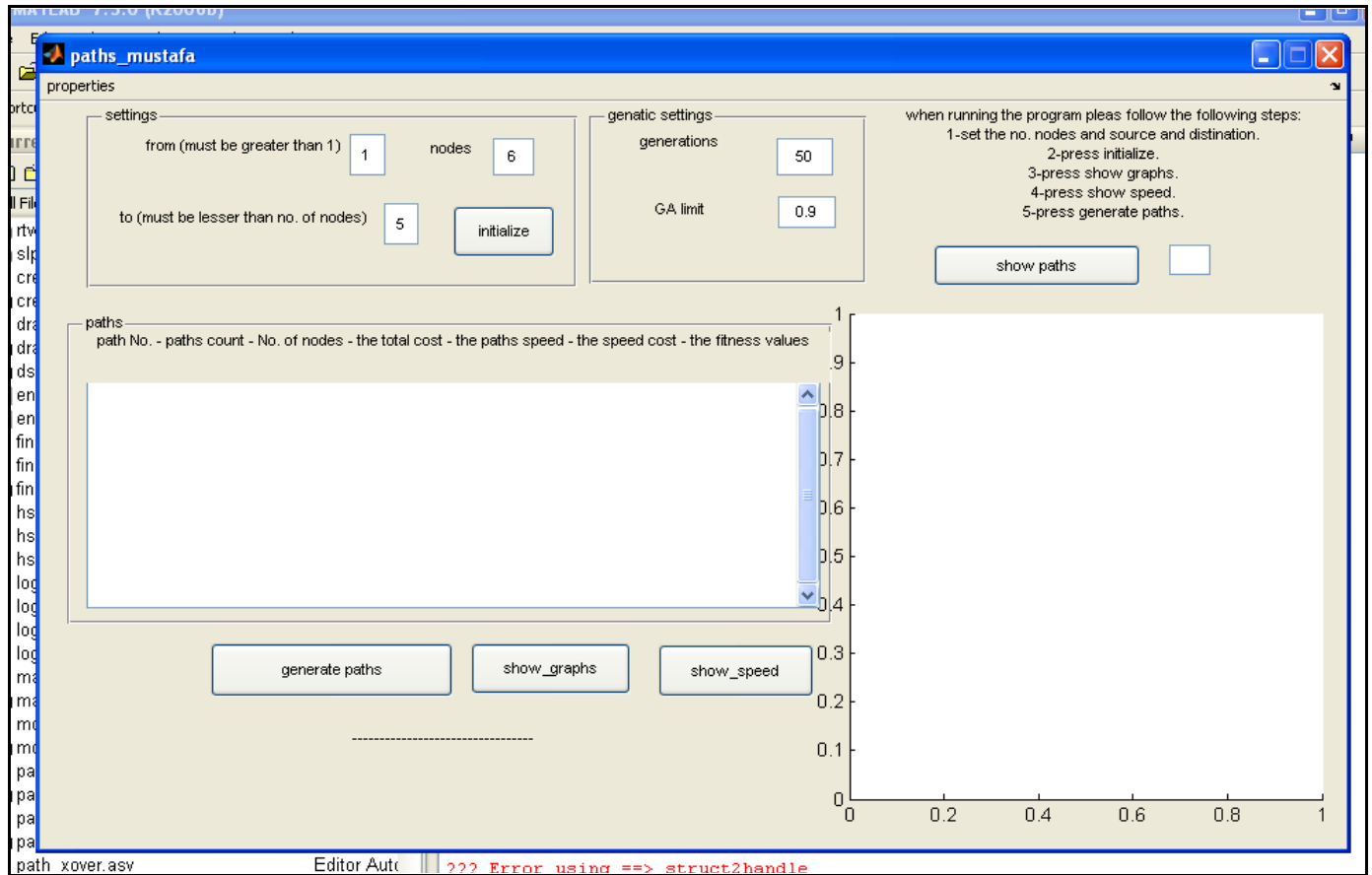
Now the program will applied mutation on the path, also first by converting the path number to binary and reading the path, based on number of path the bits will mutate after that the path number will converted back to decimal.

After finishing the GA for a normal fitness function, the same data set are applied to adaptive fitness function and the same procedure is done again, but here if the result fitness value for the path is less than the normal fitness the loop will break and returning the result.

3.9 Program User Interface

We build the simulation as simple user interface figure (3-10) that will facilitate our work. The interface is composed from three main parts:-

Figure (3-10): user interface



1. Setting parameters that will let the user input the trial parameters including the no. of nodes, the source and destinations also the no. of generation and the average fitness value to terminate.
2. Also a text area that we will display the results.
3. The third area that will draw the path by just entering the path no. and pressing on draw button to do so.

So first we will input the trail parameters and press initialize, then for conformation we will press on draw graph and draw speed graph to show the cost and the speed cost on the screen.

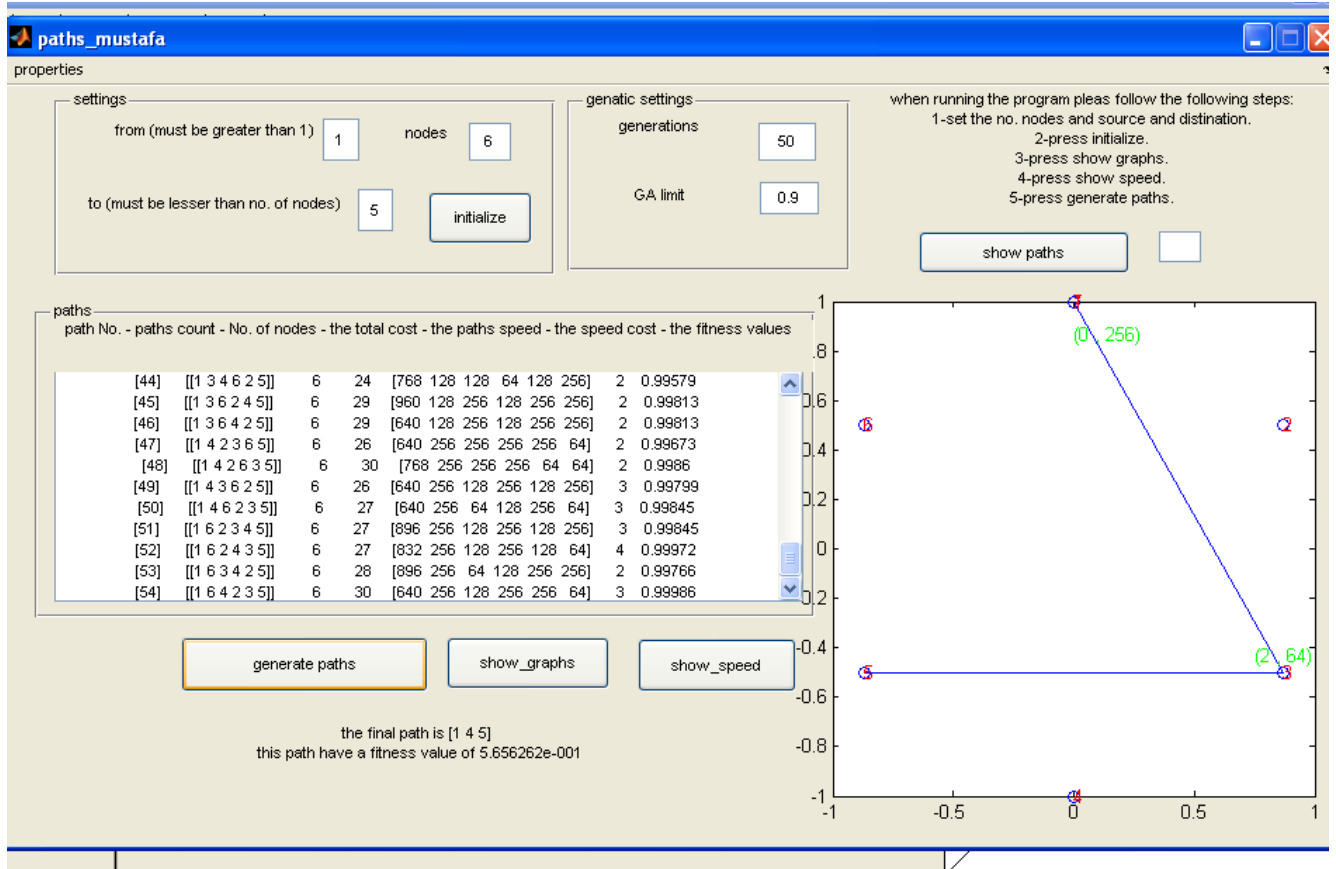


Figure (3-11): user interface in action.

Chapter Four

Results

4.1 Introduction

In this chapter we will preview and test the results under different conditions for example number of nodes, change in the source and destination values then comparing the output of standard genetic algorithm with adaptive genetic algorithm techniques.

4.2 Limitations

One of the series problems in scientific research that uses computer is the occurrence of low memory signals during program running this is a common problem. This can rise due to different things:

1. The Physical memory in the PC is not enough to continue solving the problem.
2. The processed data within the project is very huge as in our suggested method; both physical and virtual are not enough.

MATLAB generates an Out of Memory message whenever it requests a segment of memory from the operating system that is larger than what is currently available.

Our thesis is based on finding all possible paths first, say if the number of nodes is 6 so we must start with $6*6 = 36$ path then search and repeat for each of the paths approximately 36 agent after that we will filter from these possible path, these paths that reach the destination, so as we see within this stage we need at least a proximately $36*36*36$ then within the filtered path we need at least addition amount that is equal to the final possible paths.

Also each path is represented as a data structure with 6 records, also at least two of these records are an array of very large amount so that it can take any possible path No. and speed what ever its amount.

The second stage is based on re-production; crossover and mutation, in this stage the final paths are processed many times until the conditions are met, in this stage the fitness are calculated.

After the previous stage is calculated, the third stage is applied fully but here we calculate the fitness using the adaptive role. So many variables and many locations are allocated again.

So as we see above the PC will cover much large memory for the program also for its system function. By increasing the amount of nodes, the allocated memory will exceed the normal available memory, so this function will occur. Other factor also must be taken in an account, that the MATLAB is build completely over java, and as we know that java is based on virtual machine. This leads to decrease in operation speed and low memory usage efficacy.

We tried to solve the problem using the suggestion MATLAB methods. The only thing that can be used here is what is called parallel processing, i.e. many computers share there resources to do the task, it is not possible for now since here we need at least 5 PC linked to gather and use a special MATLAB Toolkit to do it called Distributed Computing Toolbox.

4.3 Results

The results was developed and implemented on MATLAB Version 2007 installed on server with the following configuration CPU P4 , 3000 MHz speed, 1 GB RAM ,by testing 6, 7, 8, 9 network nodes with different mutation values (0.3, 0.5, 0.7) and by running two suggested algorithm, first by applying standard genetic algorithm (GA), and second by adaptive genetic algorithm (AGA). From running different cases we take the important results and it will be obtained as shown in the figures below.

The number of nodes in the best path in GA and AGA decrease when the number of total network nodes increase, this indicate that the algorithm find the best path with minimum number on nodes. This mean the best path will be found in minimum time. Note that this results obtained with probability of mutation equal to (0.3). This relation is shown in figure (4-1) and figure (4-2)

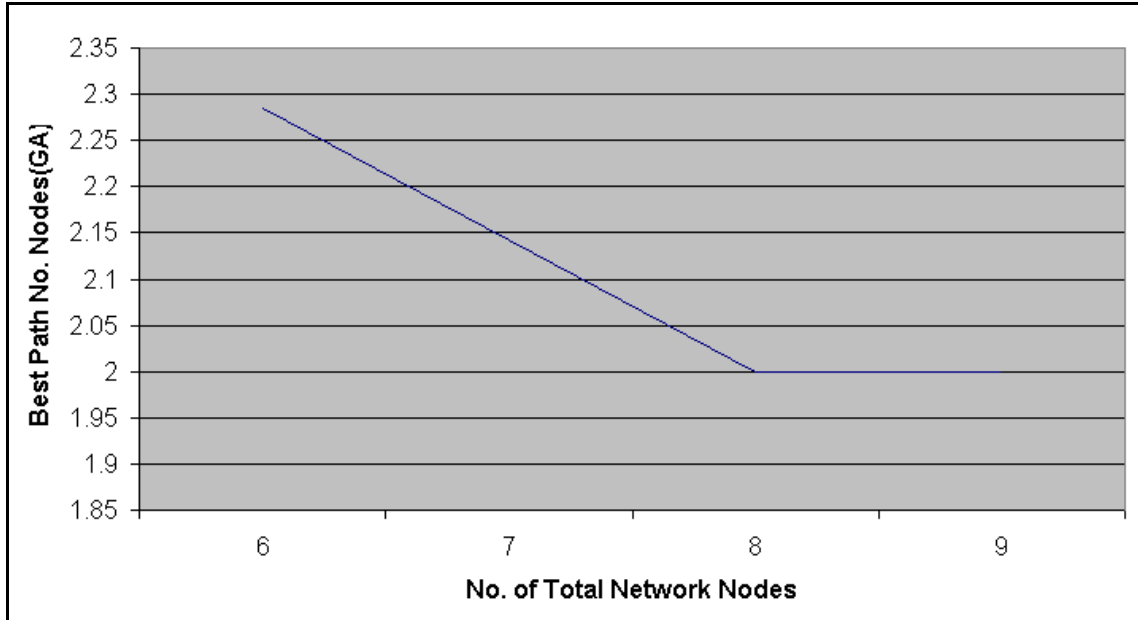


figure (4-1): no. of total network nodes versus best path no. of nodes (GA)

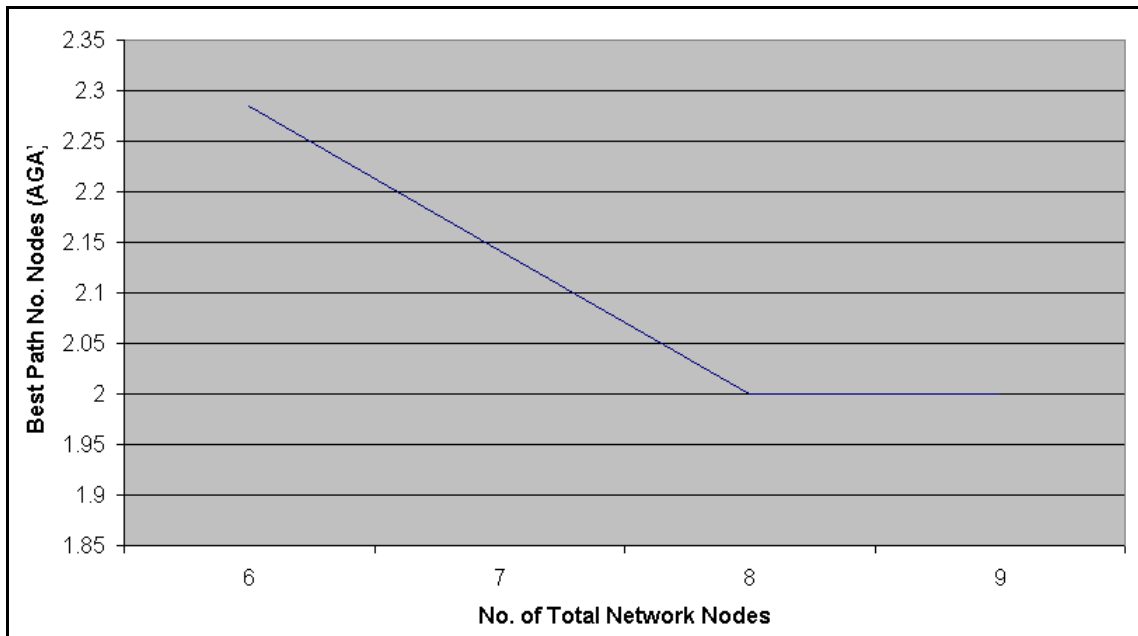
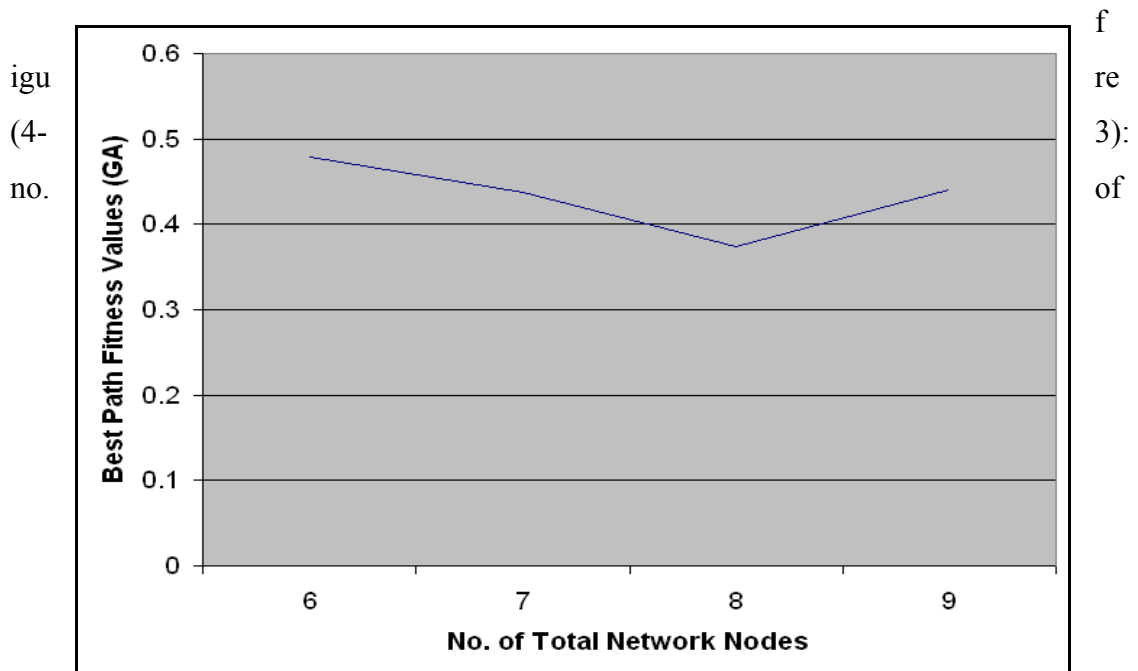


figure (4-2): no. of total network nodes versus best path no. of nodes (AGA)

The fitness values in GA may be decrease or increase for all values of mutation probability, so it does not depend on the total number of network nodes, this mean using fixed fitness function is not appropriate to find best path with best fitness value as shown in figure (4-3). This figure with mutation probability equal to (0.5)



total network nodes versus best path fitness values (GA)

When we use adaptive fitness function (changed dynamically), our algorithm give us best results, because the fitness value is always decreasing when we increase the total number no network nodes

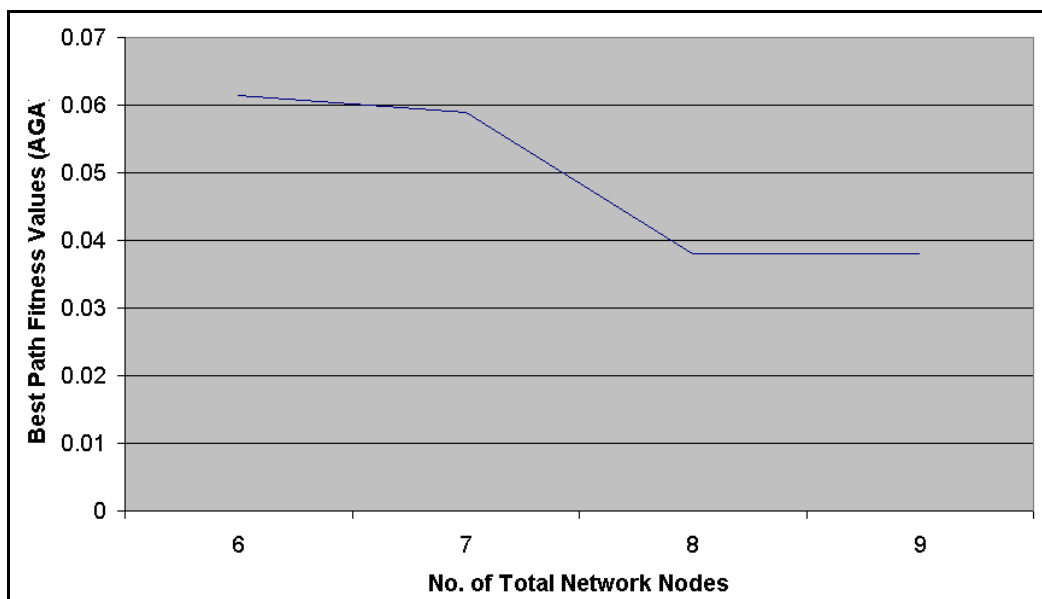


figure (4-4): no. of total network nodes versus best path fitness values (AGA)

In GA the no. of generation is fixed and equal to 50 for all values of mutation probability. This means that GA takes all the time executions time without changing, as shown in figure (4-5).

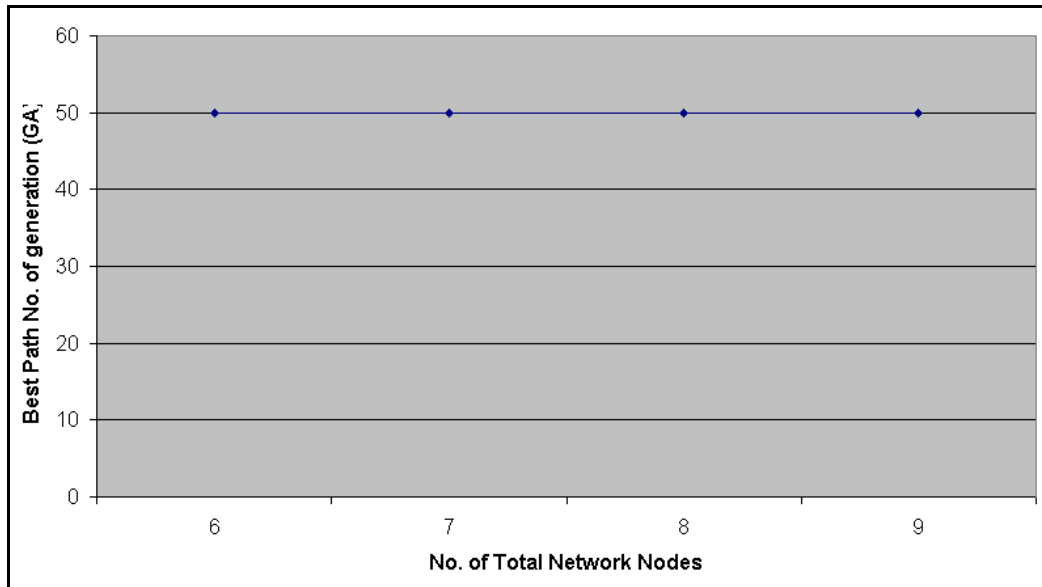


figure (4-5): No. of total Network Nodes versus Best Path No. of Generation (GA)

When number on network nodes increase, the AGA finds the optimal solution in minimum number of generation. This means the AGA is faster than GA, as shown in figure (4-6)

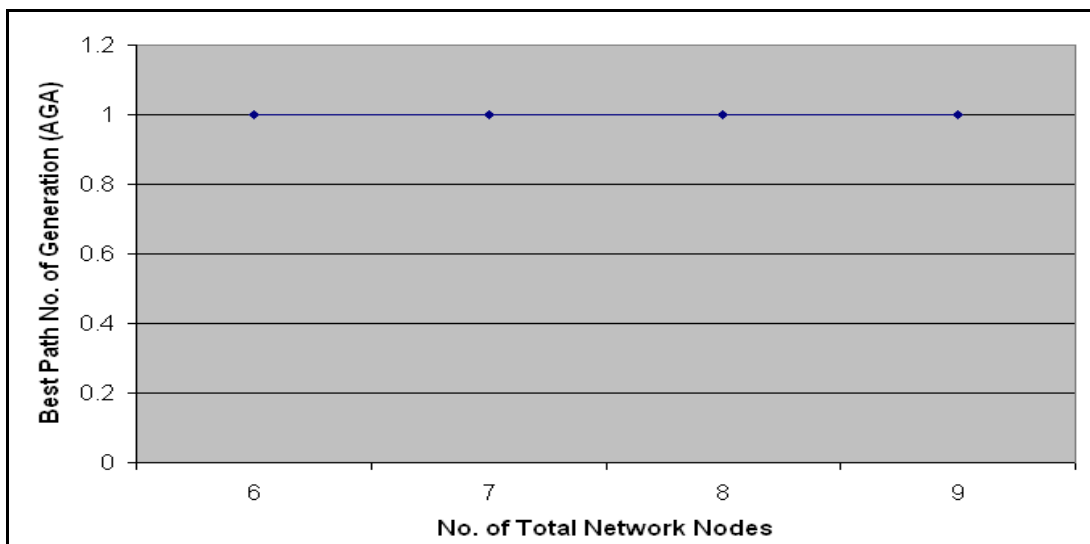
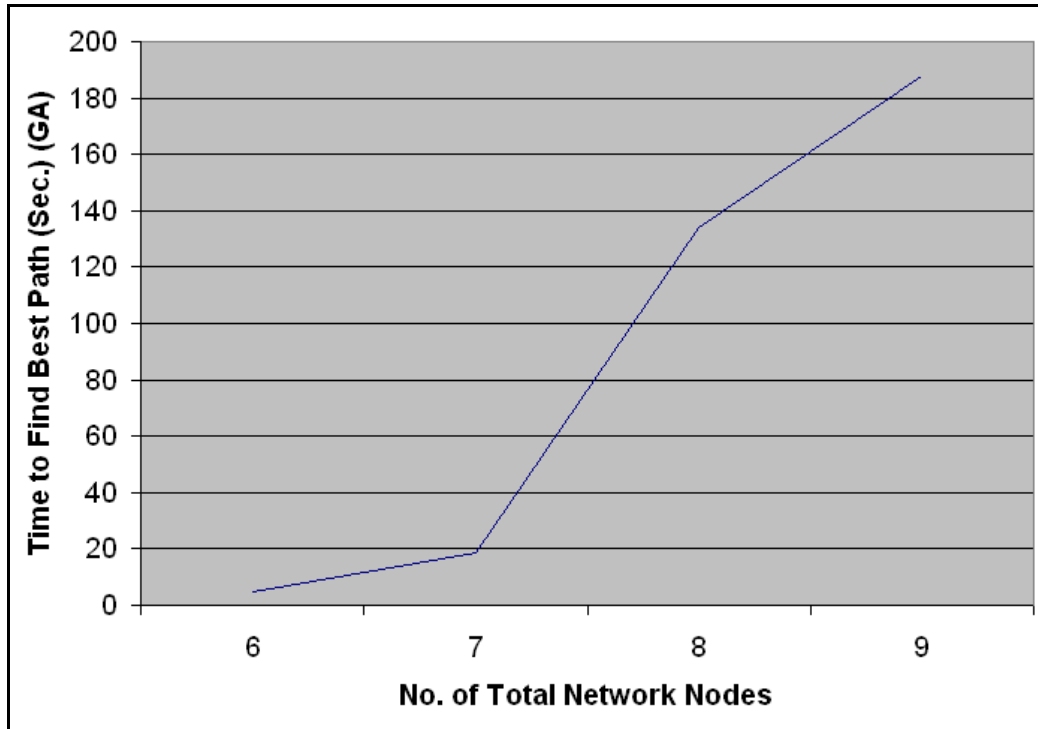


Figure (4-6): No. of total network nodes versus best path No. of generation (AGA)

We will see in figure (4-7) the time of executions when we using GA with probability of mutation equal to (0.7), it will take long time, because the fitness function is fixed. The GA will finish 50 generation to find best fitness value.

Figure (4-7): no. of total network nodes versus time to find best path (GA)



The time of execution when we using AGA are less than using GA, because the fitness function is changed dynamically for each generation to be suitable in all situations. The AGA will finish 1 generation only to finding best fitness value as shown below in figure (4-4).

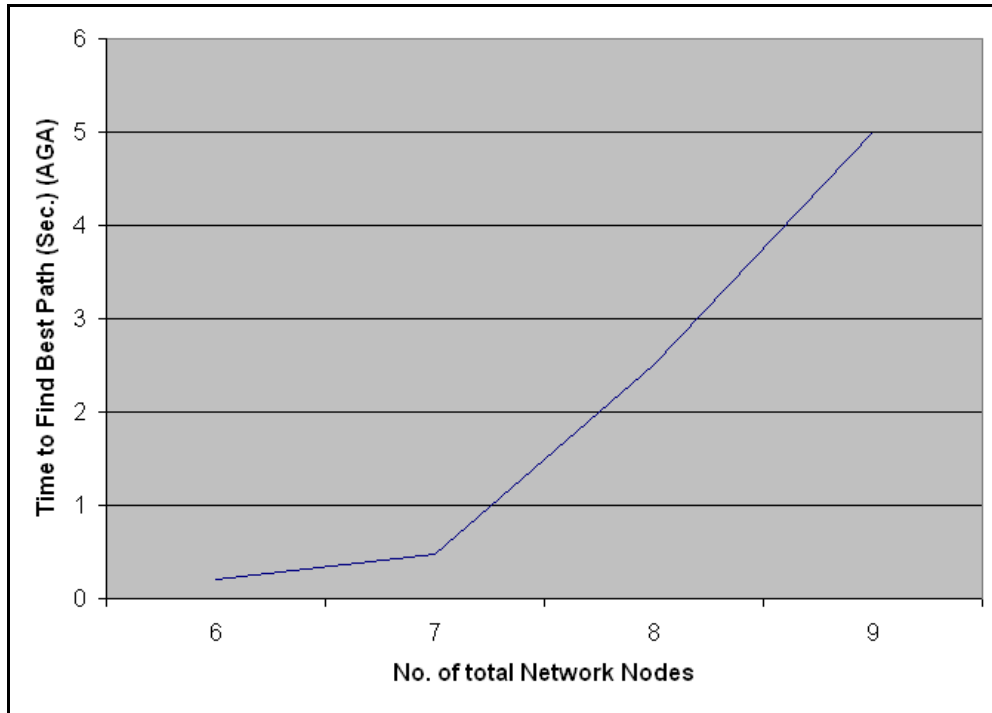


Figure (4-8): no. of total network nodes versus time to find best path (AGA)

Chapter Five

Conclusions and Future Works

5.1 Conclusions

In this thesis, we developed a hybrid dynamic routing protocol that implements and collects the principles of link state and distance vector dynamic routing protocol. We focused on selecting and implementing the following criteria: cost, number of nodes and number of increasing transmission speed on communication paths.

We implement a genetic algorithm and adaptive genetic algorithm that calculate optimal path in a minimum time, after comparing the results from the two algorithms we find that the adaptive genetic algorithm give us better results with minimum calculating time to choose optimal path from standard genetic algorithm.

Suggested adaptive algorithm is meant to be implemented with the OSPF dynamic routing protocol, which commonly implemented in internet routing protocol.

In fact the obtained solution will give more accurate results, enhance the performance when increasing the population, because the GA gives more accurate results when the population size increase that is directly connected with finding better solutions from many solutions that rise up through calculation.

5.2 Future Works

The following are some recommendation for future work to improve optimal network paths with minimum time.

Adding new criteria such delay and future Quality of Services (QoS) that have the ability to improve the calculation to find optimal paths.

Using neural network to tuning the weight in adaptive fitness function

Genetic Based Machine Learning (GBML) may be used to enhance the rules in crossover and mutation.

- [1] [Andreas Blass](#) and [Yuri Gurevich](#) (2003), [Algorithms: A Quest for Absolute Definitions](#), Bulletin of European Association for Theoretical Computer Science 81, 2003.
- [2] [Boolos, George and Jeffrey, Richard](#) (1999). Computability and Logic, Cambridge University Press, London. . cf chapter 3 Turing machines where they discuss "certain enumerable sets not effectively (mechanically) enumerable.
- [3] [Kleene, Stephen C.](#). *Introduction to Metamathematics*, Tenth Edition 1991, North-Holland Publishing Company.
- [4] Yao Zhou, "STUDY ON GENETIC ALGORITHM IMPROVEMENT AND APPLICATION" thesis 2006.
- [5] [Davis, Martin](#) . *Engines of Logic: Mathematicians and the Origin of the Computer*. New York: W. W. Norton, 2000.
- [6] Jeff Doyle," Dynamic Routing Protocols", Nov 16, 2001 Sample Chapter is provided courtesy of Cisco Press
- [7] Link state routing techniques US Patent Issued on May 16, 2006
- [8] Doyle, Jeff and Carroll, Jennifer (2005). Routing TCP/IP, Volume I, Second Ed. Cisco Press. ISBN 1587052024. Ciscopress ISBN 1587052024.
- [9] Tu-Chih Tsai, "Network Protocol Handbook", [Book Reviews], Communications Magazine, IEEE, Volume 34, Issue 1, Jan 1996 Page(s):10 -
Digital Object Identifier 10.1109/MCOM.1996.482238
- [10] Nilanjan Banerjee, Vaibhav Mehta and Sugam Pandey, "A Genetic Algorithm Approach for Solving the Routing and Wavelength Assignment Problem in WDM Networks", Department of Computer Science an Engineering Indian Institute of Technology, 2004.

- [11] Mitsuo Gen and Lin Lin, "A New Approach for Shortest Path Routing Problem by Random Key-based GA", 808-0135, JAPAN,2006.
- [12] A.E.Eiben and J.I. Van Hemert," SAW-ing adapting the fitness function for solving constrained problems", Leiden University, 1999.
- [13] Bilal Gonen, "Genetic Algorithm Finding the Shortest Path in Networks",Department of Computer Science and Engineering University of Nevada, Reno, Nevada 89502, 2003.
- [14] M. Mitchell, "An Introduction to Genetic Aglorithms", February 1998.
- [15] Stallings, William (2004). Data and Computer Communications (7th) Printice Hall, 137-138. ISBN 0-13-100681-9.
- [16] Quarteroni, Alfo; Fausto (2006). Scientific Computing with MATLAB and Octave. Springer. ISBN 978-3-540-32612-0.
- [17] D. Brelaz, New methods to color the vertices of a graph, *Communications of ACM*, 22(4):251-256, 1979.
- [18] Assefaw Hadish Gebremedhin, Parallel Graph Colouring, *Thesis*, 1999.
- [19] M. Gen and R. Cheng, *Genetic Algorithms and Engineering Optimization*. New York: John Wiley & Sons, 2000.
- [20] Gen, M., R. Cheng and S. S. Oren, "Network Design Techniques using Adapted Genetic Algorithms," *Advances in Engineering Software*, vol.32, no.9, pp. 731-744, 2001.

- [21] C.W. Ahn and R. Ramakrishna, "A Genetic Algorithm for Shortest Path Routing Problem and the Sizing of Populations", *IEEE Trans. Evol. Comput.*, vol. 6, no. 6, pp.566-579, 2002.
- [22] L. Lin, M. Gen and R. Cheng, "Priority-based Genetic Algorithm for Shortest Path Routing Problem in OSPF", *Proc. of 3rd Inter. Conf. on Information and Management Sciences*, pp. 411-418, 2004.