A GENETIC ALGORITHM BASED APPROACH TO VEHICLE ROUTE SEARCH IN EKITI STATE

BY

ODOLE, KAYODE EMMANUEL

(CPE/13/1079)

SUBMITTED TO THE

DEPARTMENT OF COMPUTER ENGINEERING

FACULTY OF ENGINEERING

FEDERAL UNIVERSITY OYE-EKITI,

NIGERIA.

IN PARTIAL FULFILMENT OF THE REQUIREMENT FOR THE AWARD

OF

BACHELOR OF ENGINEERING (B.Eng.) IN COMPUTER ENGINEERING

MARCH, 2019

CERTIFICATION

This project with the title

A GENETIC ALGORITHM BASED APPROACH TO VEHICLE ROUTE SEARCH IN EKITI STATE

Submitted by

ODOLE, KAYODE EMMANUEL

(CPE/13/1079)

Has satisfied the regulations governing the award of degree of

BACHELOR OF ENGINEERING (B.Eng) in Computer Engineering

Federal University Oye-Ekiti, Ekiti State Nigeria.

........01/04/2019........ ....01/04/2019........

Engr. (Mrs.) B.A. Omodunbi **Date**

Supervisor

....1/4/19........

Dr. (Engr.) T.O. Olaniyan **Date**

Head of Department

DECLARATION

This project is a result of my own work and has not been copied in part or in whole from any other source except where duly acknowledged. As such, all use of previously published work (from books, journals, magazines, internet and so on) has been acknowledged within the main report to an entry in the References list.

I agree that an electronic copy or hardcopy of this report may be stored and used for purposes of plagiarism prevention and detection. I understand that cheating and plagiarism constitute of a breach of University Regulations and will be dealt with accordingly.

Copyright

The copyright of this project and report belongs to Federal University, Oye-Ekiti.

Student's Full Name: ODOLE, KAYODE EMMANUEL

Sign & Date: Kayode 11/04/2019

DEDICATION

This research work is dedicated to God Almighty, for his mercy and grace over my life.

ACKNOWLEDGEMENTS

I whole heartedly want to appreciate my supervisor in person of Engr. (Mrs.) B.A.Omodunbi, the management of Federal University Oye-Ekiti, the academic and non-academic staff of the Computer Engineering department and for their guidance and support throughout my stay in the university. I also want to appreciate my parents for their support financially, emotionally and spiritually.

I also want to acknowledge my colleagues and friends namely Temitayo Bolaji, Sunday John, Tobi Alebiosu, Bunmi Alegbeye, Promise Adebayo, Samuel Ayeni, Bolaji Okunade, Tobi Adu, Monisola Jayeoba, FataiIsiaq, Oluwagbotemi Abatan, Tobi Olanitori, Michael Olaniyi, AdewunmiNiyi, Samuel Toloruntomi, Olive Nwafor, Saheed Okunlola, Folakemi Oyedokun, KafayatAina,, Esan Olorunfemi, Fatoye Pelumi, Olumide Adigun, Temitope Madamidola, Adebayo Olamilekan, Ajilore Jacob, Ayooluwa Adetunmbi, Mary Akano and many others whose names are not mentioned. God bless you all.

ABSTRACT

Vehicle routing involves searching for efficient routes for vehicles along transportation networks in an attempt to reduce travel time, route length, service cost. For vehicle drivers, the idea of getting to their destination in the shortest time possible is very appealing especially by taking the shortest path to their destination but the shortest route may not always be the optimal route. Drivers might need multiple and distinct good (near optimal routes) options which are based on multiple criteria that can make the search space too large to get the solution in real time by deterministic algorithms.

This project proposes a genetic based algorithm that combines the flexibility (producing more than one solution) of genetic algorithm and the speed of Dijkstra algorithm. This algorithm uses genetic operators including selection, crossover and incorporates Dijkstra algorithm mutation to produce optimal solutions. The processes involved include converting an actual road map into weighted graph, get origin and destination nodes, initialize the population, continue to perform genetic operation until the termination criteria is met and return best solutions. The termination criteria is implemented as the number of times the proposed algorithm performs genetic operation before returning the solutions found, at each iteration mutation and crossover operation will be performed.

The developed system was evaluated using processing time and distance. Genetic-Dijkstra algorithm has an average processing time of 5.83 seconds and average distance of 2059.09 meters compared to Genetic algorithm with processing time of 6.89 seconds and average distance of 2364.65 meters. The result gotten shows that the developed system is efficient and can be implemented in any other routing application.

TABLE OF CONTENT

LIST OF FIGURES

Figures pages

LIST OF TABLES

CHAPTER ONE

INTRODUCTION

1.1 Background

Vehicle routing involves searching for efficient routes for vehicles along transportation networks in an attempt to reduce travel time, route length, service cost. Vehicle routing has many applications within the fields of operations research, logistics, distribution, supply chain management, transportation. Vehicle drivers have a variety of preferences when it comes to the definition of an ideal route, which lead to various criteria for planning and evaluation of routes. (Bozyigit, Alankus, & Nasiboglu, 2017)

Searching for the optimal route from one place (origin point) to another place (destination point) has been one of the lifelong pursuit of vehicle drivers because the optimal route logically indicates shorter commuting time and reduced travel time which drivers find appealing. There are several shortest path algorithms namely Dijkstra algorithm, Bellman-Ford algorithm that can produce stable solutions in form the shortest path. But shortest path may not always be the best on considering factors like environmental problem, traffic congestions or personal satisfaction by users. Multiple simultaneous search for multiple short routes is difficult with the above mentioned algorithm because they only produce single routes at a time and need to be rerun to get new solutions which is computationally costly and doesn't guarantee the successive shortest paths. (Chakraborty, 2004)

This project developed a genetic algorithm that is capable of producing multiple simultaneous solutions (routes) for vehicle drivers in Ekiti State. Genetic algorithms are a type of optimization algorithm used to find optimal solution or solutions to a given computational problem that minimizes or maximizes a particular function. Genetic algorithms represents a branch of evolutionary computation mainly because biological processes natural selection and reproduction is imitated to solve for best (fittest) solutions. Genetic Algorithm is one of the approaches that can be used for probabilistic search, learning and optimization (Carr, 2014). While it maintains the population that describes candidate solutions, the entities (or individuals) in the population evolves using genetic operators of selection, crossover, and mutation. The size of the population is maintained, however superior candidates are preserved and the inferior are replaced by new individuals reproduced during evolution. Genetic algorithm has been shown to draw good solutions for a variety of problems which includes shortest path routing problem, Wireless sensor network optimization, traveling salesman problem and so on.

1.2 Statement of Problem

For vehicle drivers, the idea of getting to their destination in the shortest time possible is very appealing especially by taking the shortest path to their destination but the shortest route may not always be the optimal route. Drivers might need multiple and distinct good (near optimal routes) options which are based on multiple criteria that can make the search space too large to get the solution in real time by deterministic algorithms (Chakraborty, Maeda, & Chakraborty, 2005). Genetic algorithm is capable of returning multiple candidates solutions (near optimal routes) during search and has a possibility of always giving approximate (near optimal) solution regardless of search time (Alhalabi, Al-Qatawneh, & Samawi, 2008).

1.3 Aim and Objectives

The aim of this project is to develop a vehicle route search system using genetic based algorithm which can find the optimal route(s) between two points (from origin to destination) on the road map of Ekiti State. The specific objectives are as follows:

i. To design a vehicle route search system using genetic based algorithm

ii. To implement the designed system for some selected point of Ekiti State.

iii. To evaluate the effectiveness of the system.

1.4 Scope of Study

This project approaches vehicle route search in Ekiti State using Genetic based Algorithm. It was be tested and compared with pure genetic algorithm. The project emphasizes on exploring the use of genetic algorithm to estimate and evaluate more than one minimal cost path.

1.5 Significance of Study

Genetic algorithm can provide multiple and distinct good (near optimal) choices of routes for drivers in real world scenarios like riots, flood, fire outbreak affecting a city hence blocking some routes leading through it, in cases like this a single route isn't enough to make the right choice. Genetic algorithm can automatically get the necessary knowledge about the search space during its search process and by itself control the entire search process using random optimization techniques (Wu & Shan, 2000). The practical significance of the proposed algorithm is as follows:

i. It makes provision for dynamic environment optimization and always returns a solution.

ii. It can be applied to problems that can be defined as a graph and especially when the solution of the problem depends on the sequence of the elements

iii. Provide effective solution for geographical information system network analysis.

1.6 Methods of Study

The methods to be used in achieving this project will include the following:

i. Continual review of relevant literatures in the library and online resources related to Vehicle route search, Genetic algorithm, Road network processing, shortest path problem.

ii. Techniques of representing road transportation network as graph will be studied.

iii. Genetic algorithm will be used to estimate the minimal cost path

iv. Evaluation of the Implemented route search system using time complexity and space complexity.

CHAPTER TWO
LITERATURE REVIEW

2.1 Vehicle Route Search

Vehicle routing involves searching for efficient routes for vehicles along transportation networks in an attempt to reduce travel time, route length, service cost. Vehicle routing has many applications within the fields of operations research, logistics, distribution, supply chain management, transportation. Vehicle drivers have a variety of preferences when it comes to the definition of an ideal route, which lead to various criteria for planning and evaluation of routes (Bozyigit, Alankus, & Nasiboglu, 2017).

Optimal route search is part of the primary functions of car navigation devices. Intelligent transportation systems has made it possible for car navigation system to receive real-time information on traffic situations. Before the start of a journey the optimal route to the destination is calculated based on available information. Route search algorithms for car navigation devices make use of this information to avoid the traffic congestions. Route search algorithms should find the new optimal route efficiently when the traffic situation changes. On most cases the minimum traveling time or distance is considered to define the optimal route. It is important to note that the minimum traveling time or distance is not always what the user is looking for. The user may prefer to travel on a certain route even at the cost of traveling time or distance (Mainali, Mabu, Yu, Eto, & Hirasawa, 2017).

2.2 Algorithms

Algorithms according to computer science can be described as a finite, deterministic, and effective problem solving method that is suitable for implementation as a computer program. It can be defined by describing a procedure for solving a problem in a natural language, or by writing a computer program that implements the procedure. (Sedgewick & Wayne, 2011)

Cormen *et al.* defined an algorithm as a computational procedure that takes a set of values or some value as input and produces a set of values or some values as output. In essence an algorithm is a sequence of computational steps that transform the input into the output. Algorithms can be expressed as pseudocode. Pseudocode is different from a real code in that it can be expressed in whatever expressive method that is clear and concise to express the algorithm, such a method includes English phrases and sentences. An example to explain the use of pseudocode is the insertion sort which is an algorithm for sorting small number of elements. Figure 2.1 shows how the algorithm works in sorting an array A= (5, 2, 4, 6, 1, 3) (Cormen, Leiserson, Rivest, & Stein, 2009). Examples of algorithms include Ant Colony Optimization, Dijkstra's algorithm and Genetic algorithm, these algorithms can be used for route search.

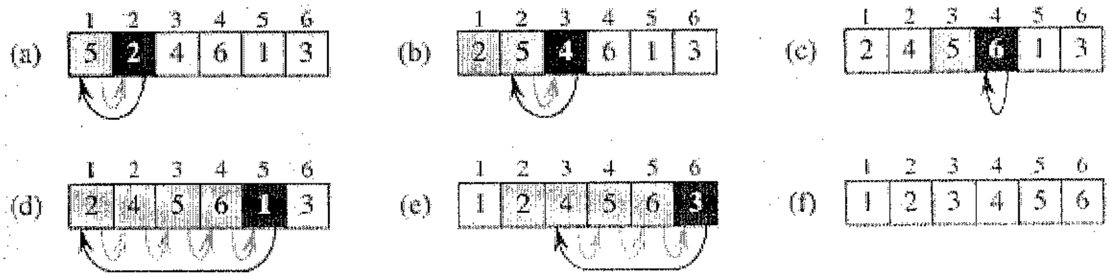


Figure 2.1: Operation of insertion sort on array A= [5, 2, 4, 6, 1, 3]. (Cormen, Leiserson, Rivest, & Stein, 2009)

2.3 Overview of Genetic Algorithm

Mankind over the years has drawn inspiration from different sources. Nature happens to be one of those sources, the evolutionary characteristic of nature gave way for Evolutionary computation a family of algorithms for global optimization. Genetic algorithms are a subset of Evolutionary computation (Kinnear, 2000), because they mimic the biological process of reproduction and natural selection to solve for the fittest solutions. These algorithm are more powerful and efficient than random search and exhaustive search algorithms (Carr, 2014).

Genetic algorithms are the most fundamental and widely known form of evolutionary computation currently in research. The underlying concepts of the GA were originally developed by John Holland at the University of Michigan beginning in the 1960s while Holland was doing research in the field of complex adaptive systems. Subsequent research efforts by Holland, his university colleagues and many other researchers have since advanced the GA into many practical forms of scientific application. These types of evolutionary algorithms have been designed as models capable of solving decision problems, classification problems and complex numerical optimization problems. (Krzanowski & Raper, 2001).

Genetic algorithm is based on a set of candidate solutions representing a solution to the optimization problem being solved (Kramer, 2017). "A genetic algorithm begins with a randomly chosen assortment of chromosomes, which serves as the first generation (initial population). Then each chromosome in the population is evaluated by the fitness function to test how well it solves the problem at hand" (Carr, 2014). The term "population" refers to all candidate solutions for a given optimization problem. The term "Chromosomes" refers to one of the candidate solutions.

2.3.1 Basic Components of Genetic Algorithms

Relevant terminology used in genetic algorithm is borrowed from biology since its process is designed to simulate a biological process. The following are basic component common to almost all genetic algorithms:

i. Fitness function for optimization

ii. A population of chromosomes

iii. Selection of which chromosomes will reproduce

iv. Crossover to produce next generation of chromosomes

v. Random mutation of chromosomes in new generation

The fitness function refers to the function that the algorithm is trying to optimize. Fitness function tests and quantifies the fitness of a potential solution. The fitness function happens to be one of the most critical parts of the algorithm.

2.3.2 Basic Terminology in Genetic Algorithm

i. Population: It is collection of all the possible solutions to the given optimization problem.

ii. Chromosomes – A chromosome is one of such solution to the given problem i.e. a single candidate solution out of the population of solutions.

iii. Gene – is one element position of a chromosome

iv. Allele – is the value a gene takes for a particular chromosome.

v. Genotype – Genotype is the population in the computation (digital) space. In the computation space, the solutions are represented in a way which can be easily understood and manipulated using a computing system

vi. Phenotype – Phenotype is the population in the actual real world solution space in which solutions are represented in a way they are represented in real world situations.

vii. Decoding and encoding – for simple problems, the phenotype and genotype spaces are the same. However, in most of the cases the phenotype and genotype spaces are different. Decoding is a process of transforming a solution from the genotype to the phenotype space, while encoding is a process of transforming from the phenotype to genotype space. Decoding should be fast as it carried out repeatedly in genetic algorithm during the fitness value calculation.

viii. Fitness function – is simply a function which takes the solution as input and produces the suitability of the solution as the output. In some cases, the fitness function and the objective function may be the same, while in others it might be different based on the problem.

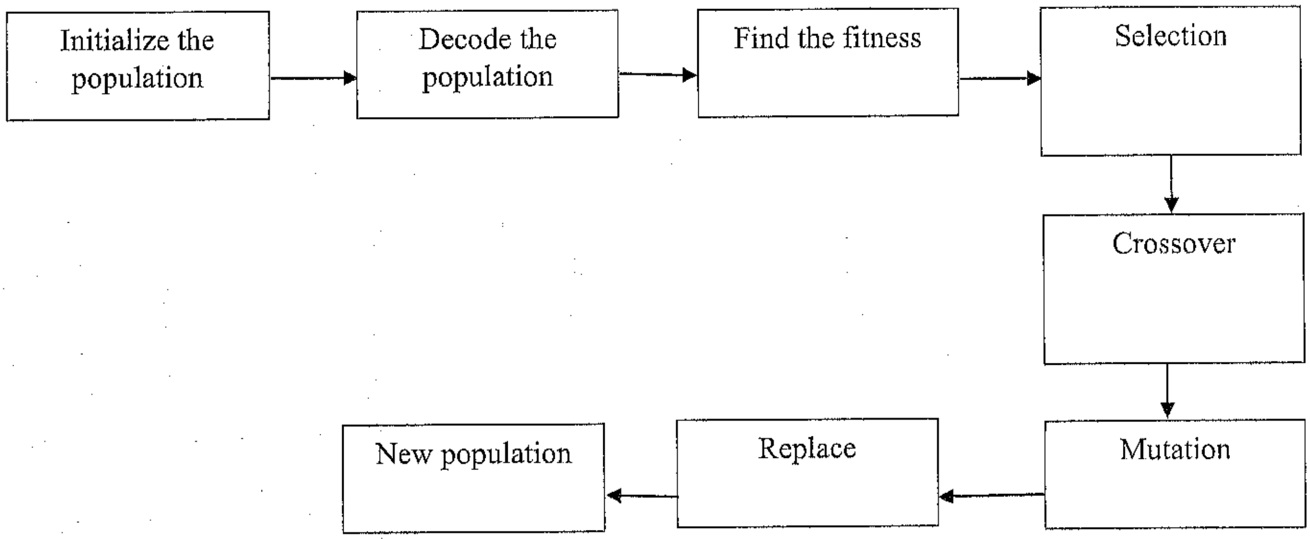


Figure 2.2: Block diagram of Genetic algorithm (Papakostas, Koulouriotis, Polydoros, & Tourassis, 2011)

2.3.3 Operators for Genetic Algorithm

2.3.3.1 Crossover

This operator allows the combination of genetic material of two or more solutions (parents). There exceptions where only one solution (parent) is required to produce new solution (offspring). The crossover operators works by mixing the genetic material of the parents. For bit string representation we have n-point crossover. This methods splits up two solution at n positions and alternately assembles them to produce a new one. The following are the most commonly used crossover operators:

a) One-Point Crossover: in this method of crossover a single random point is chosen in the genes parent chromosomes and the genetic material of both parents are swapped returning two new children chromosomes. For example given two bit-string parent chromosomes selected from the population

Parent 1: 1111 | 00001

Parent 2: 0000 | 1110

The selected crossover point chosenis indicated by " |", crossing the two chromosomes which involves swapping the genes after the chosen crossover point produces the following offspring

Offspring1: 11111110

Offspring2: 000000001

b) Two-Point Crossover: this method is quite similar to the one-point crossover method, except that two cross points are chosen instead of one. Given the parents below

Parent 1: 10 110 01

Parent 2: 01 010 00

The space in the parent genes indicate the two crossover points i.e. the first and the last two bits, crossing the parent chromosomes gives the following offspring:

Offspring 1: 10 010 01

Offspring 2: 01 110 00

Kramer opined that "The motivation for such an operator is that both strings might represent successful parts of solutions that when combined even outperform their parents. This operator can easily be extended to more points, where the solutions are split up and reassembled alternately." (Kramer, 2017). Figure 2.3 shows another example. Where two parent solutions are split into two in the middle and re-assembled to get two new solutions using one-point crossover.

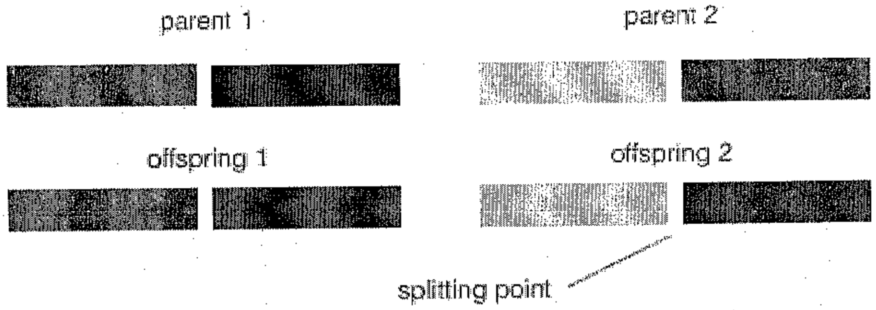


Figure 2.3: One-point crossover. (Kramer, 2017)

2.3.3.2 Mutation

Mutation operators operate by random changes to individual bits in the new chromosomes by turning 1 into 0 and vice versa. Mutation normally happens with a very low probability, such as 0.001. Some genetic algorithms implement the mutation operator before the selection and crossover operators (Carr, 2014).Mutation is based on random changes. The strength of this disturbance is called rate. In continuous solution spaces the mutation rate is also known as step size. Only a single chromosome (solution) is required to produce new offspring. For example a bit string chromosome is defined below, showing the effect of mutation before and after.

Before mutation: 1011011

After mutation: 1011001

There are three main requirements for mutation operators.

i. Reachability: Each point in solution space must be reachable from an arbitrary point in solution space. There must be minimum chance to reach every part of the solution space. If otherwise, the chance is positive that the optimum can be found. Not every mutation operator can guarantee this condition, for example decoder approaches have difficulties covering the whole solution space. (Kramer, 2017)

ii. Unbiasedness: The mutation operator should not induce a drift of the search to a particular direction.

iii. Scalability: Each mutation operator should offer the degree of freedom that its strength is adaptable. This is usually possible for mutation operators that are based on a probability distribution.

2.3.3.3 Fitness

Fitness computation of a solution is done using fitness function. The fitness function measures the quality of the solutions that the genetic algorithm has generated. Designing the fitness function is part of the modeling process of the optimization approach. "The performance of a genetic algorithm in solving a problem is usually measured in terms of the number of required fitness function evaluations until the optimum is found or approximated with a desired accuracy" (Kramer, 2017).

2.3.3.4 Selection

Convergence towards optimal solution requires that the best offspring solutions have to be selected to parent the new generation of offspring solutions. The selection process is based on the fitness values in the population. The probability for being selected depends on the fitness of a solution. The selection operator chooses some of the chromosomes for reproduction based on a probability distribution defined by the user. The fitter a chromosome is, the more likely it is to be selected. It is important to note that the selection operator chooses chromosomes with replacement so there is a possibility of choosing the same chromosome more than once. Using selection as mechanism to choose the parents of a new offspring solution is known as survival selection, so the selection operator determines which of the solutions survives or dies. "This perspective directly implements Darwin's principle of survival of the fittest" (Kramer, 2017).

2.3.3.5 Termination

The genetic algorithm runs for a predefined number of generations, but a defined termination condition can define when the main evolutionary loop should terminate. The termination condition determines when the main evolutionary loop terminates. (Kramer, 2017)

Normally the crossover, selection and mutation process continues until the number of the offspring is the same as the initial population, implying that the second generation has completely replaced the first. However some genetic algorithms let highly-fit members of the first generation survive into the second generation. The second generation is tested by the fitness function, and the cycle repeats until criteria is met then termination occurs. It is common to record chromosome that has the highest fitness along with its fitness value from each generation. (Koza, 1994).

2.3.4 Strengths of Genetic Algorithm

i. Provides a list of good solutions i.e. more than one good solution it is not restricted to a single solution

ii. It provides flexibility for different applications

iii. Has good parallel capabilities

iv. It always gets solution to a problem, and the solution keeps improving over time

v. The concept is easy to understand

vi. It is very useful when the search space is very large and there are a large number of parameters involved

vii. It does not require any derivative information

2.3.5 Limitation of Genetic Algorithm

i. Genetic algorithm is time consuming

ii. Inadequate encoding choice can affect the performance of algorithm negatively

iii. Genetic algorithm implementation is still an art.

2.4 Overview of Dijkstra's Algorithm

Dijkstra finds the shortest path in weighted graph. The nodes are divided into two groups namely tentative and permanent. The current node finds all its neighbor and assigns them as the tentative, then examines the tentative nodes if they pass the criteria that the next node to be in the tentative list must have minimum link cost or minimum weight, if the nodes passes the criteria then they are moved to the permanent list. Figure 2.4 shows the basic flow of Dijkstra algorithm. The root (source node) is chosen, the permanent list is initialized empty and the tentative list has one node which the root node. Then all the neighbors of the nodes in the tentative list are found and a cumulative cost of the edges are analyzed and the node in the tentative list is moved to the permanent list. Then all the neighbors found are placed in the tentative list, then among the neighbors which has the smallest cost is moved to the permanent list. This process is repeated until the tentative list becomes empty.

2.4.1 Applications of Dijkstra's Algorithm

i. Routing Systems
ii. Mapping e.g. Google Maps, Map Quest.
iii. Traffic Information System
iv. Open shortest path first, used in internet routing

2.4.2 Strength of Dijkstra Algorithm

i. It is used in Google maps
ii. It always finds the shortest path between the source and target node if it exists in the graph

2.4.3 Limitations of Dijkstra Algorithm

i. It does a blind search, hence it is time consuming.
ii. It cannot handle negative edges which leads to acyclic graphs and most often cannot obtain the right shortest path

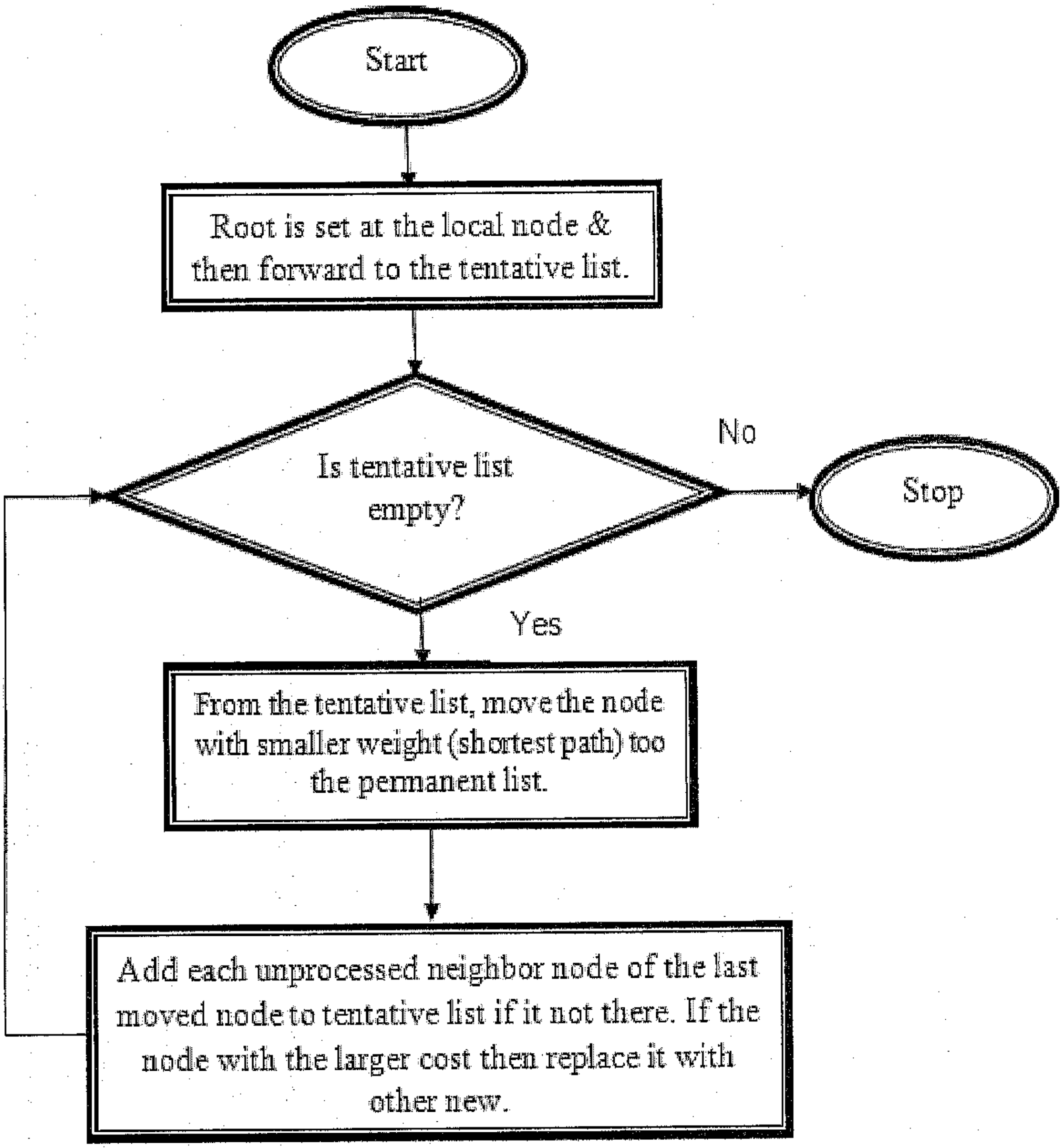

Start
Root is set at the local node & then forward to the tentative list.
Is tentative list empty?
No
Stop
Yes
From the tentative list, move the node with smaller weight (shortest path) too the permanent list.
Add each unprocessed neighbor node of the last moved node to tentative list if it not there. If the node with the larger cost then replace it with other new.

Figure 2.4: A flowchart of dijkstra's algorithm (Chip & Sandhu, 2016)

2.5 Maps

A map according to the Merriam-Webster dictionary can be defined as a representation usually on a flat surface of the whole or a part of an area (Merriam-Webster, 2018). Figure 2.5 shows the map of Ekiti State.

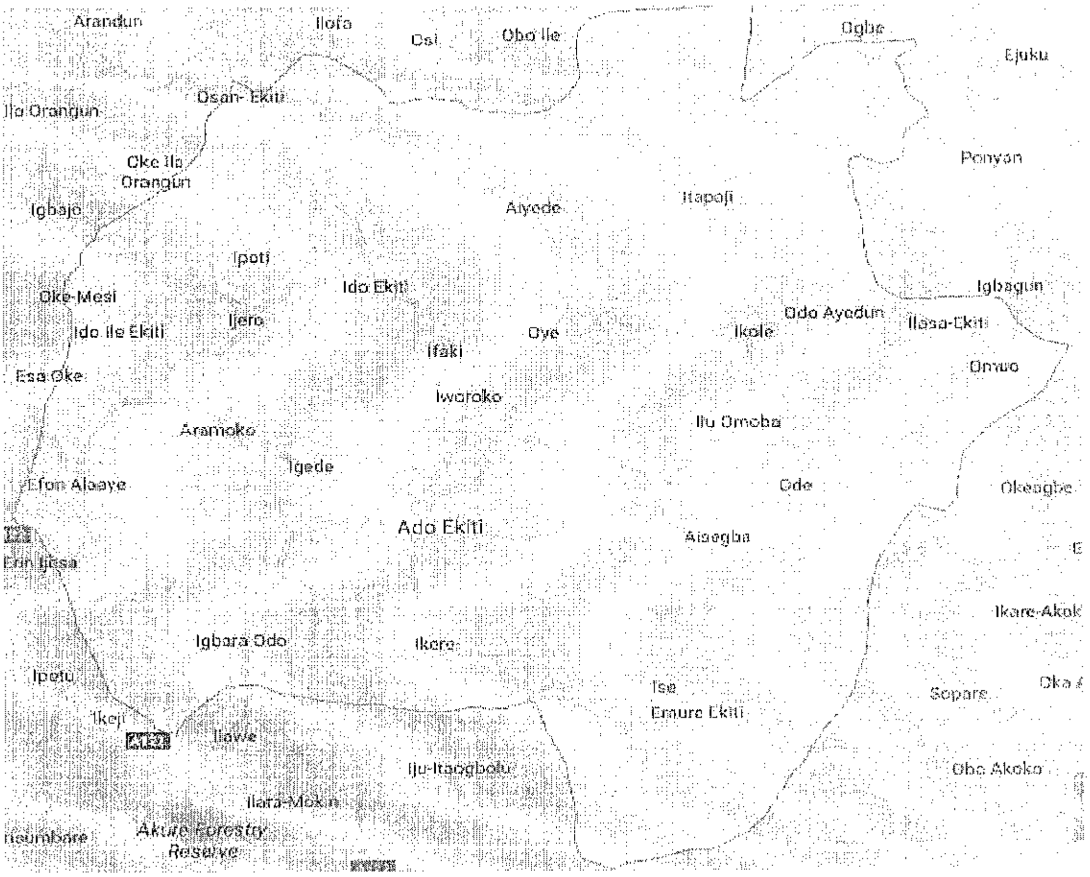


Figure 2.5: Map of Ekiti State. (Ekiti - Google Maps, 2018)

A map is used to illustrate detailed and specific features of a particular area. Maps attempt to represent a different things like roads, topography, population, climates, economic activities, natural resources, and climates. (Maps of India, 2012).

2.5.1 Uses of Map

Maps can be generally used for the following areas (Maps of India, 2012):

i. Analysis

ii. Confirmation

iii. Communication

iv. Decoration

v. Collection

vi. Investment

vii. Exploration

viii. Hypothesis Stimulation

ix. Navigation

x. Control & Planning

xi. Map Reading

xii. Storage of Information

xiii. Historical perspective

2.6 Graphs

A Graph can be defined as a set of vertices(nodes) and a collection of edges(links) that is connected to a pair of vertices. Figure 2.5 below shows a graph containing 7 vertices namely (0, 1, 2, 3, 4, 5, 6) and 8 edges. Two vertices are adjacent when they are connected by an edge. A vertex refers to a single node, so vertex 0 is adjacent to vertices 1, 2, 6 and 5 (Sedgewick & Wayne, 2011). Road networks can be modeled using graphs, in that the vertices can be used to represent the cities or junctions, and the roads linking this cities or junctions will be represented as edges. Graphs can be categorized as follows Undirected and directed graphs, weighted and un-weighted. Graphs facilitate making pairwise connections between items which plays a critical role in vast collection of computational applications.

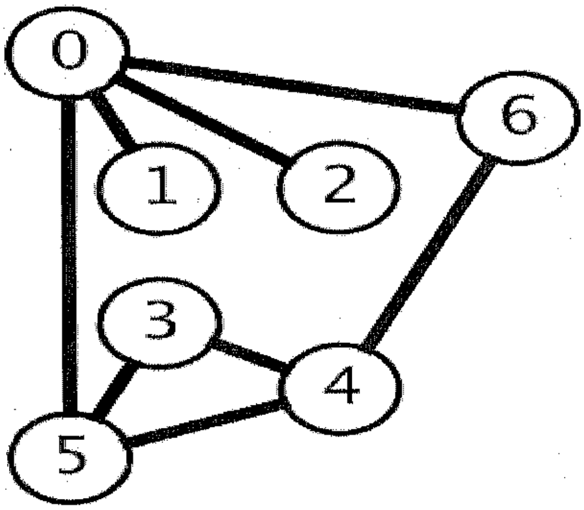


Figure 2.6: Graph composed of 7 vertices and 8 edges. (Sedgewick & Wayne, 2011)

2.6.1 Undirected and Directed Graph

A graph G = (V, E) where V denotes a finite set of vertices (nodes), and E denotes a set of edges. Graph G is undirected if an edge is an unordered pair {u, v} of nodes u, v ∈V implying that nodes u and v are elements of the finite set of vertices (nodes) V. This implies that there is no difference between the two vertices linked with each edge. While in directed Graph, edges are one way in the sense that the pair of vertices that defines each edge is an ordered par that specifies a one-way adjacency. Directed Graph which is also known as a digraph is a set of vertices and a collection of directed edges. Each directed edge connects an ordered pair of vertices. Unlike undirected graph edge (0-1) linking node 0 to node 1 isn't the same as edge(1-0) linking node 1 to node 0. (Sedgewick & Wayne, 2011) Figure 2.7 illustrates the connection of places in Ekiti state as an undirected graph.

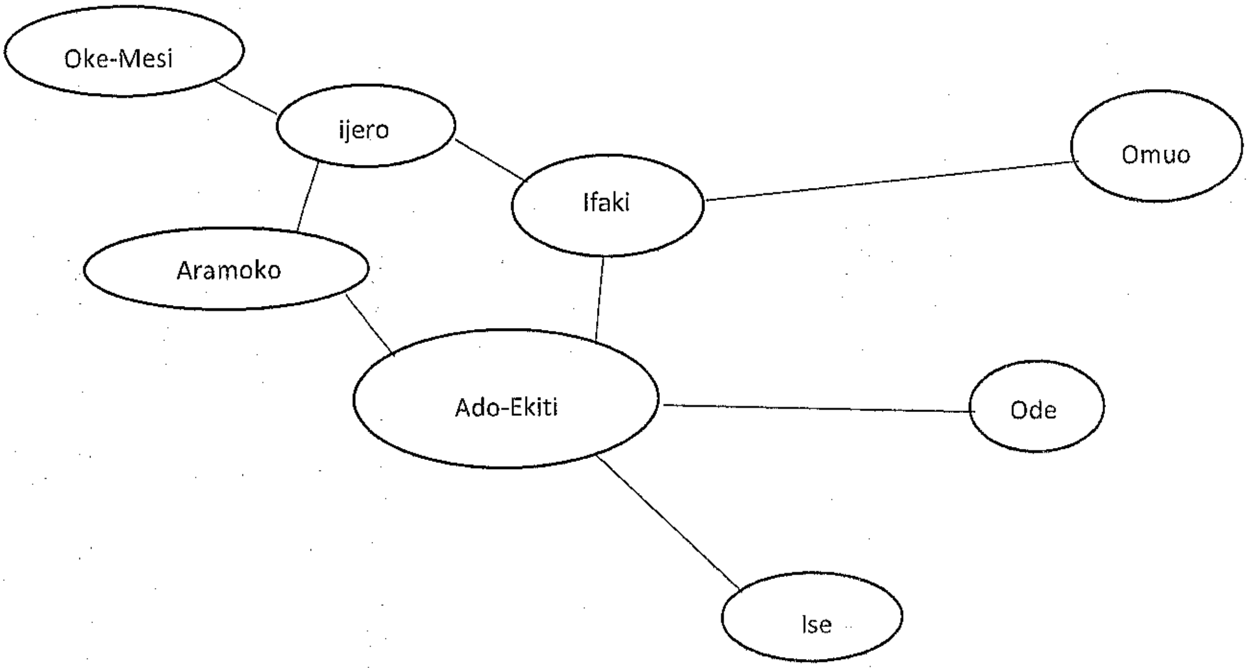


Figure 2.7: Undirected graph showing connection between places in Ekiti state map

2.6.2 Weighted and Un-weighted Graph

A graph G = (V, E) where V denotes a set of vertices, and E denotes a set of edges. In weighted graphs, each edge of G is assigned a numerical value which is known as weight. The weight on the edges of a road network graph might represent their length, drive-time or speed limit. While in un-weighted graphs, there is no weight for any of the edges. Figure 2.8 shows a weighted graph, this can be used to represent a road network where each node represent intersections or places, each edge represent the road and the weight on the edges can represent length of the road, speed limit, commute time or drive-time. Figure 2.9 shows un-weighted graph that identifies the vertex and edge.

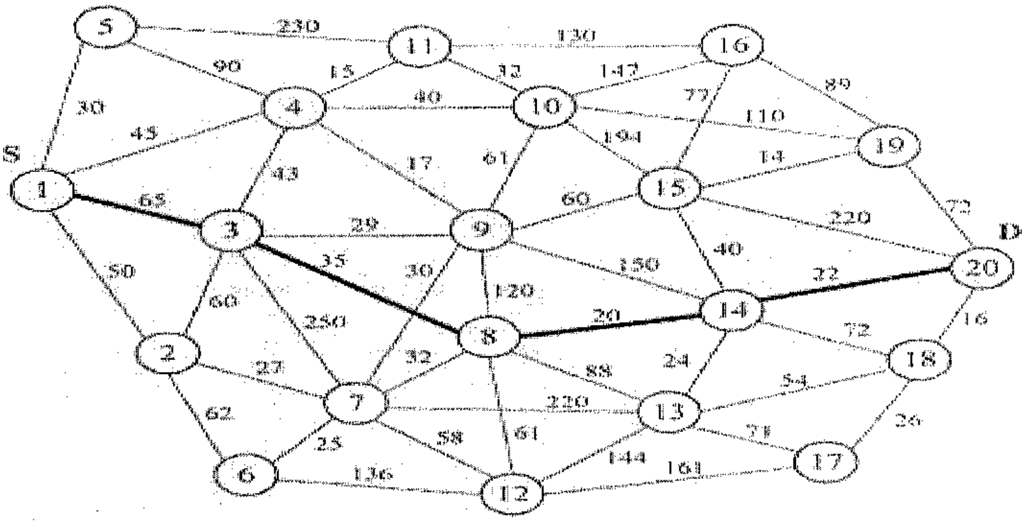


Figure 2.8: Weighted graph, each edge has a weight or cost. (Ahn & Ramakrishna, 2002)

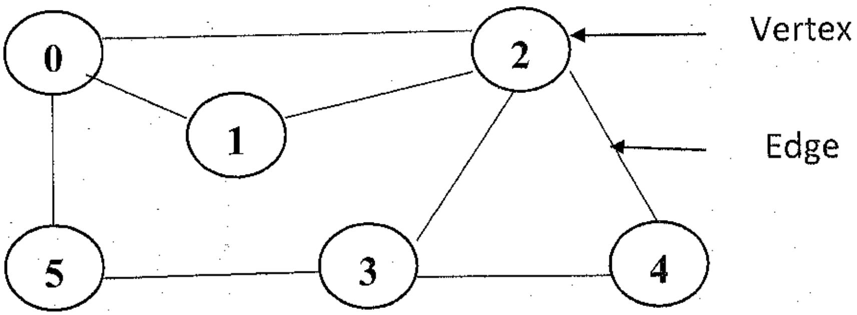


Figure 2.9: Un-weighted graph, the edges have no cost. (Sedgewick & Wayne, 2011)

2.6.3 Applications of Graph

Graphs has many practical applications, some of them include modeling many types of relations and process in physical, biological systems (Mashagi & al., 2004). In real world application the term "network" sometimes refers to a graph in which attributes like names, places are associated with the nodes and edges. The following are examples of areas of where graphs are useful, such as Maps, Web content, Computer networks, matching, software, social networks. Table 2.1 shows typical graph applications.

Table 2.1: Typical graph applications. (Sedgewick & Wayne, 2011)

Application	Item	Connection
Map	Intersection	Road
Web content	Page	Link
Circuit	device	Wire
Schedule	job	constraint
Commerce	customer	transaction
Matching	student	application
Computer network	site	connection
Software	method	Call
Social network	person	friendship

a) Maps: planning a trip for instance from one town to another, using Ekiti as a case study requires a person to answer questions like "What is the shortest route from Ado Ekiti to Ikole Ekiti? " in cases where the shortest route is not the fastest route, the question can be rephrased as "what is the fastest way to get from Ado Ekiti to Ikole Ekiti? " To answer such a question, the information about the trip can be processed by representing roads as edges and intersections or places as nodes. (Sedgewick & Wayne, 2011)

b) Web content and social network: When browsing the web, pages that contain links which are references to other pages are encountered, and navigating through the pages require clicking on the links. The entire web page can be represented as a graph, where the nodes are the pages and links are the edges. Graph processing algorithms are essential components of the search engines that assist in locating information on the web. (Sedgewick & Wayne, 2011) Search Engines like google use powerful algorithms to handle the search request of users. For social networks, explicit connections are built with friends. Nodes represent people while edges represent can represent connections as a friend or follower. Understanding the properties of these networks facilitates how best to support such networks.

c) Computer networks and software: Computer network consist of interconnected devices that send, forward, and receive messages of various types. Graph facilitates knowing about the nature of the interconnection structure which allows laying wires and building switches that can handle the network traffic efficiently. For software a compiler builds graphs to represent relationships among modules in a large software system. The nodes are the various classes or modules that makes up the system. The edges can represent the possibility that a method in one class might call another or with actual calls while the system is in operation. Analyzing the graph is necessary in order to determine how to allocate resources to the program efficiently. (Sedgewick & Wayne, 2011)

2.7 Open Street Map, Osmnx, Networkx, Pycharm and Spyder

Open street map (OSM) is the free wiki world map. It is built by a community of mappers that contribute and maintain data about roads, trails, cafes, railway stations all over the world. It emphasizes local knowledge. Contributors use aerial imagery, global positioning system (GPS), and low-tech field maps to verify that OSM is accurate and up to date. OSM is open data, implying that there is freedom to use it for any purpose as long as OSM and its contributors are credited. Any alteration or building upon on OSM data can be distributed only under the same license. (About OpenStreetMap, 2019)

OSMNX is python package that can be used to retrieve, construct, analyze, and visualize street networks from OpenStreetMap. OSMNX facilitates the download of spatial geometries and from OpenStreetMap's application programming interface (API). It has capabilities which includes automated and on-demand downloading of political boundary geometries, building footprints, and elevations, It can automate and customize the downloading of street networks from OpenStreetMap and construct them into graphs and it can correct and simplify network topology. (Boeing, 2017)

NetworkX is a python package for the creation, manipulation, and study of the structure, dynamics, and functions of complex networks. Its features includes data structures for graphs, digraphs, and multigraphs, many standard graph algorithms, network structure and analysis measures, generators for classic graphs, random graphs, and synthetic networks. It also has additional benefits from python which includes fast prototyping, easy to teach, and multi-platform. (Networkx Home, 2019)

Spyder is a powerful scientific development environment written in python, it offers a unique combination of the advanced editing, analysis, debugging, and profiling functionality of a comprehensive development tool with the data exploration, interactive execution, deep inspection and beautiful visualization capabilities of a specific package. (SPYDER, 2019)

Pycharm is an integrated development environment (IDE) used specifically for python language. It is developed by the Czech company JetBrains. It provides code analysis, a graphical debugger, an integrated unit tester, integration with version control systems (VCSes), and supports web development with Django. (Pycharm, 2019)

2.7 Relevant Works

Ahmed proposed a genetic algorithm to determine the k shortest paths from a single source node to multiple destinations with bandwidth constraints. The proposed algorithm uses the connection matrix of a given network, and the links of the bandwidth to get the k shortest paths. He used two examples of network topology and applied the proposed genetic algorithm on the network examples. Figure 2.10 illustrates the study of the effect of the mutation probability, it is obvious from figure 2.11 that the k shortest paths decrease when the mutation rate decrease in the proposed algorithm. (Hamed, 2010)

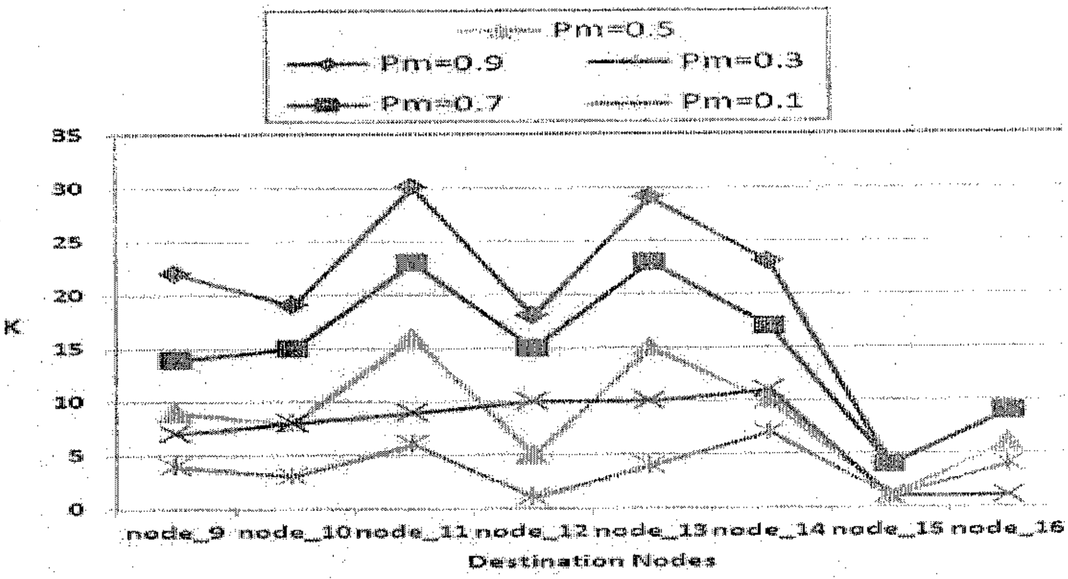
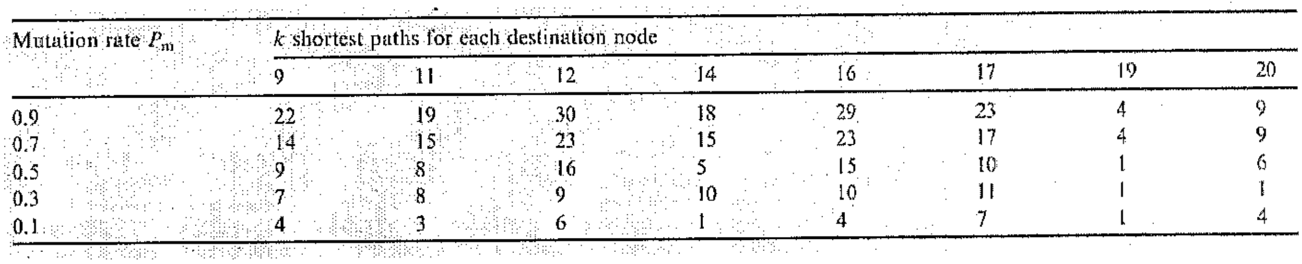


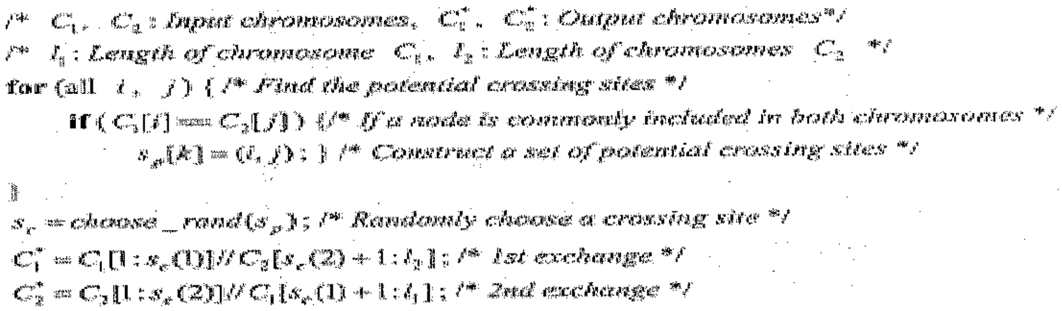
Figure 2.10: The effect of the mutation probability (Hamed, 2010)



Mutation rate P_m	k shortest paths for each destination node							
	9	11	12	14	16	17	19	20
0.9	22	19	30	18	29	23	4	9
0.7	14	15	23	15	23	17	4	9
0.5	9	8	16	5	15	10	1	6
0.3	7	8	9	10	10	11	1	1
0.1	4	3	6	1	4	7	1	4

Figure 2.11: The effect of mutation on k shortest path (Hamed, 2010)

Ahn and Ramakrishna proposed a Genetic algorithm to find the shortest path, by defining and using variable length chromosomes for encoding the problem, each gene represents a place in the map. A population-sizing equation that facilitates a solution with desired quality was also developed. The crossover operation exchanges partial chromosomes at independently chosen two crossing sites, and the mutation operation makes changes to randomly determined sub-paths between the source and the chromosomes in terms of the representational requirements for the given task of route search. However the algorithm fixes invalid chromosomes using the repair function. The proposed algorithm was evaluated to be able to search the solution space effectively and speedily compared to other extant algorithms. In addition the population sizing equation was observed to be a conservative tool to determine a population size in the routing problem. Figure 2.12 and figure 2.13 shows the Pseudocode and example of crossover procedure respectively. Figure 2.14 shows the example of mutation procedure. Figure 2.15 shows the example of repair function. Function 2.16 shows the comparison result for Inagaki's algorithm, Munemoto algorithm and the proposed algorithm (Ahn & Ramakrishna, 2002).



```
/* $C_1$ , $C_2$ : Input chromosomes, $C_1^*$ , $C_2^*$ : Output chromosomes*/
/* $l_1$ : Length of chromosome $C_1$ , $l_2$ : Length of chromosomes $C_2$ */
for (all $i$ , $j$ ) { /* Find the potential crossing sites */
        if ( $C_1[i] == C_2[j]$ ) {/* If a node is commonly included in both chromosomes */
                $s_p[k] = (i, j)$ ; } /* Construct a set of potential crossing sites */
}
$s_c$ = choose_rand($s_p$); /* Randomly choose a crossing site */
$C_1^* = C_1[1 : s_c(1)] // C_2[s_c(2) + 1 : l_2]$ ; /* 1st exchange */
$C_2^* = C_2[1 : s_c(2)] // C_1[s_c(1) + 1 : l_1]$ ; /* 2nd exchange */
```

Figure 2.12: Pseudo-code for crossover. (Ahn & Ramakrishna, 2002)

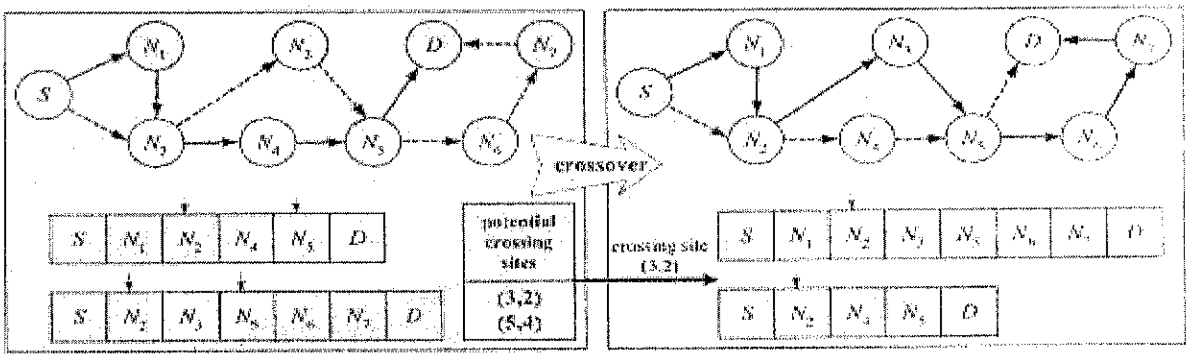


Figure 2.13: example of crossover procedure. (Ahn & Ramakrishna, 2002)

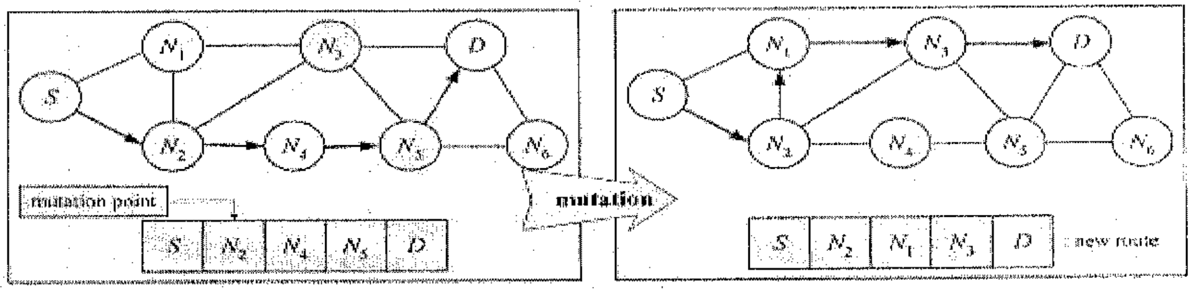


Figure 2.14: Example of mutation procedure (Ahn & Ramakrishna, 2002)

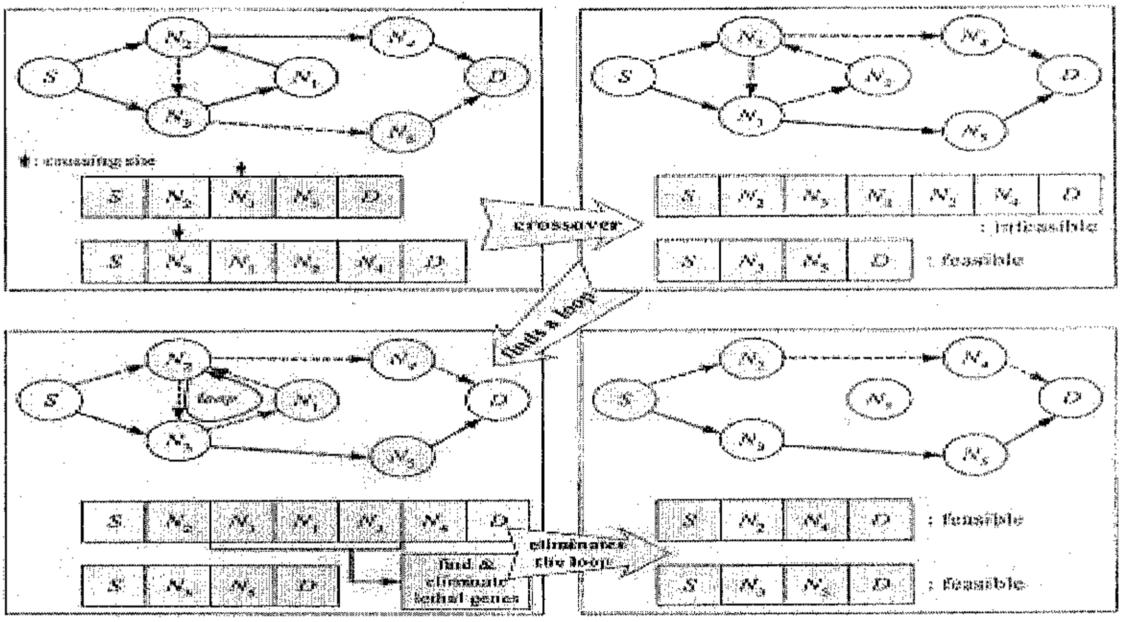


Figure 2.15: Example of repair function (Ahn & Ramakrishna, 2002)

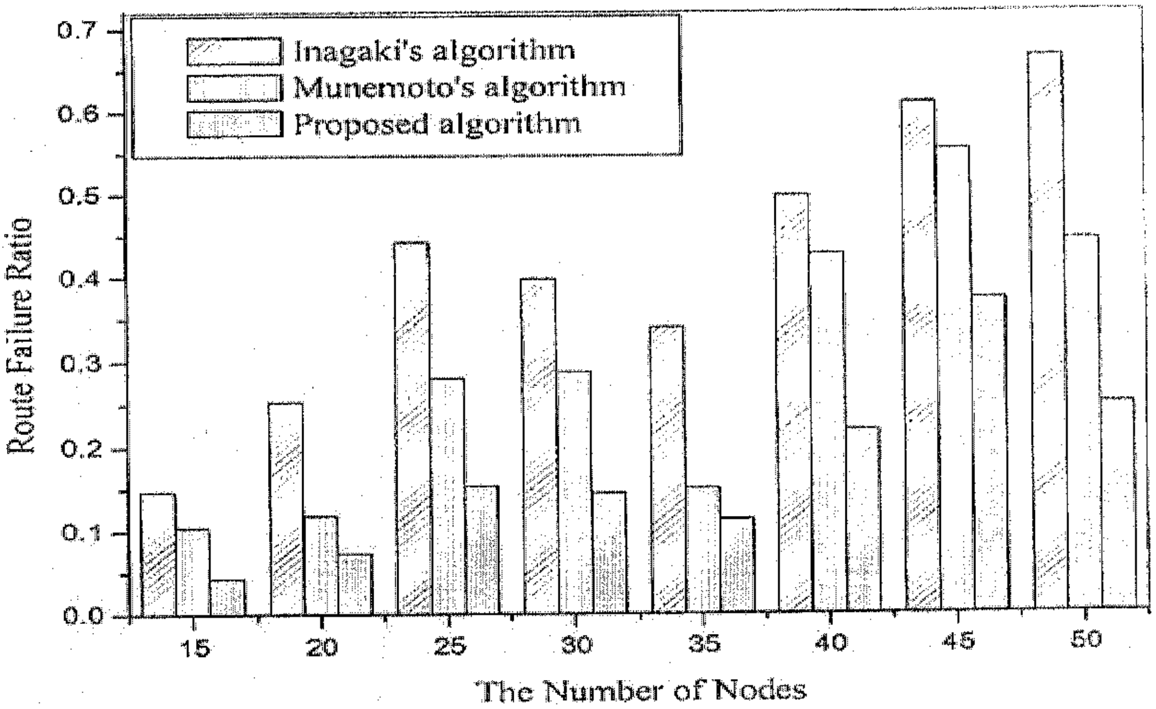


Figure 2.16: Comparison results of the quality of solution for each algorithm. (Ahn & Ramakrishna, 2002)

Wu and Shan (2000) proposed a genetic algorithm which provides solution for geographical information system (GIS) network analysis which provides strong decision support for users in searching for the nearest facility and determining the service area. The knapsack problem was used to introduce the concept of genetic algorithm and describe its various operations such as encoding, crossover, inversion and mutation. The algorithm was tested using a network with eighty nodes to demonstrate the efficiency of the genetic algorithm and its application potential. It was able to solve the best route selection problem in the network analysis through efficient encoding, selection of fitness function and various genetic operations. The genetic algorithm implementation method also showed to be effective in terms of computation time and complexity. However further efforts is required on the integration of genetic algorithm with commercial GIS packages. (Wu & Shan, 2000)

Sur and Shukla proposed a modified version of discrete bat algorithm that suits discrete domain problems. In this method the Bat algorithm has utilized three famous variable dependent WeibullCummulative Distribution Function as Weibull Coded Binary Bat Algorithm (WCBBA), Real Bat algorithm(RBA) and the third as the hybrid of the two. For search process with its modeling according to a road network management system where it is used to optimize the travel route and produce a vehicle load balancing structure of the network through optimized path establishment. The proposed Bat algorithm is modeled to handle multi-objective optimization model where each bat tries to optimize its own route criteria in other words each bat tries to find its likely prey. The enhanced search criteria of bats helps in reducing the number of bats for the search process, but as the echolocation process spreads out it increases the complexity of the search

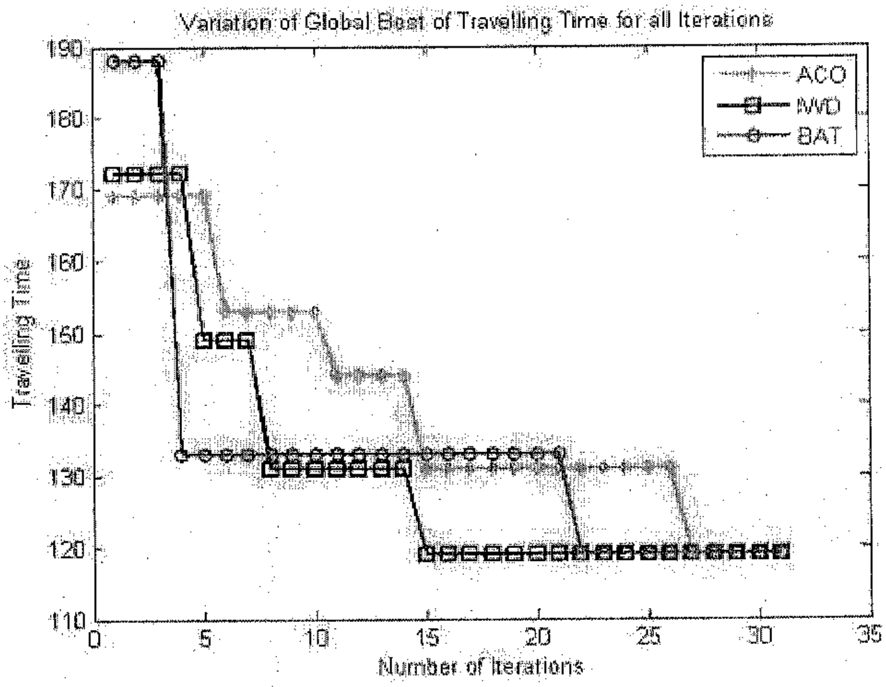


Figure 2.17: Variation of global best of traveling time for all iterations. (Sur & Shukla, 2013)

process making it more complex. The result show that the algorithm has potential for better results and has been compared with the converging rate of Ant Colony Optimization(ACO) & Intelligent Water Drops(IWD) algorithms. Figure (2.17, 2.18, 2.19) clearly shows Real Bat Algorithm(RBA) is well suited for discrete search space problems and road network graph. (Sur & Shukla, 2013)

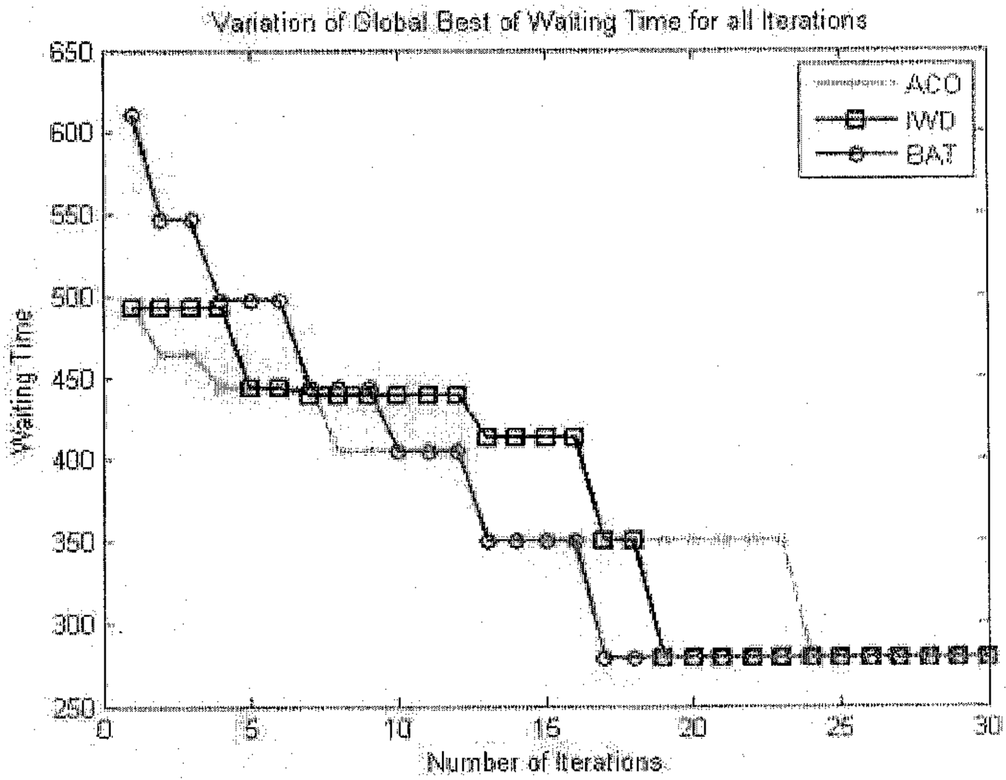


Figure 2.18: Variation of global best waiting time for all iterations. (Sur & Shukla, 2013)

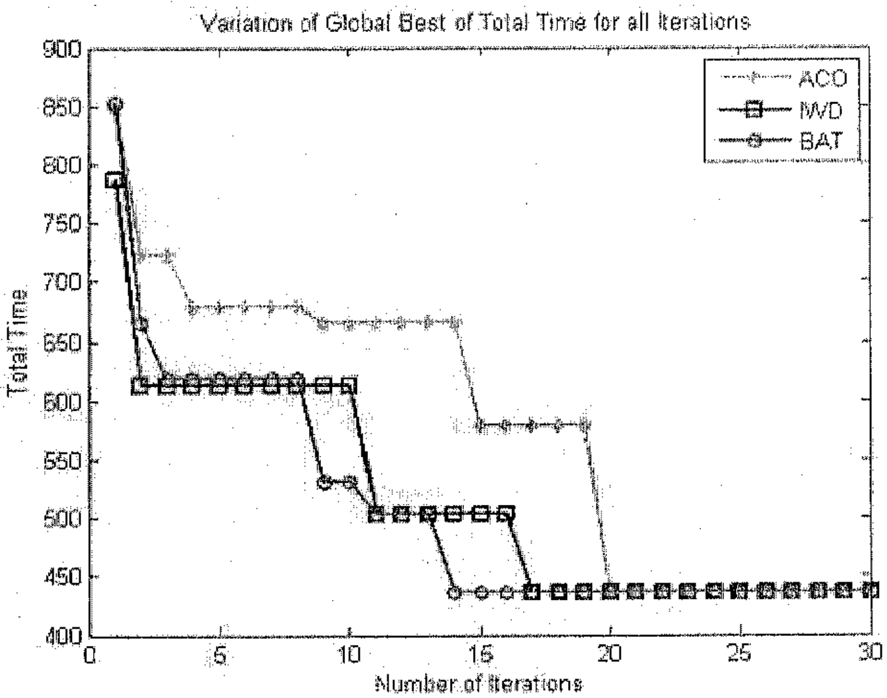


Figure 2.19: Variation of global best of total time for all iteractions. (Sur & Shukla, 2013)

Inagaki, Haseyama, & Kitajima proposed an algorithm that uses fixed length chromosomes. The chromosomes in the algorithm are sequences of integers and each gene represents a node identity that is selected randomly from the set of nodes connected with the node corresponding to its locus number. All the chromosomes have the same length which are fixed. During the crossover phase, one of the genes from two parent chromosomes is selected at the locus of the starting node identity and placed in the same locus of an offspring. One of the gens is then selected randomly at the locus of the previously chosen gene's number. This process is continued until the destination node is reached. The details of mutation are not explained in the algorithm. The algorithm requires a large population to attain an optimal or high quality of solution due to its inconsistent crossover mechanism. Some offspring may generate new chromosomes that resemble the initial chromosomes in fitness, thereby retarding the process of evolution (Inagaki, Haseyama, & Kitajima, 1999).

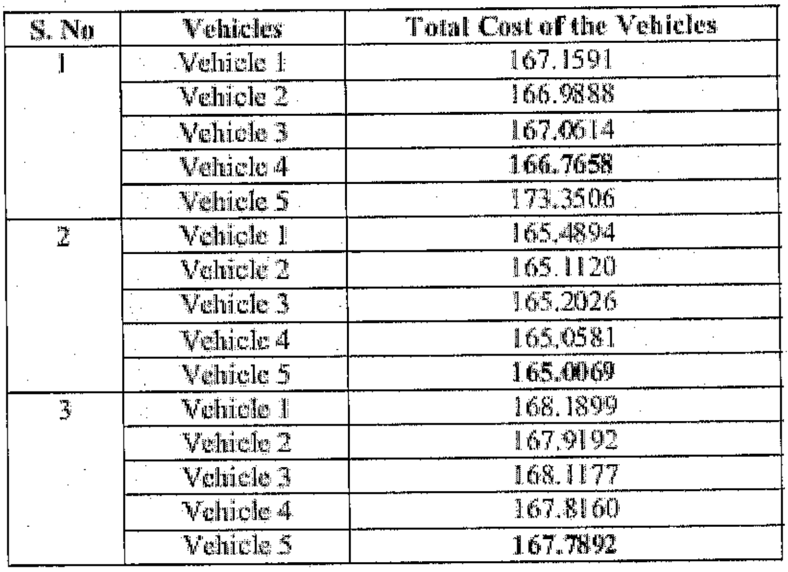
Rui-qing and Zhao proposed a monkey algorithm inspired by the mountain climbing behavior of monkeys the search space of the problem is the field with several mountains with food sources. The monkeys move in groups in search of food over the mountains. In the process of searching for food for their family, they perform climbing, watching, jumping and somersaulting processes. All these behaviors are exploited to form the steps of the monkey algorithm. In monkey algorithm (Kumar, devi, & Sathya, 2017), it is assumed that finding the highest mountaintop is compared to the maximum objective function. The monkeys from their respective positions climb to reach the mountaintop. When the monkeys reach the top of the mountain, they will wait and spend time to check about the better mountaintop from their current positions. Once they find the better positions they will jump to their respective better positions. This process is simulated in the watch jump process of the monkey algorithm. This is followed by repeated climbing process till the mountaintop is reached for all the monkeys. The three basic processes in monkey algorithm namely Climb, Watch-Jump and Somersault is explained briefly below (Rui-qing & Zhao, 2008) :

a) Climb Process: Monkeys starting from the initial position, climb over the mountain step-by-step carefully. This behavior is simulated in the climb process of monkey algorithm to search for the local optimal solution based on the pseudo-gradient information of the objective function. Step size of the monkeys to increase their position in the climb process is defined step length parameter.

b) Watch-Jump: On reaching the mountain top, the monkey will watch around from the mountain top position. The maximum distance the monkey can watch is defined by the eyesight parameter. During the watch process if any close higher mountain is found compared to the current position, the monkey will jump to that adjacent mountain and try to climb up. The process described above is simulated in the watch-jump process to accelerate the search course. The current solution is updated with the new and better solution.

c) Somersault process: This process is used to introduce monkeys to new global search domains. During this process a barycenter which is called the pivot is selected from the current positions of all monkeys. The monkeys will somersault to a new position either forward or backward with respect to the pivot.

The monkey algorithm was applied to open vehicle routing problem and the results is shown in figure. The open vehicle routing problem has been run for 3 time and the minimum optimal cost of the vehicles obtained from each run is displayed in figure 2.20.



S. No	Vehicles	Total Cost of the Vehicles
1	Vehicle 1	167.1591
	Vehicle 2	166.9888
	Vehicle 3	167.0614
	Vehicle 4	**166.7658**
	Vehicle 5	173.3506
2	Vehicle 1	165.4894
	Vehicle 2	165.1120
	Vehicle 3	165.2026
	Vehicle 4	165.0581
	Vehicle 5	**165.0069**
3	Vehicle 1	168.1899
	Vehicle 2	167.9192
	Vehicle 3	168.1177
	Vehicle 4	167.8160
	Vehicle 5	**167.7892**

Figure 2.20: Total cost of the vehicles in three runs. (Rui-qing & Zhao, 2008)

Byonghwa *et al* proposed a new route search algorithm which is based on genetic algorithm called Alternative route search (ARS) algorithm, in an unexpected events of system failure or traffic slowdowns and gridlock due to accidents this algorithm is capable of searching alternative routes for users dynamically. The algorithm was reported to be survivable and adaptable in a variety of dynamic traffic situations compared to Dijkstra algorithm and TPEG. The result of their experimentation via simulation showed that the ARS got more vehicles to their destinations compared to Dijkstra and TPEG. They expect to validate the usefulness of their algorithm by further adjusting parameters in more sophisticated ways and enlarging the scale of simulations, in the end hoping to produce a new navigation system incorporating their ARS and to deploy it in Korea. (Byonghwa, *et al.*, 2010). Figure 2.22 shows the infrastructure of the navigation system by car-to-car communication. Figure 2.21 shows the dynamic route generation by ARS.Figure 2.23 shows the result of the experiment. It display the simulation time in seconds and the number of vehicles that successfully reached the destination by each algorithm.

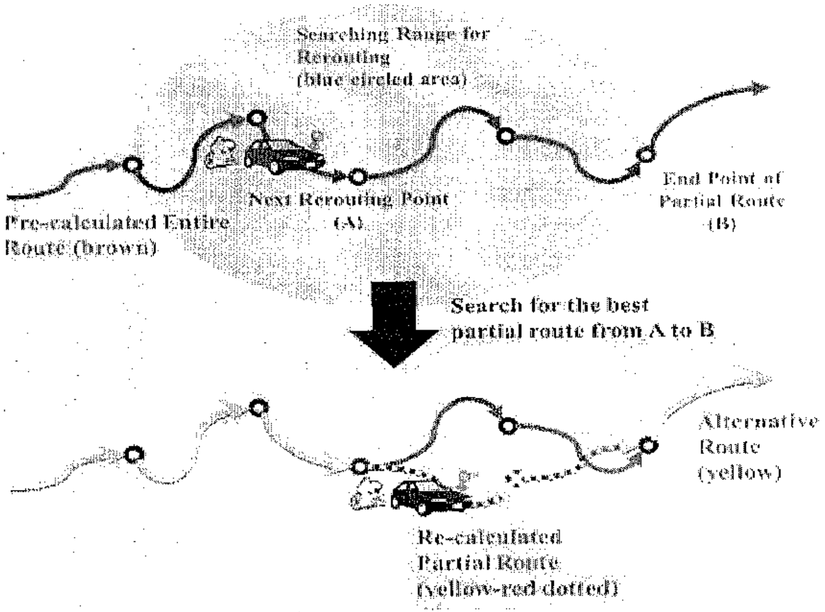

Searching Range for Rerouting (blue circled area)
Pre-calculated Entire Route (brown)
Next Rerouting Point (A)
End Point of Partial Route (B)
Search for the best partial route from A to B
Alternative Route (yellow)
Re-calculated Partial Route (yellow-red dotted)

Figure 2.21: Dynamic route generation by ARS (Byonghwa, *et al.*, 2010)

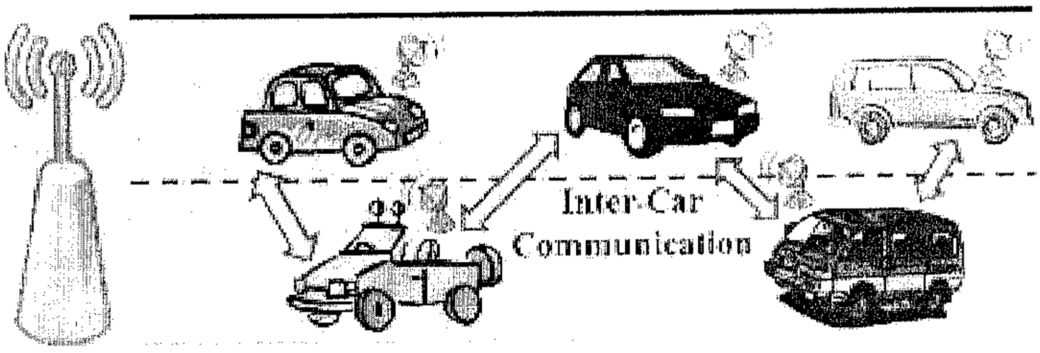
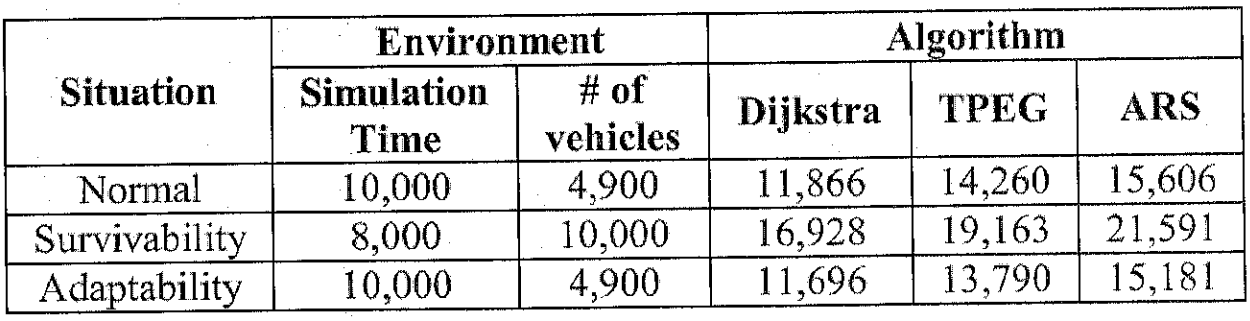


Figure 2.22: Navigation system based on inter-car communication. (Byonghwa, *et al.*, 2010)



Situation	Environment		Algorithm		
	Simulation Time	# of vehicles	Dijkstra	TPEG	ARS
Normal	10,000	4,900	11,866	14,260	15,606
Survivability	8,000	10,000	16,928	19,163	21,591
Adaptability	10,000	4,900	11,696	13,790	15,181

Figure 2.23: The number of vehicles reaching their destination. (Byonghwa, *et al.*, 2010)

Charkraborty proposed a genetic algorithm for searching multiple non overlapping routes from origin point to the destination point in a road map to be used in a car navigation system. He also proposed a new fitness function that ranks the likely solutions. This algorithm was reported to have been evaluated against Inoue's method (Inoue, 2001) which was proposed by Inagaki (Inagaki, 2002). The performance of the proposed algorithm was compared against Dijkstra, and reported that algorithms like Dijkstra find it difficult to produce multiple route simultaneously but the proposed GA based algorithm can produce multiple routes simultaneously with minimal overlapping efficiently. He reported that the usefulness of the proposed algorithm can be better understood in the problem of finding out multiple routes in dynamic environment. The process involved in the proposed algorithm include Coding solution space, setting fitness function, initialize population and genetic operation. (ChakraBorty, 2004)

Wen and Hsu proposed a route navigation system with a new revised shortest path routing algorithm for solving road traffic problems. The proposed system when it determines the shortest paths from source to destination can avoid selecting no left(right) turns, one-way roads and congested roads. The proposed new algorithm was compared numerically with existing algorithms such as the Dijkstra algorithm and the A* algorithm. For experimentation purpose a road network containing 4000 nodes which have 200 no left turn situations, the result obtained was that the road traffic problem of a 4000 node traffic network can be solved within only 0.651 seconds on average. The prototype system was built and some extra functions were added which provided benefits that include offering the shortest path and providing information and security services for drivers. Figure 2.24 below shows the average execution time of four algorithms, the A* algorithm's execution time 0.099 seconds was very close to that of Wen-Hsu's algorithm, 0.098 seconds. From the experiment the speed of the execution time of the A* algorithm and the Wen-Hsu's algorithm is superior to that of Dijkstra and Dijkstra+'s algorithm. Wen and Hsu reported that a better

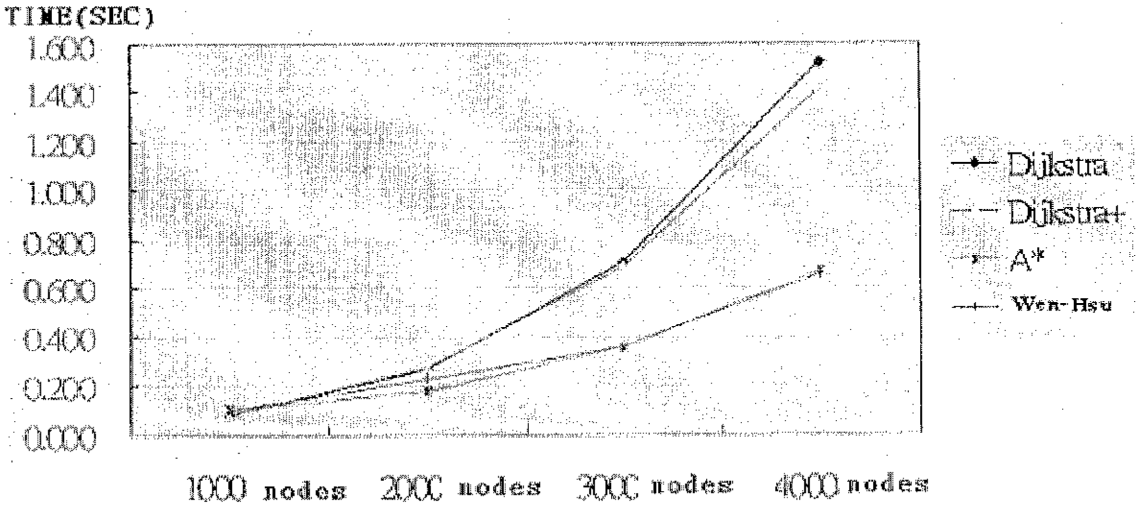


Figure 2.24: Average execution time of the four algorithms (Wen & Hsu, 2005)

efficiency can be generated when a large number of nodes are chose. However the route navigation system's function developed can be further improved by exetending its function to have an automatic reasoning mechanism using a knowledge-based inference engine. Future work involves Building an efficient searching and learning mechanism for the route navigation system. (Wen & Hsu, 2005)

Alhalabi, Al-Qatewneh, and Samawi proposed a route navigation system which efficiently guides users to multiple road destinations so the user can know several path to take to get the destinations. They proposed a genetic algorithm solution to a problem of selecting route to a given start and destination on an actual map under user defined environment. The algorithm implements variable length chromosomes meaning that chromosomes size can be dynamic due to path length which is the number of cities between source and destination. The population size of the routes is generated with respect to the chromosome size. Two selection methods was used and compared namely tournament selection method and a suggested selection method. They recorded that there is a relationship between the length of chromosome and the optimal solution, suggesting that the optimal solution with best value can be derived by using the maximum length of chromosome which contains all elements of the DB genes. They used larger population size which allowed to have better and faster results throughout the crossover and mutation operation. However using larger population size slowed down the machine which means cost in terms of memory and time, in addition deciding adequate population size is crucial for efficiency. They suggested that future work should deal with situation of providing a balance between the importance of using full length of chromosome(the full content of population) to get the optimal solution and the importance of reducing the size of the instance content(contents belonging to specific population and its chromosomes) to reduce the throughput time in order to speed up getting the optimal solution not semi-optimal solution (Alhalabi, Al-Qatawneh, & Samawi, 2008).

El-Sayed, Zhang, Thandavarayan and Hawas proposed a method called “Partittion based Framework for Distance and Shortest Path Queries (ParDiSP) for road networks. The proposed method is a combination of both bounded-hop and hierarchical approaches. ParDiSP is divided into two phases which is preprocessing phase and query phase. In the preprocessing phase the input network is divided into four components. The components are extended component, transit network, in-component distances tables (IDT), component distance matrix (CDM). Each component has its own functionality e.g. recording the closes path in between end node, storing the distance of every node with end node. In the query phase which is the second phase it is divided into two cases: the first is when starting and end places are present in one component then ALT algorithm is applied. On the other hand for queries where start and end places are in distance queries by combining distances from three pre-calculated tables two in-component distance tables(IDTs) and one entry component distance matrix(CDM). (El-Sayed *et al.*, 2016)

Katre and Thakare proposed a smart system for a device called Quick response device. This device provides current location of a place and also provides the shortest path for rescue vehicle to reach to the destination in time. The proposed system can be used for web application to send location, display route and maintain database which stores all the nearest place details. The quick response device is divided into three sub components, which includes the User interface component, control unit component and receiver component. The user interface component which allows the user to interact with the system has three buttons: Red, Blue and Green. The Red button indicates the police emergency button, green button indicates the hospital emergency button and blue button indicate the fire bridged emergency. The Control Unit component which communicates with google map cloud service to find the nearest required location then routing algorithm is applied to find an effective route between the source and destination. The receiver end component displays the emergency message and the shortest path to its current location. Figure 2.25 shows the system architecture.Katre and Thakare also reviewed classical shortest path algorithm with their advantage and disadvantages, Table 2.2 shows the analysis of the algorithms.

Table 1.2: Analysis of the algorithm

No	Algorithm	Advantages	Disadvantage	Time Complexity	Space complexity
1.	Dijkstra's Algorithm	Find shortest path in $O(\lvert E\rvert + \lvert V\rvert Log(\lvert V\rvert))$ if you use a min priority queue	Fails in cases where you have a negative edge	$O(\lvert E\rvert+\lvert V\rvert Log\lvert V\rvert)$	$O(\lvert v\rvert^2)$
2.	A* Algorithm	A* is faster than using Dijkstra and uses best-first-search to speed things up.	The main drawback of A* Algorithm and indeed of any best first search is its memory requirement	Depends on the heuristic	Depends on the heuristic
3.	Floyd-warshall Algorithm	Much easier to code and All pairs of shortest paths are solved	Slower and harder to understand	$O(n^3)$	$O(n^3)$
4.	Johnson algorithm		Fails in cases where you have a negative edge	$O(\lvert V\rvert^2 log\lvert V\rvert + \lvert V\rvert\lvert E\rvert)$	

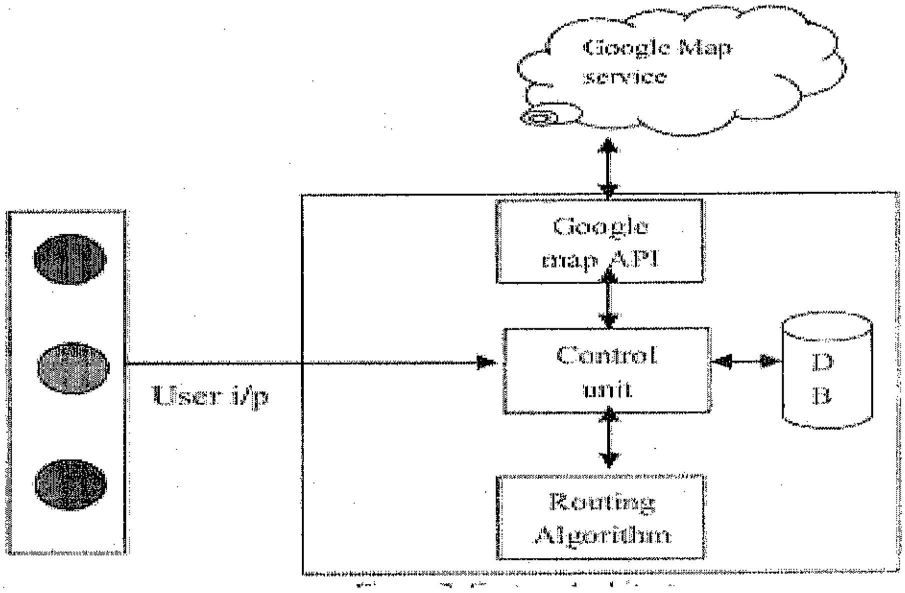


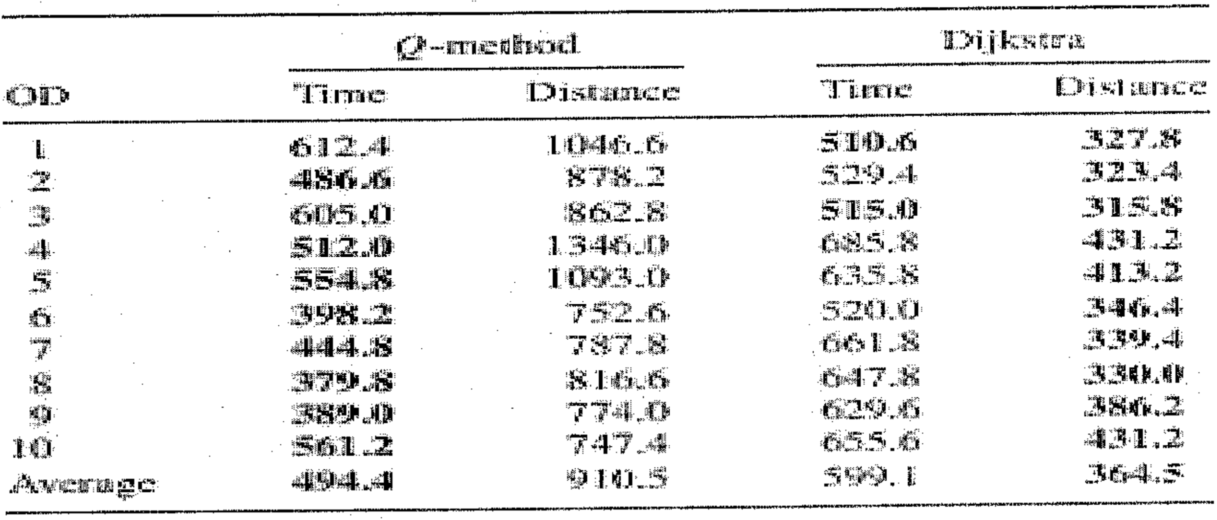
Figure 2.25: System Architecture (Katre & Thakare, 2017)

Mainali *et al.* proposed a dynamic programming algorithm to find the optimal route considering that it should deal with the changes of the traffic situation and multiple criteria. Information from the previous computation is used to find the new optimal route considering user preferences when the traveling time of the road section changes. The proposed method was applied to a real road network to find the optimal route and the results obtained shows that the proposed method can find the user-preferred optimal route. The digital map of the road network of Kitakyushu city was used for the evaluating the proposed method. The traveling time of the road sections is calculated based on the distance of the road section and the maximum speed allowed. The speed shown in Table 2.3 below was used if the maximum speed information is not available. Simulation results showed that the proposed algorithm has better calculation time compared to the Dijkstra algorithm. Figure 2.24 shows computational time comparison.

Table 2.3: Maximum speed of the road section (Mainali, Mabu, Yu, Eto, & Hirasawa, 2017)

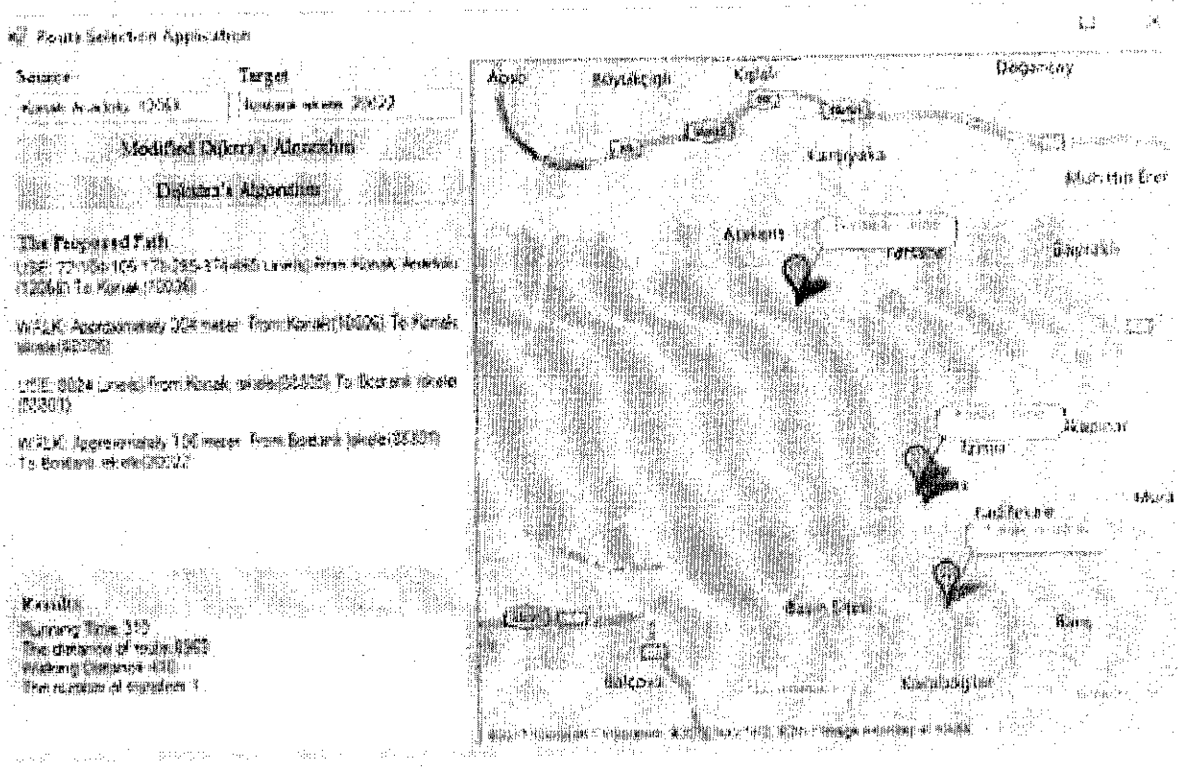
Road type	Speed (km/h)
Highway	80
Urban Highway	80
National road	60
Main prefecture road	50
Main city road	50
General prefecture road	40
General city road	40
Others	30
Not defined	30

Table 2.5: Computational time comparison of the Q-method and the Dijkstra algorithm (ms) (Mainali, Mabu, Yu, Eto, & Hirasawa, 2017)



	Q-method		Dijkstra	
OD	Time	Distance	Time	Distance
1	612.4	1046.6	510.6	327.8
2	486.6	878.2	529.4	323.4
3	605.0	862.8	515.0	315.8
4	512.0	1346.0	685.8	431.2
5	554.8	1093.0	635.8	413.2
6	398.2	752.6	520.0	346.4
7	444.8	787.8	661.8	339.4
8	379.8	816.6	647.8	330.0
9	389.0	774.0	629.6	386.2
10	561.2	747.4	655.6	431.2
Average	494.4	910.5	599.1	364.5

Bozyigit, Alankus and Nasiboglu proposed a modified Dijkstra's algorithm, which was observed to be efficient for route planning in the public transport network in terms of the number of transfers, distance of proposed route and walking distance compared to Dijkstra's algorithm which isn't efficient for public transport route planning, because it doesn't take into account the number of transfers and walking distances. The proposed modified Dijkstra's algorithm was tested on the real world transport network of Izmir and the result was compared with the result of Dijkstra's Algorithm. The two weakness of Dijkstra's algorithm namely ignoring number of transfers and walking distance was overcomed by adding a "small penalty cost" to the cost of the path in cases of the stated weaknesses. In essence the number of repeated walking and the number of transgers are minimized by slightly increasing the distance of the proposed path in the new route selection method.Figure 2.26 shows the route selection application, the application includes GMap.NET(map component), the application uses real-world transport network of Izmir(Turkey) as the dataset and this dataset is obtained from the web page of ESHOT. Table 2.6 shows the average result of the proposed modified Dijkstra's algorithm and Dijkstra's algorithm. (Bozyigit, Alankus, & Nasiboglu, 2017)



Source: Konak [illegible]

Target: [illegible]

The Proposed Path

LINE: [illegible] Line(s) From Konak [illegible] To Konak [illegible]

WALK: Approximately [illegible] meter From Konak[illegible] To Konak [illegible]

LINE: [illegible] Line(s) From Konak [illegible] To [illegible] [illegible]

WALK: Approximately [illegible] meter From [illegible] To [illegible]

Results

Running Time: [illegible]

The distance of route: [illegible]

Walking Distance: [illegible]

The number of transfers: 1

Figure 2.26: The route selection application (Bozyigit, Alankus, & Nasiboglu, 2017)

Table 2.6: The average results of the algorithms (Bozyigit, Alankus, & Nasiboglu, 2017)

Algorithm	***Average Running Time***	***Average Distance of Routes***	***Average Number of Transfers***	***Average Walking Distance***
Dijkstra's Algorithm	242 ms	30.974 km	4.72	0.482 km
Modified Dijkstra's Algorithm	309 ms	33.598 km	2.48	0.396 km

CHAPTER THREE

METHODOLOGY

3.1 Overview of Research Approach

The following are the steps that was taken for this project.

i. Implementation of selected algorithm(Dijkstra and Genetic Algorithm)

ii. Development of Genetic-Dijkstra algorithm

iii. Implementation of the developed algorithm on vehicle route search

iv. Evaluation of the algorithms.

The proposed genetic algorithm approach uses genetic operations namely crossover, selection and mutation to obtain the best optimal route for the vehicle.

3.2 Implementation of Selected Algorithms

Python programming language was used for implementing the data structure and logic of Dijkstra and Genetic algorithm. Ekiti's road network data was downloaded from open street map, and represented as a weighted graph using networkx python package.

3.2.1 Implementation of Dijkstra

Dijkstra's algorithm that was used for this project was already implemented in the Networkx package as functions. The function "dijkstra_path(...)" uses Dijkstra's method to compute the shortest weighted path between two nodes in a graph, It has four parameters namely:

i. Graph

ii. Source node

iii. Target node

iv. Weight

The returns the shortest weighted path from source to target in the graph, in the form of a list of nodes. As illustrated below in figure 3.3, a sample weighted non-directed graph named G has six nodes and 7 edges each of which has a weight (cost). On applying the function **dijkstra_path(G, 1, 5, 'weight')** the function returns [1, 4, 5] which is the shortest path to from node 1 to node 5, because it has the lowest length(cost) i.e. from node 1 to node 4 the length is 2, from node 4 to the destination node 5 the cost is 2, so the total cost of the path is 4.

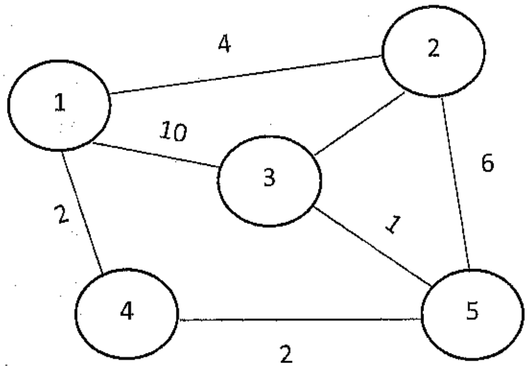


Figure 3.1: A weighted non-directed graph labeled G

3.2.2 Implementation of Genetic Algorithm

Genetic Algorithm was used to search for multiple routes from Ekiti state road map, the generated routes is referred to as solutions. The solutions are ranked from shortest to the farthest, for this project the maximum number of generated solutions is restricted to 10 solutions. Ekiti road map was converted to graph where each road intersection, point of interest or place is considered a node and all the nodes are numbered. The roads in the map are represented as edges in the graph. The distance between the nodes is the length (weight) of the edge between the corresponding nodes.

The solutions provided is made up of several vertices (nodes) arranged in particular order to represent the path in the transportation network. Each solution is a possible path which can be called a chromosome consisting of genes (nodes) where the first gene is the origin node and the last gene represents the destination node. The chromosome (individual solution) length represented mathematically as $L \in [2, n]$ where L is the length (total cost) of the path and n is the total number of the nodes in the road network graph, and each chromosome represents the path from the origin node to the destination node without any loop or duplicate nodes where the maximum length of the chromosome is at most n which is the total number of nodes in the transportation network.

The general genetic algorithm steps for finding out optimal route(s) is described as follows

a. Represent map as weighted graph

b. Input the origin and destination

c. Generate initial population

d. Evaluate initial population using fitness function

e. Perform genetic operation(selection, crossover and mutation)

f. Return best solutions.

Pseudocode for the proposed algorithm is listed as follows:

```
Input network graph
Input origin and destination
Generate initial population of individuals (routes).
Evaluate the initial parent population (using the fitness function)
Loop until termination criteria is satisfied
    Filter population to avoid overpopulation( removing less fit individuals)
    Select chromosomes for reproduction (selection)
    Create offspring using reproduction via operators such as crossover and mutation
    Replace parent population by offspring population and elite parent
Return solution(s)
```

3.2.2.1 Generate Initial Population

Population of chromosomes representing the solution paths was generated. Genetic operations was performed iteratively to generate better generations until the termination condition is satisfied. The origin node and target node are chosen, then the algorithm randomly picks path leading to the target, until the defined initial population size is met. Suppose the network graph of the road network is represented in Figure 3.1 above, using genetic algorithm, randomly generated path from 1 (source node) to 5 (target node) can include [1,3,2,5] , [1.2.3.5] and [1,4,5].

3.2.2.2 Genetic Operations

The operations that was used to evolve better solutions are explained below, they include selection, crossover and mutation

a. **Selection**: The fitness of individual chromosomes was used to rank them. Therefore the fittest of the solutions had higher chance of been selected to produce new offsprings with the intention of improving the fitness.

b. **Cross over operation**: involves picking two or more parent chromosomes to produce new offspring chromosomes. With two chromosomes s1 and s2, two crossover positions in chromosome s1 and s2 was generated randomly, the crossover positions must not be the source node or destination node. Assuming that the gene at position p1 in chromosome s1 is g1, search for g1 in chromosome s2 from position p2. All the genes between position p1 and the destination node in chromosome s1 was replaced by those between gene g1 and destination node in chromosome s2. If the gene g1 is not in chromosome s2 the genes in chromosome s1 remain unchanged. Assuming that the gene at position p2 in chromosome s2 is g2, search for gene g2 from position p1 in chromosome s1. All the genes between position p2 and destination node in chromosome s2 was replaced by those between gene g2 and destination node in chromosome s1. If the gene g2 can't be found in chromosome s1, the genes in chromosome s2 remains unchanged. Figure 3.2shows chromosome s1 and chromosome s2, the crossover position p1=4, p2 = 3, where S = source (origin) node and D = destination node.

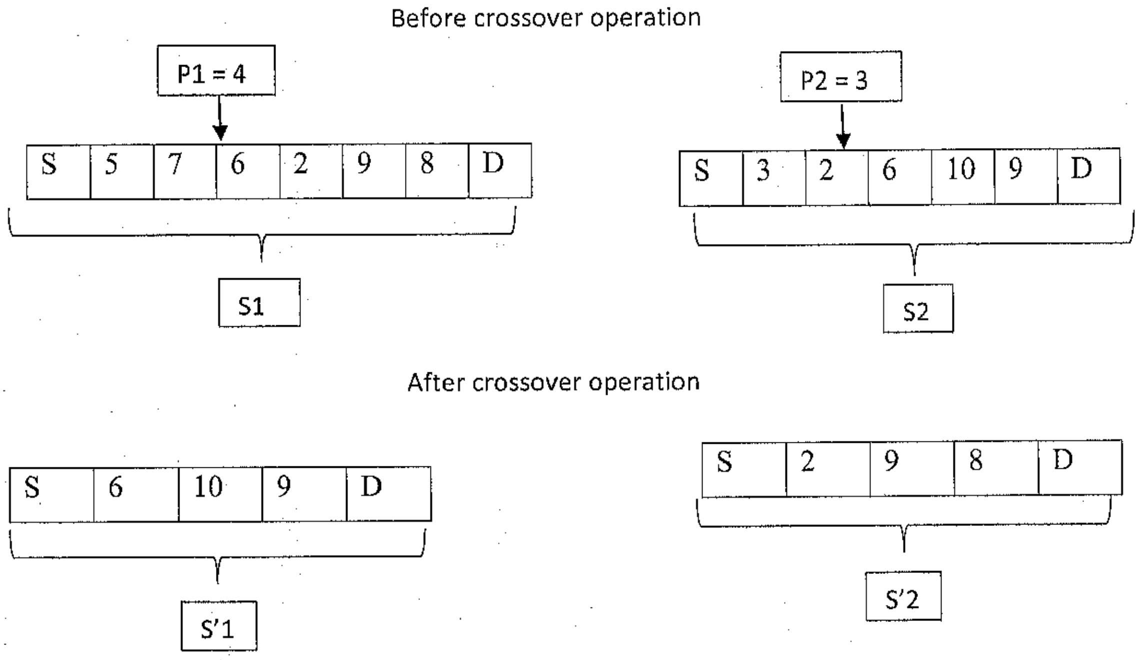

Before crossover operation
P1 = 4
S | 5 | 7 | 6 | 2 | 9 | 8 | D
S1
P2 = 3
S | 3 | 2 | 6 | 10 | 9 | D
S2
After crossover operation
S | 6 | 10 | 9 | D
S'1
S | 2 | 9 | 8 | D
S'2

Figure 3.2: Cross-Over

c. **Mutation operation:** unlike the crossover operation, new offspring is generated from a single parent. Two mutation positions p1 and p2 in chromosome randomly. Make the value at position q1 of gene is x1, and the value at position q1 is x2. A path between x1 and x2 randomly.

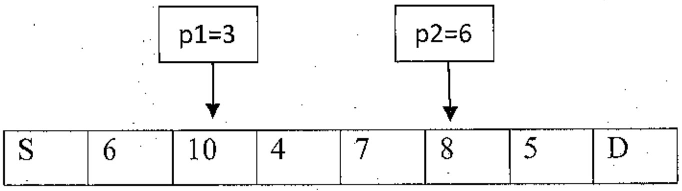


Figure 3.3: Chromosome before mutation

Figure 3.3 shows a sample chromosome with mutation positions p1 = 3 and p2 = 6, the corresponding nodes at mutation position p1 and p2 are node 10 and node 8 respectively. A random new path was generated between node 10 and node 8 to produce a new sub-path. Figure 3.3 shows the chromosome in figure 3.2 after mutation operation.

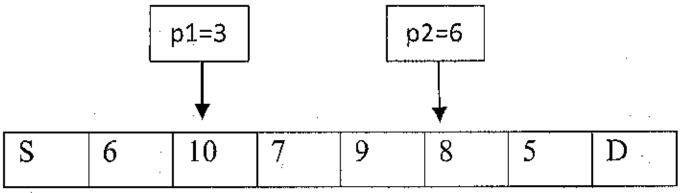


Figure 3.4: Chromosome after mutation

d. **Fitness evaluation:** Fitness function is used to evaluate how fit a solution is, Where the fitness function of the k*th* chromosome mathematically is represented as

$$F(k) = \sum_{i=1}^{n-1} W_E(k[i], k[\, i + 1])$$

n = number of nodes (genes) in the kth chromosome (individual solution), W_E is the weight of the edge between the gene at index ***i*** and the gene at index ***i+1***. Every member (chromosome) of the population is passed to the fitness function the value returned by the fitness function represents is the cost of the path represented by the chromosome.

3.2.2.3Flowchart of Genetic Algorithm

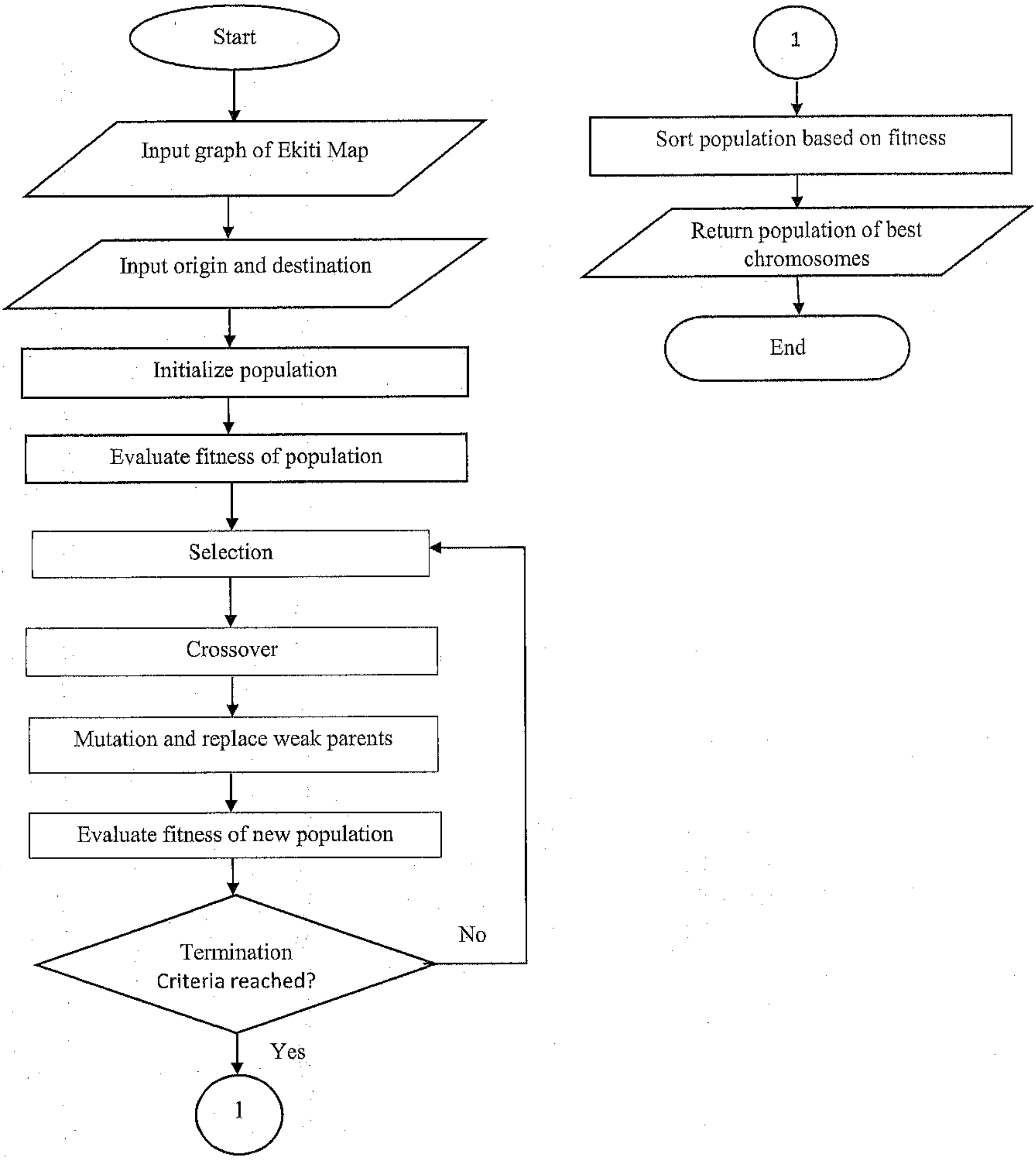

Start
Input graph of Ekiti Map
Input origin and destination
Initialize population
Evaluate fitness of population
Selection
Crossover
Mutation and replace weak parents
Evaluate fitness of new population
Termination Criteria reached?
No
Yes
1
1
Sort population based on fitness
Return population of best chromosomes
End

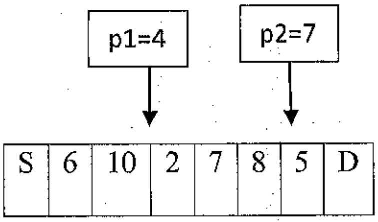
Figure 3.5: Flowchart of Genetic algorithm.

3.3 Development of Genetic-Dijkstra Algorithm

Genetic-Dijkstra algorithm is genetic algorithm based, where Dijkstra algorithm was integrated in the mutation operation of Genetic Algorithm described earlier to improve the speed of producing viable solutions and the quality of solutions produced. Solution in this context means a path from the origin node to the target node, parent means a path in the population list and offspring refers to new path produced from crossover or mutation. This combination produces more than one optimal path from origin to target node and reduced computation time compared to using genetic algorithm alone. The mutation operation of the Genetic-Dijkstra is described below

3.3.1 Mutation of Genetic-Dijkstra Algorithm

Mutation operation doesn't require two parent to produce an offspring like crossover operation, on selecting the parent for example a sample path illustrated in figure 3.6 from the population of solutions. a. Two mutation positions p1 and p2 in chromosome randomly. Make the value at position q1 of gene is x1, and the value at position q1 is x2. A path between x1 and x2 randomly.



S	6	10	2	7	8	5	D

Figure 3.6: Chromosome before mutation

S	6	10	2	9	5	D

Figure 3.7: Chromosome after mutation

Dijkstra algorithm defined as a function in the NetworkX package defined in the implementation of Dijkstra algorithm described earlier is called by passing node at position p1 which is 2 as the source and the node at position p2 which is 5 as the target node. A list of nodes which represent the shortest path from 2 (origin node) to 5 (target node) is returned i.e. [2, 9, 5] which means starting from node 2, passing through 9, then 3 and finally 5 which is the target node.

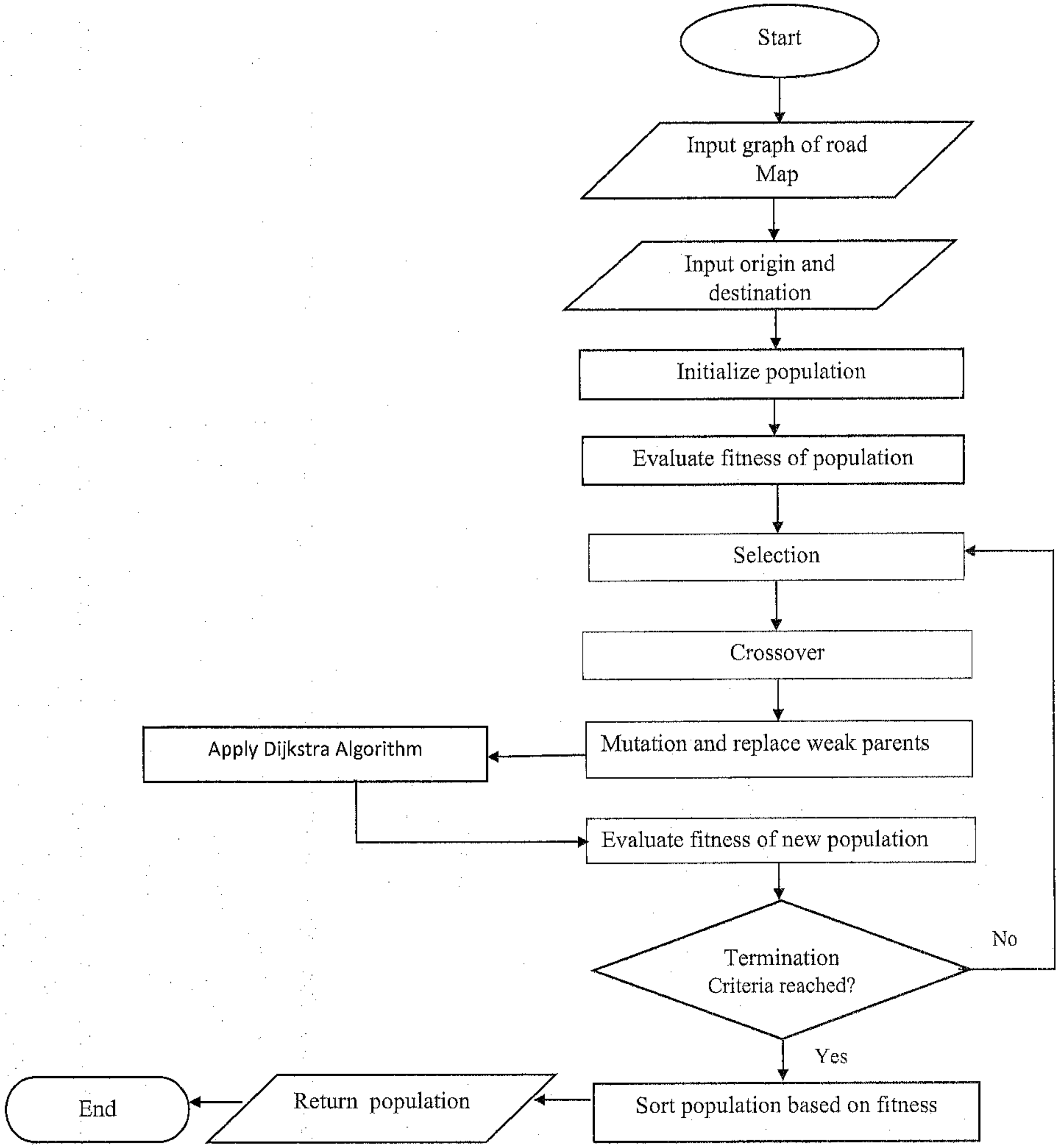

Start
Input graph of road Map
Input origin and destination
Initialize population
Evaluate fitness of population
Selection
Crossover
Mutation and replace weak parents
Apply Dijkstra Algorithm
Evaluate fitness of new population
Termination Criteria reached?
No
Yes
Sort population based on fitness
Return population
End

Figure 3.8: Flowchart of Genetic-Dijkstra algorithm

3.4 System Design of Genetic Algorithm Based Vehicle Route Search

The graphical user interface (GUI) illustrated in figure 3.9 was designed using python programming language, the library used for the design is PyQt5 package which is python package used for GUI designs, in the Pycharm integrated development environment. The GUI uses a combo-box used for selecting the source and target nodes, contains two buttons, the button labeled as Genetic uses pure genetic algorithm to find the optimal routes from the source to the target node while the second button labeled as Genetic+Dijkstra uses the Genetic-Dijkstra algorithm to find the optimal routes from the source to the target node. The population list which is a table at the left side of the interface displays the solutions produced by the algorithm after

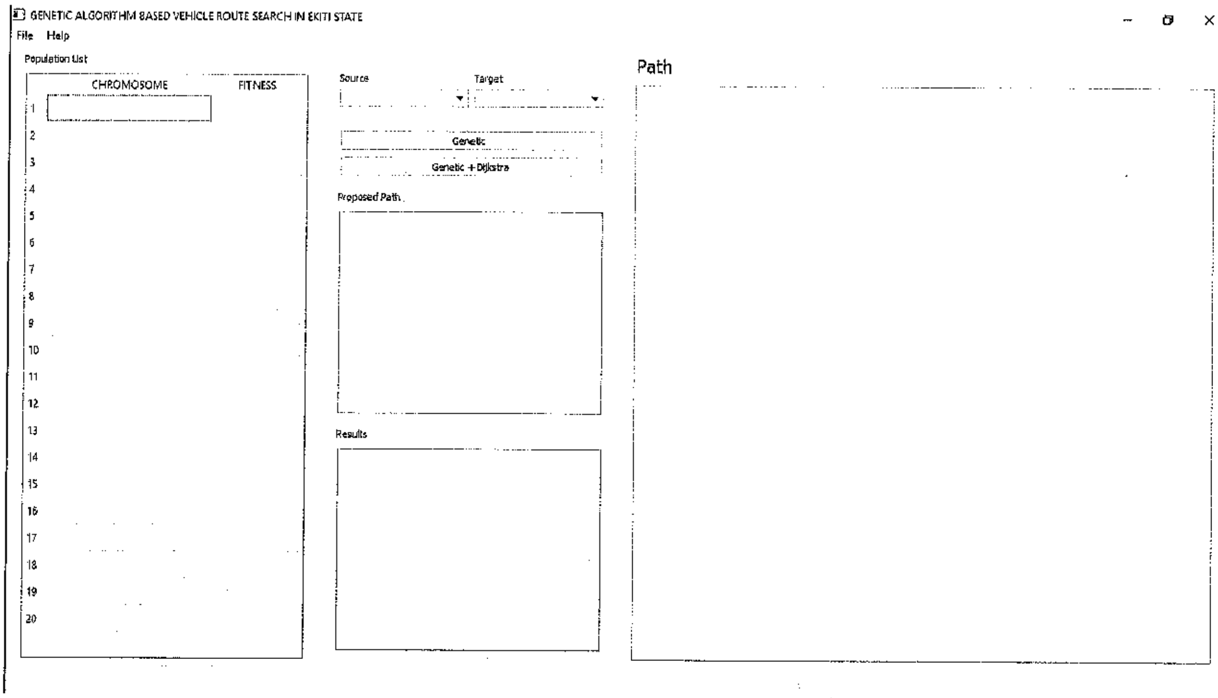
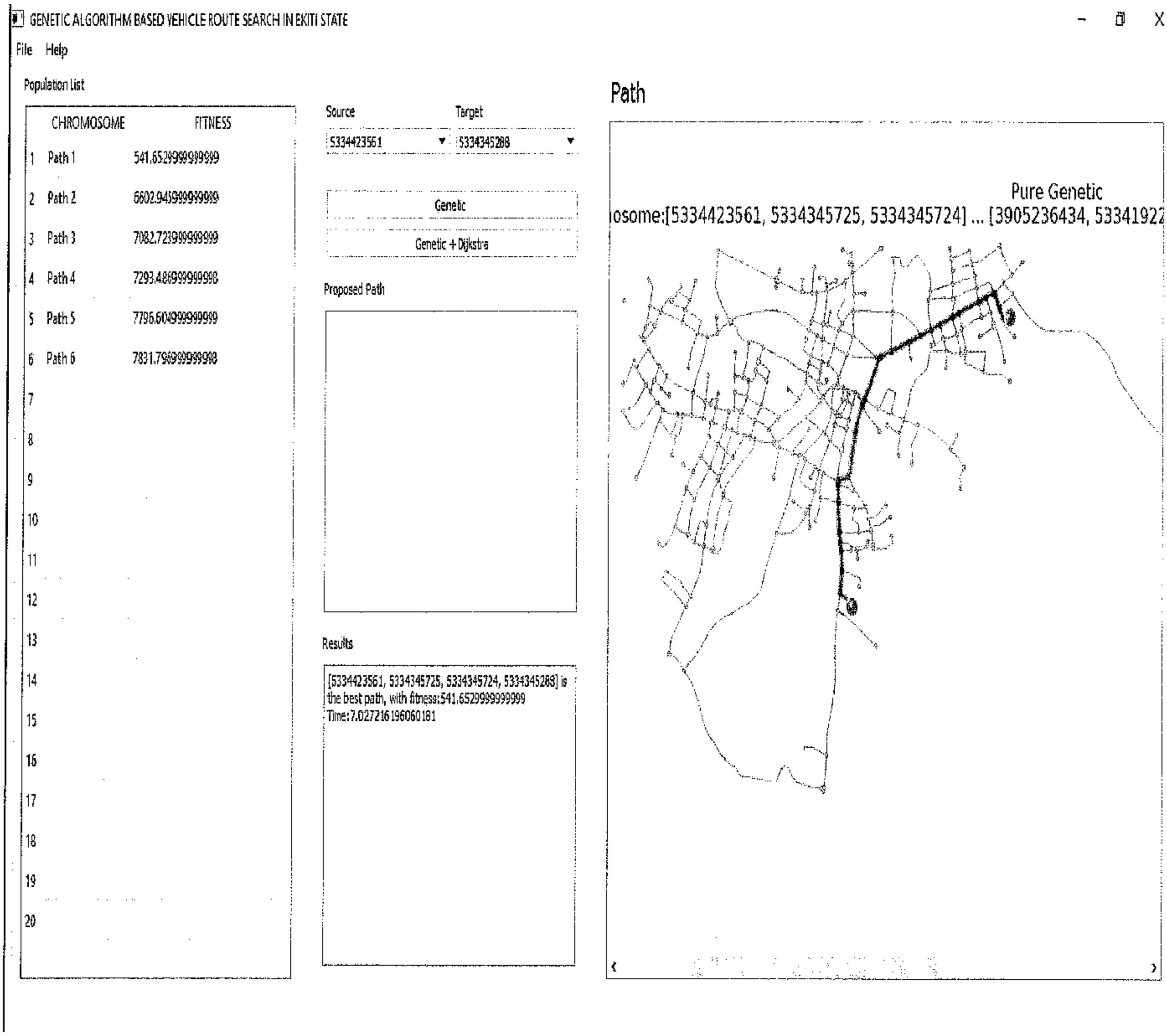


Figure 3.9: Graphical user interface of system design

clicking the buttons Genetic or Genetic+Dijkstra, the fitness of each path which is the distance of the path in metres is also displayed. While the path panel displays a graphical representation of the path from the source node to the destination node as illustrated in figure 3.10



Population List

	CHROMOSOME	FITNESS
1	Path 1	541.6529999999999
2	Path 2	6602.9459999999999
3	Path 3	7082.7239999999999
4	Path 4	7293.4869999999998
5	Path 5	7796.6049999999999
6	Path 6	7831.7969999999998
7		
8		
9		
10		
11		
12		
13		
14		
15		
16		
17		
18		
19		
20		

Source: 5334423561 ▼

Target: 5334345288 ▼

Proposed Path

Results

```
[5334423561, 5334345725, 5334345724, 5334345288] is
the best path, with fitness:541.6529999999999
Time:7.027216196060181
```

Path

Pure Genetic

osome:[5334423561, 5334345725, 5334345724] ... [3905236434, 53341922

Figure 3.10: System showing the shortest path from source node (blue) to target node (red)

3.5 Performance Evaluation of the Developed Algorithm

The coding and simulation activities was performed using SPYDER an integrated development environment (IDE) using the python programming language. The results are presented in tabular format and charts for better graphical and statistical presentation. The following metric was used to evaluate the performance of the algorithms:

a. Processing Time: how long it takes to find the path measured in seconds
b. Distance: measure of the distance of the path found measured in meters

CHAPTER FOUR

RESULT AND DISCUSSION

4.1 Result

In achievement of objective 1 of this research work, the performances of the genetic algorithm and Dijkstra-genetic algorithm investigated in Section 3.2.2 and Section 3.3 respectively. Simulation was performed on a personal computer with intel pentium 1.6GHz processor and 4 Gigabyte Random Access Memory (RAM) using the spyder integrated development environment (IDE).

The steps utilized in the Genetic-Dijkstra algorithm is listed as follows:

i. Initialize the network graph

ii. Create the population

iii. Calculate the fitness function value of each chromosome in the population

iv. Apply selection operation

v. Apply crossover operation

vi. Apply mutation operation (using Dijkstra method)

vii. if termination criteria is reached terminate, otherwise algorithm

The Genetic-Dijkstra algorithm developed is capable of discovering more than one solution for a given source (origin) and destination, providing an option of exploring the next shortest path which exists other than the optimal path.

4.2 Analysis of Genetic Algorithm Compared to Genetic-Dijkstra

Table 4.1 shows the raw result obtained from 72 simulation runs of the Genetic algorithm and Genetic-Dijkstra algorithm, where the time taken for each algorithm to calculate the route and distance of the route is displayed.

Table 4.1a: Table of the simulated data

S/N	Origin Node	Destination Node	Genetic Algorithm		Genetic-Dijkstra algorithm	
			Time	Distance	Time	Distance
1	5332838393	5334215766	89.22744	4677.64	90.8463	4414.75
2	5342819321	5334345659	0.937617	1305.54	0.202966	1269.72
3	3905236357	5238876947	0.656346	1942.64	0.453157	1807.06
4	5342635943	5342635970	2.438287	2187.12	2.343739	1746.16
5	5334192385	5334166805	1.109451	995.35	0.609476	926.3
6	5334423570	5342926360	0.828188	3695.08	0.671921	2922.44
7	5342819415	5334275999	1.218836	2250.75	1.156565	2250.75
8	5342926362	5334192439	4.43786	4964.5	4.095679	4876.93
9	5332838377	5334215729	7.78562	3842.31	6.552325	2930.48
10	5332897615	5334192119	1.234457	1584.46	0.308006	1584.46
11	3905236400	5334275999	0.265694	928.68	0.128005	928.68
12	5334345651	5334276140	0.828233	1582.85	0.348012	1488.77
13	5342635938	5334166813	0.437471	915.15	0.062501	915.15
14	3905236266	5332898134	0.296842	865.37	0.109382	730.55
15	5334276038	3905236275	1.328264	1638.88	0.437529	1532.24
16	5238942608	3905236352	0.343726	842.99	0.078174	842.99
17	5342815694	5334345268	0.171889	1306.28	0.046879	1306.28
18	5334215816	5334424956	0.500187	1195.08	0.109384	1195.08
19	3905236316	3905245329	0.359319	812.51	0.140631	812.51
20	317747160	5342815720	0.781384	1456.59	0.32612	1510.17
21	5332898156	5334275926	0.156292	517.35	0.078132	517.35
22	5334424938	3905236395	0.375052	1389.14	0.140682	974.59
23	3905236266	5334423550	0.203217	990.36	0.031195	522
24	5334490403	3905236282	0.499968	1595.1	0.140636	1370.08
25	5334192383	5334345724	0.875015	1236.06	0.171887	1190.38
26	5342926360	5332898134	0.484453	2179.77	0.093755	2179.77
27	5238942618	5334345712	1.875199	2003.49	0.468778	1894.9

Table 4.1b: Table of the simulated data

S/N	Origin Node	Destination Node	Genetic Algorithm		Genetic-Dijkstra algorithm	
			Time(s)	Distance(m)	Time(s)	Distance(m)
29	5334215806	3905236313	9.401477	3542.35	11.3133	3064.13
30	3905236433	5239110486	13.54391	2974.44	4.594077	2705.72
31	317746932	5239109193	0.218815	795.4	0.0625	795.4
32	5334275968	5239109193	1.88665	2943.45	1.468864	2805.52
33	5334275999	5334345705	2.664407	3192.74	1.890752	1792.68
34	317747170	1098696298	0.750007	3528.37	0.265642	2819.11
35	5334276065	5334215729	1.29692	2559.8	0.919656	2559.8
36	5334215615	5334192444	15.6205	3282.26	14.21349	3239.77
37	5334424957	3905236434	22.40937	4092.45	20.00533	3369.55
38	5334345626	3905236299	0.406336	1320.92	0.124925	1300.98
39	5334345636	5334215812	0.70313	1075.99	0.218767	1075.99
40	3905236275	5334455439	0.328099	681.73	0.125007	676.96
41	5334345268	5334192103	91.71001	4325.67	88.15147	3931.11
42	321720319	5334192119	0.578191	1284.38	0.578173	1284.38
43	317747160	5334192420	7.728369	2245.11	13.50901	2269.95
44	5334192352	5334275615	15.08662	4375.01	28.5516	3407.21
45	5238942618	5239110271	0.322924	979	0.374969	979
46	5334424956	5334192254	0.489278	1656.88	0.60936	1656.88
47	5334424966	5742544792	1.093826	1763.91	0.827951	1527.53
48	5334192420	5334166721	0.062508	174.1	0.062601	174.1
49	5238876947	5332898156	2.265784	2681.71	4.969857	2610.74
50	5334275946	5332838637	3.839296	2678.32	2.437675	2678.32
51	5334192439	3905236352	0.937541	1649.7	2.218919	1649.93
52	5334214617	5332838384	0.406224	840.19	0.54696	840.19
53	5334166780	5334166721	0.109518	150.89	0.062681	150.89
54	5342926362	3905236313	0.358712	3804.41	0.406193	3800.13
55	5334423588	5334345650	0.171885	956.92	0.312706	956.92

Table 4.1c: Table of the simulated data

S/N	Origin Node	Destination Node	Genetic Algorithm		Genetic-Dijkstra algorithm	
			Time(s)	Distance(m)	Time(s)	Distance(m)
56	5238876948	5334491029	7.531951	3946.22	6.548814	3197.07
57	5742544793	5239110486	0.812617	1226.5	0.71868	1224.59
58	3905236332	5334345713	10.14572	3018.35	10.11001	2243.87
59	5342926361	3905236441	15.01596	4872.59	22.71413	4611.66
60	5238942628	5334345714	6.470767	2899.91	6.219337	2531.4
61	5334275999	5334345658	2.046974	2388.24	5.016153	2211.58
62	5334276009	5334275926	0.20309	695.66	0.281319	695.66
63	5334345712	3905236445	2.937562	2372.86	1.531412	2236.44
64	5334455432	5334276038	0.593838	1435.57	1.250105	1435.57
65	5334192444	5342819321	1.109512	1432.97	0.937873	1432.97
66	3905236413	5334491042	1.53138	2529.61	2.109605	2379.22
67	5334166780	5334424972	3.70453	2625.41	1.502941	2613.49
68	5332898125	5334215525	3.452182	2542.67	8.641234	2175.61
69	3905236447	3905236444	0.17178	533.34	0.109383	533.34
70	5239110486	1098698286	0.156305	744.46	0.10932	744.46
71	5334276009	5238942620	0.546911	1624.42	0.468792	1580.79
72	5334276074	5334455438	0.656297	1055.11	0.484238	1002.11

Table 4.2 shows the instances where genetic algorithm was able to get produce solutions faster than genetic-dijkstra algorithm but this doesn't guarantee that it found the shortest path. Table 4.3 shows the instances where genetic algorithm was able to get the shorter path compared to genetic-dijkstra algorithm. Table 4.4 shows the instances where genetic algorithm was able to get shorter distances faster than genetic-dijkstra algorithm.

Table 4.2: Genetic algorithm best time

S/N	Origin Node	Destination Node	Genetic Algorithm		Genetic-Dijkstra algorithm	
2	5334215806	3905236313	9.401477	3542.35	11.3133	3064.13
3	317747160	5334192420	7.728369	2245.11	13.50901	2269.95
4	5334192352	5334275615	15.08662	4375.01	28.5516	3407.21
5	5238942618	5239110271	0.322924	979	0.374969	979
6	5334424956	5334192254	0.489278	1656.88	0.60936	1656.88
7	5334192420	5334166721	0.062508	174.1	0.062601	174.1
8	5238876947	5332898156	2.265784	2681.71	4.969857	2610.74
9	5334192439	3905236352	0.937541	1649.7	2.218919	1649.93
10	5334214617	5332838384	0.406224	840.19	0.54696	840.19
11	5342926362	3905236313	0.358712	3804.41	0.406193	3800.13
12	5334423588	5334345650	0.171885	956.92	0.312706	956.92
13	5342926361	3905236441	15.01596	4872.59	22.71413	4611.66
14	5334275999	5334345658	2.046974	2388.24	5.016153	2211.58
15	5334276009	5334275926	0.20309	695.66	0.281319	695.66
16	5334455432	5334276038	0.593838	1435.57	1.250105	1435.57
17	3905236413	5334491042	1.53138	2529.61	2.109605	2379.22
18	5332898125	5334215525	3.452182	2542.67	8.641234	2175.61

Table 4.3: Genetic algorithm best distance

S/N	Origin Node	Destination Node	Genetic Algorithm		Genetic-Dijkstra algorithm	
			Time	Distance	Time	Distance
1	317747160	5342815720	0.78138423	1456.59	0.326119661	1510.17
2	317747160	5334192420	7.728368998	2245.11	13.50900507	2269.95
3	5334192439	3905236352	0.937541246	1649.7	2.218919277	1649.93

Table 4.4: Genetic algorithm best distance and time

S/N	Origin Node	Destination Node	Genetic Algorithm		Genetic-Dijkstra algorithm	
			Time	Distance	Time	Distance
1	317747160	5334192420	7.728369	2245.11	13.50901	2269.95
2	5334192439	3905236352	0.937541	1649.7	2.218919	1649.93

4.3 Analysis of Genetic-Dijkstra Algorithm compared to Genetic Algorithm

Table 4.5 shows the instances where genetic-dijkstra algorithm was able to produce solutions faster than genetic algorithm. Table 4.6 shows instances where genetic-dijkstra algorithm was able to produce solutions with shorter distance compared to genetic algorithm. Table 4.7 shows instances where genetic-dijkstra algorithm was able to produce solutions with shorter distance faster than genetic algorithm. From the data presented genetic-dijkstra algorithm was observed to perform better than genetic algorithm. Genetic-Dijkstra outperforms genetic algorithm in most cases based on processing time and average distance of the solutions produced.

The results obtained from the simulation, Genetic-Dijkstra algorithm average time is 5.83 seconds which is greater than Genetic algorithm's average time 6.89 seconds and Genetic-Dijkstra algorithm average distance found is 2059.09 meters which is less than Genetic algorithm average distance 2364.65 meters, implying that Genetic-Dijkstra found shorter distance than Genetic algorithm.

Table 4.5a: Genetic-Dijkstra algorithm best time

S/N	Origin Node	Destination Node	Genetic Algorithm		Genetic-Dijkstra algorithm	
			Time	Distance	Time	Distance
1	5342819321	5334345659	0.937617	1305.54	0.202966	1269.72
2	3905236357	5238876947	0.656346	1942.64	0.453157	1807.06
3	5342635943	5342635970	2.438287	2187.12	2.343739	1746.16
4	5334192385	5334166805	1.109451	995.35	0.609476	926.3
5	5334423570	5342926360	0.828188	3695.08	0.671921	2922.44
6	5342819415	5334275999	1.218836	2250.75	1.156565	2250.75
7	5342926362	5334192439	4.43786	4964.5	4.095679	4876.93
8	5332838377	5334215729	7.78562	3842.31	6.552325	2930.48
9	5332897615	5334192119	1.234457	1584.46	0.308006	1584.46
10	3905236400	5334275999	0.265694	928.68	0.128005	928.68
11	5334345651	5334276140	0.828233	1582.85	0.348012	1488.77
12	5342635938	5334166813	0.437471	915.15	0.062501	915.15
13	3905236266	5332898134	0.296842	865.37	0.109382	730.55
14	5334276038	3905236275	1.328264	1638.88	0.437529	1532.24
15	5238942608	3905236352	0.343726	842.99	0.078174	842.99
16	5342815694	5334345268	0.171889	1306.28	0.046879	1306.28
17	5334215816	5334424956	0.500187	1195.08	0.109384	1195.08
18	3905236316	3905245329	0.359319	812.51	0.140631	812.51
19	317747160	5342815720	0.781384	1456.59	0.32612	1510.17
20	5332898156	5334275926	0.156292	517.35	0.078132	517.35
21	5334424938	3905236395	0.375052	1389.14	0.140682	974.59
22	3905236266	5334423550	0.203217	990.36	0.031195	522
23	5334490403	3905236282	0.499968	1595.1	0.140636	1370.08
24	5334192383	5334345724	0.875015	1236.06	0.171887	1190.38
25	5342926360	5332898134	0.484453	2179.77	0.093755	2179.77
26	5238942618	5334345712	1.875199	2003.49	0.468778	1894.9

Table 4.5b: Genetic-Dijkstra algorithm best time

S/N	Origin Node	Destination Node	Genetic Algorithm		Dijkstra-Genetic Algorithm	
			Time(s)	Distance(m)	Time(s)	Distance(m)
27	3905236433	5239110486	13.54391	2974.44	4.594077	2705.72
28	317746932	5239109193	0.218815	795.4	0.0625	795.4
29	5334275968	5239109193	1.88665	2943.45	1.468864	2805.52
30	5334275999	5334345705	2.664407	3192.74	1.890752	1792.68
31	317747170	1098696298	0.750007	3528.37	0.265642	2819.11
32	5334276065	5334215729	1.29692	2559.8	0.919656	2559.8
33	5334215615	5334192444	15.6205	3282.26	14.21349	3239.77
34	5334424957	3905236434	22.40937	4092.45	20.00533	3369.55
35	5334345626	3905236299	0.406336	1320.92	0.124925	1300.98
36	5334345636	5334215812	0.70313	1075.99	0.218767	1075.99
37	3905236275	5334455439	0.328099	681.73	0.125007	676.96
38	5334345268	5334192103	91.71001	4325.67	88.15147	3931.11
39	321720319	5334192119	0.578191	1284.38	0.578173	1284.38
40	5334424966	5742544792	1.093826	1763.91	0.827951	1527.53
41	5334275946	5332838637	3.839296	2678.32	2.437675	2678.32
42	5334166780	5334166721	0.109518	150.89	0.062681	150.89
43	5238876948	5334491029	7.531951	3946.22	6.548814	3197.07
44	5742544793	5239110486	0.812617	1226.5	0.71868	1224.59
45	3905236332	5334345713	10.14572	3018.35	10.11001	2243.87
46	5238942628	5334345714	6.470767	2899.91	6.219337	2531.4
47	5334345712	3905236445	2.937562	2372.86	1.531412	2236.44
48	5334192444	5342819321	1.109512	1432.97	0.937873	1432.97
49	5334166780	5334424972	3.70453	2625.41	1.502941	2613.49
50	3905236447	3905236444	0.17178	533.34	0.109383	533.34
51	5239110486	1098698286	0.156305	744.46	0.10932	744.46
52	5334276009	5238942620	0.546911	1624.42	0.468792	1580.79
53	5334276074	5334455438	0.656297	1055.11	0.484238	1002.11

Table 4.6a: Genetic-Dijkstra algorithm best distance

S/N	Origin Node	Destination Node	Genetic Algorithm		Genetic-Dijkstra algorithm	
			Time	Distance	Time	Distance
1	5332838393	5334215766	89.22744	4677.64	90.8463	4414.75
2	5342819321	5334345659	0.937617	1305.54	0.202966	1269.72
3	3905236357	5238876947	0.656346	1942.64	0.453157	1807.06
4	5342635943	5342635970	2.438287	2187.12	2.343739	1746.16
5	5334192385	5334166805	1.109451	995.35	0.609476	926.3
6	5334423570	5342926360	0.828188	3695.08	0.671921	2922.44
7	5342926362	5334192439	4.43786	4964.5	4.095679	4876.93
8	5332838377	5334215729	7.78562	3842.31	6.552325	2930.48
9	5334345651	5334276140	0.828233	1582.85	0.348012	1488.77
10	3905236266	5332898134	0.296842	865.37	0.109382	730.55
11	5334276038	3905236275	1.328264	1638.88	0.437529	1532.24
12	5334424938	3905236395	0.375052	1389.14	0.140682	974.59
13	3905236266	5334423550	0.203217	990.36	0.031195	522
14	5334490403	3905236282	0.499968	1595.1	0.140636	1370.08
15	5334192383	5334345724	0.875015	1236.06	0.171887	1190.38
16	5238942618	5334345712	1.875199	2003.49	0.468778	1894.9
17	5334215806	3905236313	9.401477	3542.35	11.3133	3064.13
18	3905236433	5239110486	13.54391	2974.44	4.594077	2705.72
19	5334275968	5239109193	1.88665	2943.45	1.468864	2805.52
20	5334275999	5334345705	2.664407	3192.74	1.890752	1792.68
21	317747170	1098696298	0.750007	3528.37	0.265642	2819.11
22	5334215615	5334192444	15.6205	3282.26	14.21349	3239.77
23	5334424957	3905236434	22.40937	4092.45	20.00533	3369.55
24	5334345626	3905236299	0.406336	1320.92	0.124925	1300.98
25	3905236275	5334455439	0.328099	681.73	0.125007	676.96
26	5334345268	5334192103	91.71001	4325.67	88.15147	3931.11

Table 4.6b: Genetic-Dijkstra algorithm best distance

S/N	Origin Node	Destination Node	Genetic Algorithm		Dijkstra-Genetic Algorithm	
			Time(s)	Distance(m)	Time(s)	Distance(m)
27	5334192352	5334275615	15.08662	4375.01	28.5516	3407.21
28	5334424966	5742544792	1.093826	1763.91	0.827951	1527.53
29	5238876947	5332898156	2.265784	2681.71	4.969857	2610.74
30	5342926362	3905236313	0.358712	3804.41	0.406193	3800.13
31	5238876948	5334491029	7.531951	3946.22	6.548814	3197.07
32	5742544793	5239110486	0.812617	1226.5	0.71868	1224.59
33	3905236332	5334345713	10.14572	3018.35	10.11001	2243.87
34	5342926361	3905236441	15.01596	4872.59	22.71413	4611.66
35	5238942628	5334345714	6.470767	2899.91	6.219337	2531.4
36	5334275999	5334345658	2.046974	2388.24	5.016153	2211.58
37	5334345712	3905236445	2.937562	2372.86	1.531412	2236.44
38	3905236413	5334491042	1.53138	2529.61	2.109605	2379.22
39	5334166780	5334424972	3.70453	2625.41	1.502941	2613.49
40	5332898125	5334215525	3.452182	2542.67	8.641234	2175.61
41	5334276009	5238942620	0.546911	1624.42	0.468792	1580.79
42	5334276074	5334455438	0.656297	1055.11	0.484238	1002.11
43	5334424938	5334425013	26.57278	3289.41	22.23536	3027.83

Table 4.7a: Genetic-Dijkstra algorithm best time and distance

S/N	Origin Node	Destination Node	Genetic Algorithm		Genetic-Dijkstra algorithm	
			Time	Distance	Time	Distance
1	5342819321	5334345659	0.937617	1305.54	0.202966	1269.72
2	3905236357	5238876947	0.656346	1942.64	0.453157	1807.06
3	5342635943	5342635970	2.438287	2187.12	2.343739	1746.16
4	5334192385	5334166805	1.109451	995.35	0.609476	926.3
5	5334423570	5342926360	0.828188	3695.08	0.671921	2922.44
6	5342926362	5334192439	4.43786	4964.5	4.095679	4876.93
7	5332838377	5334215729	7.78562	3842.31	6.552325	2930.48
8	5334345651	5334276140	0.828233	1582.85	0.348012	1488.77
9	3905236266	5332898134	0.296842	865.37	0.109382	730.55
10	5334276038	3905236275	1.328264	1638.88	0.437529	1532.24
11	5334424938	3905236395	0.375052	1389.14	0.140682	974.59
12	3905236266	5334423550	0.203217	990.36	0.031195	522
13	5334490403	3905236282	0.499968	1595.1	0.140636	1370.08
14	5334192383	5334345724	0.875015	1236.06	0.171887	1190.38
15	5238942618	5334345712	1.875199	2003.49	0.468778	1894.9
16	3905236433	5239110486	13.54391	2974.44	4.594077	2705.72
17	5334275968	5239109193	1.88665	2943.45	1.468864	2805.52
18	5334275999	5334345705	2.664407	3192.74	1.890752	1792.68
19	317747170	1098696298	0.750007	3528.37	0.265642	2819.11
20	5334215615	5334192444	15.6205	3282.26	14.21349	3239.77
21	5334424957	3905236434	22.40937	4092.45	20.00533	3369.55
22	5334345626	3905236299	0.406336	1320.92	0.124925	1300.98
23	3905236275	5334455439	0.328099	681.73	0.125007	676.96
24	5334345268	5334192103	91.71001	4325.67	88.15147	3931.11
25	5334424966	5742544792	1.093826	1763.91	0.827951	1527.53

Table 4.7b: Genetic-Dijkstra algorithm best time and distance

S/N	Origin Node	Destination Node	Genetic Algorithm		Genetic-Dijkstra Algorithm	
			Time(s)	Distance(m)	Time(s)	Distance(m)
26	5238876948	5334491029	7.531951	3946.22	6.548814	3197.07
27	5742544793	5239110486	0.812617	1226.5	0.71868	1224.59
28	3905236332	5334345713	10.14572	3018.35	10.11001	2243.87
29	5238942628	5334345714	6.470767	2899.91	6.219337	2531.4
30	5334345712	3905236445	2.937562	2372.86	1.531412	2236.44
31	5334166780	5334424972	3.70453	2625.41	1.502941	2613.49
32	5334276009	5238942620	0.546911	1624.42	0.468792	1580.79
33	5334276074	5334455438	0.656297	1055.11	0.484238	1002.11
34	5334424938	5334425013	26.57278	3289.41	22.23536	3027.83

Table 4.7 shows that Genetic–Dijkstra algorithm outperforms Genetic algorithm, figure 4.1 shows the Genetic-Dijkstra has a better processing time than Genetic algorithm where the blue marker represent the processing time of Genetic-Dijkstra and the orange marker represents the Genetic algorithm and figure 4.2 shows that Genetic-Dijkstra was able to find shorter path than Genetic algorithm where the blue marker represent the processing time of Genetic-Dijkstra and the orange marker represents the Genetic algorithm.

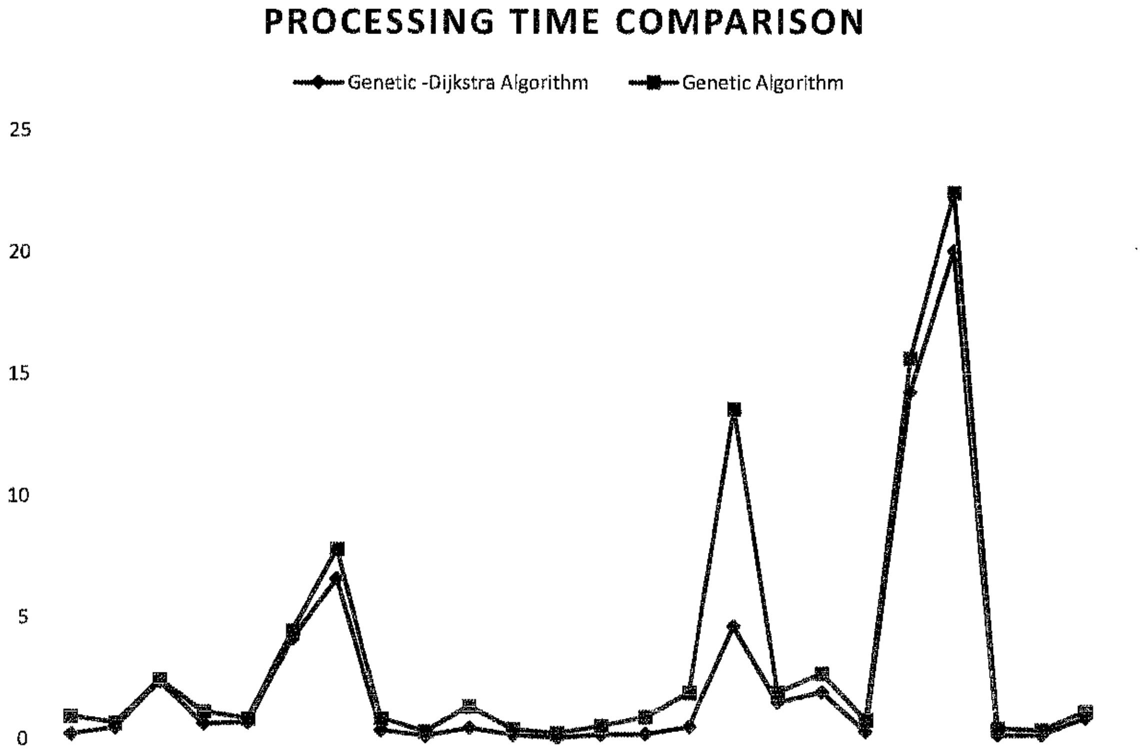


Figure 4.1 Processing Time Chart of Genetic and Genetic Dijkstra algorithm

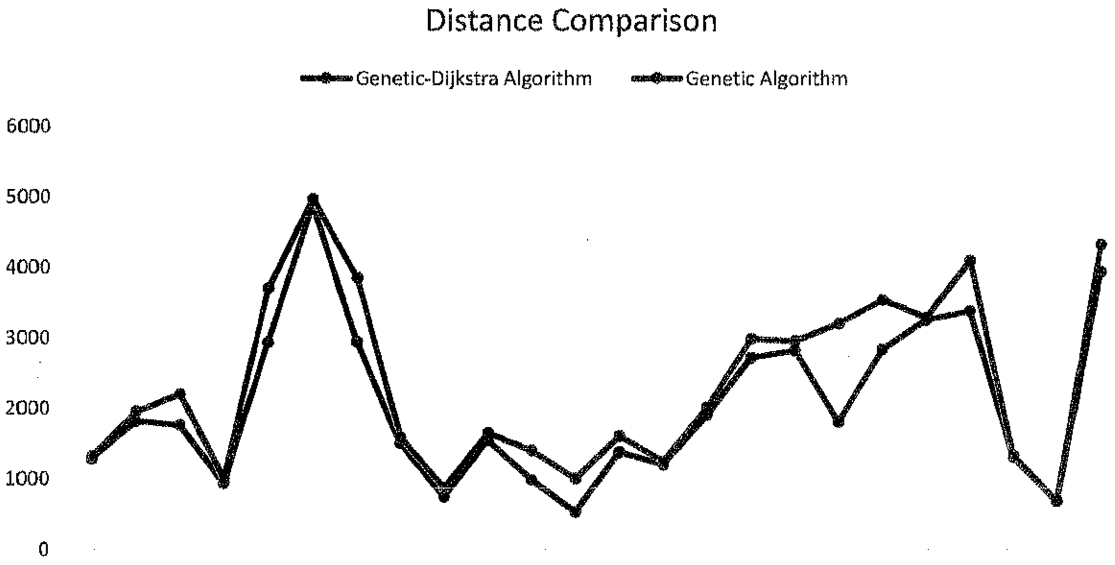


Figure 4.2: Distance chart of Genetic and Genetic-Dijkstra

Figure 4.3 illustrates the road network used for the simulation, the road network is represented as weighted non directed graph, the blue annotated node is the origin node, the red annotated node is the target node. This graph contains 339 nodes.

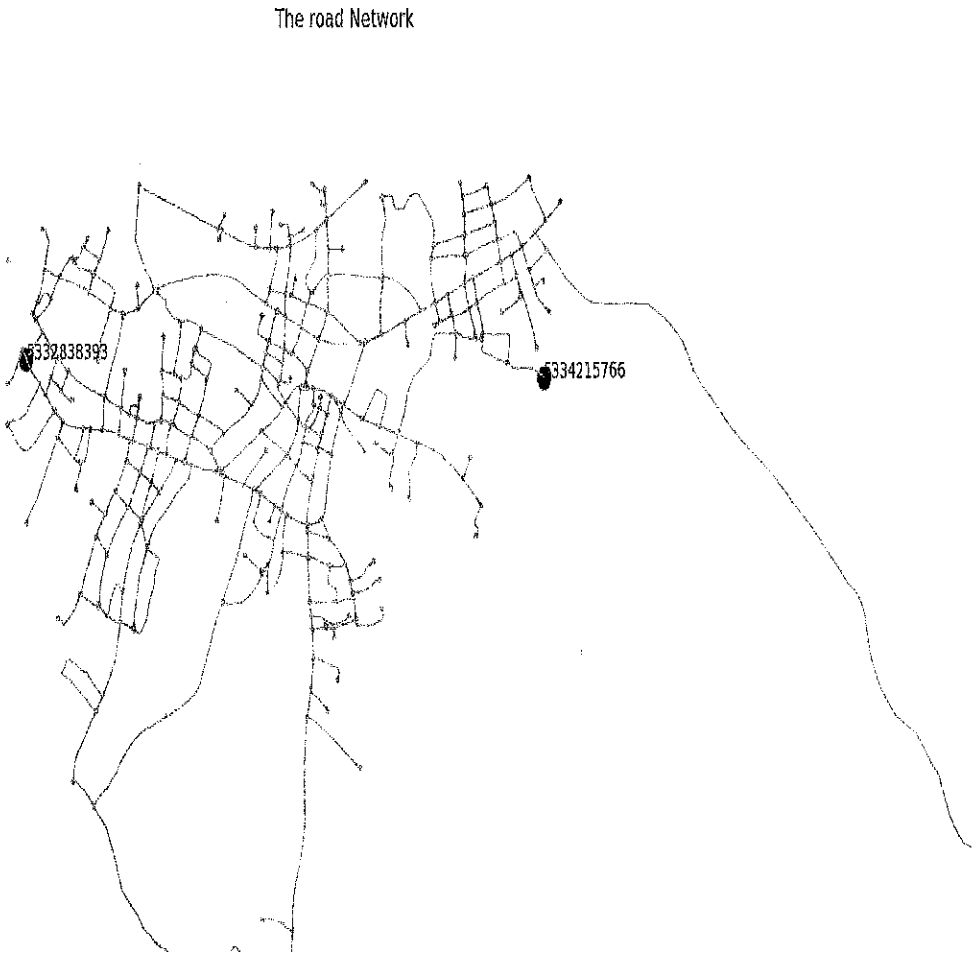


Figure 4.3: The Road network represented as a weighted non-directed graph

Figure 4.4a and Figure 4.4b shows a path derived by using Genetic algorithm from the origin to the destination with a longer cost, while figure 4.5 shows the path derived by using Genetic-Dijkstra algorithm.

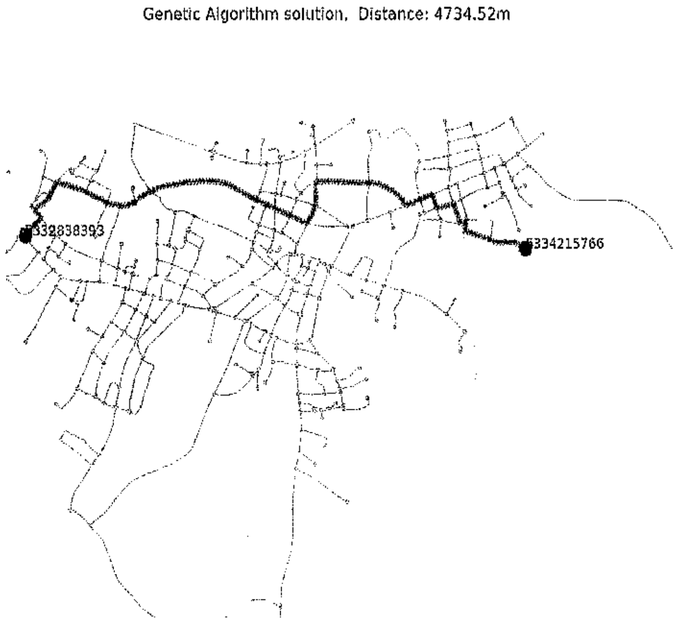


Figure 4.4a: Shortest path derived by Genetic algorithm.

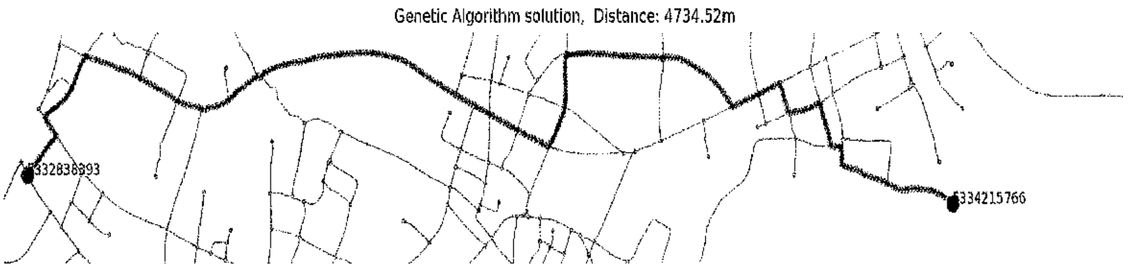


Figure 4.4b: Shortest path derived by Genetic algorithm (zoomed in).

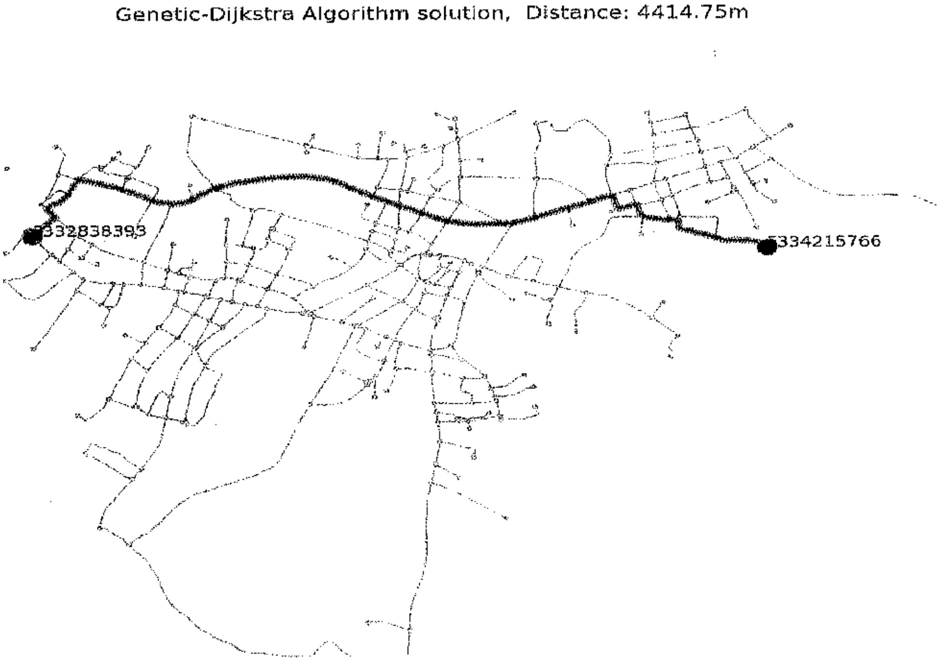

Genetic-Dijkstra Algorithm solution, Distance: 4414.75m
5332838393
5334215766

Figure 4.5a: Shortest path derived by Genetic Dijkstra algorithm.

Genetic-Dijkstra Algorithm solution, Distance: 4414.75m
5332838393
5334215766

Figure 4.5b: Shortest path derived by Genetic Dijkstra algorithm (zoomed in).

CHAPTER FIVE

CONCLUSION AND RECOMMENDATION

5.1 Conclusion

This research work proposed and implemented a genetic based algorithm called Genetic-Dijkstra algorithm for vehicle route search in Ekiti state. This was achieved by investigating the concept of genetic algorithm, Dijkstra algorithm, and how Ekiti road network can be represented as a non-directed weighted graph. This investigation was achieved by implementing two algorithms which pure genetic algorithm and Genetic-Dijkstra algorithm, and representing Ekiti road map as a graph using the python language in the spyder integrated development environment. The reason for implementing Genetic-Dijstra algorithm is to combine the speed of Dijkstra algorithm and flexibility of Genetic algorithm. Genetic algorithm and Genetic-Dijkstra algorithm was simulated 72 times by picking two nodes (origin and target node) for each iteration the nodes were passed to the algorithms and their performance illustrated in table 4.2 to table 4.7 shows that Genetic-Dijkstra algorithm performs better than genetic algorithm. The Genetic-Dijkstra algorithm is capable of producing more than one optimal path (in a short period of time) from the origin node to the destination node which makes it more flexible than using Dijkstra algorithm and faster than using Genetic algorithm.

5.2 Recommendations and Future Research

This research has demonstrated the effect of incorporating Dijkstra algorithm with genetic algorithm, however more work still needs to be done in:

i. Investigating how to improve the Genetic-Dijkstra algorithm performance in larger graphs with more than 300 nodes

ii. Optimizing the generation of initial solution (improving the time taken to create initial population

REFERENCES

Ahn, C. W., & Ramakrishna, R. S. (2002). A genetic algorithm for shortest path routing problem and the sizing of populations. *IEEE Transactions on Evolutionary Computation, volume. 6, NO. 6*, 567 - 579.

Alhalabi, S. M., Al-Qatawneh, S. M., & Samawi, D. V. (2008). *Developing A Route Navigation System.* Southamp: VIJ Press.

Boeing, G. (2017). OSMnx: New methods for acquiring, constructing, analyzing, and visualizing complex street networkx. *Computers, Environment and Urban Systems*, 126-139.

Bonabeau, E., Dorigo, M., & Theraulaz, G. (1999). *From Natural to Artificial Swarm Intelligence.* New York: Oxford University Press.

Bozyigit, A., Alankus, G., & Nasiboglu, E. (2017). Public transport route planning: Modified Dijkstra's Algorithm. *(UBMK'17) 2nd International Conference on Computer Science and Engineering* (pp. 502-505). IEEE.

Byonghwa, O., Yongchan, N., Jihoon, Y., Sungyong, P., Jongho, N., & Jihwan, K. (2010). Genetic Algorithm-based Dynamic Vehicle Route Search using Car-to-Car Communication. *Advances in Electrical and Computer Engineering Volume 10, Number 4*, 1-6.

Carr, J. (2014, May 16). *An Introduction to Genetic Algorithm.* Winchesster: Press LLC.

ChakraBorty, B. (2004). Simultaneous search for multiple routes using genetic algorithm. *IEEE International Conference on computational intelligence for measurement systems and applications.*

Chakraborty, B., Maeda, T., & Chakraborty, G. (2005). Multiobjective route selection for car navigation system using genetic algorithm. *Proceedings of IEEE Mid-Summer Workshop*, (pp. 190 - 195, 28-30).

Chip, R., & Sandhu, A. S. (2016). A design and analysis of EOM for energy saving and effective routing protocol: AODV and AOMDV. Indian Journal of Science and Technology. *Indian Journal of Science and Technology*, 9-19.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). *Introduction to Algorithms.* Cambridge, Massachusetts: The MIT Press.

Data, G. M. (2018, July 4). *Google map.* Retrieved from Ekiti - Google Maps: https://www.google.com/maps/place/Ekiti/@7.697504,5.0467762,10z/data=!4m5!3m4!1s0x1047e00de70f4df3:0xdd1fbccfbb2e1e35!8m2!3d7.6655813!4d5.3102505

Ekiti - Google Maps. (2018, February 22). Retrieved from Google: https://www.google.com/maps/place/Ekiti/@7.6972311,4.7664906,9z/data=!3m1!4b1!4m5!3m4!1s0x1047e00de70f4df3:0xdd1fbccfbb2e1e35!8m2!3d7.6655813!4d5.3102505

El-Sayed, H., Zhang, L., Thandavarayan, G., & Hawas, Y. (2016). ParDiSP: A partition-based framework for distance and shortest path queries on road networks. *IEEE ICC 2016 Ad-hoc and Sensor Networking Symposium.*

Goldberg, D. E. (1989). *Genetic Algorithms in Search Optimization, and Machine Learning.* Upper Saddle River, NJ: Addison-Wesley.

Hamed, A. Y. (2010). A genetic algorithm for finding the k shortest paths in a network. *Egyptian Informatics Journal*, 75-79.

Inagaki, J. (2002). A method of determining various solutions for routing application with a genetic algorithm. *Trans of IEICE, J82-D-1*, 1102 - 1111.

Inagaki, J., Haseyama, M., & Kitajima, H. (1999). A genetic algorithm for determinig multiple routes and its application. *IEEE international symposium circuits and systems*, (pp. 137-140).

Inoue, Y. (2001). Exploration method of various routes with genetic algorithm. *Master's Thesis, Information System Engineering.* Institute of Technology, Kochi.

Jones, M. T. (2008). *Artificial Intelligence.* New Delhi: Infinity Science press LLC.

Katre, P. R., & Thakare, D. A. (2017). A Survery on shortest path algorithm for road network in emergency services. *2017 2nd International Conference for Convergence in Technology*, (pp. 393 - 396).

Khan, S. A., & Abd-El-Barr, M. I. (2018). A Hybrid Ant Colony Optimization Algorithm for Topology Optimization of Local Area Networks. *2018 International Conference on Computer Sciences and Engineering*. IEEE Xplore.

Kinnear, K. E. (2000). *Advances in Genetic Programming.* Cambridge: MIT Press.

Koza, J. R. (1994). *Introduction to Genetic Programming.* Cambridge: MIT Press.

Kramer, O. (2017). *Genetic Algorithms Essentials, Studies in Computational Intelligence.* Springer International Publishing AG.

Krzanowski, R., & Raper, J. (2001). *Spatial Evolutionary Modeling.* New York: Oxford UP.

Kumar, N., devi, R. V., & Sathya, S. S. (2017). Monkey algorithm for robot path planning and vehicle routing problems. *International conference on information, communication & embedded systems*, 405-468.

Liu, Z., Kong, Y., & Su, B. (2016). An Improved Genetic Algorithm Based on the Shortest Path Problem. *International conference on information and automation* (pp. 328-332). Nimgbo,China: IEEE.

Mainali, M. K., Mabu, S., Yu, S., Eto, S., & Hirasawa, K. (2017). Dynamic Optimal route search algorithm for car navigation systems with preferences by dynamic programming. *IEEJ Transactions on electrical and electronic engineering*, 14-22.

Maps of India. (2012, June 18). *What is a Map, Definition of Map.* Retrieved from MapsofIndia: https://www.mapsofindia.com/what-is-map.html

Mashagi, A., & al., e. (2004). Investigation of a protein complex network. *European Physical Journal,* 113-121.

About OpenStreetMap. (2019, February 26). Retrieved from OpenStreetMap: https://www.openstreetmap.org/about

Merriam-Webster. (2018, JUne 28). *Map | Definition of Map by Merriam-Webster.* Retrieved from Merriam-Webster: https://www.merriam-webster.com/dictionary/map

Networkx Home. (2019, February 25). Retrieved from Networkx: https://networkx.github.io

Papakostas, G., Koulouriotis, D., Polydoros, A., & Tourassis, V. (2011). *Evolutionary Feature Subset Selection for Pattern Recognition Applications.*

Pycharm. (2019, February 26). Retrieved from Pycharm: https://www.jetbrains.com/pycharm/features/

Rui-qing, & Zhao, W.-s. T. (2008). Monkey algorithm for global numerical optimization. *Journal of Uncertain systems*, 165-176.

Sedgewick, R., & Wayne, K. (2011). *Algorithms Fourth Edition.* New York: Addison-Wesley.

SPYDER. (2019, February 25). *Spyder Website.* Retrieved from Spyder: spyder-ide.org

Stalling, W. (1998). *TCP/IP and ATM Design Principles.* Englewood Cliffs, NJ: Prentice-Hall.

Sur, C., & Shukla, A. (2013). Adaptive & discrete real bat algorithms for route search optimization of graph based network. *2013 International Conference on Machine Intelligence and Research Advancement*, (pp. 120-124).

Uppalancha, B., & Bachupally, N. (2015). Optimizing the Robot's path using Dijkstra algorithm. *International Journal of Innovative Research in Science, Engineering and Technology*, 4423-4430.

Vaira, G. (2014). *Genetic algorithm for vehicle routing problem.* Vilnius University.

Wen, W., & Hsu, S. W. (2005). A route navigation system with a new revised shortest path routing algorithm and its performance evaluation. *WIT Transactions on the Built Environment, Vol 77* (pp. 733 - 743). WIT Press.

Wu, Q., & Shan, J. J. (2000). *The application of genetic algorithm in geographical information system network analysis.* West Lafayette: Purdue University.