UNIVERSITY OF CAPE TOWN

FACULTY OF ENGINEERING AND THE BUILT ENVIRONMENT

DEPARTMENT OF CIVIL ENGINEERING

THE DESIGN OF PUBLIC TRANSIT NETWORKS WITH HEURISTIC ALGORITHMS: CASE STUDY CAPE TOWN

THIS THESIS IS SUBMITTED TO THE DEPARTMENT OF CIVIL ENGINEERING, IN

PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR AN

MSc DEGREE IN CIVIL ENGINEERING

Prepared by: **Obiora A. Nnene**

Supervisor: **A/Prof. Mark Zuidgeest**

Co-supervisor: **Dr. Edward Beukes**

October, 2014

Plagiarism Declaration

Supervisor: Associate Professor Mark Zuidgeest

Co Supervisor: Dr. Edward Beukes

1. I know the meaning of plagiarism and declare that all the work in the document, save for that which properly acknowledged, is my own.

2. I have used the Harvard Convention for citation and referencing. Each significant contribution to and quotation in this research forms the work or works of other people that has been attributed and has been cited and referenced.

3. I have not allowed and will not allow anyone to copy my work with the intension of passing it as his or her own work.

Name: Obiora A. Nnene Student Number: NNNOBI002

Date: 10/14/2014 Signature: Signed by candidate

Dedication

To God Almighty and my precious family.

Abstract

The Transit Network Design Problem (TNDP) is well-researched in the field of transportation planning. It deals with the design of optimized public transportation networks and systems, and belongs to the class of non-linear optimization problems. In solving the problem, attempts are made to balance the tradeoffs between utility maximization and cost minimization given some resource constraints, within the context of a transportation network. In this dissertation, the design of a public transit network is undertaken and tested for Cape Town. The focus of the research is on obtaining an optimal network configuration that minimizes cost for both users and operators of the network. In doing so, heuristic solution algorithms are implemented in the design process, since they are known to generate better results for non-linear optimization problems than analytical ones. This algorithm which is named a Bus Route Network Design Algorithm (BRNDA) is based on genetic algorithms. Furthermore, it has three key components namely: 1) Bus Route Network Generation Algorithm (BRNGA) – which generates the potential network solutions; 2) Bus Route Network Analysis Procedure (BRNAP) – which evaluates the generated solutions; 3) Bus Route Network Search Algorithm (BRNSA) – which searches for an optimal or near optimal network option, among the feasible ones. The solution approach is tested first on a small scale network to demonstrate its numerical results, then it is applied to a large scale network, namely the Cape Town road network.

Keywords: Transit Network design, Heuristic Algorithm, Objective function, Network Analysis, Genetic Algorithm, Meta Heuristic

Acknowledgement

I would like to express my sincere thanks and appreciation to the following:

- My supervisor A/Prof Mark Zuidgeest and co-supervisor Dr. Edward Beukes. I appreciate the opportunity to work with and learn from two excellent gentlemen. Your work ethic greatly inspired me, and there is no way the dream of this work would have been realized without your technical guidance.
- My parents Mr. and Mrs. C. E. Nnene, sisters (Onyinye, Amaka and Chi) and my dearest 'uga' (Kelechi). You all constantly prayed for and supported me while I worked on this thesis. Thank you very much and I love you all.
- My friends Tosin Oladele, Mphoentle Kgatshe and the unseen ones at stackoverflow.com. Tosin and Mpho were constant sources of encouragement as we worked on our respective dissertations. The guys at Stackoverflow, though I never met them helped me troubleshoot knotty programming challenges when they arose. Thank you all
- The people at the Center for Transport Studies for their kindness and support to me while I worked on this research.
- Lastly the University of Cape Town's ICTS High Performance Computing facility were some of my preliminary computations were performed.

Table of Contents

List of Figures

List of Tables

Abbreviations

AI	Artificial Intelligence
AON	All-or-Nothing
BFSA	Breadth First Search Algorithm
BRNAP	Bus Route Network Analysis Procedure
BRNDA	Bus Route Network Design Algorithm
BRNGA	Bus Route Network Generation Algorithm
BRNSA	Bus Route Network Search Algorithm
BRT	Bus Rapid Transit
BTND	Bus Transit Network Design
BTNDP	Bus Transit Network Design Problem
CBD	Central Business District
CoCT	City of Cape Town
CoCTTA	City of Cape Town Transportation Area
CoCTMA	City of Cape Town Metropolitan Area
DOT	Department of Transport
ECS	Evolutionary Computation Strategy
EMME	Equilibre Multimodal Multimodal Equilibrium
ESRI	Environmental Systems Research Institute
GA	Genetic Algorithm
GIS	Geographic Information System
IPTN	Integrated Public Transit Network
ITP	Integrated Transport Plan
MBT	Mini-bus Taxi
MNLM	Multinomial Logit Model
NHTS	National Household Travel Survey
NDM	Network Design Model

O-D	Origin - Destination
PRASA	Passenger Rail Agency of South Africa
PSF	Python Software Foundation
PTN	Public Transit Network
PTND	Public Transit Network Design
SA	Simulated Annealing
TAZ	Transportation Analysis Zone
TCT	Transport for Cape Town
TNDP	Transit Network Design Problem
TS	Tabu Search

CHAPTER ONE

1. General Overview

1.1 General Introduction

Public transportation remains one of the most viable options for meeting South Africa's increasing mobility need. This need spurns from the steady growth of the country's population and economy over the years. However, data available in the National Household Travel Survey (Statistics South Africa – NHTS, 2013) shows a decline in the use of public transport in cities across South Africa. The report states that in the period between 2003 and 2013, private car ownership increased from 22.9% to 28.5%. It further reveals the fact that private cars were used for more work trips than any single public transit mode in 2013. This development does not augur well for the country as an increase in private car usage will only exacerbate transportation related exclusion, waste of energy and land resources, as well as encourage congestion, traffic accidents and environmental pollution.

High quality Public Transit Networks (PTN) foster economic development – they facilitate the effective mobility of people, goods and the exchange of information (Rodrigue & Notteboom, 2013). However, PTNs in cities across South Africa score low in terms of quality and efficiency. In Fletterman, (2008), the author opines that this condition can be attributed to the decades of neglect these networks have suffered. A case study is the metropolitan city of Cape Town, which is located in South Africa's Western Cape Province (see Figures 1.1 & 1.2). According to City of Cape Town – ITP, (2006), its public transport network were designed at a time when the Central Business District (CBD) was the sole hub of economic activities in the city. As a result of this, the network has a radial configuration which is focused towards the CBD. Over the years however, the city has experienced a significant redistribution of its population and activity bases, which has in turn altered its urban form. The aforementioned, necessitates a redesign of the city's public transit network to cater for this transformation. However, many years later the network has not been fully upgraded to address these changes and the resultant alterations in travel demand patterns.

Furthermore, Cape Town's PTN(s) – road and rail, which was designed during the pre-democratic era had selective network coverage. As a result, they fostered transport related inequity and exclusion along racial and socio economic lines (Behrens & Behrens, 2004). One consequence of this was that poorer sections of the populace which depended more on public transit for their mobility – captive riders, had to walk longer distances than the more affluent choice riders to access public transit. Notably, the City of Cape Town – ITP, (2006) reports that this trend still exists today, with a number of the inhabitants in the city's southeast-northwest axis still having to walk long distances to work or to access public transport. This trend is validated, by the NHTS report which reveals that 10.2% of workers had to walk for over 15mins to get to their closest public transit node.

The issues discussed above negatively impact the overall efficiency of public transit networks in Cape Town. It is therefore pertinent, that considerable attention must be paid to upgrading the city's public transportation networks, which can potentially improve the attractiveness of public transit utilization within the city. One major step that can be taken in this direction, is in the optimized design or redesign of public transit networks in Cape Town. This is a welcome development, given the reality of economic resource constraints South Africa is confronted with as an emerging economy. Furthermore, it will add value to the nation's economy, by facilitating the provision of more desirable transit services for a greater number of people and goods while consuming less land and energy resources. It is noteworthy, that the South African National Department of Transport (Department of Transport – Public Transport Strategy, 2007) and the City of Cape Town (City of Cape Town – ITP, 2013) both highlight this as one of the main objectives in their plan to provide effective public transportation for the populace. The goal of this research is therefore, to develop an optimization model which will improve the public transit network design process in the City of Cape Town (CoCT), with a focus on bus-based transportation systems. It is expected that the model can be used in the future review and redesign of the Integrated Public Transport Network (IPTN) in the city

Figure 1.1: Map showing the location of Cape Town in the Western Cape Province (Google Earth)

Figure 1.2 Map of Cape Town adapted from Google Earth

1.2 Research Problem

Public transit networks in Cape Town like those in many other South African cities, impose high costs on users and operators of the networks. This condition stems from the resultant inefficiencies linked to their inability to respond to changes in travel demand pattern that has occurred over time. For users, costs typically come in the form of:

- Long travel times.
- Prohibitive walking distances to access points on the networks.
- Indirectness of travel.
- Unsatisfied demand due to poor service coverage.

While operators consider costs in the form of operational inefficiency regarding:

- Number of vehicles operated.
- Cost of personnel remuneration.
- High fuel and maintenance costs.

Some reasons for this present conditions are:

- The techniques used to design the networks, which depended solely on the network designer's previous experience and manual computations.
- The lack of re-design of these networks despite changing demand patterns and their maximum capacities being exceeded long ago.

This current situation poses a challenge to the effective utilization of public transport networks in Cape Town. With recent advances in the field of operations research and computing, it is therefore important that computer based optimization techniques are integrated into the transit network design process, since they are known to yield better results. With the aforementioned discussion as a backdrop, this work proposes to design an optimized bus transit network for Cape Town, using a network design model, based on machine learning principles. It is intended that the model will help generate an optimized bus route network configuration which minimizes the generalized cost of travel for users and operators of public transit networks in the city.

1.3 Research Objective and Questions

The main research objective is to develop a Public Transport Network Design (PTND) model solution based on heuristic techniques, which will design optimized bus networks for the City of Cape Town (CoCT). The optimal network should:

- Minimize travel costs for both users and operators of the network in the city.
- Maximize overall network utilization – satisfied travel demand.

The central research question therefore is, "how can an optimized bus network be designed for Cape Town using heuristic techniques?"

The sub questions are stated as follows:

1) How is the Transit Network Design Problem (TNDP) defined?

- What techniques can be used to solve the Transit Network Design Problem?
- What heuristic techniques have been used to solve the Transit Network Design Problem?
- Which of these techniques is suitable for the design of large transit networks?

2) How will an objective function for the design of public transit network in a city like Cape Town be formulated?

- What are the component terms used in the objective function of a TNDP?
- Will the objective function be minimized or maximized?
- What decision variables were used in formulating the objective function?

3) How will the heuristic algorithm be used to design a bus network?

- How can the initial bus routes be generated in the design of a bus transit network?
- How can the generated bus networks be analyzed?
- How can the bus network objective function values be evaluated?

4) How will the heuristic network design algorithm be tested?

- What input parameters are used in the heuristic algorithm to get the optimized network?

- How can optimal parameter values for the heuristic search algorithm be obtained?
- Does the results show that the final network solution is an optimal or near optimal solution?

5) Conclusion and evaluation

- What are the innovations introduced in this research?
- What are the limitations of the research?
- What are future research directions that can be taken from this research?

1.4 Research Design

The Transit Network Design Problem (TNDP) is well known in literature, it focuses on how to produce optimized transportation networks. Some aspects where the subject finds application are route design, frequency setting, bus-stop placement (Fletterman, 2008) and more recently it was used to minimize transit network emissions, see (Beltran et al., 2009). Since the goal of solving TNDP is to produce enhanced transportation networks, a route alignment design model is proposed in this research. A detailed review of previous approaches used by researches in solving the problem will be presented in chapter two. However, an important research carried out in the area of bus route network design is (Pattnaik, Mohan & Tom, 1998), which will form the basis for this research. In this article, the authors proposed a two-stage model solution that implements a heuristic[1] route generation procedure in its first phase to generate candidate networks. In the second stage of the model, a genetic algorithm (GA) was used to search for the optimal bus network from the set of candidate networks. The model was then tested on a small case study network. Some similarities between the model by (Pattnaik, Mohan & Tom, 1998), and that proposed in this work, are the use of a heuristic algorithm to generate the bus networks and the formulation of an objective function[2] to minimize the generalized cost of transport. The model proposed in this work is a three stage model which comprises: 1) A route generation phase; 2) A network analysis procedure; 3) A solution search algorithm.

[1] Heuristic procedures are algorithms based on machine learning principles, which seek to generate acceptable results to an optimization problem in least amount of time possible.

[2] A mathematical expression which represents the problem to be optimized, given certain constraints (Cormen, T.H et al. 2001)

The key innovations brought to bear in this research are as follows:

- Application of the TNDP – route alignment design knowledge to a large scale public transit network within a developing city context.
- Utilizing a network analysis procedure which implements both a traffic assignment model and a Breadth First Search Algorithm (BFSA)[3] to get the potential and satisfied transit demand on each network solution, which has not been attempted in the literature.

An exploration of the literature reveals that earlier solution models, were either tested on idealized transit networks, or on subsets of a large scale network. Fletterman, (2008) applied the TNDP to a large scale network in South Africa (City of Tswhane), however, the focus of his work was on the optimal placement of bus stops within the network. Another common thread seen in those researches was that the models were only applied to cities of first world countries, which have transportation realities different from those of a developing nation like South Africa. This research intends to address this gap by proposing a model solution that will be applied to a large scale network in a South African city – Cape Town. The network design process will also incorporate measures, to address the unique transportation features of the City of Cape Town discussed in Section 1.1. Details of the steps taken to address these will be presented in chapter four which elaborates on the modelling process. The model, interfaces with Geographical Information System (GIS) data obtained from the city's transportation authority – Transport for Cape Town (TCT) and the Python programming language to generate its final output. A brief outline of the model's components, their data interactions with GIS and the individual roles of the components will be presented in Sections 1.4.1–1.4.2. Figure 1.3 shows the model's components and data exchange.

1.4.1 Geographic Information System (GIS)

ArcGIS® by the Environmental Systems Research Institute (ESRI) is used to manipulate and store geographic data. The GIS is involved in the first step of the model design process. It plays the role of

[3] A Graph search algorithm which searches for all the neighboring nodes to a specified node in a graph. In this work, it is used to obtain the transit demand supply routes for each generated network.

storing, handling and exchanging the data that will be used as input in the model. Key inputs for the GIS software are: the road network, zonal and census survey data. The complete description of the data analysis will be presented in (Section 3.5).

1.4.2 Network Design Model

The proposed network design model is termed a Bus Route Network Design Algorithm (BRNDA). It comprises of three major component phases namely: a network generation phase, network analysis and lastly, a procedure used to search for an optimized network. The model interfaces with the data provided from the GIS to generate the optimized network see (Fig 1.3). Details of the BRNDA are discussed in chapter four.

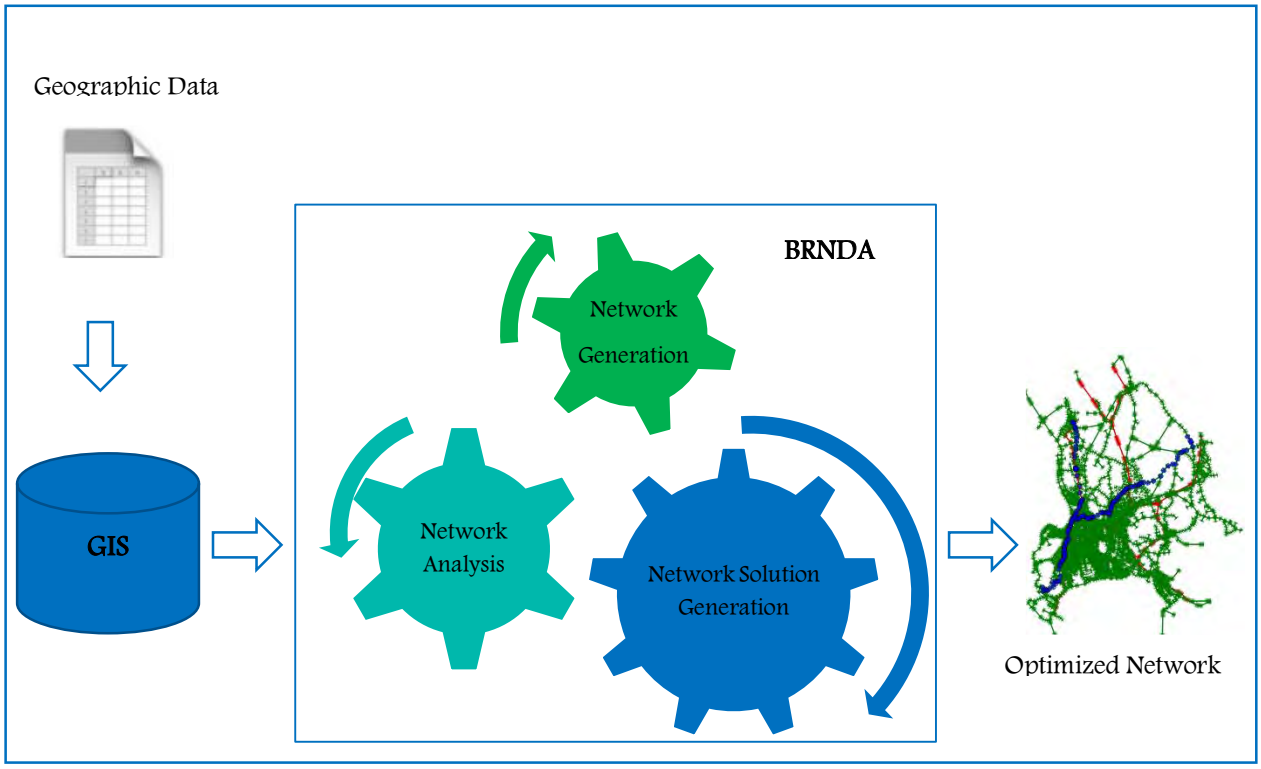

Figure 1.3 The BRNDA components, processes and data interactions

1.4.2.1 Network Generation Phase

This is the first phase of the BRNDA. The candidate routes are generated by randomly selecting routes from a study network – road network of the city of Cape Town. Corresponding bus networks are then

constructed using heuristic algorithms. The generated bus routes are then stored accordingly as potential candidate solutions from which the optimal network solution will be chosen in phase three

1.4.2.2 Network Analysis Phase

This is the intermediate stage between network generation and the search for an optimal solution. Herein, the ridership figure for each network generated in phase one is determined using a traffic assignment algorithm. The output from the traffic assignment is then used to evaluate the given objective function, and the results are assigned to the candidate networks as fitness scores.

1.4.2.3 Solution Generation Phase

The solution generation stage of the solution model is its final stage of the model, wherein it uses an Evolutionary Computation Strategy[4] (ECS) to compare the candidate network fitness scores previously generated in Section 1.4.2.2. The final network solution generated will be the one with the best (smallest) fitness score. This is because the problem is a minimization problem (see Section 1.4).

1.5 Scope of Research

The design of a public transit network is a very complex process. The Transit Network Design Problem (TNDP) broadly comprises two main sub-problems namely route alignment design and frequency setting. This work is focused on route alignment design, which essentially deals with getting the best network configuration that minimizes or maximizes the network designer's stated objective. To achieve this, the network design model (BRNDA) detailed in the research design (Section 1.4.2) is developed. The BRNDA should be capable of designing a large scale bus network for Cape Town that addresses the issues highlighted in Section 1.1. The bus network should also be capable of simultaneously addressing both user and operator perspectives, by ensuring adequate network coverage for the users and minimizing operational inefficiency therewith. The model solution is developed using the Python programming language which was developed by the Python Software Foundation (PSF). Other standard Python libraries were incorporated into the development for this

[4] ECS is a strategy used to solve optimization problems that involves the simulation of biological evolution. The technique, is based on the random selection and variation of populations to arrive at a solution. (Bäck, Fogel & Michalewicz, 2000).

research. No proprietary software was used in the development effort for the model. For the final visualizations, the model is interfaced with the Python visualization tool known as Matplotlib (Hunter, John D, 2007).

1.6 Document Structure

The remainder of the thesis is outlined as follows:

- A review of relevant literature is done in chapter two.
- In Chapter three, the data collection and analysis process is discussed.
- Chapter four deals with the modelling approach used to design the bus transit network.
- Chapter five, focuses on testing the solution model and applying it to the Cape Town network.
- Finally, the work is summarized and conclusions are drawn in chapter six.

A detailed discussion of the model outlined in this chapter will be presented throughout the body of the work. The design flow can be seen in the (Figure 1.4) below.

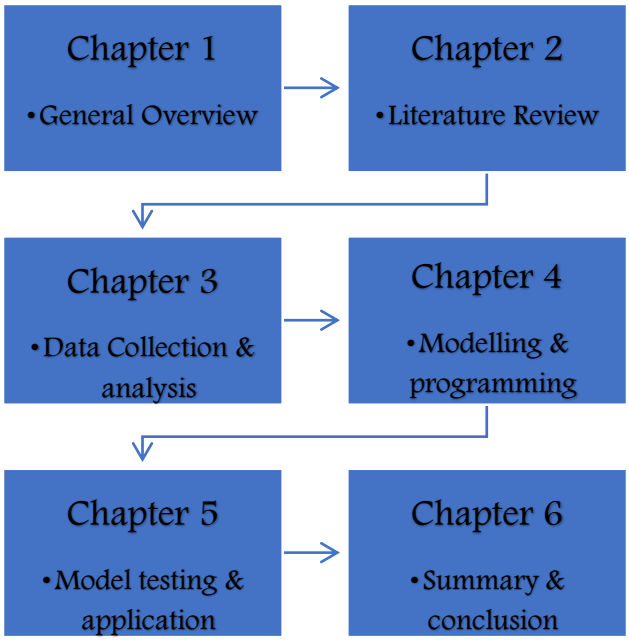

Figure 1.4 Document Structure

CHAPTER TWO

2. Literature review

This review outlines the theoretical basis for the proposed network design model (NDM). The NDM is a Bus Route Network Design Algorithm (BRNDA) which uses heuristic techniques to produce an optimized network. This requires the synthesis of knowledge from several research areas within the field of transportation network design. The broader TNDP and its features are discussed in section 2.1. In section 2.2, the BTNDP is introduced as a sub class of the TNDP. Section 2.3, looks at previous works done in the BTND literature, highlighting some of the notable researches carried out in the period between 1970s till date. A tabulated summary of these works is presented at the end in (Table 2.1). Section 2.4, discusses solution approaches used to solve the problem namely conventional and heuristic techniques. Furthermore, a case is made for heuristic algorithms as a better approach for solving the BTND. Section 2.5, expands on the heuristic techniques previously introduced. It highlights a sub class of Heuristic algorithms known as meta-heuristics, which are considered more effective than traditional heuristic algorithms, and have been widely applied to different aspects of the BTNDP. Three of these algorithms namely Genetic Algorithms (GA), Simulated Annealing (SA) and Tabu Search (TS) are introduced and compared. The choice of genetic algorithm for use in this research is also discussed. Lastly, a conclusion of the chapter is drawn in (Section 2.6).

2.1 Transit Network Design Problem (TNDP)

The Transit Network Design Problem is well-researched in transportation planning literature. It deals with the design of multi-modal public transportation networks such as rail, road and integrated public transit networks. The nature of the problem is that of a multi-objective optimization one, which attempts to balance the tradeoff between utility maximization and cost minimization given resource constraint, within the context of a transportation network. In mathematical programming, it is considered a non-convex[5] problem. Hence, it is usually modelled as a non-linear programming

[5] Optimization problems that have several local optimum solutions and one global optimum solution which may take a lot of time to find.

problem due to its features which include many to many trip distribution, non-linear constraints, and its classification in computational complexity theory as NP-hard[6], see (Magnanti & Wong, 1984; Newell, 1979). The two main components of the TNDP are transit route network design and frequency setting as Chakroborty, (2003) states. Many other applications of the network design problem have been attempted such as: optimal route scheduling (Chakroborty & Wivedi, 2002; Chakroborty, Deb & Srinivas, 1998), intermodal network design (Peng & Fan, 2007), rail frequency optimization (Gallo, Montella & D'Acierno, 2011) and road pricing revenue maximization (Yang & H. Bell, 1998). This research will focus on the design of bus route networks and will be presented in (Section 2.2). However, a discussion of the main features of the TNDP being transit demand, feasibility constraints, objective functions, decision variables and user behaviour, will first be discussed in (Sections 2.1.1 – 2.1.5).

2.1.1 Transit Demand

Variations in the demand for transportation greatly influence the need to either plan for a new transit route/network or make modifications to existing ones. An exploration of previous attempts to solve the TNDP reveals a commonly used assumption that transit demand is fixed. This assumption is simplistic, but makes it easier to resolve the TNDP. Shih & Mahmassani, (1994), asserts that variable travel demand is more reflective of real life scenarios but are more difficult to model. In addition, he criticizes existing demand models as not being completely reliable given their "questionable accuracy". Therefore, despite being a more realistic approach, an implementation of the variable demand is not only extremely complex but also not dependable. This notwithstanding, some researchers have attempted this approach in their proposed solution to the TNDP, see (Kov, Fukuda & Yai, 2006; Lee & Vuchic, 2005; Fan & Machemehl, 2004).

[6] Problems for which a solution cannot be proven to exist in polynomial time (Newell, 1979)

2.1.2 Feasibility Constraints

Constraints are parameters which reflect the limiting conditions of the decision variable in a transit network design problem (TNDP). They generally define the feasibility of the optimization problem and ensure that solutions are obtained within reasonable resource limitations. Some of those commonly used in the literature are maximum/minimum service frequencies, maximum load factor, route length, fleet size and operational cost. Pattnaik, Mohan & Tom, (1998), used a combination of maximum frequencies, allowable fleet size and maximum load factor, while Cipriani, Gori & Petrelli, (2012), used route length and service frequency, to ensure that the length of the resulting routes and their service frequencies does not exceed an acceptable threshold. As stated earlier, these constraints ensure that network solutions are generated within realistic limits, defined by the availability of resources or their scarcity thereof.

2.1.3 Objective Function

Optimization problems are mostly multi-objective in nature and the same can be said of the transit network design problem. They typically attempt to simultaneously optimize two or more conflicting objectives. Van Nes & Bovy, (2000), investigated six different approaches in which the objective function has been used namely: minimization of total travel time and total cost. Others are the maximization of transit operator profits, cost effectiveness, total cost and total passengers. Typically in the literature, a single goal, or a combination of objectives are used in objective function formulations. Mandl, (1980) minimized total travel cost while Fan & Mumford, (2010), used a combined minimization of travel time and user cost.

2.1.4 Decision Variables

In the Transit Network Design Problem literature, a decision variable is seen as a resource which is subject to the transit stakeholder's choice in terms of its allocation see (Curtin, 2004). The limits or bounds of their availability is usually defined by a feasibility condition. Some of the most commonly used decision variables in the design of bus transit networks are route alignment, route frequency,

route length, headway, service frequency and timetables. An investigation of the literature reveals that several researchers have used either a single decision variable, like (Pattnaik, Mohan & Tom, 1998; Newell, 1979), who both used route, while LeBlanc, (1988) used frequency. Others such as (Ngamchai & Lovell, 2003a; Shih & Mahmassani, 1994; Van Nes, Hamerslag & Immers, 1988) all combined route and frequency as their decision variables.

2.1.5 User Behavior

In Transit Network Design literature, trip assignment is considered as a proxy for user behavior. The two predominant classifications of trip assignment in the literature are single and multi-path assignments. Dial, (1971) criticized the single path assignment for its characteristic inability to reflect real passenger behavior, since it assigns all traffic to the single shortest path between selected origin and destination node pairs. The technique is generally known as an All-or-Nothing (AoN) trip assignment. On the other hand, Shih & Mahmassani, (1994) states that multi-path assignments models select a set of acceptable route equivalents based on the probability that the first vehicle to arrive serves that path. He further asserts that in this way, the multi-path assignment accounts for the waiting time at transit terminals since there are multiple acceptable routes. This is considered to be more reflective of passenger behavior in reality. Mandl, (1980) and Rea, (1972) both used the single path assignment method, while most other researchers have implemented a multi-path trip assignment in their proposed solution (Shih & Mahmassani, 1994).

2.2 Bus Transit Network Design Problem (BTNDP)

The Bus Transit Network Design Problem is a sub-class of the Transit Network Design Problem which focuses strictly on the design of route networks and frequency setting for bus transit systems. Among the earliest published research on Bus Network Design was the work done by Lampkin & Saalmans, (1967), who undertook the redesign of a bus route network in North England. They proposed a heuristic model to determine optimal bus networks and their corresponding frequencies. Fan & Mumford, (2010) report that, though their work was the first published attempt to solve the BTND, it

was considered an ad-hoc case study (applicable only to their test case) rather than a generic solution to the broader BTNDP. Since then there was little published research until 1979, which marked the beginning of several attempts to propose standardized solutions to the Bus Network Design Problem. Next, some notable BTND research done in the epoch spanning 1970s till date will be reviewed.

2.3 Previous Research on the Bus Transit Network Design Problem

2.3.1 Dubois, Bel & Llibre, 1979

The authors in Dubois, Bel & Llibre, (1979), proposed a hybrid two stage model, which comprised both route network design and frequency setting. Route and frequency were used as the decision variables, while the optimization objective was to minimize generalized time of travel[7], while using investment cost as constraints. The total cost comprised of investment cost and cost of fleet. The first stage of the model, included two sub-problems, namely: 1) to select a set of streets; 2) to choose their corresponding bus lines. This was done with the aid of a heuristic algorithm. In the second stage, optimal line frequencies were calculated, using a precise analytic model that assumed the streets and bus lines were fixed and took passenger waiting time into account. To get this optimal frequency, a variable demand context was proposed in which the total trip matrix was first determined. Next, a diversion curve based on expected travel times was used to get the public transit share from the previously forecast total trip matrix. The transit demand between each origin-destination pair was treated as a variable that responded to the network design solution. The work included transit trips that require transfers. The model was used in about ten French towns

2.3.2 Leblanc, L. J. 1987

Leblanc, proposed an analytical optimization model, which had only one decision variable namely, frequency. The goal of the model was to improve the service frequency of individual routes within a predetermined network. In doing so, the overall network utilization was increased due to the cumulative increase of users on individual routes which had their service frequency improved.

[7] Cost of travel measured in terms of time.

Operator cost was used as the constraint in this model as increased frequency had a direct relationship with the total operating cost. Therefore, the objective was to minimize operator cost and maximize transit usage. The solution proposed in the work was an implementation of the Hooke Jeeves algorithm which LeBlanc & Abdulaal, (1984), had earlier shown to produce a good solution for optimization problems albeit those with few decision variables. A modal split assignment model which recognizes the frequency of each separate line was implemented in the solution approach used by (LeBlanc, 1988). This was an improvement on the modal split assignment models of that time, as existing ones lacked this feature (Fan & Machemehl, 2004).

2.3.3 Constantin & Florian, 1995

Constantin and Florian proposed an analytical model. According to Fan & Machemehl, (2004), the objective of their model was to minimize waiting time and total travel time while satisfying fleet size constraints. Fan & Machemehl, (2004), further states that the problem was formulated as a non-convex model (see footnote 5 on page 11), while a sub-gradient algorithm[8] was used in the solution technique for the model. Route frequency was used as the decision variable in their model and the solution was applied to a sample transit network in the United States.

2.3.4 Pattnaik, Mohan & Tom, 1998

In Pattnaik, Mohan & Tom, (1998), the authors formulated a BTND model which determines an optimized network alignment for a set of bus routes. The objective function minimizes a total cost expression that was a function of the total travel time representing the user perspective and total bus kilometers which denoted the operator perspective. This approach was previously used in Baaj & Mahmassani, (1995). The decision variable for the model was route alignment while the feasibility constraints used included minimum and maximum frequency, maximum load factor and allowable fleet size. The solution approach comprised a two-stage model that involved generating feasible routes

[8] An iterative method for solving convex minimization problems which was developed by Naum Z Shor. They are particularly useful for solving problems with non-differentiable objectives. (Boyd & Stephen, 2003)

and determining the best options among them, using a heuristic procedure This procedure was first used in Mandl, (1980). While in the second stage, the authors implemented a genetic algorithm. Binary codes were used to denote each route and the GA was coded using two different techniques namely, fixed and variable string coding. Their model was tested on a small network in south India.

2.3.5 Lee and Vuchic, 2005

In their work, the authors Lee and Vuchic proposed a three stage model which comprised: 1) A heuristic route generation procedure; 2) A network analysis and; 3) Artificial Intelligence (AI) based algorithm for route improvement in the last step. They stated that transit demand should depend on the network arrangement and their corresponding route frequencies. Their objective was to minimize user total travel time subject to frequency constraints. To estimate the transit demand and generate the optimal transit network at the same time, they used a mode split model that was adapted from the solution approach used by Rea, (1972). In the last stage, the authors implemented a heuristic route improvement algorithm similar to the AI based model (Transit Route Analyst) designed earlier by Baaj & Mahmassani, (1991). Lee & Vuchic, (2005), considered variable transit demand under a fixed total travel demand. They also carried out a Sensitivity analyses to examine the relationship between the optimal transit network and the design parameters.

2.3.6 Cipriani, Gori & Petrelli, 2012

Cipriani, Gori & Petrelli, (2012) proposed a two stage model comprising a route generation phase and a parallel implementation of genetic algorithm. Their model dealt with a simultaneous determination of a sub-optimal set of routes and their frequencies. The objective function was to minimize total costs while their decision variables were route and frequency. Constraints used in the model were route length, load capacity and maximum line frequency. In the first phase of their model, three types of routes (types A, B and C) were initially generated from the study network (Rome's public transit network) using different criteria. A-type routes connected high demand nodes and addressed the user perspective of minimized transfers and increased trip directness. B-type routes on the other hand,

represented the operator's perspective. These routes connected major transit centers like rail stations. They were generated by a flow concentration procedure used earlier by Carrese & Gori, (2002). This is essentially an iterative All or Nothing Assignment, which aggregates demand volumes on links, and those with the highest volumes considered the best. C-type routes in addition were the existing bus network routes. The second stage of the model was a parallel implementation of a GA to get the optimal network from the pool of generated routes. A tabulated summary of the various researches just discussed is presented in Table 2.1 below.

Year	Author	Objective Function	Decision Variable	Solution
1979	Dubois, Bel & Llibre	Min. Generalized Time	Route & Frequency	H & A
1987	Le Blanc L. J.	Max. transit Utilization	Frequency	A
1995	Constantin and Florian	Min total travel and Wait time	Frequency	A
1998	Pattnaik Mohan & Tom	Min total operator & user cost	Route	H
2005	Lee and Vuchic	Min travel time	Route & Frequency	H & A1
2012	Cipriani et al	Min total operator & user cost	Route & Frequency	H

Table 2.1 Summary of earlier BTNDP solution approaches

A = Analytical, AI = Artificial Intelligence, H = Heuristic

2.4 BTNDP Solution Approaches

An investigation of the literature reveals that Bus Transit Network Design Problem (BTNDP) model solutions are grouped either as conventional or heuristic techniques. Conventional models employ analytical procedures, while heuristic models utilize machine learning principles according to Kepaptsoglou & Karlaftis, (2009). Both groups of solution techniques are discussed below. A descriptive chart of the techniques is also presented in (Figure 2.1) below.

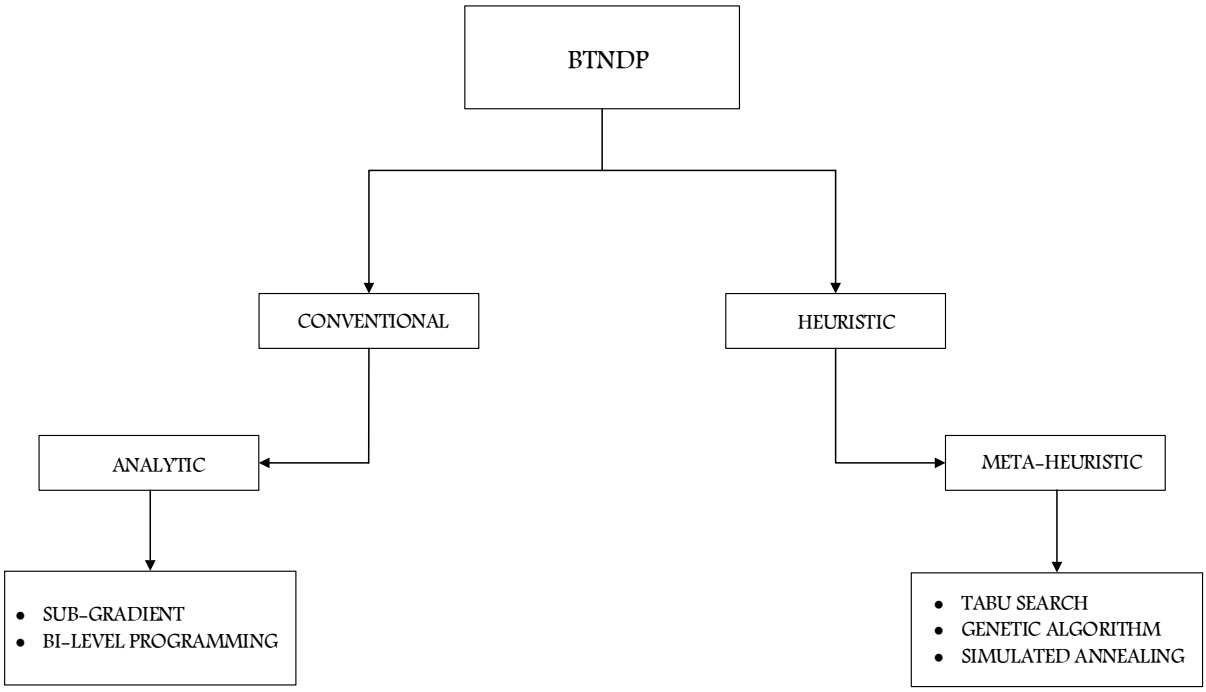

Figure 2.1 Classification of Bus Transit Network Design Problem solution Techniques

2.4.1 Conventional Models

Zhao & Zeng, (2008), identified conventional models as those that use analytical solution techniques to solve the BTNDP. Fan & Machemehl, (2004), further asserts that such models are useful to get one or several design parameters like the route spacing, route length or service frequency of a transit network, but that they are incapable of simultaneously defining the network configuration and fixing service parameters. Instances of such models can be seen in the works of Newell, (1979) and Chien & Spasovic, (2002). Shih & Mahmassani, (2004), also points out that minimizing a total cost function is the main goal of a conventional model and that the cost or objective function reflects user and operator costs but could be expanded to accommodate other cost elements such as the cost of unsatisfied demand. The components of the user cost function, as seen in most literature, combines any of the following parameters:

- Transit fare
- In-vehicle travel time cost
- Waiting time cost

- Time costs of transfers
- Time cost of boarding and alighting the vehicle

In addition, the operator cost is usually estimated in terms of total vehicle time or total vehicle distance. In terms of feasibility constraints, the most commonly used are presented below:

- Maximum/minimum route length
- Maximum load factor
- Maximum frequencies
- Maximum fleet size
- construction or operational cost

Many early approaches such as LeBlanc, (1988), formulated the objective function as a single criterion function or used a multi criteria weighting as in to combine several conflicting objectives into one function, see (Cipriani, Gori & Petrelli, 2012). However, this approach was criticized by Costelloe, Mooney & Winstanley, (2001a), who argues that the approach does not allow for an extensive analysis of trade-offs between the various criteria. More recently multi-objective approaches have been attempted, such as in the work done by Chen et al., (2010), where the BTNDP was represented by three stochastic multi-objective models presented in a bi-level programming framework (A multi-function optimization procedure wherein an inner function known as the lower level problem is nested in the outer or higher level problem (see Shimizu & Kiyotaka, 1997). Their solution comprised of optimizing all the objective functions by simultaneously generating a set of optimal solutions known as a Pareto front. The main limitation of conventional solution approaches as seen in the literature, is that given the difficult and non-convex nature, of the BTNDP as stated in Newell, (1979), it requires extremely expensive resources in terms of computational time and will still not guarantee that a global solution is found. Chakroborty, (2003), also asserts that "the mechanism of describing a problem only through functions (objective function) and inequalities (constraints) in a theoretical setting that is more comfortable with handling real (continuous) variables rather than discrete variables is the primary reason traditional mathematical programming techniques fail to solve problems such as the Transit Route Problem". Shih & Mahmassani, (1994), therefore infers that these

reasons make most analytical optimization techniques applicable only to idealized networks rendering them unable to solve realistic network problems. This has necessitated the development of heuristic techniques which are better suited to solve such realistic network problems.

2.4.2 Heuristic Models

Heuristic approaches were developed in response to the limitations inherent in analytical models as cited in section 2.4.1 above. Pearl, (1984), defines them as schemes for deciding the most effective course of action to take among several options, thereby making it possible to solve complex optimization problems like the TNDP which otherwise could not be solved. A review of the literature recognizes heuristic models as incapable of ensuring that a global optimal solution will be found, hence the near optimal ones generated are considered acceptable. Furthermore, they enable the generation of a network and the computation of its service parameters simultaneously. The inability to do the aforementioned, is considered one of the major shortcomings of conventional models. This has led to a greater acceptance of heuristic methods in solving large scale network design problem. However, heuristic algorithms are problem specific as was the case pointed out earlier in (Section 2.2) where it was reported that Fan & Mumford, (2010) identified Lampkin and Saalmans' model as ad-hoc. This implies that heuristic models can be used to solve only specific problem to which they are applied. A heuristic algorithm applied to a problem, will therefore not find a broad based application in other similar problems. Next, a class of heuristic algorithms known as meta-heuristics, which improves on this limitation, is discussed.

2.5 Meta-Heuristic Algorithms

Meta-heuristic algorithms are a sub-class of heuristic techniques, which can be applied on a broader scale than ordinary heuristics. According to Talbi, (2009), they are problem independent and are capable of being applied to a wide range of optimization problems, by adapting their parameters to the features of a problem to be solved. More researchers rely on these methods due to the better result they give within reasonable time frames and their applicability to large scale network problems. Meta-heuristics, are able to find a global (optimum) or local (near optimum) solution, within a

solution search space and are able to avoid being trapped at a local optimum in the process, see (Fletterman, 2008). The solution space of a typical BTNDP consists of one global optimum solution and several local optimum solutions, located in the troughs of the fictitious solution space illustrated in (Figure 2.2), which has been adapted and modified from (Fletterman, 2008). The troughs represent the solutions due to the nature of the BTNDP as a typical cost minimization problem. As the algorithm searches for an optimal solution, it encounters a local optimum solution, which it temporarily holds until a better one is found. According to Talbi, (2009), this might entail the algorithm's acceptance of poorer solutions, as it converges towards the global optimal solution. He however points out that this mechanism helps to prevent the algorithm from getting stuck at a local optimum. Lastly, Cipriani, Gori & Petrelli, (2012) points to the recent attraction to meta-heuristic solution techniques, owning to improvements in operations research and increased power of computing machines. Sample meta-heuristic algorithms are: 1) Genetic algorithms (GA); 2) Tabu search (TS); 3)Simulated Annealing (SA). These techniques will be discussed in detail next.

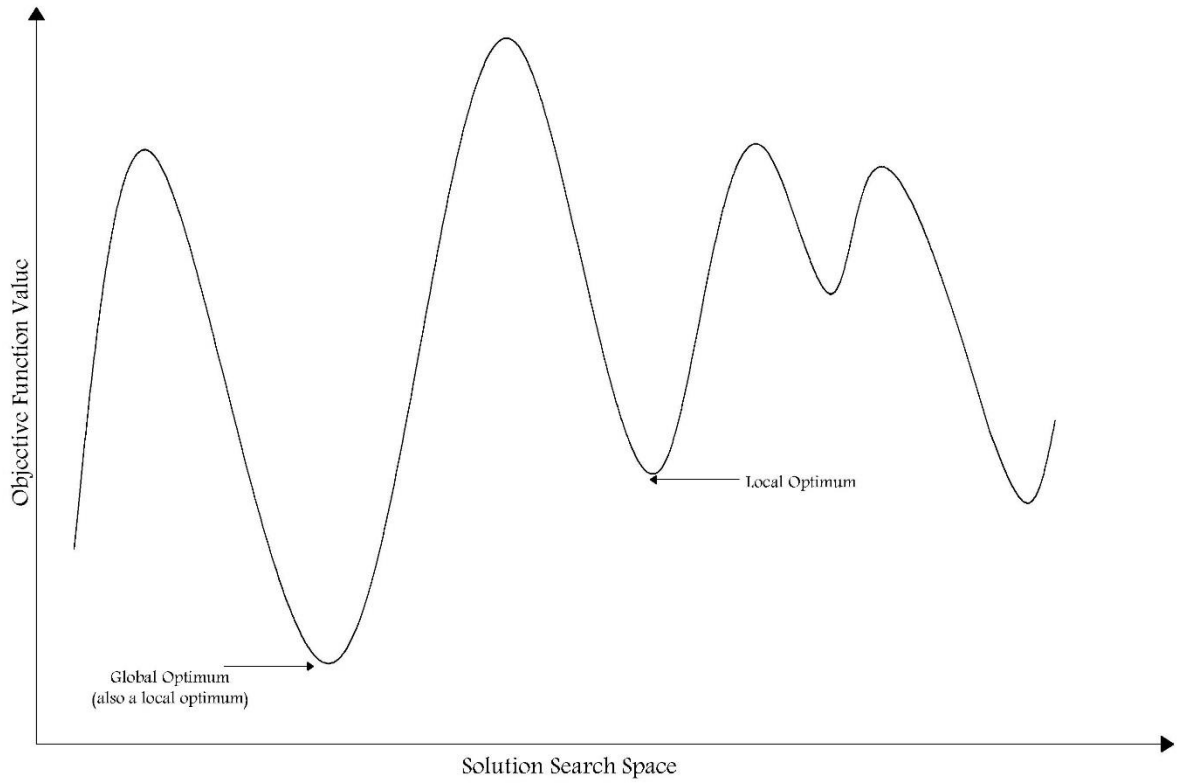

Figure 2.2 Global and local optima in a solution space modified from Fletterman, (2008)

2.5.1 Tabu Search (TS)

Tabu search is a local search algorithm that explores a given solution space by moving among respective solutions in a given neighbourhood. Glover, (1986), describes TS as a "metaheuristic that is super imposed on another heuristic". This description is seen in the literature as being due to the fact that the Tabu search enhances the performance of a local search algorithm by using flexible or adaptive memory structures. In Hillier, F.S. (1995), the basic idea of the technique is identified as being to prevent getting stuck in cycles while exploring a solution space. In addition, it accomplishes this by updating a list of previously selected solutions, marking them as 'forbidden or tabu' and restricting moves, that take the solution in the superseding iteration to those previously marked. However, the Tabu status of a solution might be overridden if the newer solution is better. According to Fan & Machemehl, (2004), this new and better solution is known as 'Aspiration'. Some characteristics of the algorithm include diversification and intensification strategies. Diversification enables the algorithm to maintain the uniqueness of each new solution generated from the previous, while intensification ensures that 'good attributes' are incorporated into the generated solution. A flow chart for a basic Tabu Search algorithm is shown in (figure 2.3) below

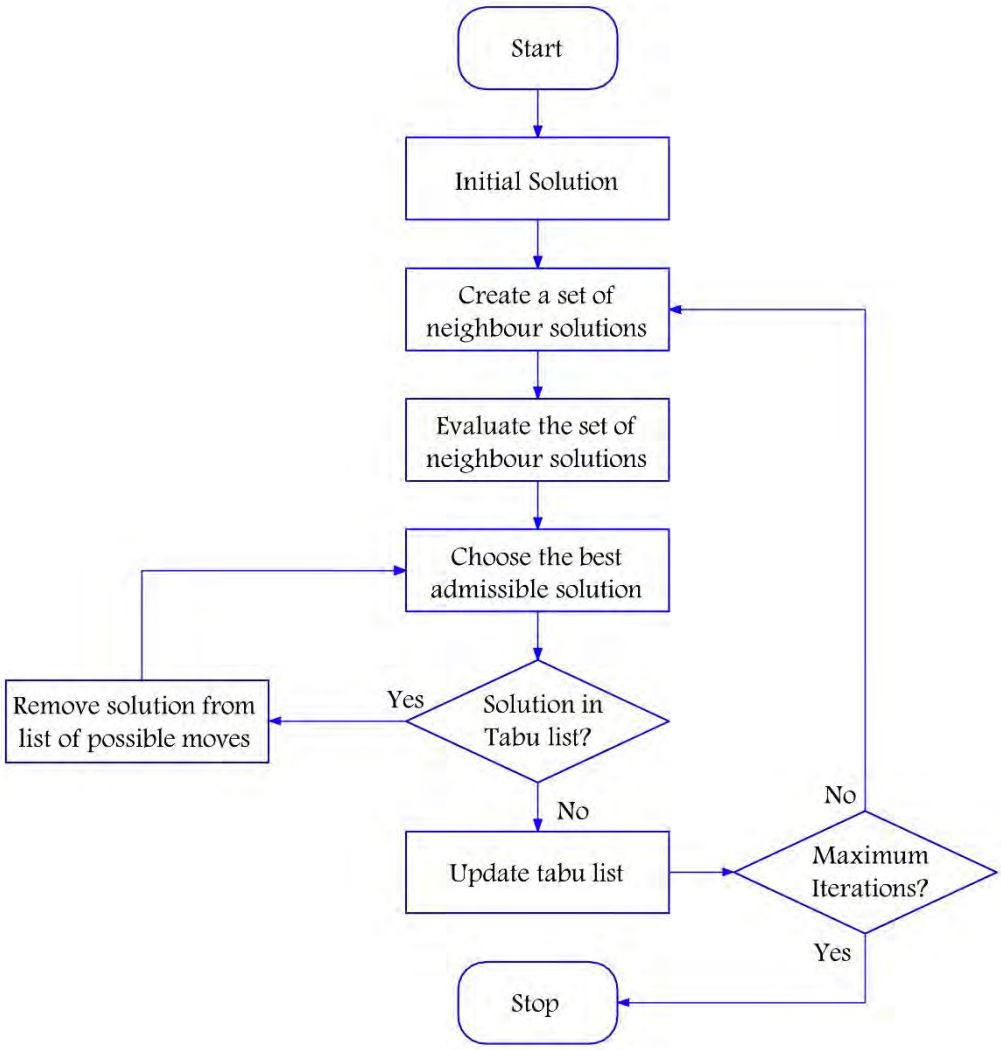

Figure 2.3 Flow chart for a basic Tabu Search algorithm modified from Fletterman, (2008)

2.5.2 Simulated Annealing (SA)

This meta-heuristic algorithm was inspired by the annealing process in metallurgy. One of its uses as seen in the literature is to get acceptable solutions to optimization problems that are near approximations of a global optimum, in solution space which has several local optima. Henderson, Jacobson & Johnson, (2003) points out, that the mode of implementing this algorithm bears some resemblance with the initial heating and controlled or slow cooling of materials during the annealing process. They further state that this process can be observed, in the gradual decrease of its likelihood to accept worse solutions, which is based on the 'temperature' of the algorithm's cooling schedule as it iteratively explores the solution search space. Additionally, the aforementioned is based on the

'temperature' of the algorithm's cooling schedule. Fletterman, (2008), asserts that the initial large probability of accepting non-improving solutions enables the algorithm to evade being stuck at a local optimum, while the slow decrease of this probability allows it to ultimately generate a solution with a low objective function value which is hoped to be the global optimum. A value of temperature (T) depends on its initial value (T_0), the temperature reduction factor (T_{reduc}) the number of iterations done at each temperature count (T_{count}) and the least temperature (T_{min}) needed to be attained before terminating the algorithm, see (Fletterman, 2008). Figure 2.4, shows the flow process for a basic Simulated Annealing process. A generic expression for SA is shown in equation 2.1 below.

$$P_{accept} = e^{\Delta obj / T} \qquad (2.1)$$

Where:

Δobj = Difference between the objective function value of the previous and current solutions

T = Current temperature of the SA cooling schedule

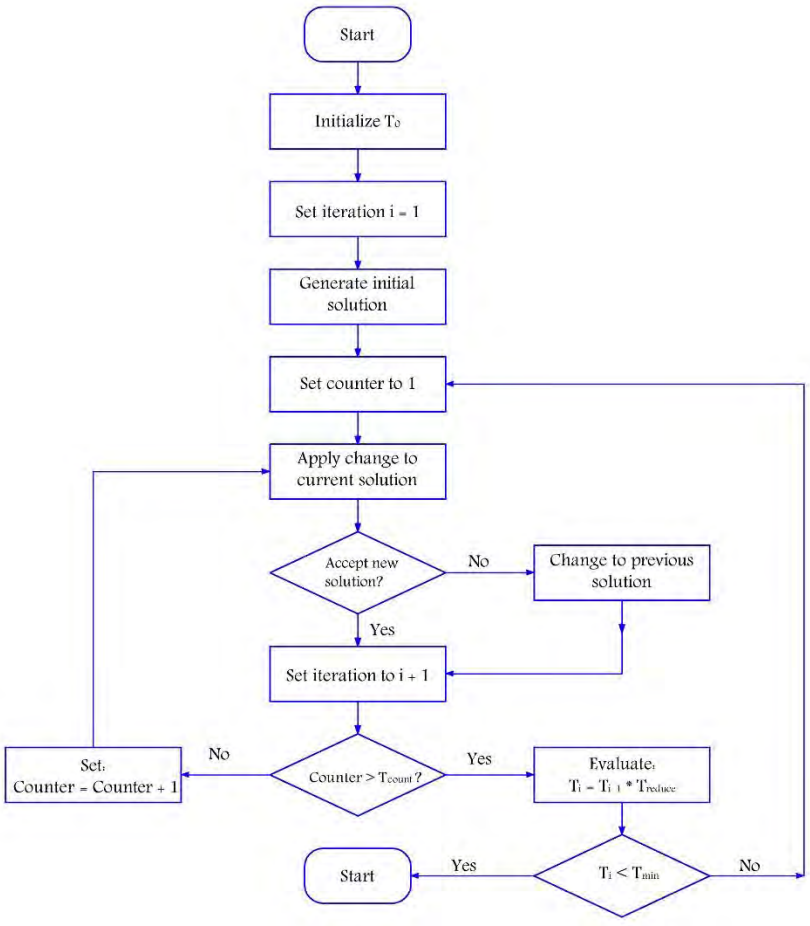

Figure 2.4 Flow for a basic Simulated Annealing algorithm modified from Fletterman, (2008)

2.5.3 Genetic Algorithm (GA)

Poli et al., (2008) identifies genetic algorithms as meta-heuristic search procedures, which are founded on the principle of natural selection and natural genetics. They work by enabling the realization of new and better generations of individuals from existing ones. A typical genetic algorithm, consists of a set of chromosomes[9]. In a typical BTND problem, individual bus routes or networks which make up the population represent the chromosomes. The route/network options or chromosome with the best attributes, represents the global optimum solution. This consideration can be applied to various types of optimization problems, by coding their variables as GA chromosomes. An initial population of the chromosomes is randomly generated in the first step of the GA. Next, each individual in the population is evaluated to determine their fitness or objective function value. The individuals with the best objective function values, then have a higher possibility of being selected as parents for the creation/reproduction, of newer populations which will be initialized in the subsequent generation. These processes are achieved by the action of the GA operators namely: selection, crossover and mutation. This procedure continues, until a predefined termination criteria is attained. Figure 2.5 below, shows the flow chart for a basic genetic algorithm. Genetic algorithms find application in diverse disciplines. Over the years, GA based-models have become one of the most efficient methods for solving the BTND, (Chakroborty, 2003; Chakroborty & Wivedi, 2002; Chakroborty, Deb & Srinivas, 1998; Fonseca & Fleming, 1993; Goldberg & Holland, 1988) are good instances.

[9] Individual members of a population that make up the solution search space.

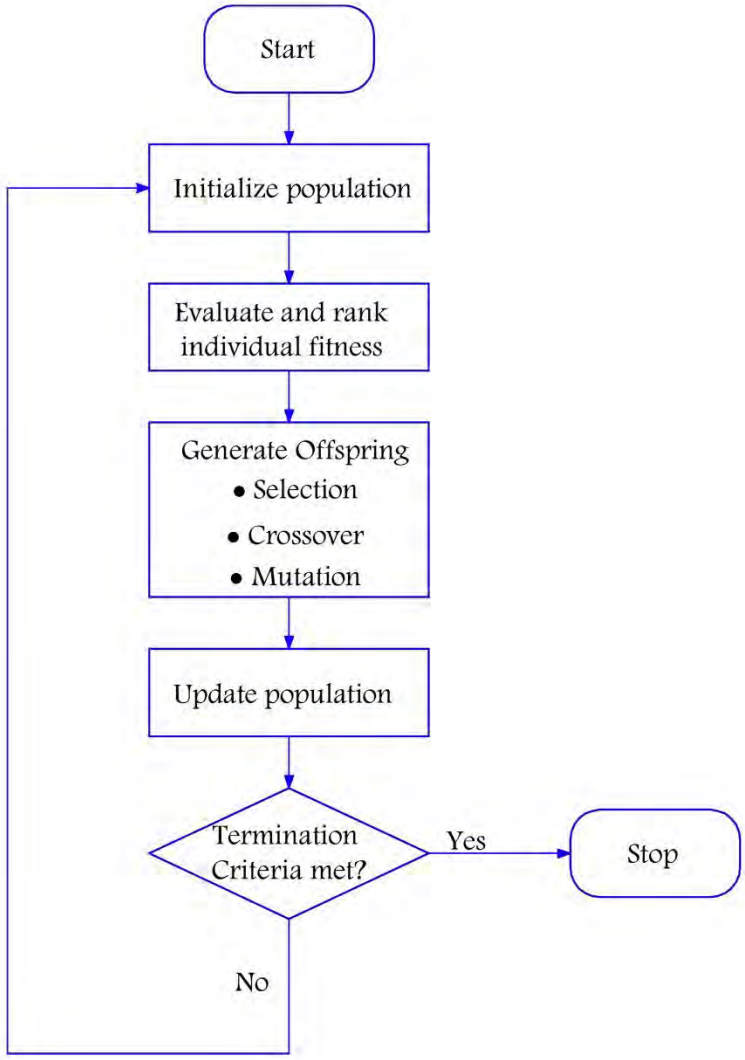

Figure 2.5 Flow chart of a basic Genetic Algorithm

2.5.4 Comparison between TS, SA and GA

In this section, a comparison of the three meta-heuristic algorithms discussed earlier will be done. This is to highlight some of their main similarities and differences it must however be pointed out that their performance is dependent on the unique nature of the problem to be solved, and the size of its solution search space. Talbi, (2009), identifies that they display similar behaviour in their tendency to avoid being stuck at a local optimum while traversing a solution space. In addition, they cannot determine from the onset that a global solution may exist or not for a given problem. Furthermore, they are improvements on traditional heuristic methods discussed in (Section 2.4.2). Lastly, they are

applicable to optimization problems, with both discreet (integer) and continuous (decimal) variables. According to Talbi, (2009), TS and SA are single solution meta-heuristics (S-Metaheuristic); they only work to improve single solution in successive iterations, while GA is population based (P-Metaheuristic) hence it improves a set of solutions at each iteration. Another key difference, between these approaches is found in the strategy they employ to guide their search of a given solution space. Tabu Search employs an intensification and diversification strategy to search for neighbourhood solutions, while genetic algorithms maintain the history of sub-populations and selects new parents in each generation by randomly altering new solutions with the aid of its selection, crossover and mutation operators see (Fan & Machemehl, 2004). Simulated annealing uses parameters like temperature and stopping criteria to handle neighbourhood solution search. Lastly, the GA and SA are referred to as stochastic meta-heuristic algorithm in the literature because they use probabilistic operators like crossover and mutation in the case of GA and the probability of acceptance in the SA. This is different in TS, which does not have randomized variables but depends on its current state saved in memory to perform its search. Hence, TS is also referred to as a deterministic algorithm.

For this research, GA is selected as the preferred solution search algorithm. This is because, genetic algorithms have been used to solve several Bus Transit Network Design problems, see (Bielli, Caramia & Carotenuto, 2002; Cipriani et al., 2005; Ngamchai & Lovell, 2003a; Chien, Yang & Hou, 2001; Fan & Machemehl, 2006). Also, Michalewicz, (1996) states that genetic algorithms search a group of possible points simultaneously within the solution space, rather than a single point as is the case with other meta-heuristic algorithms. In the literature, this is considered an advantage the GA has over TS and SA, as it allows GA(s) to perform more robust and pervasive search across a very large solutions space and in faster time see (Fan & Machemehl, 2004). They are also versatile, in that the parameters of several problems can easily be coded to represent the chromosomes of a genetic algorithm. Lastly, GA(s) use the objective function information of a problem to directly determine the fitness of each individual chromosome in the population. This is contrary to other meta-heuristics like TS that rather uses indirect measures like the cost of making a move. The ease of its operation and the ability to find good solutions, makes the GA a very attractive choice for implementation in this model.

2.6 Conclusion

An extensive literature review has been done in this chapter. The intention is to layout the theoretical foundation for the network design proposed in this dissertation. The characteristic features of the BTNDP was discussed alongside some solution approaches available in the literature. Genetic Algorithm was identified as best suited to implement in the proposed bus network design model. A synthesis of the knowledge acquired in this review will facilitate the design of a bus network that responds to the challenges highlighted in Section 1.1. In the next chapter, a description of the collection and analysis process of the data that will be used in the model is presented. This will however be preceded by a detailed discussion of the transportation context of the City of Cape Town Metropolitan Area (CoCTMA) which is used to test the model.

CHAPTER THREE

3. Data Collection and Analysis

A literature review of the Bus Transit Network Design Problem was presented in Chapter two. In this chapter, the focus will be on the process of collecting and analyzing the data which will be utilized in the design of the proposed bus transit network, using the GIS and the Python scripting language discussed earlier in Section 1.4.1. Some of the key input data required for the network design are:

- Travel demand matrix (Origin – Destination matrix)
- Cape Town public transport network topological data (Nodes and links)
- Transit operational data (Speed, headway, frequency, travel-time).

However, prior to discussing the aforementioned, it is essential to present the transportation context of Cape Town. Therefore, in section 3.1, the characteristic features of Cape Town such as its land mass and population count is presented. In section 3.2, a description of the transportation networks in the city is given, while sections 3.3 and 3.4, focus on the travel demand patterns of the study area alongside its trip generation and attraction characteristics. Lastly, section 3.5, presents the data collection and analysis.

3.1 City of Cape Town Municipal Area (CoCTMA)

The City of Cape Town transportation Area is located within the Cape Town metropolitan municipality, having an estimated land area of 2455 square kilometers. According to the 2011 census data found in Statistics South Africa - Census, (2011), the population of the city is 3.74 million people with an average household size of 3.5, and housing density of 15.3 average dwelling units per hectare. It is reported that 55% of the inhabitants rely on public transportation for their mobility. The current landscape of public transport in the city reveals a non-integrated structure, with rail services being provided by a state owned company, bus services operated by a municipal company and various commercial companies, while minibus-taxis (MBT) are operated privately. Figure 3.1 below shows the population density in the Cape Town metropolitan area.

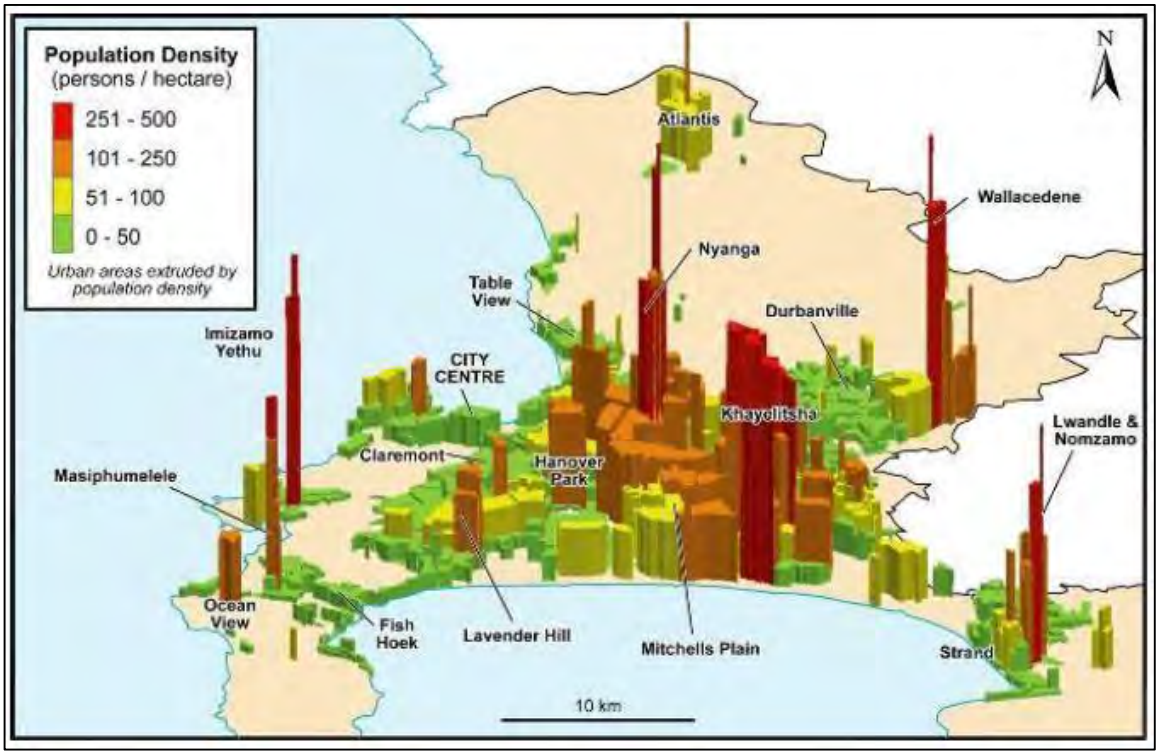

Figure 3.1 Population density of the Cape Town Municipal Area (City of Cape Town – ITP, 2013)

3.2 City of Cape Town Transportation Networks (CoCTTN)

The land-based transportation network in Cape Town consists of two key components, namely road and rail. These are shown in (figure 3.2) below. The road network serves transport vehicles, freight, pedestrian and cyclists while the rail network provides only passenger and freight services.

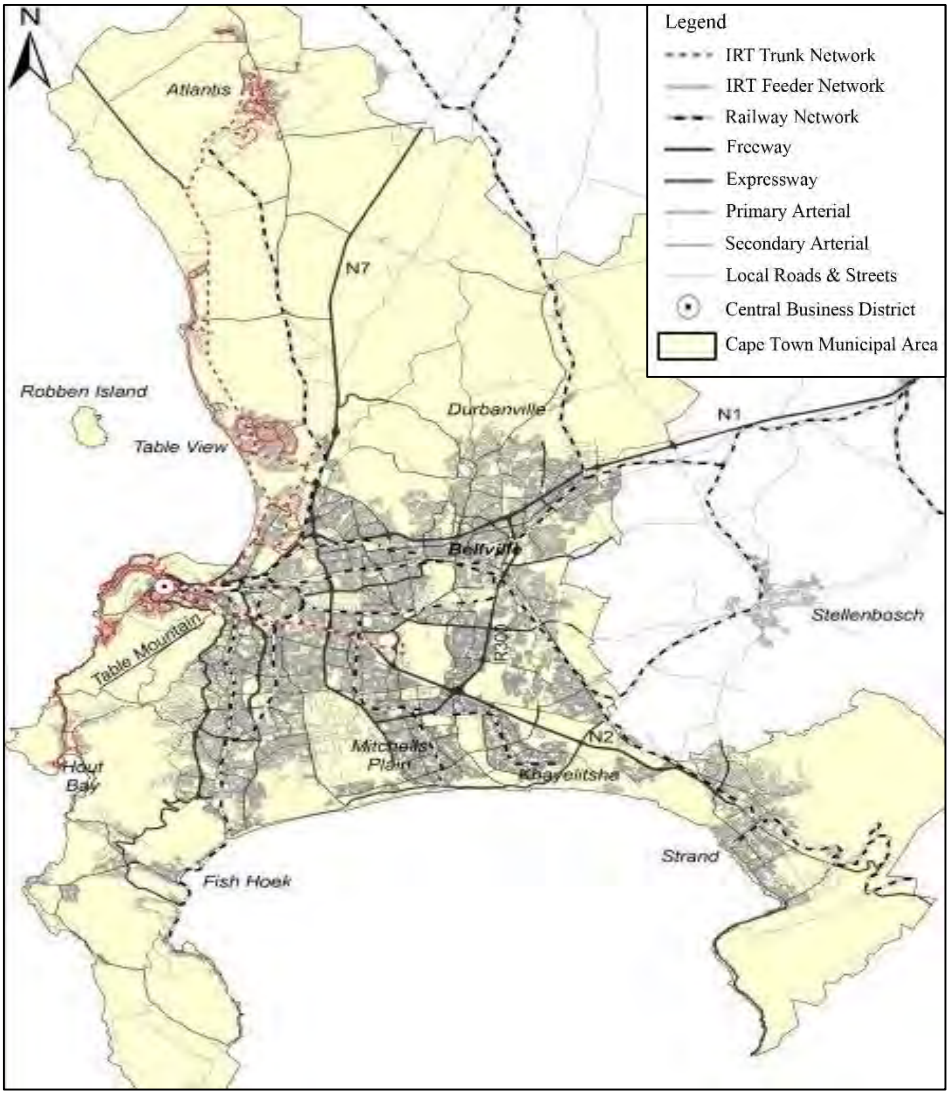

Figure 3.2 Existing Cape Town transportation network (City of Cape Town – ITP, 2013)

3.2.1 Rail Network

According to the City of Cape Town – ITP, (2013), the total length of rail lines in the city is 914km, which comprises of both passenger and freight rail lines. The passenger rail network is owned and operated by Metrorail for the Passenger Rail Agency of South Africa (PRASA) while Transnet Freight Rail owns and operates the freight lines.

3.2.2 Road Network

The 2013 Integrated Transport Plan (ITP) for Cape Town indicates that the road network in the city of Cape Town is characterized by radial routes focused on the CBD. It can also be seen in the report that there exists two main freeways, namely the N1 and N2, which runs in the North-Easterly and South-Easterly directions respectively from the CBD. The total length of the road network in the Cape Town metropolitan area is 9836km, see (City of Cape Town - ITP, 2013). The total length of dedicated BRT busways and minibus-taxi (MBT) lanes are 25km and 20.5km respectively. The network also comprises of feeder routes feeding into the N2. The details of the roads are presented in the (Tables 3.1 and 3.2) below according to two classifications namely functional class of the road and type of surfacing.

Name	Class Number	Length (Km)	Percentage of Total
Freeway	1	133	1.4
Expressway	1	213	2.3
Primary Arterial	2	553	5.9
Secondary Arterial	3	983	10.5
Tertiary	4	1,443	15.4
Minor Roads	5	6,067	64.6
Private Roads	-	-	-
Total		**9392**	**100**

Table 3.1 Functional classification of road types in Cape Town (City of Cape Town - ITP, 2013)

Surfacing type	Length (Km)	Percentage of Total
Bituminous	9,392	95.5
Block Paving	107	1.1
Concrete	123	1.3
Gravel	214	2.2
Tertiary	9,836	100

Table 3.2 Road classification in Cape Town by type of surfacing (City of Cape Town - ITP, 2013)

3.3 Travel Demand

Information in the Integrated Transport Plan (City of Cape Town – ITP, 2013) reveals a private/public modal split of 52% and 48% for all trips entering the CBD, excluding Non-Motorized Transport (NMT) and cycling, in 2012. The highest number of these trips occurred during the morning peak hour. The report further states that rail has the highest number of passenger trips among the land based public transit modes (See Table 3.3) below. The information in the Statistics South Africa – NHTS, (2013), are aggregated into a matrix of 20 sub-regional zones. Inter zonal desire lines which have been determined from passenger trip origins and destinations are grouped into different trip volume category for the morning peak (See Figure 3.3) below. Corridors with trips more than 10000 persons/hour indicate the potential for the development of commuter rail in such corridors where they do not presently exist.

Transport mode	Modal Split (%)	Daily Passenger trips
Car	52	1,338,450
Rail	25	635,000
Contracted Buses	10	240,000
BRT*	< 1	22,000
Mini-Bus Taxis	13	320,000

Table 3.3 Modal split and passenger trip data 2012 (City of Cape Town – ITP, 2013)

* Bus Rapid Transit

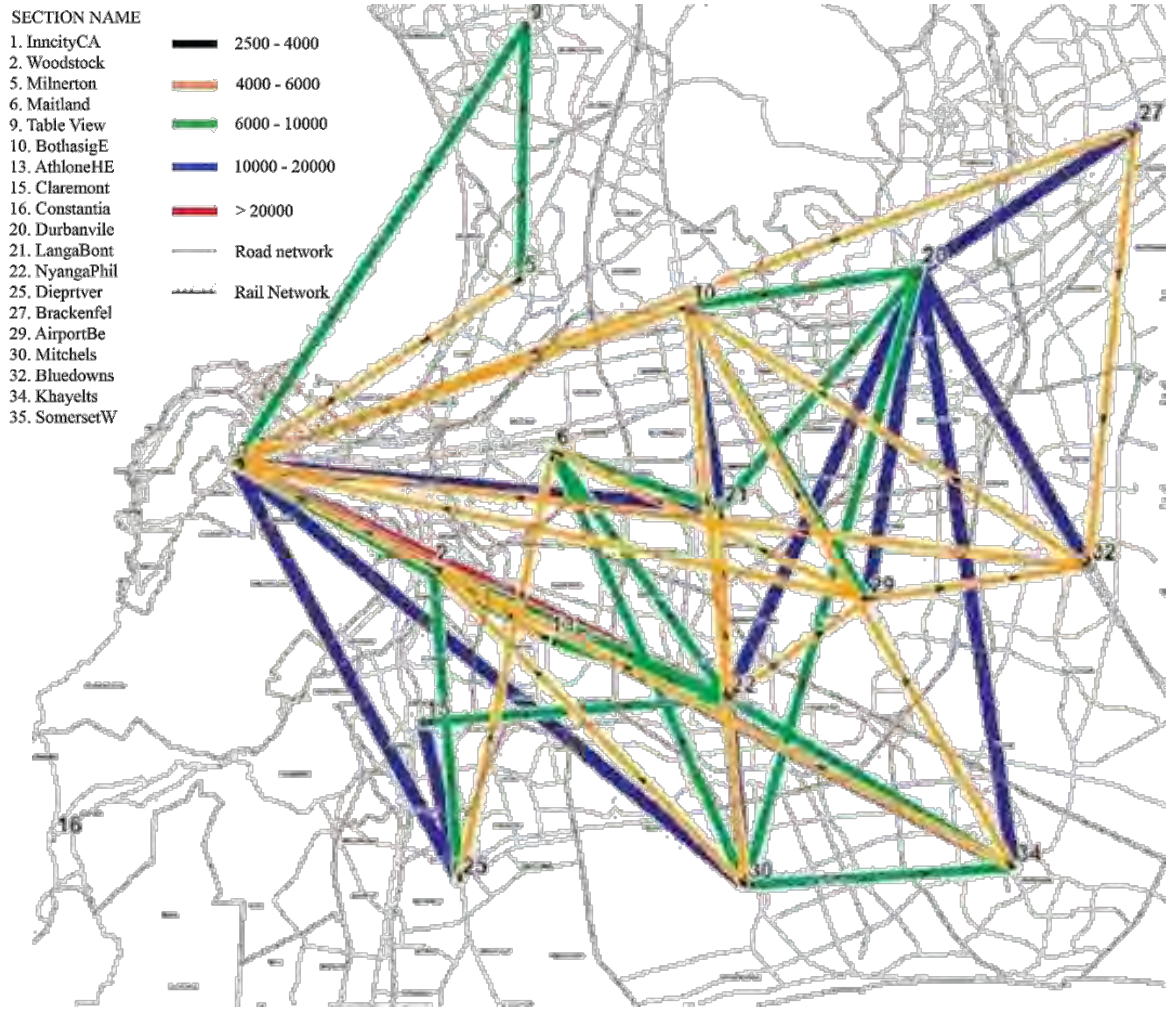

Figure 3.3 Am Peak travel desire lines for inter-zonal trips (City of Cape Town – ITP, 2013)

3.4 Trip Generation and Attraction

Data is also available in the ITP which reveals the nodes and corridors which generate and attract the most trips in the city. An observation of this data reveals that the CBD is the highest attractor node in Cape Town. However, the transportation corridors north of the CBD up to Atlantis also attract a significant number of trips. The major trip generator corridors are located in south-eastern part of the city, for details see (Figure 3.4) below wherein the generator corridors are yellow in colour, the attractors blue and the mature corridors which serve as connector between generators and attractors, are green in colour. The transport network data required for the design of the model was obtained from the City of Cape Town Department of transport (DOT).

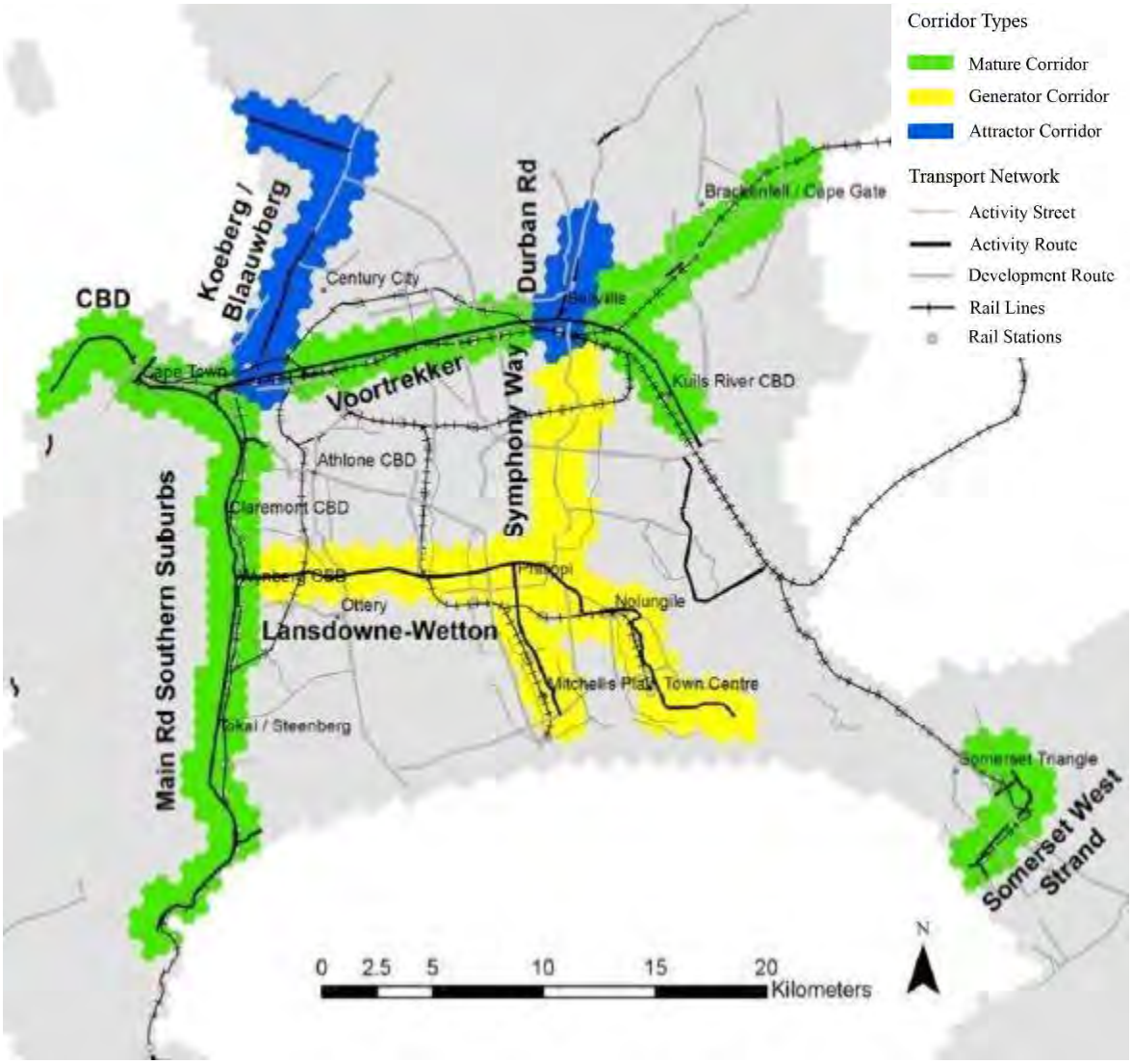

Figure 3.4 Corridor types in Cape Town (City of Cape Town – ITP, 2013)

3.5 Database Building

Having elaborated on the city of Cape Town transportation area and its travel demand patterns, attention will now be focused on the data collection and analysis process. Earlier in Chapter 1, the Geographic Information Systems software (GIS) and python scripting language were identified as the tools that were utilized in collating and formatting the input data for the model proposed in this research. The transport network topological data required for the design of the model was obtained from the City of Cape Town Department of transport (DOT). The city's department of transport builds Transport Analysis Zone (TAZ) data from the Census 2013 data. The TAZ data is then used to model

traffic for the city using the four step modelling process. The EMME/3[10] transportation modelling software is used in the City Transportation department to perform this task. The origin – destination matrix (O-D matrix) data which is the output of the transportation modelling process was obtained as the final data from the city's transport authority. The transport network topology was then disaggregated into 20 sub-regional zones, 1791 Traffic analysis zones with about 8842 nodes in the transportation area. For the model input, the obtained data was formatted with Python programming language in order to build it into a format which is readable by the proposed model solution. It was also crucial to eliminate redundant O-D pairs with no demand for travel between them. The formatting resulted in 1.36million O-D pairs with trip demand. This value represented a significant reduction from the initial number of O-D pairs which was 3.21million. The flow process of the data collection and analysis is presented in figure 3.5 below.

3.5.1 Data Organization

The python program standard library NetworkX was used to create a topological representation of the Cape Town network. A forward star representation (See Appendix B) of the study network is employed namely:

- Nodes data which contains the node identifier and X, Y coordinate of the nodes in this format; {Node_label: (X-coordinate, Y-coordinate)}
- Link data which consists of the start and end nodes of that link in addition to the link weight or attribute represented like this; [(Start_node, End_node, {Link Attribute})].

The trip demand data used is the 2013 city of Cape Town O-D trip demand matrix. Itcontains travel demand between different origin and destination pairs on the network see (Section 3.5). The demand data is stored in the designated format. The raw and formatted data used in the design and testing of the bus network, can be seen in (Appendices A and B respectively).

[10] EMME model version 3

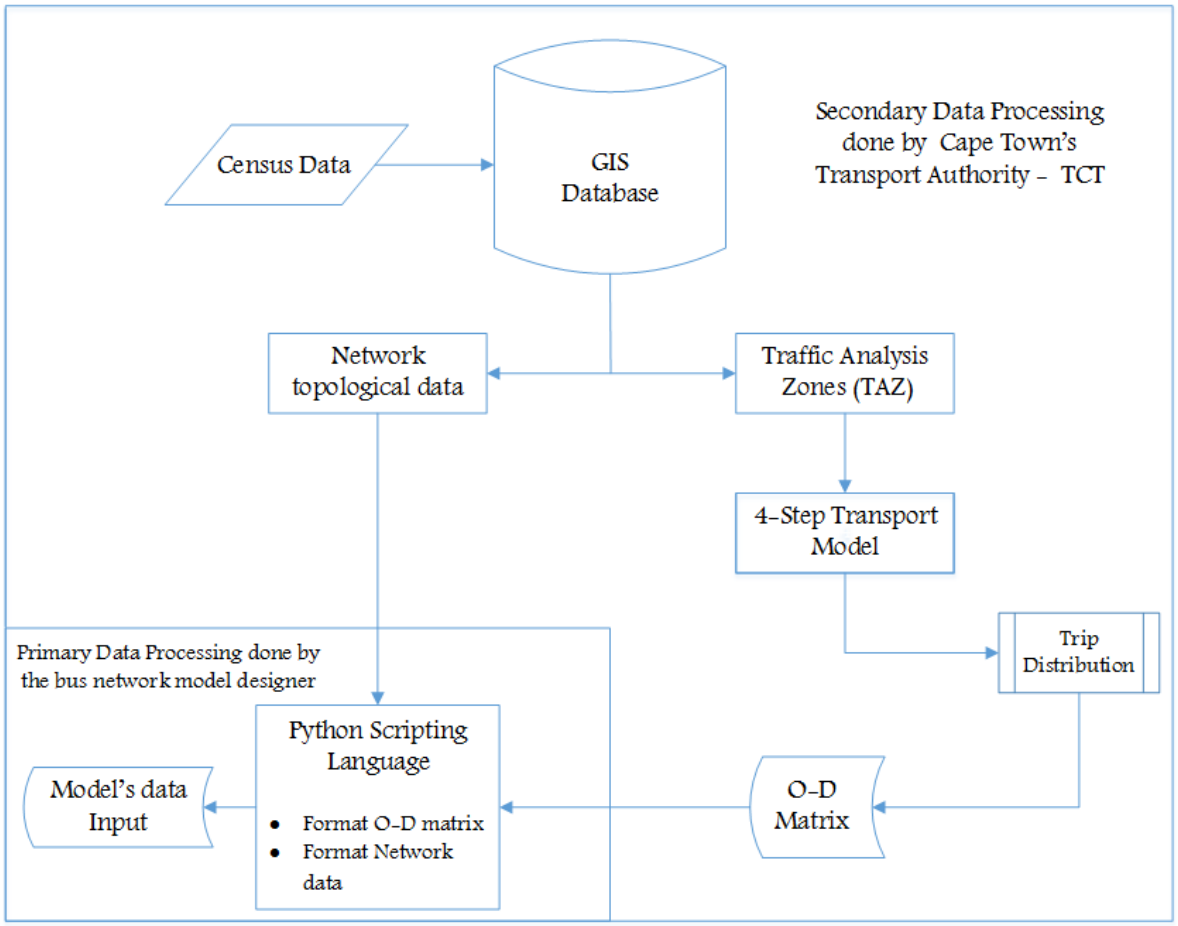

Figure 3.5 Data collection and analysis process

CHAPTER FOUR

4. Modelling and Programming

In the preceding chapter, the data collection and analysis process for the model was discussed. This data will be used as input when the model solution is tested in Chapter five. However, this chapter gives a progressive description of the algorithms used to design the bus network in this research. In section 4.1, a description of a bus network and a topological illustration of its components is presented. Section 4.2 focuses on the mathematical formulation of the objective function used to design the bus network. In section 4.3, the heuristic algorithm used to design the bus network earlier called a Bus Route Network Design Algorithm (BRNDA) in (Section 1.4.2) is presented. It is an adaptation of a genetic algorithm. The GA comprises of three characteristic procedures namely: 1) A generator function which generates the candidate network solutions; 2) An evaluator function, which analyses them; 3) An Evolutionary Computation Strategy (ECS) which selects the best performing network option. For the design of the bus network proposed in this work, custom versions of the aforementioned functions are coded. The generator is coded as a Bus Route Network Generation Algorithm (BRNGA), while the custom evaluator is coded as a Bus Route Network Analysis Procedure (BRNAP). Lastly, the ECS is coded as a Bus Route Network Search Algorithm (BRNSA). The genetic algorithm implementation available in the Python Inspyred library is utilized in the development of the BRNDA. The networkX library (Python Graphing library), was also used to write codes represent the topology of the study network in graphical format. Python programming codes used to develop the Network Design Algorithm can be seen in (Appendix D).

4.1 Bus Route Network Representation

The bus route network is represented as a weighted graph comprising nodes or vertices and edges or links. Nodes are usually geographic ingress and egress points on the network. They act like terminals which enable the flow of passengers into or out of the bus transit network, (Rodrigue, Comtois & Slack, 2006). In graph theory, an arbitrary graph $G\{N, L\}$ is a multi-connection of a finite set of N nodes and set of L links that connect each pair of nodes. Graphs are abstract representations of the

arrangement of nodes and links in a real network. When correctly arranged the graph is referred to as the topology of the network. Rodrigue, Comtois & Slack, (2006) also states that topological models of transportation networks are helpful for performing transit assignments as they enable the distribution of flow on the basis of transit node and link relationships, which are to be modelled on the graph. The bus route network in this work is represented as a weighted graph, with the length of each link denoting the weight of that link.

4.1.1 Nodes

Three types of nodes are described in Fan & Machemehl, (2004) namely: Centroid nodes which represent centroids within the traffic zones from which the trips are generated. The second type of nodes identified are intersection nodes which represent the point of overlap between two roads. The last type are called distribution node, and they connect the centroid node with other parts of the network. This connection is achieved with the aid of centroid connectors. An illustration of the different node types can be seen in (Figure 4.1) below.

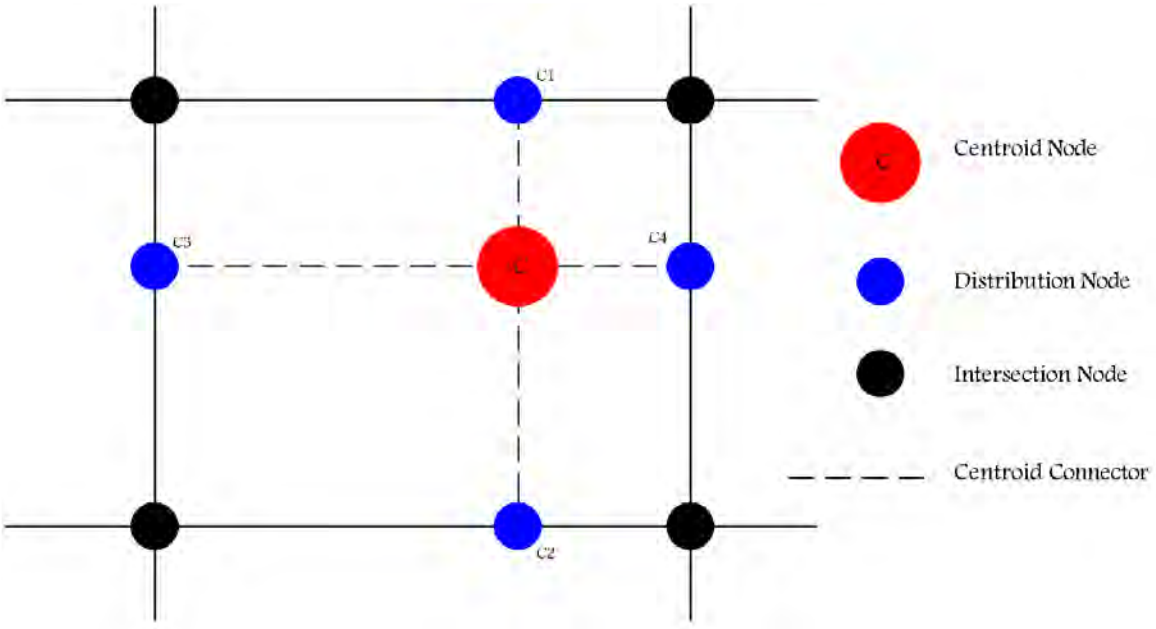

Figure 4.1 Representation of zones and nodes adapted from Fan and Machemehl, (2004)

4.1.2 Links

In transit networks, links are the connection between any node pair. They are represented by road segments on real networks and are weighted on graphs with a given attribute of that road segment, for instance, length or speed. Each link usually represents a particular mode of transport hence if two nodes A and B are accessible by more than one transport mode, the nodes will be connected by the requisite number of links, on a graph. A sequence of contiguous links is known as a route, this can be seen in Figure 4.2 below with the dotted lines representing a specified number of intermediate nodes.

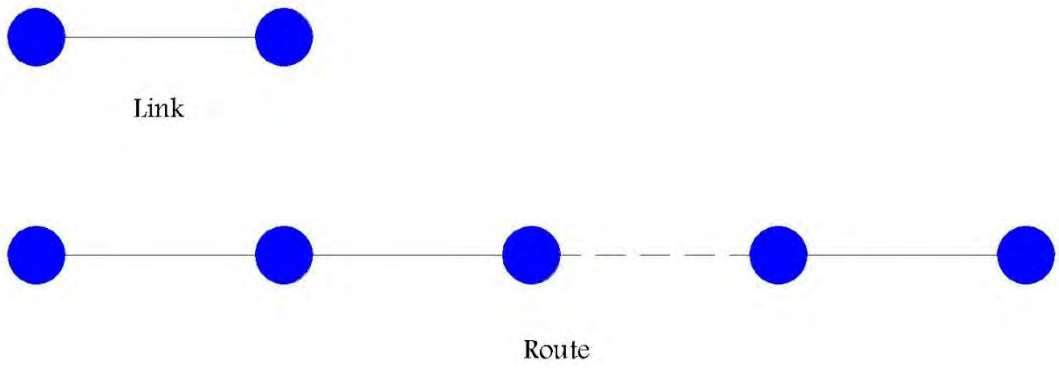

Figure 4.2 Schematic representation of a link and route

4.2 Mathematical formulation

The Bus Transit Network Design Problem is classified as Non-deterministic Polynomial-hard (NP-hard) which Newell, (1979) identifies as being extremely complex. Additionally it is conjectured in the literature that currently no solution exists in polynomial time for NP-hard problems. It is therefore framed as a non-linear optimization problem, with the objective of minimizing user, operator and unsatisfied demand costs. The only decision variable used in the proposed solution algorithm is route configuration. The objective function Z (Equation 4.1), comprises of two terms z_1, z_2 (see Equation 4.1), which represent user cost and operator cost respectively. The third term z_3 used in the objective function, represents unsatisfied demand, and it was adapted from Cipriani, Gori & Petrelli, (2012). The three terms adequately reflect the current state of public transit network in Cape Town which were discussed in (Sections 1.1 – 1.2). Therefore, by minimizing the objective function, the network will be optimized in terms of cost incurred for all stakeholders – users and operators. The variables

W_1 and $W_2$and $W_3$are introduced to represent the weighting strategy in terms of importance, for the conflicting objectives between the three cost components in the objective function. The objective function is presented below as follows:

$$\text{min: } Z = \left(W_1 . \sum_{m,n=1} d_{mn} . t_{mn} . C_t + W_2 . \sum_{j \in R_{SR}} f_j . t_j . C_{op} + W_3 . (P_u . D_u . C_t) \right) \quad (4.1)$$

Subject to the following constraints

$$B_j = Q_j^{max} / (f_j . C) \leq B_{max} \quad (4.2)$$

$$K_r \leq R_{max} \quad (4.3)$$

$$D_{min} \leq D_j \leq D_{max} \quad (4.4)$$

$$f_{min} \leq f_J \leq f_{max} \quad (4.5)$$

Also:

$$W_1 + W_2 + W_3 = 1 \quad (4.6)$$

Where:

Z = Objective or cost function

W_1 = Weight factor reflecting the relative importance of user cost.

R_{SR} is the set of routes in the network

d_{mn} = Transit demand between any node pair (m, n) on the network.

t_{mn} = Travel time between any node pair (m, n) on the network.

C_t = Equivalent cost of user time

W_2 = Weight reflecting the relative importance of operator cost.

f_j = Frequency of route j and f_{min} and f_{max} are its minimum and maximum value

t_j = Round trip time of route j

C_{op} = Unit cost of operating the bus per kilometer

W_3 = Weights reflecting the relative importance of unsatisfied demand cost.

P_u = Time penalty associated unsatisfied demand

D_u = Unsatisfied demand

B_j = load capacity of route j and B_{max} is its maximum value

Q_j^{max} = Maximum passenger volume on any link of route j

C = Bus capacity

K_r = number of routes and R_{max} is its maximum value

D_j = length of route j and D_{min} and D_{max} are its minimum and maximum value

z_1, z_2, z_3 = Cost terms of objective function representing user, operator and unsatisfied demand costs

The objective function is defined in (Equation 4.1) above. In addition, the feasibility constraints for bus capacity, route number, route length, and headway respectively, are presented in (Eq. 4.2 – 4.5) respectively. These are put in place to ensure that the maximum and minimum allowable values for frequency of service and route length are not exceeded. A weighted sum of the in-vehicle time, access time and waiting time gives the value of the transit users' cost, while the operator cost is a sum of the total vehicle distance traveled and total vehicle travel time. The third element in the objective function expression is a penalty for the level of unsatisfied demand. This is introduced to guarantee that service is provided to as many users as possible. Lastly, it should be noted that all the weights add up to 1 (see Equation 4.6 above)

For the simplicity of designing this bus network, the following assumptions are made;

- The total travel demand is fixed, which implies that the O-D matrix is static and constant for all periods see (Fan & Machemehl, 2004).
- All buses on particular routes have a constant speed. This implies that the in-vehicle travel time are not affected by congestion or other prevalent traffic conditions on the road that consists of the bus networks.
- Buses chosen to run on particular routes have equivalent capacity and load factor.

4.2.1 Transit costs

4.2.1.1 Transit user cost

This is a combination of three different cost components namely, access time, waiting time and in-vehicle travel time. Access cost is the cost in time the passenger must pay to get unto the bus network, a proxy commonly used for this is the distance between the user's origin and the closest transit stop on the network. The distance to the distribution node, depends on the location of the passenger relative to the centroid, it also depends on the configuration of the network. Waiting time is the amount of time transit users have to spend at transit nodes like bus terminals. For this research, non-transfer waiting times are assumed, hence, the resulting mean waiting time for this assumption is half the bus route headway[11]. This is meant to account for the random arrival patterns of passengers. However, this is only applicable when the bus headway is short say less than 10mins. Fan & Machemehl, (2002) observed that as headway increases the passengers tend to synchronize their arrival at the terminal to match a time closer to the bus scheduled departure time thereby reducing their waiting time. Lastly, the in-vehicle time is defined as the total time the user spends riding in the bus. The in vehicle travel time maintains a direct relationship with the length of the bus route and an inverse relationship with the speed at which the bus is travelling on the network. See (Equation 4.7) for the user cost expression.

$$z_1 = W_1 . \sum_{m,n=1} d_{mn} . t_{mn} . C_t \qquad (4.7)$$

4.2.1.2 Transit operator cost

This includes all cost components involved in operating the buses including fuel cost, maintenance, driver salaries and other personnel emoluments. See (Equation 4.8) for the operator cost expression.

$$z_2 = W_2 . \sum_{j \in R_{SR}} f_j . t_j . C_{op} \qquad (4.8)$$

[11] Headway is the time difference between the faces of two vehicles within a public transit system, when one is preceding and the other trailing

4.2.1.3 Unsatisfied Demand Cost

Normally it is impossible to satisfy the transit need of every single user that has a need to travel. An instance would be introducing a bus route to run in locations with very sparse population density. This could lead to a waste of resources even though the tradeoff for this situation is that some users may forfeit their trip, or walk longer distances to access the bus service. Due to the conflict inherent in this scenario, minimizing the amount of unsatisfied demand, is therefore a more attainable goal set by the transit authorities. The last term in the objective function is introduced to reflect this case and impose a cost penalty which has a multiplier effect, depending on the level of unsatisfied demand. See Equation 4.9 for the unsatisfied demand cost expression.

$$z_3 = W_3 \,.\, (P_u \,.\, D_u \,.\, C_t) \tag{4.9}$$

4.2.2 Feasibility Constraints

As discussed in (Section 2.1.2), these are parameters which reflect the limiting conditions of the decision variable in a bus network design problem (BNDP), they also define the feasibility of resource allocation in optimization problems. The constraints used in the model's objective function are discussed below and their mathematical expression can be seen in (Equations 4.10 – 4.14).

4.2.2.2 Load Factor Constraint

The load factor constraint is dependent on the travel demand and bus capacity. The load factor is generally used to denote vehicle capacity. A load factor of 1.0 indicates that all the seats are utilized. A load factor greater than 1 indicates the presence of standing passengers in the bus. See (Equation 4.10) below for the mathematical expression of load factor constraint.

$$B_j = Q_j^{max} / (f_j \,.\, C) \leq B_{max} \tag{4.10}$$

4.2.2.4 Route Number Constraint

It is common practice in transit network design for transit authorities to fix the maximum number of routes the network will have. This number is usually determined by the vehicle fleet size available. See the mathematical expression for route number constraint in (Equation 4.11) below.

$$K_r \leq R_{max} \tag{4.11}$$

4.2.2.1 Route length constraint

A route length constraint indicates an upper and lower bound outside which it would be illogical to operate the bus service. One such scenario could be operating a bus service on a route where walking is preferred because of its short distance another scenario could be operating buses on routes which are extremely long, which could discourage the maintenance of adequate bus schedules. See the mathematical expression for route length constraint in (Equation 4.12) below.

$$D_{min} \leq D_j \leq D_{max} \tag{4.12}$$

4.2.2.3 Frequency Constraint

The frequency feasibility constraint is introduced in this objective function formulation, to represent the maximum and minimum operable frequency on each route within the bus network. It is usually dependent on, the available fleet size and transit demand for each route. See the mathematical expression for frequency constraint in (Equation 4.13) below.

$$f_{min} \leq f_J \leq f_{max} \tag{4.13}$$

4.3 Solution Approach

The Bus Route Network Design Algorithm (BRNDA) is implemented by the three custom functions earlier named in the introductory section of this chapter as: 1) Bus Route Network Generation Algorithm (BRNGA); 2) Bus Route Network Analysis Procedure (BRNAP); 3) Bus Route Network Search Algorithm (BRNSA), In the first stage, the BRNGA generates the initial candidate networks which

represents the solution search space. The BRNAP is executed in the second stage, to evaluate the generated bus networks. In the last phase of the bus network design, implements the BRNSA, to get the best performing network option. A detailed description of the aforementioned stages in the network design algorithm, will be discussed in (Sections 4.3.1 – 4.3.3). See (Figure 4.3) below for a schematic representation of the BRNDA.

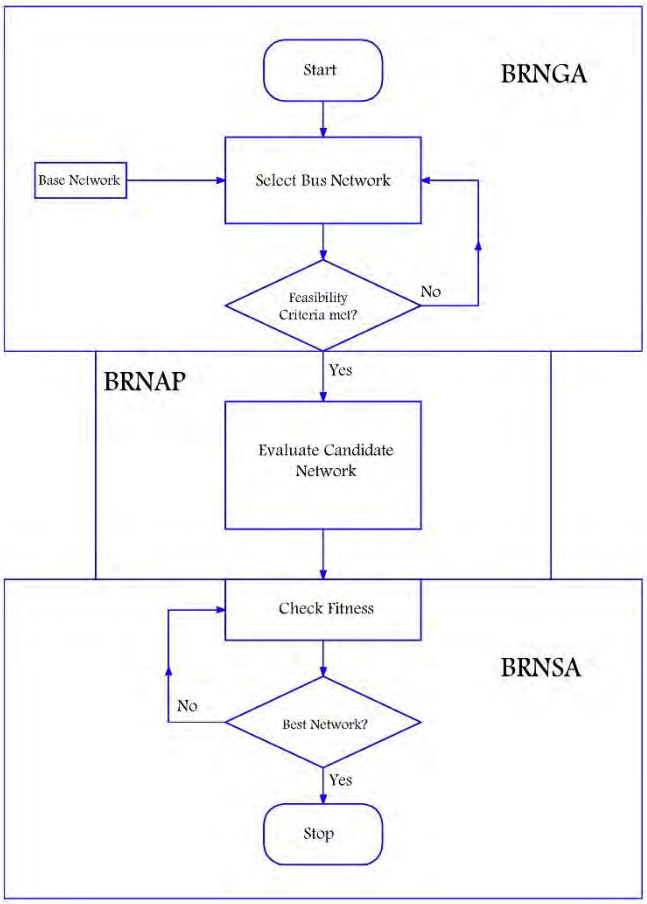

Figure 4.3 Schematic presentation of the Bus Route Network Design Algorithm (BRNDA)

4.3.1 Bus Route Network Generation Algorithm (BRNGA)

The BRNGA is first stage of the model being implemented in this research work, it is a heuristic design algorithm which generates the candidate bus network options from the study network – Cape Town road network, according to user specified guidelines.

4.3.1.1 BRNGA Procedure

The BRNGA generates bus networks, by randomly selecting a stated number of terminal node pairs on the study network. Next, it obtains the shortest path/route between the selected node pairs. The Dijkstra's shortest path algorithm, is utilized for this task (See Appendix C.2 for details of the algorithm). The generated study network routes are then stored. Hereafter, the bus route nodes are created at an offset,[12] from the corresponding study network nodes which were stored earlier. The bus routes are likewise created by generating the shortest path between the newly created bus route nodes. Furthermore, boarding and alighting links, are created, to facilitate access between the bus and study networks. The created bus networks are then stored. This process is repeated, until the user specified number of candidate networks solutions (population) is generated. These generated bus networks, represent the initial candidate solutions from which the optimized network will be selected. The final step in this procedure is the generation of rail lines from the rail network data. Despite the focus of this work being on bus networks, the rail networks are introduced in order to have a realistic traffic assignment. The network generation process is guided by user defined feasibility parameters such as maximum and minimum route length and maximum route number. The flow chart depicting the processes of the BRNGA can be seen in (Figure 4.4 below). In (Section 4.3.1.1), a description of the input data and user parameters utilized in the BRNGA is discussed.

4.3.1.2 BRNGA Input Data

The Cape Town road and rail networks are used as the key inputs for the BRNGA. Freeways are excluded from the study network, due to the fact that the bus network is intended to operate in an urban environment. Hence, the buses would not operate on expressways, even though some express bus services use them. The user parameters utilized in the BRNGA are maximum number of route as well as route length constraints which are subject to the transit authority's discretion.

[12] Equivalent distance in time required to board or alight from the bus network.

4.3.1.3 BRNGA Pseudo code

A pseudo-code for the BRNGA is presented here, representing the step taken to generate the candidate bus networks.

Step1: Read in base road network data

Step2: Randomly select origin and destination centroid nodes from the study network

Step 3: Check that the generated terminal nodes are unique. If they are not repeat step 2, else proceed.

Step 4: Compute the shortest path between the selected node pair using Dijkstra's algorithm.

Step 5: Compute the length of the generated shortest path.

Step 6: Check that the generated routes meet the length feasibility criteria. If they do accept the route else loop until the desired length is achieved.

Step 7: Save the study network routes.

Step 8: Make parallel bus route networks node at specified offset from those of the study network

Step 9: Make bus route links.

Step 10: Make bus network access links.

Step 11: Loop till the user defined number of routes is acquired

Step 12: Store the generated feasible bus routes.

Step 13: Read in the rail network data.

Step 14: Create rail lines.

Step 15: Stop the procedure

4.3.1.4 BRNGA Output

The output from the BRNGA, is the generated candidate networks. They represent the potential solution search space from optimized solution will be selected. The generated networks also play the role as one of the main inputs for the network analysis procedure (BRNAP) which will be discussed next in (Section 4.3.2), however, the flow process for the BRNGA is depicted in (Figure 4.4) below.

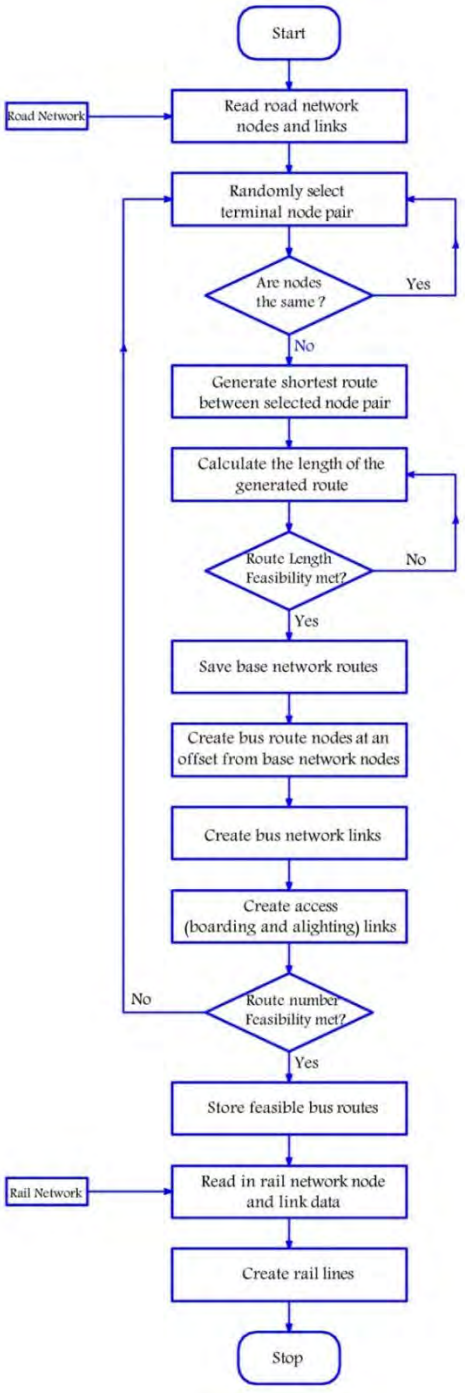

Figure 4.4 Bus Route Network Generation Algorithm (BRNGA)

4.3.2 Bus Route Network Analysis Procedure (BRNAP)

This is essentially an iterative heuristic algorithm which comprises three sub-processes namely:

- Trip assignment,
- Determination of bus network performance indicators
- Objective function evaluation.

4.3.2.1 BRNAP Procedure

In the first step of the BRNAP, an all-or-nothing assignment model (see Appendix C.3), also known as a single path trip assignment model in Dial, (1971), is used to perform the traffic assignment. This is executed in order to assign traffic volumes, to the routes in the candidate networks generated by the BRNGA. The traffic volumes subsequently assigned to each candidate's constituent routes are then summed up to get the amount of travel demand it can satisfy – its network capacity. Next, a Breadth First Search algorithm (see Appendix C.1 for details), is used to obtain all the feeder links, at a distance equal to the maximum allowed walking distance of 1km from for each candidate bus networks. Additionally, the feeder link volumes are summed to get the potential demand for each candidate network. The volume of unsatisfied travel demand accruing to each candidate network can then be obtained by subtracting its capacity from the total demand volume it attracts. In the final sub-process of the BRNAP, route performance indicators like network capacity, route speed and unsatisfied demand are used to compute the objective function value for each of the candidate bus networks. It should be noted that the raw objective function values, are transformed using the widely known fitness function seen below in (Equation 4.14). Pattnaik, Mohan & Tom, (1998) states that the transformation of the raw objective function value is necessary in order to determine each individual's fitness relative to the entire population. To this end, fitness values rather than raw objective scores will be used to report the results at the testing stage of the algorithm later in chapter five.

$$F(i) = N - \frac{P.O_i}{\sum_{k=i}^{p} O_i} \qquad (4.14)$$

Where:

N = A large non-negative integer; P = population size; O_i = objective function of the i_{th} individual and F(i) = Fitness function

4.3.2.2 BRNAP Input Data

The first input of the BRNAP are the candidate networks, generated in the first phase of the model. This consists of nodes and links presented in a format which is readable by the model see (Section 3.5.1). The next input is the transit demand matrix (O-D matrix) which represents the demand for passenger trips between traffic zones in the network. Another crucial user input is the maximum walking distance to the bus network, which is used to determine how much travel demand will potentially be attracted to the bus network. Other inputs are route link attributes such as speed, travel time, equivalent cost of user time, operational cost, and the weighting components for objective function.

4.3.2.3 BRNAP Pseudocode

The respective steps taken to analyze the candidate bus networks can be seen in the BRNAP pseudo-code below.

Step1: Read in input data (generated bus networks)

Step2: Initialize the bus routes and access links

Step 3: Implement breadth first search algorithm.

Step 4: Perform Traffic Assignment.

Step 5: Compute bus network performance measures

Step 6: Evaluate objective function to get fitness values.

Step 7: Loop to evaluate all networks.

Step 8: Stop the procedure

4.3.2.4 BRNAP Output

The output from this stage of the model comprises network description data as follows:

- Candidate network fitness function values computed from the objective function expression.
- Feeder link potential travel demand for each network.
- Link attribute data like link flow, headway, frequency, travel time among others
- Route information such as route round trip time and passengers volume per routes.

The fitness function values for the candidate networks (see Section 4.3.2.1) will be used as one of the key inputs in the BRNSA which is discussed next in (Section 4.3.3), however the flow chart for the BRNAP is presented below in (Figure 4.5).

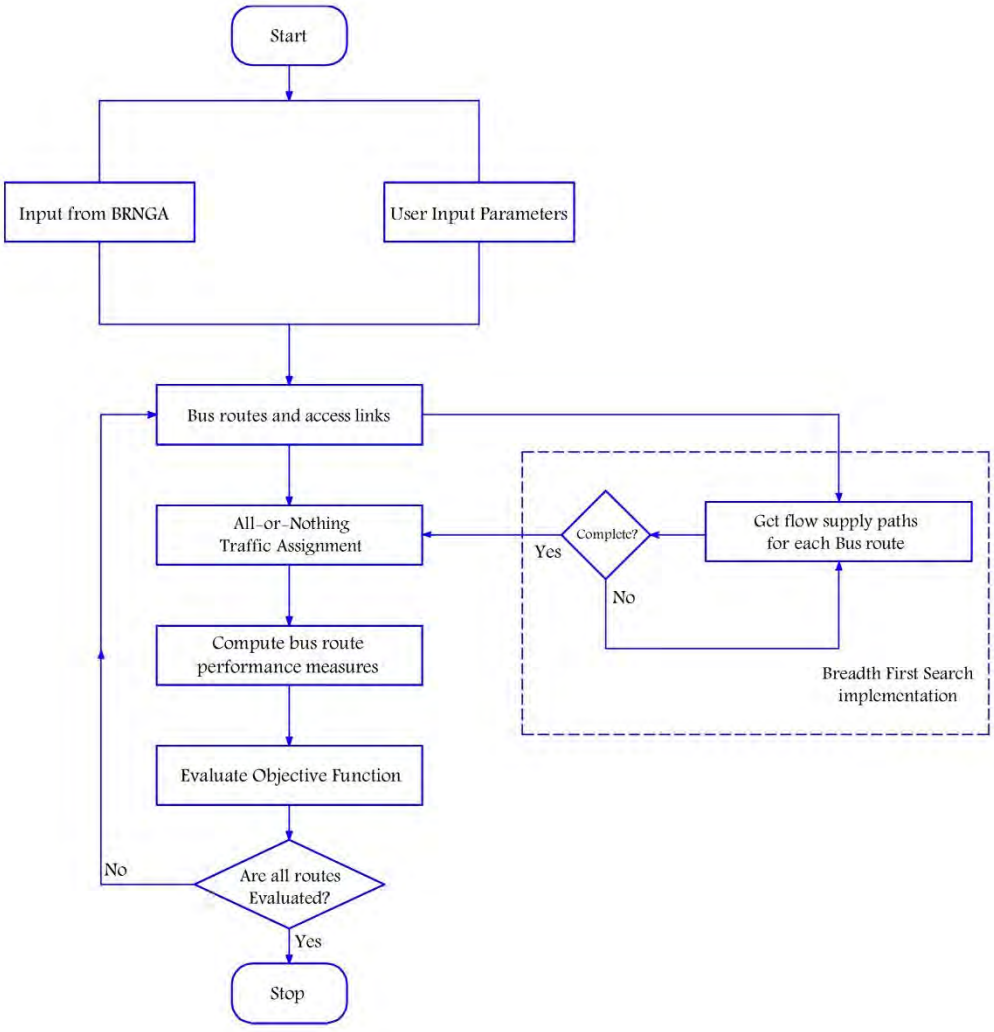

Figure 4.5 Bus Route Network Analysis Procedure (BRNAP)

4.3.3 Bus Route Network Search Algorithm (BRNSA)

In this section, the third component of the Bus Route Network Design Model is discussed. The network Search algorithm used to search for the best network, is based on an Evolutionary Computation strategy of the GA (See introductory section of Chapter four). The fitness value returned by the BRNAP in phase two of the bus route network design algorithm, constitutes the metric with which the candidate networks will be ranked. The flow process for the genetic algorithm, is seen in (Figure 4.6).The input data for this phase are the candidate networks, and their fitness function values. Other User inputs are the genetic algorithm operators like mutation and crossover which will be used to perform the search for the optimized network.

4.3.3.1 BRNSA Procedure

The BRNSA starts by calling the BRNGA and BRNAP respectively. The former generates the initial candidate networks, while the latter evaluates them. The generated candidate networks are initialized as the first parent population. Each individual network in the population, is assigned their respective fitness value, which was obtained in the Bus Route Network Analysis Procedure through the evaluation of the objective function expression. At each iteration, the algorithm uses the fitness values to determine the survival capacity of each individual in the population (network) and attempts to improve the fitness values, of each offspring population. Each preceding generation, contributes to the creation of the offspring populations otherwise known as reproduction. This is achieved, by the respective actions of the genetic operators namely: selection, crossover and mutation. A termination criteria is usually introduced in a genetic algorithm. This ensures the algorithm will terminate once that criteria is met. The stopping criteria used in this work is number of generations. A description of the GA reproduction process used in this stage of the BRNDA, will be discussed next.

4.3.3.1.1 Selection

This is the first of the three operators used in the reproduction strategy of genetic algorithms. Its objective, is to choose the individuals in the population which will generate offspring population for

the next generation. The selection is done on the basis of the fitness of each individual network in the population. The selected individuals are referred to in the literature as a mating pool, see (Kumar, 2012). The mating pool contributes their characteristic, towards the evolution of the next generation of solutions. When the selection, is not appropriately done, as to reflect the diversity of the parent population, a bias for either the fittest or weakest individuals may arise. The effect of this could be a quick convergence to a local solution in the case of the former, or a very slow convergence rate in the latter scenario. Hence, the effectiveness of the search for an optimized network solution can be impacted either positively or negatively by the selection operator. Roulette wheel, rank based and tournament selection are some of the commonly used selection techniques seen in literature. For this research, the Python Inspyred library default selection function is used. It simply selects and returns the entire generated population (candidate bus network) as the parent that will produce the offspring generation.

4.3.3.1.2 Crossover

Crossover is the combination of two randomly designated parent networks from the mating pool created in the selection process in order to generate a new offspring network. A crossover probability, is a user defined parameter, introduced to determine the rate at which a crossover operation will occur. The crossover probability multiplied by the population size determines the number of individuals in a population that will undergo crossover. This implies, that if the crossover rate is set as 0.7, and population size 100, seventy percent of the population will go through crossover. Siriwardene & Perera, (2006), observes that a high crossover probability, encourages a good combination of parent solutions. This would guarantee that offspring, actually inherit the attribute of the parent network. Despite the optimum crossover probability value being problem dependent, Different authors have proposed some typical values for the crossover probability. De Jong, (1975) proposed 0.6, while Grefenstette, (1986), proposed 0.95. Lastly, Schaffer et al., (1989) suggested that typical values lie between 0.75 and 0.95. However in this work, the crossover probability used, will be determined from the sensitivity analysis carried out later in (section 5.2). Some types of crossover

available in the literature are single point, multi point and uniform crossover methods. A multi-point (2-point crossover) crossover operator is used in this research. The 2-point crossover randomly chooses two points on the parent networks and then swaps every data between the two points, thereby creating an offspring network solution. A simple algorithm for the cross over operator is presented in (Appendix C.4).

4.3.3.1.3 Mutation

This is the last operator in the genetic algorithm reproduction process. It introduces diversity into the population, by altering the configuration of some networks within the parent population. This is done, by randomly identifying the networks to be altered in the first step, then replacing some routes within them with those generated from outside the solution search space (from the study network). This helps to prevent premature convergence, by reducing an excessive cluster of similar routes. The mutation of individual routes is controlled by the mutation probability parameter. This is the probability that mutation will occur. This means that if a sample solution space has a population size of 20 and each individual in the population has a network size of 10 routes, a mutation rate of 0.1 implies that on average 20 routes in the total populations will be mutated. This is obtained by (20 x 10 x 0.1), see (Siriwardene & Perera, 2006). Holand, (1975), reports that a high mutation probability increases the possibility of searching more areas in the solution search space but decreases the possibility of convergence to the global optimum solution. Since the mutation probability is problem specific, the sensitivity analysis performed in (section 5.2) will yield the values that will be applied in this model. The pseudo code for the mutation operator is presented in (Appendix C.5)

4.3.3.2 BRNSA Input Data

The main input data for the BRNSA are the generated candidate networks from the generation stage, their fitness values from the analysis stage and other parameters like: crossover probability, mutation probability, number of populations, number of generations and number of crossover points

4.3.3.3 BRNSA Pseudo code

Detailed below is the pseudo code for the BRNSA:

Step 1: Call Bus Route Network Generation function.

Step 2: Initialize all performance measures

Step 3: Initialize population.

Step 4: Set generation = 0

Step 5: Call Bus Route Network Analysis Procedure function.

Step 6: Evaluate objective function.

Step 7: Keep Current best Population

Step 8: Generate next population

- selection,
- crossover
- mutation

Step 9: If termination criteria is not met, increment generation by 1 and go to step 4 else update the current best solution if improved

Step 10: if route number criteria is not met, increment route number by 1 and go to step 2 else proceed to output

Step 11: Output the best route set from the best solution found.

4.3.3.4 BRNSA Output

The output from the BRNSA, is the final optimized network. This would be either a local optimum or global optimum solution, depending on the parameter values with which the search is conducted. A summary of Bus Route Network Design Algorithm's (BRNDA) input, output and algorithms are

presented in tabular form in (Table 4.6) below, while Figure 4.7 depicts the flow process for the BRNDA.

BRNDA Component	Input Data	User parameters	Algorithm Used	Output
Bus Route Network Generation Algorithm (BRNGA from section 4.3.1)	Study network data (links and nodes)	Number of Routes Maximum route length Minimum route length	Dijkstra's Shortest Path Algorithm	Generated Candidate Networks
Bus Route Network Analysis Procedure (BRNAP from section 4.3.2)	Trip Demand Matrix	Travel speed Frequency Operator cost User cost weight Unsatisfied demand cost Weight Operator cost Weight Value of time	Breadth First Search Algorithm (BFSA) All-or-Nothing traffic assignment algorithm	Candidate Network's: Fitness values. Link flows Headways Frequencies Trip time
Bus Route Network Heuristic Search Algorithm (BRNSA from section 4.3.3)	Evaluated Networks from the BRNAP and their fitness values.	Maximum Generation Population size Mutation probability Cross over probability Number of Cross over points	Evolutionary Computation Strategy (ECS)	Final Optimized Network

Table 4.1 Summary of BRNDA components, their input and outputs

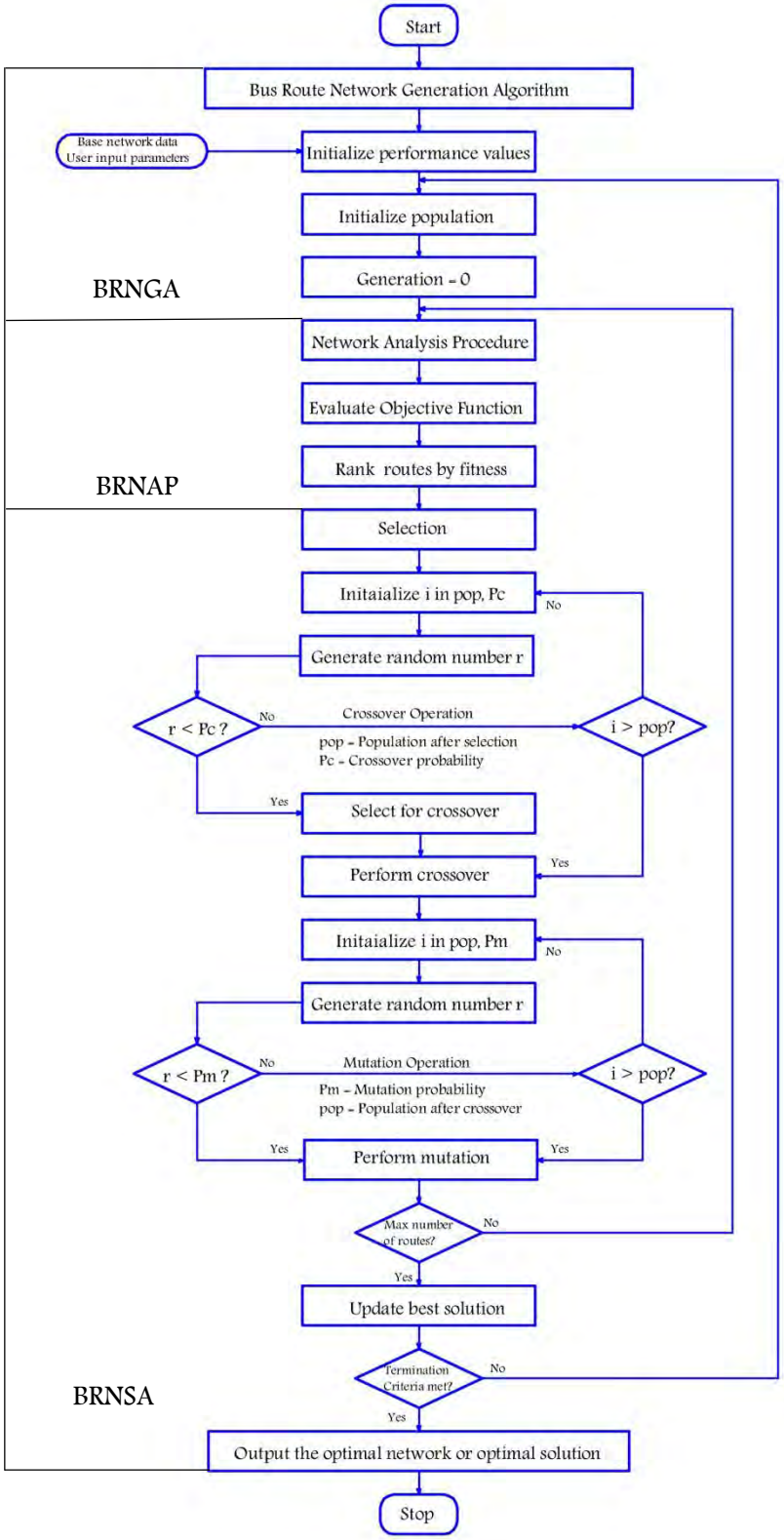

Figure 4.6 The genetic algorithm based Bus Route Network Design Algorithm (BRNDA)

CHAPTER FIVE

5 Testing and Application

5.1 Introduction

The BRNDA has been developed by synthesizing knowledge from various knowledge areas within the TNDP literature, and these have been detailed in preceding chapters. Chapter five, will now focus on testing the BRNDA and applying it to Cape Town. First a sensitivity analysis is done to determine the optimum values for the design parameters. After the sensitivity analysis, the model is used to design a small network. The result from this application to the small network, will show the complete numerical results obtained from the design. Finally, the model will be used to design a large-scale public transport network for Cape Town.

5.2 Sensitivity Analysis

In this section, a sensitivity analysis is done on the main design parameters. The sensitivity analysis can be used to determine their optimum values. In addition, it is important to understand the behaviour of these parameters across a varying range of values. At this stage, the model's best parameter values are not known, hence, the need to determine them through the sensitivity analysis. It should be noted that the results were obtained within a limited computation time frame of 120 hours for each parameters. Some of the parameters are mutation and crossover probability, number of generations and number of population. Some of the parameters such as mutation and crossover probability, are decimal (continuous) and hence cover a very large range. This makes it practically impossible to test all combinations of these parameters. Furthermore, the optimal/near optimal parameters may be test case dependent, hence they may not be optimum given a different scenario than that for which they are obtained. While a parameter is being tested, its values are varied over a range, but those of the other parameters are kept constant. The results of the sensitivity analysis are presented and discussed, showing the effect of different parameters used in the model. It should be noted that the raw objective function values have been transformed using the fitness function expression earlier discussed in (Section 4.3.2.1). Furthermore, the optimal/near optimal fitness value

occurs at the point where the fitness value and standard deviation is lowest. The results of the sensitivity analysis are presented in graphical form, showing the fitness values and standard deviation of each parameter that was tested.

5.3 Sensitivity Analysis Results

5.3.1 Effect of Population Size

The effect of population size is examined by varying its value between 5 and 100. This reveals that the population size has an inverse relationship with the fitness value. This trend is consistent with work done in Fan & Machemehl, (2004). It is observed that the local optimal fitness value occurs at the point where the fitness value and standard deviation is lowest. From this, it can be seen that the best (lowest) number of population is 60. The result is illustrated below in (Figure 5.1).

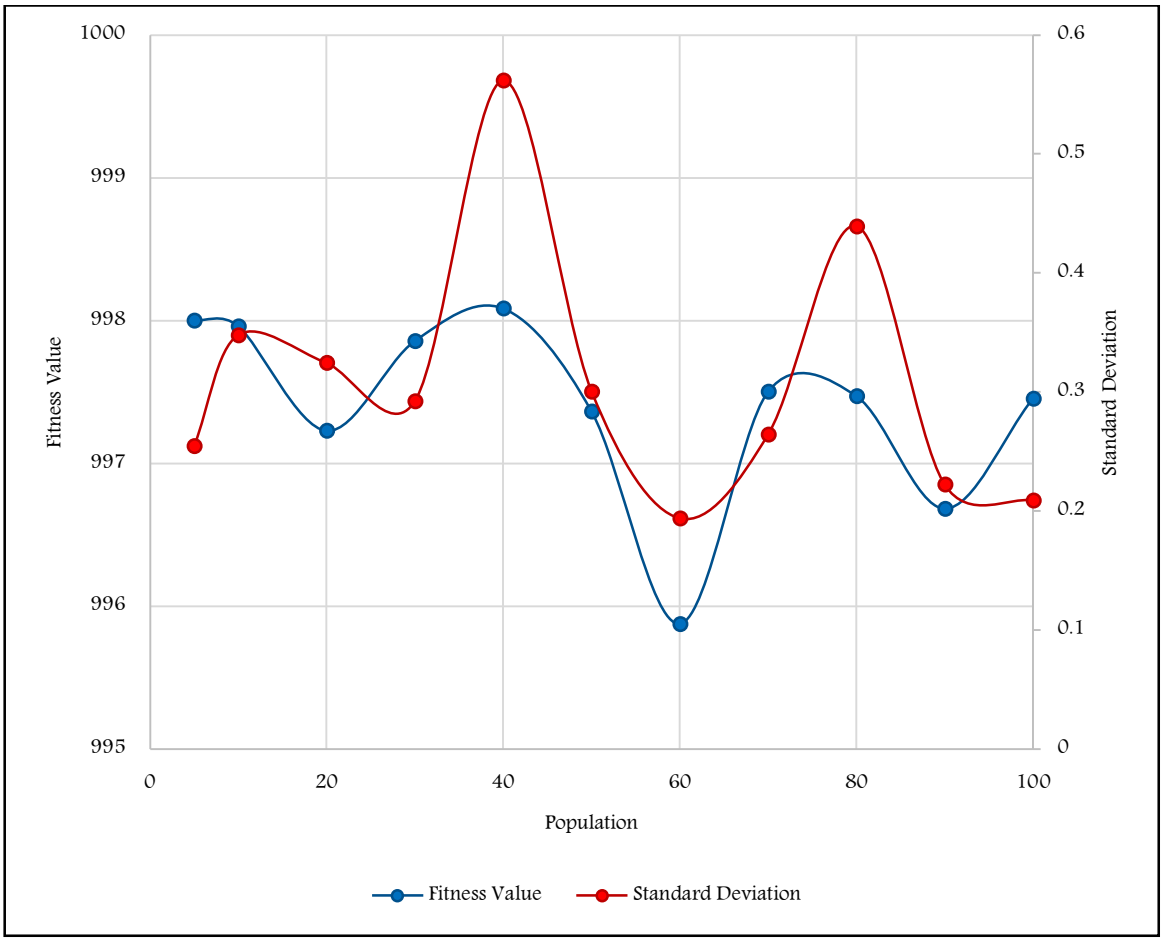

Figure 5.1 Combined plot of fitness value and standard deviation against population

5.3.2 Effect of Generations

Maximum number of generations is used as the stopping criteria in this research and its effect on the fitness value is observed by changing its value between 5 and 50. The result shows that the lowest or best fitness value occurs at 50 generations, for this run. However, a local optimum value occurs after 10 generations, which can be seen in (Figure 5.2) below. A plot of the best, worst and median individual networks in the population can also be seen below in (Figure 5.3)

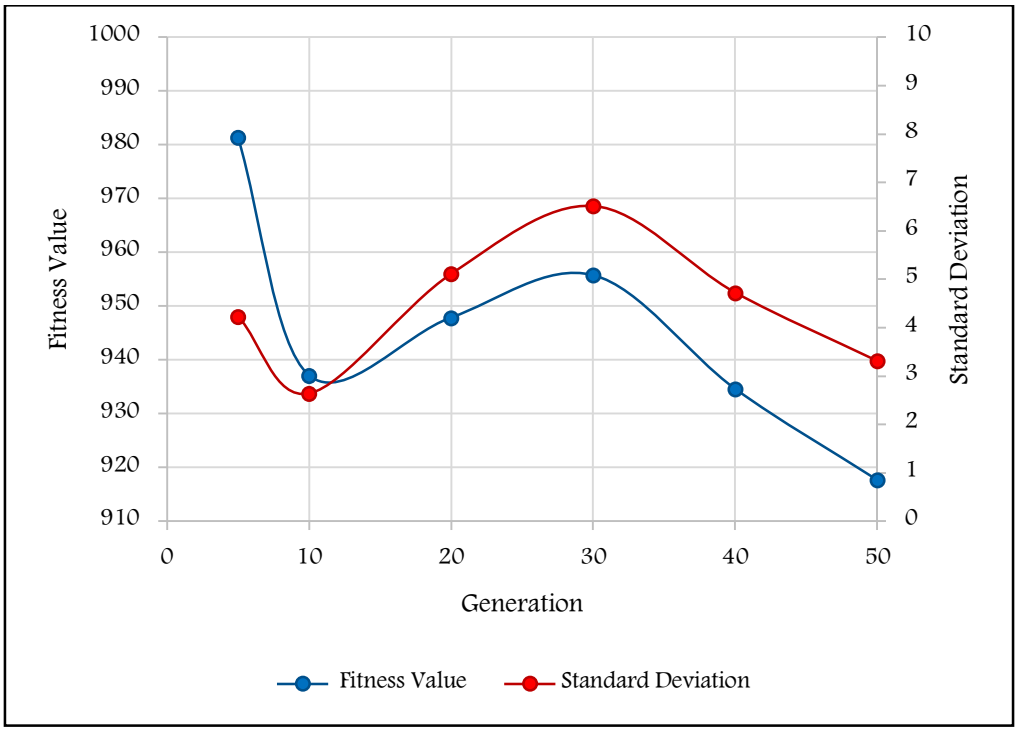

Figure 5.2 Combined plot of fitness value and standard deviation against generation

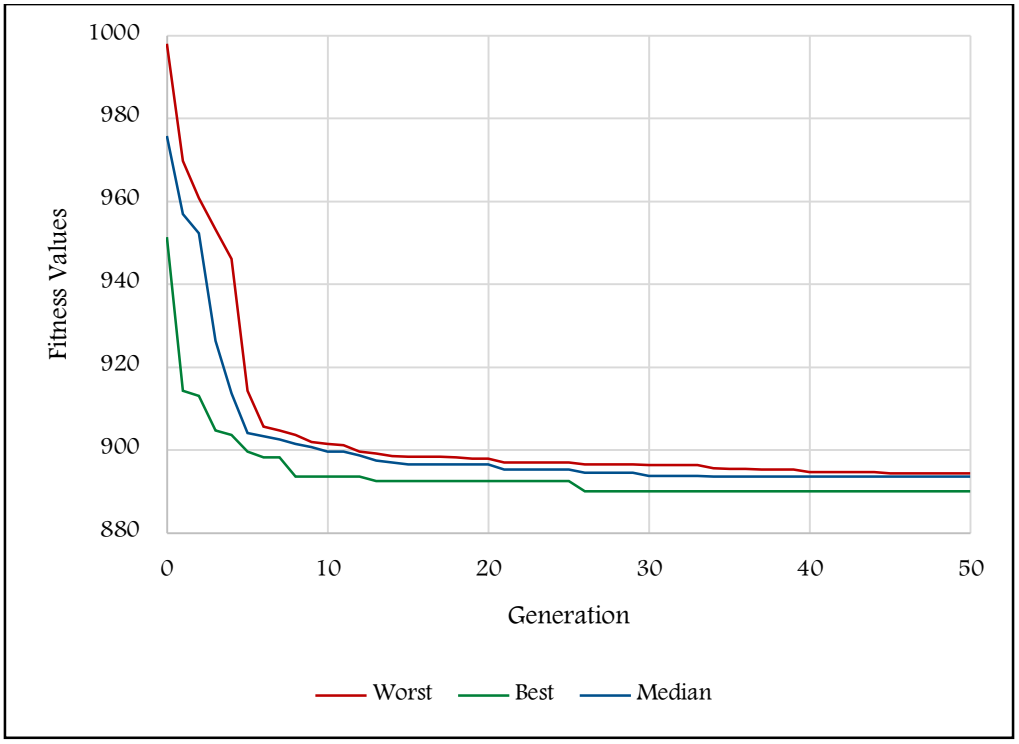

Figure 5.3 Combined plot of the worst, best and median network solutions against generation

5.3.3 Effect of Crossover Probability

The effect of crossover probability is studied by varying the value between 0.1 and 0.9. The results illustrated in (Figure 5.4) below, show that the lowest fitness value and highest standard deviation occurs at 0.7. Therefore, it can be inferred that 0.7 is the optimal/near optimal value.

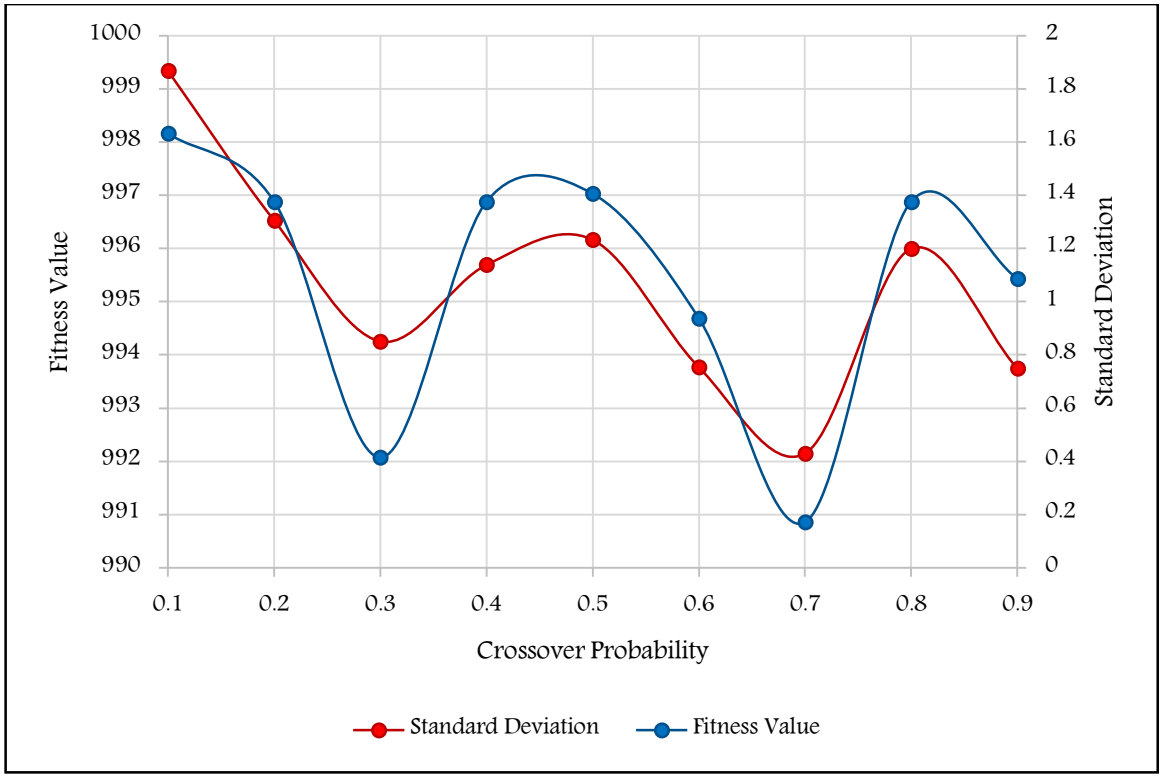

Figure 5.4 Combined plot of fitness value and standard deviation against crossover probability.

5.3.4 Effect of Mutation Probability

The effect of crossover probability is studied by varying the value between 0.1 and 0.18. The results which can be seen in (Figure 5.5) below, show that the local optimal value i.e. lowest fitness value occurs at 0.15.

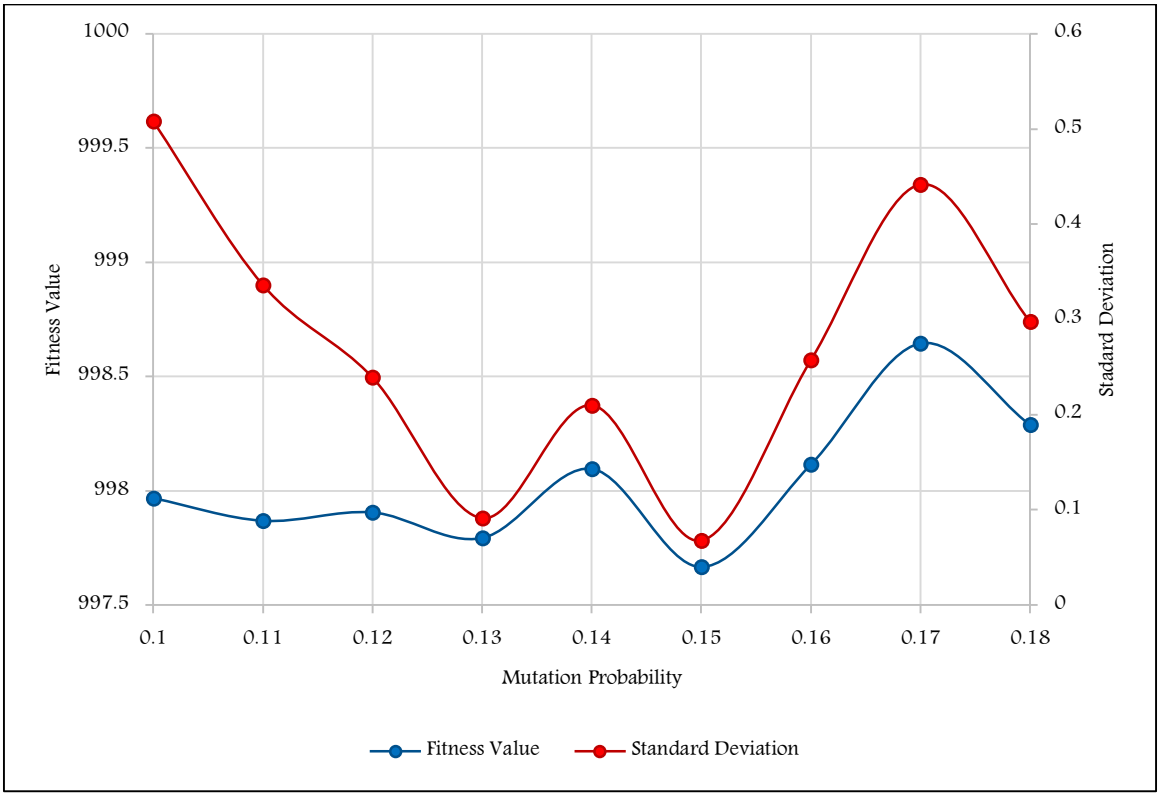

Figure 5.5 Combined plot of fitness value and standard deviation against mutation probability.

5.3.5 Effect of Network Size

An observation of the effect of network size (number of bus routes in the network) was done by varying the number of routes between 10 and 50 as seen in (Figure 5.6). This shows the near optimum fitness value occurs at the point where the network size is 40 routes. This behavior may indicate that for this scenario, the network efficiency will depreciate beyond 40 bus routes. Computation time was also observed to have a direct relationship with network size, (see Figure 5.7). This indicates an increase in time as the network size increased.

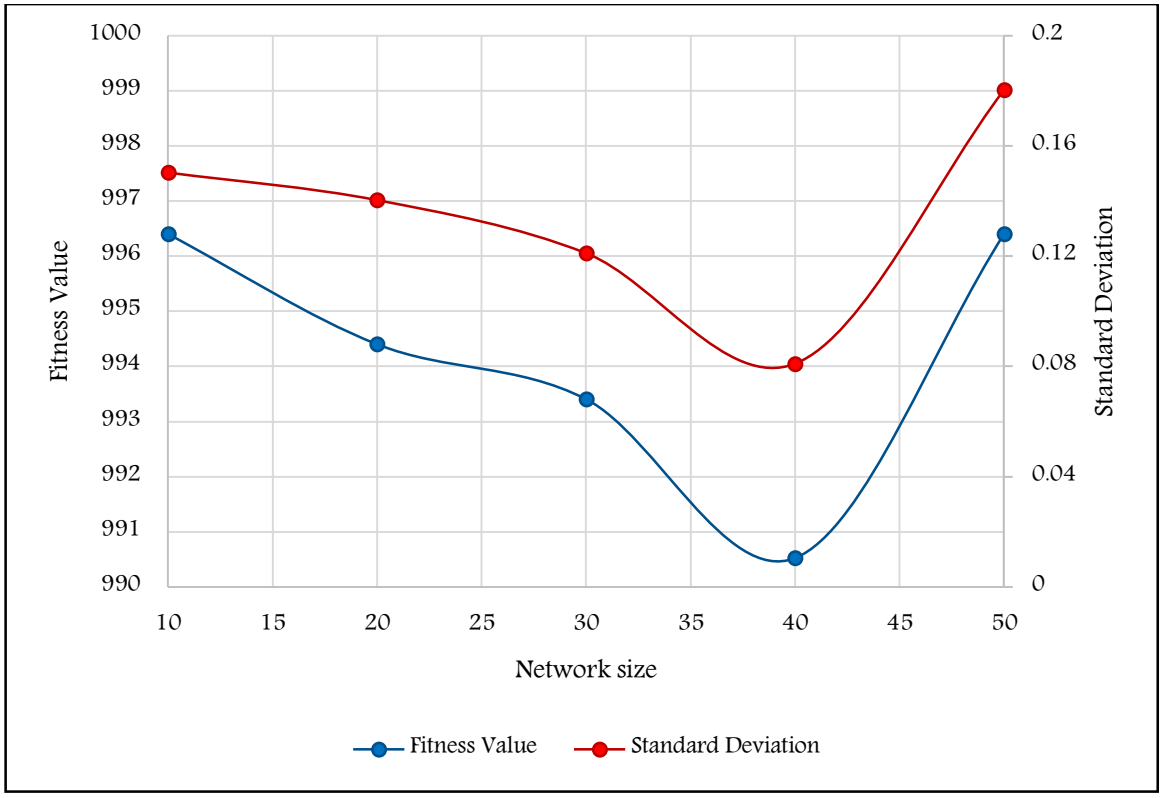

Figure 5.6 Combined plot of fitness value and standard deviation against network size

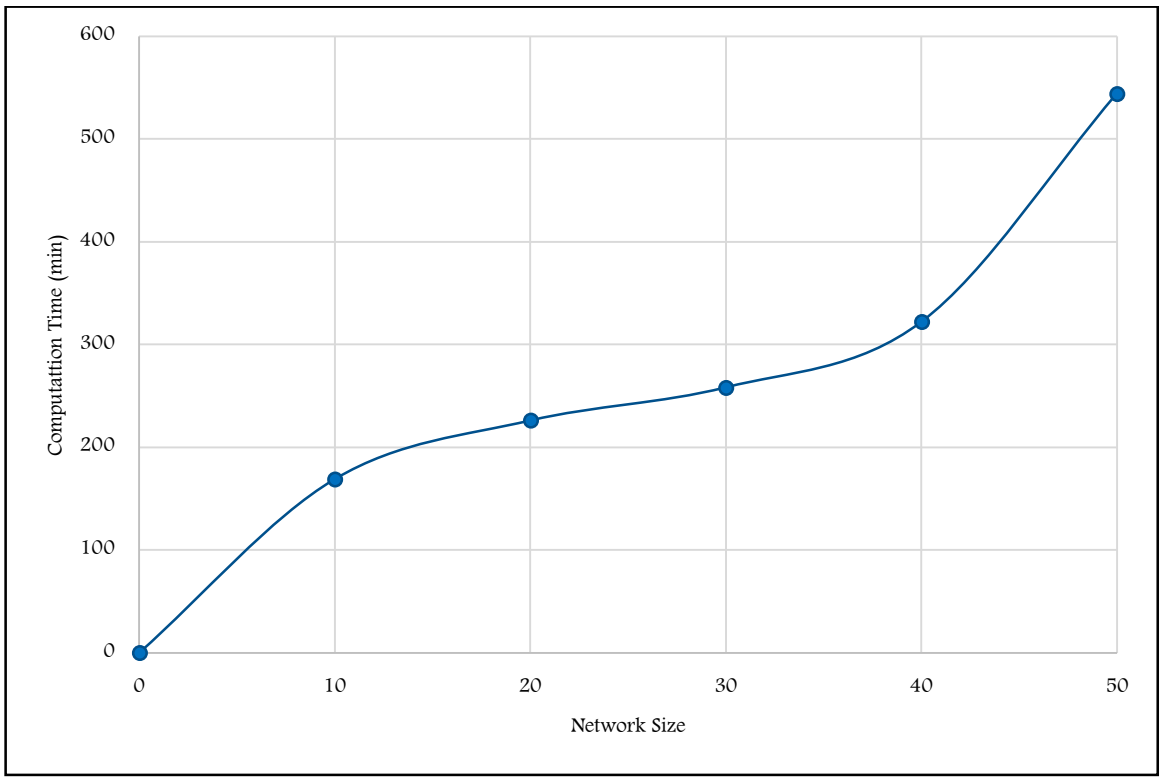

Figure 5.7 Plot of network size vs computation time

5.4 Design of a small network

In this section, a demonstration of the BRNDA complete numerical results from a typical run is presented. The test is done on the small network shown in (figure 5.8), using the parameter values from the result of the sensitivity analysis. The network consists of 25 nodes and 39 links. It was used in Ngamchai & Lovell, (2003b). The essence of this test is to get the complete run of the model's numerical results within a reasonable run time. Numerical performance indicators of the worst, median and best network options selected in each generation are shown. The final optimized network is the best option after the stopping criteria of 5 generations is reached, albeit a local optimum.

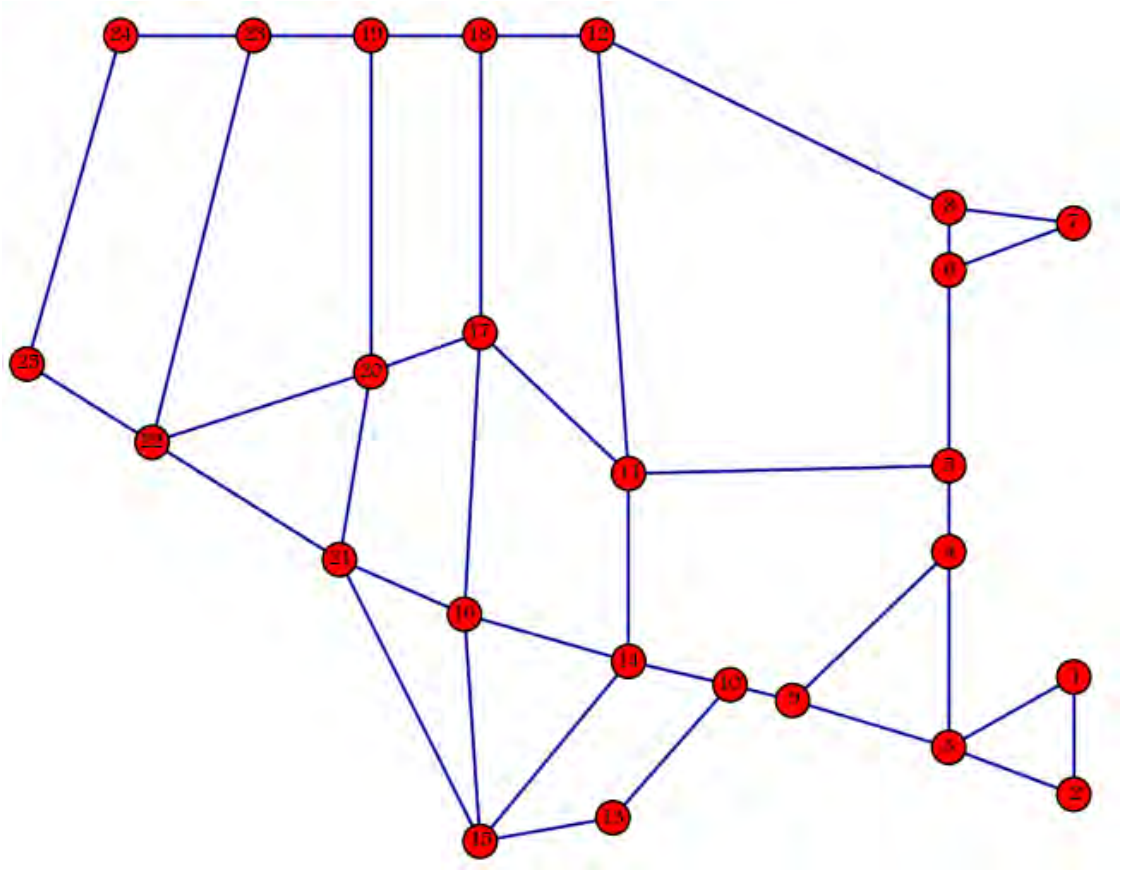

Figure 5.8 Network used to test the model adapted from Ngamchai and Lovell, (2003)

Table 5.2 below, shows the parameters used in this test case, it also indicates the component of the model where they are used. An equal cost weighting has been applied to the three components of cost

in the objective function namely user, operator and unsatisfied demand costs. This weighting reflects the objective of this research, which is to minimize the total cost, hence none of these parties are given a preference in assigning the weights. It should be noted that this can also be altered to reflect the perspective of a given stakeholder, depending on the objective of the scenario being tested. The other parameters used are from the results of the sensitivity analysis. However, five generations were selected for this trial run in order to demonstrate a complete sample of the model's output within a reasonable amount of computation time.

Component	Parameter	Value
BRNGA from Section 4.3.1	Network Size	39 routes
	Minimum Route Length	5km
	Maximum Route Length	10km
BRNAP from Section 4.3.2	Operation cost per km	R15
	Value of time in Rand	R15
	Weight of User Cost	0.333
	Weight of Operator Cost	0.333
	Weight of Unsatisfied demand Cost	0.333
	Bus Travel Speed	Randomly chosen from a predefined list
	Bus Travel Time	Computed from the value of speed
	Bus Route headway	Randomly chosen from a predefined list
BRNSA from Section 4.3.3	Crossover Probability	0.70
	Mutation Probability	0.15
	Number of Cross over points	2
	Number of Populations	5
	Number of Generations	5

Table 5.1 Description of the parameters used in the model's sample run.

5.4.1 Numerical Results from the Design of the Small Network

The results of the model's test run are demonstrated in the tables below Tables 5.2(a-f). They reveal that the model chooses a network solution in each generation, which is either an improvement on the one selected in the previous generation, or a slightly better approximation (See Generations 0 and 1 in table 5.2a and 5.2b). Given this trend it can be stated that the final network option generated in the

fifth generation (termination criteria) of this sample run, is better than others generated in previous iterations. This indicates that the model has the capability of converging towards a global optimum in the solution search space, yielding better or more acceptable results as it does. The aforementioned observable trend, seen in the results, is confirmed by the fact that the number of satisfied demand in every successive generation constantly increases or does not necessarily get worse. Furthermore, in the first generation seen in (table 5.2b), the best network satisfies the highest number of demand, but does not necessarily have the lowest unsatisfied demand. This shows that the model makes a choice of the best network solution on the basis of the defined objective which is to minimize total cost of transport and by implication maximize network utilization. While this attests to the model's effectiveness, relative to its objective, it also highlights one of the limitations of simplifying a multi-objective problem through a weighted combination of multiple criteria. See (Costelloe, Mooney & Winstanley, 2001b), where it is stated that this approach does not ensure a complete analysis of all the trade-offs which may arise from optimizing that objective function.

Table 5.2 (a-f) Numerical results from a typical run

Table 5.2a: - Generation 0			
	Worst	**Best**	**Median**
Demand	125637	141115	159602
Satisfied	54410	117112	56728
Unsatisfied	71226	24003	102874
Objective Function Value	9580096.65	19909659.60	9992710.04
Fitness Function Value	999.198	998.333	999.164

Table 5.2b - Generation 1			
	Worst	**Best**	**Median**
Demand	159602	205408	121497
Satisfied	56728	117941	113677
Unsatisfied	102874	87467	7821
Objective Function Value	9992710.04	20190959.98	19420989.48
Fitness Function Value	999.164	998.31	998.374

Table 5.2c – Generation 2			
	Worst	**Best**	**Median**
Demand	138433	195446	141115
Satisfied	110870	119396	117941
Unsatisfied	27563	76050	23173
Objective Function Value	18933017.00	20514862.90	19909659.6
Fitness Function Value	998.415	998.282	998.333

Table 5.2d – Generation 3			
	Worst	**Best**	**Median**
Demand	211934	151257	211678
Satisfied	116199	128165	119934
Unsatisfied	95734	28092	91744
Objective Function Value	19980088.42	21529757.06	2041590.39
Fitness Function Value	998.33	998.197	998.291

Table 5.2e – Generation 4			
	Worst	**Best**	**Median**
Demand	151257	200506	194910
Satisfied	128165	202787	131411
Unsatisfied	28092	*	63499.9
Objective Function Value	21529757.06	34782799.43	22594829.9
Fitness Function Value	998.197	997.088	998.108

Table 5.2f – Generation 5			
	Worst	**Best**	**Median**
Demand	186439	154750	194910
Satisfied	129171	251647	131411
Unsatisfied	57268	*	63499.9
Objective Function Value	21868822.66	41230989	22594829.9
Fitness Function Value	998.169	996.548	998.108

- indicates that the generated network has a supply capacity greater than the demand on the network

5.5 Design of Large Scale Network (Cape Town Network)

To determine the ability of the proposed model solution to design a large scale public transportation network, it is applied to the city of Cape Town transportation area. A detailed description of the area has previously been given in chapter three. The network consists of 8842 nodes and about 1.36 million O-D pairs based on the 2013 Traffic Analysis Zones data. This test case will simulate the trunk network of the Integrated Public Transport Network (IPTN) which is planned for the city of Cape Town. The input parameters for this test case will be those of the IPTN. A brief description of the IPTN will be given in (Section 5.5.1).

5.5.1 City of Cape Town Integrated Public Transport Network (CoCT IPTN)

The Integrated Public Transport Network (IPTN), is a public transit network planned in anticipation of the future effect of urban growth on travel demand in Cape Town. The network comprises of 33 trunk lines and over 200 feeder routes excluding rail. It is a long term (18 years) plan which is expected to be fully functional in 2032. The plan involves a significant expansion of the city's current public transportation network. This is logical as the population of the city is expected to grow by approximately 37% by the target year. It is expected that bus rapid transit (BRT) and Rail services will form the backbone of the IPTN. Other features of the network includes an introduction of 10 additional BRT trunk lines and an expansion of the existing rail network. A key objective of the IPTN is to ensure that at least 80% of the inhabitants in the city of Cape Town will live within 500m of a BRT trunk or rail line by the target year.

Figure 5.9 Network of IPTN trunk lines

5.5.2 Network Design

Since this test case simulates the IPTN trunk network depicted in (Figure 5.9) above, the latter will be a good means of assessing how realistic the test outputs are. While a detailed comparison in terms of operational features of the two networks is not intended, it is crucial to see if the model produces a network that has realistic similarities with the IPTN. This indicates that an enhancement of the model would make it a helpful tool for the future planning and design of optimized transportation networks in the City of Cape Town. In this test case, only the trunk lines of the IPTN will be simulated. This is due to the exponential increase in the amount of time required to test the whole network (trunk and feeder). The number of routes for the test run will be the same as the size of IPTN trunk network. The other parameter values used will be taken from the result of the sensitivity analysis done previously.

A list of all the parameters used for this test case, are shown in (Table 5.4) below. Once again an equivalent objective function weighting approach is adopted, for details see (Section 5.4).

5.5.2.1 Network Generation

The initial population is generated by the model's route generation algorithm (BRNGA). The input data for this algorithm comprises the nodes and link data for the city of Cape Town road network respectively. The generation algorithm will randomly select bus routes from the 8842 nodes on the road network and generate the candidate bus routes which will populate the solution search space. Another vital input is the route length constraint, which in this case, is set between 25 and 70km.

5.5.2.2 Network Analysis

In this stage, the generated network is analyzed and fitness values for the network is estimated. The main inputs here are the travel demand matrix obtained from the Cape Town Department of Transport (DOT). A transit assignment is carried out here in order to assign trip volumes to the generated bus networks, see (Appendix C.3). After the traffic assignment is completed, fitness values are evaluated, and stored to be used in the final phase of the model. Other parameters that are useful at this stage are mostly operational ones like the bus travel speed, headways and travel time.

5.5.2.3 Network Solution Search Stage

In this final phase of the model, the network search module of the BRNDA is used to search for an optimal/near optimal bus network. As previously discussed in (section 2.5.3), this search is guided by the GA operators. The input parameters used for this stage are crossover probability which was set at 0.7, number of populations which was set at 60, mutation probabilities set at 0.15 and number of generations which was set at 100 etc. Other parameters were used but can be seen in the table below.

Component	Parameter	Value
BRNGA from section 4.3.1	Network Size	33 routes (equal to IPTN trunk network size)
	Minimum Route Length	25km
	Maximum Route Length	70km
BRNAP from section 4.3.2	Operation cost per km	R15
	Value of time in Rand	R15
	Weight of User Cost	0.333
	Weight of Operator Cost	0.333
	Weight of Unsatisfied demand Cost	0.333
	Bus Travel Speed	Randomly chosen from a predefined list
	Bus Travel Time	Computed from the value of speed
	Bus Route headway	Randomly chosen from a predefined list
BRNSA from section 4.3.2	Crossover Probability	0.70
	Mutation Probability	0.15
	Number of Cross over points	2
	Number of Populations	60
	Number of Generations	100

Table 5.3 Details of parameters used to design the large scale network

5.6 Results

Due to time limitations, a limit of 100 generations was used as the termination criteria. This proved insufficient for the model to converge to a global optimum solution. However, a local optimum could be realized.

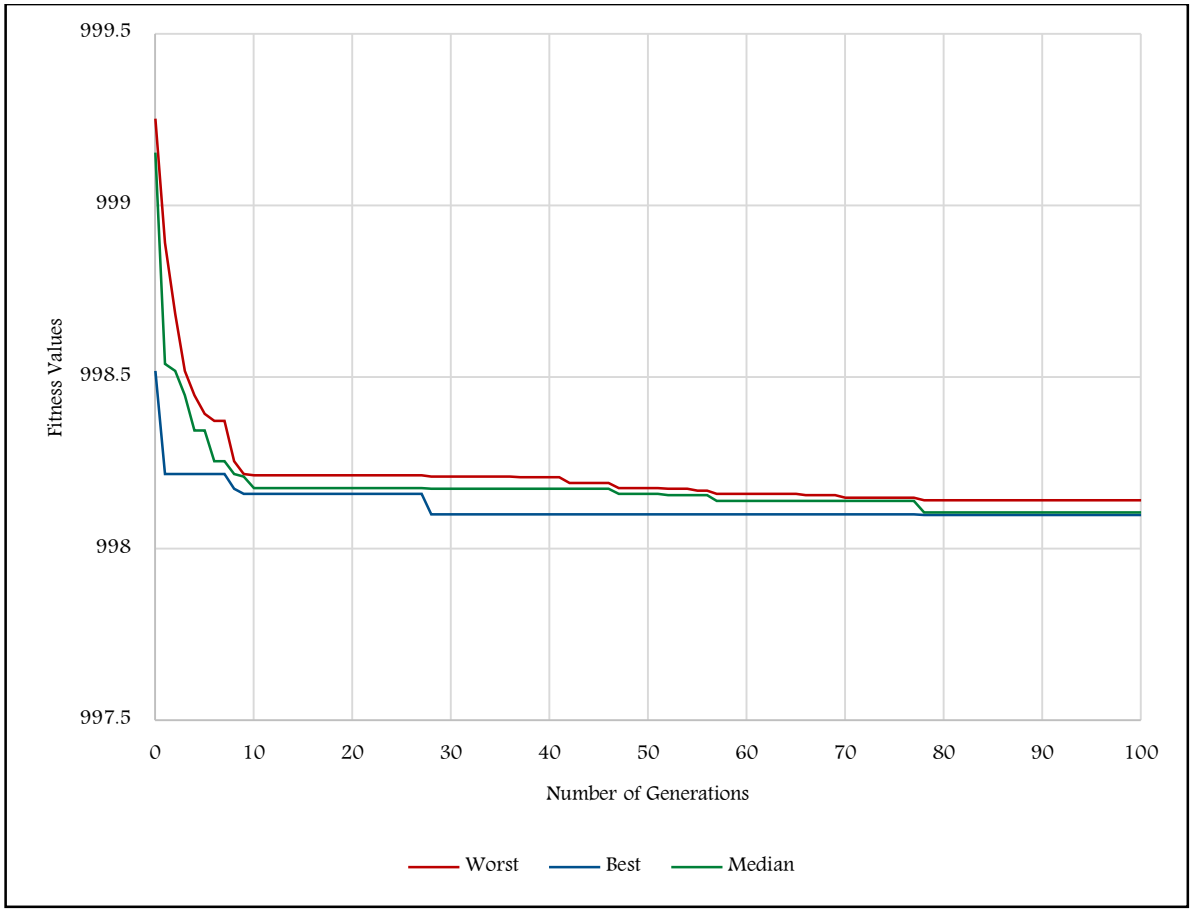

Figure 5.10 Plot of best, worst and median network after IPTN simulation

The graph in (figure 5.10 above) depicts the performance measure (fitness values) of the best median and worst networks from the test run. An observation of the graph, reveals a decrease in the fitness values, as the number of generations increase. In addition, it is observed that there is a convergence towards an optimum or near optimum solution. Given the aforementioned, it can therefore be deduced that the model is capable of generating better networks in each successive iteration. The final network was retrieved after 6060 evaluations and 360hours (15 days). The model, however, requires a substantial amount of computational resources in order to find an optimal solution. The results presented here were tested using available computer resources with a Microsoft Windows 7 system, 2.5GHz clock speed, 8GB of RAM and 750GB hard drive. Enhancements on the efficiency of the model will improve the quality of results obtained and the use of better computer hardware will reduce the computation time needed to get results. The optimized network after 100 generations, is presented in the figure 5.11 below.

Figure 5.11 Optimized network representing a local optimum solution after the IPTN simulation

The figure shows that the model is capable of creating a network which provides public transport for a reasonable portion of the inhabitants of Cape Town. This is due to the fact that the near optimal networks provided by the model has routes that are concentrated along the corridors where travel demand is high, see (sections 3.3 and 3.4). It is also interesting to note that the final network bears some resemblance with the proposed IPTN trunk lines when both are compared. Routes in the final network output, run from the CBD northwards towards Atlantis as well as south and south eastern part of the city. While these results are in their preliminary stages and hence relatively coarse, it shows that the model's output is logical. The model however needs substantially more computing resources and extended time for full testing. It is also believed that the effectiveness of the algorithm can be improved, which will ultimately improve the results generated results.

Chapter Six

6 Summary and Conclusion

6.1 Summary

One of the major ways of tackling the problems associated with transportation in South Africa is through improving the quality of public transportation networks. This has a potential of stimulating increased ridership on these networks. The direct benefit of increased public transit utilization is the reduction of air pollution, traffic congestion and energy consumption. The main goal of this research has been to design a bus transit network optimization model which uses heuristic techniques to generate an optimized bus network for Cape Town. The Bus Transit Network Design Problem involves the minimization of generalized transportation costs subject to constraints which reflect system performance conditions. In this work, an attempt is made to introduce Bus Transit Network Design knowledge to a South African context. This is crucial as the transit networks in the country are increasingly requiring redesign, occasioned by an inefficient utilization of their current design capacity. The Transit Network Design Problem (TNDP) is well researched in literature, and its nature as an NP-Hard problem makes it an extremely difficult problem to solve. Some of the characteristics which increases its complexity are many conflicting objectives, its non-convexity and non-linear constraints as well as combinatorial complexity that results from its discreet nature. In literature, attempts to solve the problem have broadly been classified as either analytical or heuristic approaches. The analytical solution approach is limited in its ability to solve the Transit Network Design Problem due to the computational complexities involved. Furthermore, they can only be applied to ideal networks, rendering them of little importance in transit network design problems with real life application. The heuristic solution methods were developed in response to this problem. They apply machine learning principles in solving NP-Hard problems and are known to generate optimal or near optimal solutions that are acceptable. The model proposed in this research is a heuristic model that consists of three major component algorithms – Bus Route Network Generation Algorithm (BRNGA), Bus Route Network Analysis Procedure (BRNAP) and Bus Route Network Search Algorithm (BRNSA). The component phases of the model randomly generates potential bus route network solutions,

analyses them and searches for the most optimized network respectively. To achieve this, different heuristic algorithms are employed at each stage of the model. Better solutions evolve with each successive generation, till a predefined stopping criteria for the model is satisfied. A sensitivity analysis is then carried out on the parameters of the model, using the city of cape Town Road network as test case. The results show that the optimal value for each of the tested parameter, is scenario dependent implying that as the network sizes change, there will be slightly different optimal values. The values gotten here are however within acceptable limits as they show consistency with those seen in literature. Further testing of the model will however yield other results that can improve the parameter optimum values. Lastly, the model is tested on both a small network and on a more realistic one namely, the Cape Town IPTN.

6.2 Conclusion

One of the major contributions, of this work has been the systematic application of the Bus Network Design knowledge to a real life network in a South African city context. This has been achieved by developing a network design solution which minimizes transport costs and maximizes bus transit utilization. The procedures employed in carrying out this research and the results obtained shows that the research questions which we set to explore have been adequately responded to as follows:

1) How is the Transit Network Design Problem (TNDP) defined?

- This question is responded to in the work, by identifying two main techniques used to solve the TNDP – analytical and heuristic techniques. Three heuristic algorithms (GA, SA, TS), which have been widely used to solve the problem were also identified. After a comparison of the three methods, genetic algorithms was highlighted as the most suited for the design of large scale networks.

2) How will an objective function for the design of public transit network in a city like Cape Town be formulated?

- This question has been responded to, by formulating an objective function which reflects the transportation context of Cape Town. This was done, by modelling it to minimize the generalized cost of transport for both the users and operators of the network in the city. Three cost components are used in the objective function which represent the user and operator perspective as well as unsatisfied demand. These sufficiently addresses the issues of network coverage and operational inefficiency discussed in the introductory chapter. Lastly, route configuration has been used as singular decision variable in the bus network design carried out in this research

3) How will the heuristic algorithm be used to design a bus network?

- This question has been responded to, by generating the bus network with the aid of heuristic algorithms such as the Dijkstra's shortest path algorithm. The generated networks were analyzed, by implementing a network analysis procedure, which performs a traffic assignment and assigns performance measures to them. Lastly, the results of the network analysis, were then used as variables to evaluate the objective function for each respective candidate bus network.

4) How will the heuristic network design algorithm be tested?

- This question has been responded to, by identifying some input data used in the network design algorithm as follows: decision variables, travel demand matrix, transport network data and feasibility constraints like route length and route number. Next, a sensitivity analysis was carried out to obtain the optimal parameter values. Lastly, the algorithm was tested by applying it to the design of both small and large scale networks. The results obtained have demonstrated that the final generated outputs, are near optimal solutions. This is because, as observed, the fitness value decreases as the number of iterations increase indicating convergence towards a global optimum solution. This is typical with minimization problems. However a substantially longer test period and computing resources will be needed to achieve a global optimum.

5) Conclusion and evaluation

The innovative significance of this work, has been the application of the TNDP knowledge to a large scale network within the context of a developing city. This has not been previously attempted in the literature. Additionally, the network analysis procedure used in this work, implemented both a traffic assignment and a Breadth First Search Algorithm to get the potential and satisfied transit demand on each network solution, this has not been attempted in the literature. The main limitations of the current model includes, its inability to deal with the optimal node/bus-stop placement which is a part of the bus planning process. Also, the reliability of the results obtained from this model could have been better, if a multi-path traffic assignment was used rather than the all-or nothing assignment utilized in the work. The major area which could be a subject to further research is to incorporate a service frequency setting module into the current algorithm.

References

Baaj, M.H. & Mahmassani, H.S. 1991. An AI-based approach for transit route system planning and design. *Journal of Advanced Transportation.* 25(2): 187-209.

Baaj, M.H. & Mahmassani, H.S. 1995. Hybrid route generation heuristic algorithm for the design of transit networks. *Transportation Research Part C: Emerging Technologies.* 3(1): 31-50.

Bäck, T., Fogel, D.B. & Michalewicz, Z. 2000. *Evolutionary computation 1: Basic algorithms and operators.* CRC Press.

Behrens, R. & Behrens, R. 2004. Understanding travel needs of the poor: towards improved travel analysis practices in South Africa. *Transport Reviews.* 24(3): 317-336.

Beltran, B., Carrese, S., Cipriani, E. & Petrelli, M. 2009. Transit network design with allocation of green vehicles: A genetic algorithm approach. *Transportation Research Part C: Emerging Technologies.* 17(5): 475-483.

Bielli, M., Caramia, M. & Carotenuto, P. 2002. Genetic algorithms in bus network optimization. *Transportation Research Part C: Emerging Technologies.* 10(1): 19-34.

Boyd, S., Xiao, L. & Mutapcic, A. 2003. Subgradient methods. *Lecture Notes of EE392o, Stanford University, Autumn Quarter.* 2004.

Carrese, S. & Gori, S. 2002. An urban bus network design procedure. In *Transportation planning.* Springer. 177-195.

Chakroborty, P. 2003. Genetic algorithms for optimal urban transit network design. *Computer-Aided Civil and Infrastructure Engineering.* 18(3): 184-200.

Chakroborty, P., Deb, K. & Srinivas, B. 1998. Network-Wide Optimal Scheduling of Transit Systems Using Genetic Algorithms. *Computer-Aided Civil and Infrastructure Engineering.* 13(5): 363-376.

Chakroborty, P. & Wivedi, T. 2002. Optimal route network design for transit systems using genetic algorithms. *Engineering Optimization.* 34(1): 83-100.

Chen, A., Kim, J., Lee, S. & Kim, Y. 2010. Stochastic multi-objective models for network design problem. *Expert Systems with Applications.* 37(2): 1608-1619.

Chien, S.I. & Spasovic, L.N. 2002. Optimization of grid bus transit systems with elastic demand. *Journal of Advanced Transportation.* 36(1): 63-91.

Chien, S., Yang, Z. & Hou, E. 2001. Genetic algorithm approach for transit route planning and design. *Journal of Transportation Engineering.* 127(3): 200-207.

Cipriani, E., Fusco, G., Gori, S. & Petrelli, M. 2005. A procedure for the solution of the urban bus network design problem with elastic demand. *Proceedings of 10th EWGT Meeting on Advances in Modeling, Optimization and Management of Transportation Processes, Poznan, Poland.* 681.

Cipriani, E., Gori, S. & Petrelli, M. 2012. Transit network design: A procedure and an application to a large urban area. *Transportation Research Part C: Emerging Technologies.* 20(1): 3-14.

City of Cape Town - ITP 2006. *Comprehensive Integrated Transport Plan 2006 - 2011 (Review and Update 2009).*

City of Cape Town - ITP 2013. *Final Draft 2013 - 2018 Integrated Transport Plan.* South Africa: Royal HaskoningDHV (Pty) Ltd.

Constantin, I. & Florian, M. 1995. Optimizing Frequencies in a Transit Network: a Nonlinear Bi- level Programming Approach. *International Transactions in Operational Research.* 2(2): 149-164.

Costelloe, D., Mooney, P. & Winstanley, A. 2001a. Multi-Objective Optimisation on Transportation Networks. *Proc. of 4th AGILE Conference, Brno, Czech Republic.*

Costelloe, D., Mooney, P. & Winstanley, A. 2001b. Multi-Objective Optimisation on Transportation Networks. *Proc. of 4th AGILE Conference, Brno, Czech Republic.*

Curtin, K.M. 2004. Operations Research. *Encyclopedia of Social Measurement.* 925-931.

De Jong, K.A. 1975. Analysis of the behavior of a class of genetic adaptive systems.

Department of Transport - Public Transport Strategy. 2007. *Public Transport Strategy.* South Africa:

Dial, R.B. 1971. A probabilistic multipath traffic assignment model which obviates path enumeration. *Transportation Research.* 5(2): 83-111.

Dubois, D., Bel, G. & Llibre, M. 1979. A set of methods in transportation network synthesis and analysis. *Journal of the Operational Research Society.* 797-808.

ESRI 2011. ArcGIS desktop: release 10.

Fan, L. & Mumford, C.L. 2010. A metaheuristic approach to the urban transit routing problem. *Journal of Heuristics.* 16(3): 353-372.

Fan, W. & Machemehl, R.B. 2002. Characterizing bus transit passenger waiting times. *2nd Material Specialty Conference of the Canadian Society of Civil Engineering, Montreal, Quebec, Canada.*

Fan, W. & Machemehl, R.B. 2004. Optimal transit route network design problem: Algorithms, implementations, and numerical results. *Transport Research Board.*

Fan, W. & Machemehl, R.B. 2006. Optimal transit route network design problem with variable transit demand: genetic algorithm approach. *Journal of Transportation Engineering.* 132(1): 40–51.

Fletterman, M. 2008. *Designing Multimodal Public Transport Networks using Metaheuristics.*

Fonseca, C.M. & Fleming, P.J. 1993. Genetic Algorithms for Multiobjective Optimization: FormulationDiscussion and Generalization. *ICGA.* 416.

Gallo, M., Montella, B. & D'Acierno, L. 2011. The transit network design problem with elastic demand and internalisation of external costs: An application to rail frequency optimisation. *Transportation Research Part C: Emerging Technologies.* 19(6): 1276–1305.

Garrett, A. Python Libraries for Evolutionary Computation. *Genome.* 1(G1DList): 20.

Glover, F. 1986. Future paths for integer programming and links to artificial intelligence. *Computers & Operations Research.* 13(5): 533–549.

Goldberg, D.E. & Holland, J.H. 1988. Genetic algorithms and machine learning. *Machine Learning.* 3(2): 95–99.

Grefenstette, J.J. 1986. Optimization of control parameters for genetic algorithms. *Systems, Man and Cybernetics, IEEE Transactions On.* 16(1): 122–128.

Hagberg, A., Swart, P. & S Chult, D. 2008. *Exploring Network Structure, Dynamics, and Function using NetworkX.*

Henderson, D., Jacobson, S.H. & Johnson, A.W. 2003. The theory and practice of simulated annealing. In Handbook of metaheuristics. Springer. 287–319.

Hillier, F.S. 1995. *Introduction to operations research.* Tata McGraw-Hill Education.

Holand, J.H. 1975. Adaptation in natural and artificial systems. *Ann Arbor: The University of Michigan Press.*

Hunter, J.D. 2007. Matplotlib: A 2D graphics environment. *Computing in Science & Engineering.* 9(3): 0090–95.

INC, I.C. 2010. EMME/3 User's Manual. *INRO Consultants Inc.*

Kepaptsoglou, K. & Karlaftis, M. 2009. Transit route network design problem: review. *Journal of Transportation Engineering.* 135(8): 491–505.

Kov, M., Fukuda, D. & Yai, T. 2006. Frequency Optimization of Bus Network in Urban Mixed Traffic.

Kumar, R. 2012. Jyotishree: Blending Roulette Wheel Selection & Rank Selection in Genetic Algorithms. *Proceedings of International Conference on Machine Learning and Computing.* 197.

Lampkin, W. & Saalmans, P. 1967. The design of routes, service frequencies, and schedules for a municipal bus undertaking: A case study. *OR.* 375-397.

LeBlanc, L.J. 1988. Transit system network design. *Transportation Research Part B: Methodological.* 22(5): 383-390.

LeBlanc, L.J. & Abdulaal, M. 1984. A comparison of user-optimum versus system-optimum traffic assignment in transportation network design. *Transportation Research Part B: Methodological.* 18(2): 115-121.

Lee, Y. & Vuchic, V.R. 2005. Transit network design with variable demand. *Journal of Transportation Engineering.* 131(1): 1-10.

Magnanti, T.L. & Wong, R.T. 1984. Network design and transportation planning: Models and algorithms. *Transportation Science.* 18(1): 1-55.

Mandl, C.E. 1980. Evaluation and optimization of urban public transportation networks. *European Journal of Operational Research.* 5(6): 396-404.

Michalewicz, Z. 1996. *Genetic algorithms data structures= evolution programs.* springer.

Newell, G. 1979. Some issues relating to the optimal design of bus routes. *Transportation Science.* 13(1): 20-35.

Ngamchai, S. & Lovell, D.J. 2003a. Optimal time transfer in bus transit route network design using a genetic algorithm. *Journal of Transportation Engineering.* 129(5): 510-521.

Ngamchai, S. & Lovell, D.J. 2003b. Optimal time transfer in bus transit route network design using a genetic algorithm. *Journal of Transportation Engineering.* 129(5): 510-521.

Pattnaik, S., Mohan, S. & Tom, V. 1998. Urban bus transit route network design using genetic algorithm. *Journal of Transportation Engineering.* 124(4): 368-375.

Pearl, J. 1984. *Heuristics.* Addison-Wesley Publishing Company Reading, Massachusetts.

Peng, S. & Fan, H.S. 2007. A New Computational Model for the Design of an Urban Inter- modal Public Transit Network. *Computer- Aided Civil and Infrastructure Engineering.* 22(7): 499-510.

Poli, R., Langdon, W.B., McPhee, N.F. & Koza, J.R. 2008. *A field guide to genetic programming.* Lulu. com.

Python Software Foundation 2012. The python programming language.

Rea, J.C. 1972. Designing urban transit systems: an approach to the route-technology selection problem. *Transport Research Board.*

Rodrigue, J., Comtois, C. & Slack, B. 2006. Transport Systems and Networks - (Chapter 2). In *The Geography of transport systems.* 2nd ed. London: Routledge.

Rodrigue, J. & Notteboom, T. 2013. Transporttaion and Economic Development: (Chapter 7 - Section 1). In *The geography of transport systems.* Routledge.

Schaffer, J.D., Caruana, R.A., Eshelman, L.J. & Das, R. 1989. A study of control parameters affecting online performance of genetic algorithms for function optimization. *Proceedings of the Third International Conference on Genetic Algorithms.* Morgan Kaufmann Publishers Inc. 51.

Shih, M. & Mahmassani, H.S. 1994. A design methodology for bus transit networks with coordinated operations. *Transport Research Board.*

Shimizu, K., Ishizuka, Y. & Bard, J.F. 1997. *Nondifferentiable and two-level mathematical programming.* Kluwer Academic Publishers Boston.

Siriwardene, N. & Perera, B. 2006. Selection of genetic algorithm operators for urban drainage model parameter optimisation. *Mathematical and Computer Modelling.* 44(5): 415-429.

Statistics South Africa - Census 2011. Census 2011.

Statistics South Africa - NHTS 2013. *National Household Travel Survey.* Pretoria, South Africa:

Talbi, E. 2009. *Metaheuristics: from design to implementation.* John Wiley & Sons.

Van Nes, R., Hamerslag, R. & Immers, B.H. 1988. Design of public transport networks. *Transportation Research Record.* 120274-83.

Van Nes, R. & Bovy, P.H. 2000. Importance of objectives in urban transit-network design. *Transportation Research Record: Journal of the Transportation Research Board.* 1735(1): 25-34.

Van Rossum, G. 2007. Python Programming Language. *USENIX Annual Technical Conference.*

Yang, H. & H. Bell, M.G. 1998. Models and algorithms for road network design: a review and some new developments. *Transport Reviews.* 18(3): 257-278.

Zhao, F. & Zeng, X. 2008. Optimization of transit route network, vehicle headways and timetables for large-scale transit networks. *European Journal of Operational Research.* 186(2): 841-855.

Appendices

Appendix A: Raw Input data for the small test network

Table A.1: X and Y coordinates of nodes in the small test network

Node	X	Y
1.00	13.40	2.10
2.00	13.40	0.60
3.00	11.80	1.20
4.00	11.80	3.70
5.00	11.80	4.80
6.00	11.80	7.30
7.00	13.40	7.90
8.00	11.80	8.10
9.00	9.80	1.80
10.00	9.00	2.00
11.00	7.70	4.70
12.00	7.30	10.30
13.00	7.50	0.30
14.00	7.70	2.30
15.00	5.80	0.00
16.00	5.60	2.90
17.00	5.80	6.50
18.00	5.80	10.30
19.00	4.40	10.30
20.00	4.40	6.00
21.00	4.00	3.60
22.00	1.60	5.10
23.00	2.90	10.30
24.00	1.20	10.30
25.00	0.00	6.10

Table A.2: O-D matrix of small test network

		DESTINATIONS NODES																							
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
	1	0	100	150	150	10	20	15	125	115	50	150	115	50	21	115	60	100	60	50	100	35	10	10	10	10
	2	100	0	10	2	2	1	3	3	4	5	3	3	3	1	3	3	3	3	2	5	4	2	1	1	1
	3	150	10	0	150	10	150	250	150	50	150	25	10	115	25	10	10	15	5	5	150	25	10	5	50	5
	4	150	2	150	0	25	25	25	10	150	100	50	30	20	35	7	80	12	25	30	20	30	10	20	50	5
	5	10	2	10	25	0	30	25	10	15	10	10	15	25	40	45	30	10	15	10	25	10	10	25	50	20
ORIGIN NODES	6	20	1	150	25	30	0	15	30	10	15	5	10	10	15	25	30	25	20	30	30	20	10	25	100	20
	7	15	3	250	25	25	15	0	2	2	2	1	2	3	5	4	3	3	1	1	3	2	1	3	5	3
	8	125	3	150	10	10	30	2	0	30	30	25	10	10	10	10	10	15	50	25	25	5	10	40	75	25
	9	115	4	50	150	15	10	2	30	0	10	10	5	7	7	15	15	15	15	15	50	5	5	10	65	25
	10	50	5	150	100	10	15	2	30	10	0	3	3	3	3	3	3	3	3	3	2	5	10	10	100	25
	11	150	3	25	50	10	5	1	25	10	3	0	4	4	4	4	4	2	2	3	5	5	5	5	35	25
	12	115	3	10	30	15	10	2	10	5	3	4	0	40	40	40	40	40	40	40	50	5	10	5	40	25
	13	50	3	115	20	25	10	3	10	7	3	4	40	0	35	25	30	25	20	20	50	5	15	15	25	25
	14	21	1	25	35	40	15	5	10	7	3	4	40	35	0	40	40	35	30	30	10	5	20	10	25	25
	15	115	3	10	7	45	25	4	10	15	3	4	40	25	40	0	45	100	50	50	15	5	35	25	25	25
	16	60	3	10	80	30	30	3	10	15	3	4	40	30	40	45	0	10	25	25	20	5	40	25	25	20
	17	100	3	15	12	10	25	3	15	15	3	2	40	25	35	100	10	0	10	5	100	10	40	30	25	40
	18	60	3	5	25	15	20	1	50	15	3	2	40	20	30	50	25	10	0	10	100	10	40	20	25	40
	19	50	2	5	30	10	30	1	25	15	3	3	40	20	30	50	25	5	10	0	100	15	25	5	5	40
	20	100	5	150	20	25	30	3	25	50	2	5	50	50	10	15	20	100	100	100	0	10	25	10	5	40
	21	35	4	25	30	10	20	2	5	5	5	5	5	5	5	5	5	10	10	15	10	0	10	10	5	40
	22	10	2	10	10	10	10	1	10	5	10	5	10	15	20	35	40	40	40	25	25	10	0	10	5	40
	23	10	1	5	20	25	25	3	40	10	10	5	5	15	10	25	25	30	20	5	10	10	10	0	5	40
	24	10	1	50	50	50	100	5	75	65	100	35	40	25	25	25	25	25	25	5	5	5	5	5	0	40
	25	10	1	5	5	20	20	3	25	25	25	25	25	25	25	25	20	40	40	40	40	40	40	40	40	0

Appendix B: Formatted node data

The formatted data presented below shows the format in which node input data is read into in the Bus Route Network Design Algorithm. The format contains the node identifier and X, Y coordinate of the nodes in this set-up: {Node_label: (X-coordinate, Y-coordinate)}

B.1 Formatted nodes data:

{

1: (13.4, 2.1), 2: (13.4, 0.6),

3: (11.8, 1.2), 4: (11.8, 3.7),

5: (11.8, 4.8), 6: (11.8, 7.3),

7: (13.4, 7.9), 8: (11.8, 8.1),

9: (9.8, 1.8), 10: (9.0, 2.0),

11: (7.7, 4.7), 12: (7.3, 10.3),

13: (7.5, 0.3), 14: (7.7, 2.3),

15: (5.8, 0.0), 16: (5.6, 2.9),

17: (5.8, 6.5), 18: (5.8, 10.3),

19: (4.4, 10.3), 20: (4.4, 6.0),

21: (4.0, 3.6), 22: (1.6, 5.1),

23: (2.9, 10.3), 24: (1.2, 10.3),

25: (0.0, 6.1)

}

B.2 Formatted links data:

The formatted data presented below shows the format in which link input data is read into in the Bus Route Network Design Algorithm. It consists of the start and end nodes of that link in addition to the link weight or attribute represented like this; [(Start_node, End_node, {Link Attribute})].

[

(1, 2, {'length': 0.07}), (1, 3, {'length': 0.1}),

(2, 3, {'length': 0.1}), (3, 4, {'length': 0.15}),

(3, 9, {'length': 0.11}), (4, 5, {'length': 0.05}),

(4, 9, {'length': 0.16}), (5, 6, {'length': 0.15}),

(5, 11, {'length': 0.15}), (6, 7, {'length': 0.09}),

(6, 8, {'length': 0.02}), (7, 8, {'length': 0.08}),

(8, 12, {'length': 0.29}), (9, 10, {'length': 0.02}),

(10, 14, {'length': 0.05}), (10, 13, {'length': 0.13}),

(11, 14, {'length': 0.19}), (11, 12, {'length': 0.39}),

(11, 17, {'length': 0.15}), (12, 18, {'length': 0.07}),

(13, 15, {'length': 0.09}), (14, 15, {'length': 0.18}),

(14, 16, {'length': 0.12}), (15, 16, {'length': 0.20}),

(15, 21, {'length': 0.26}), (16, 21, {'length': 0.09}),

(16, 17, {'length': 0.23}), (17, 18, {'length': 0.25}),

(17, 20, {'length': 0.07}), (18, 19, {'length': 0.06}),

(19, 20, {'length': 0.28}), (19, 23, {'length': 0.08}),

(20, 22, {'length': 0.18}), (20, 21, {'length': 0.14}),

(21, 22, {'length': 0.17}), (22, 23, {'length': 0.36}),

(22, 25, {'length': 0.1}), (23, 24, {'length': 0.08}),

(24, 25, {'length': 0.29})

]

Appendix C: Algorithms Utilized in the Bus Route Network Design Algorithm

C.1 Breadth First Search Algorithm (BFSA)

This is one of the most basic graph transversal algorithm. Cormen, T. H, et al. (2009) states that other algorithms like Prim's minimum-spanning tree algorithm and Dijkstra's single-source shortest-paths algorithm, use ideas similar to those in breadth-first search when the search essentially consists of two procedures namely visit and examine a node of a graph followed by visit the neighbor nodes of the presently visited node. The BFSA starts at a root node and inspects all the adjacent nodes. Then for each of those neighbor nodes in turn, it inspects their previously unvisited neighbor nodes and so on. Mathematically, the breadth first algorithm is presented thus:

Given a graph $G = (V, E)$ and a distinguished source vertex s, breadth-first search systematically explores the edges of G to "discover" every vertex that is reachable from s. It computes the distance (smallest number of edges) from s to each reachable vertex. It also produces a "breadth-first tree" with root s that contains all reachable vertices. For any vertex v reachable from s, the simple path in the breadth-first tree from s to v corresponds to a "shortest path" from s to v in G, that is, a path containing the smallest number of edges. The algorithm works on both directed and undirected graphs. Breadth-first search is so named because it expands the frontier between discovered and undiscovered vertices uniformly across the breadth of the frontier. That is, the algorithm discovers all vertices at distance k from s before discovering any vertices at distance k + 1. The breadth-first-search algorithm BFSA assumes that the input graph $G = (V, E)$ is denoted using adjacency lists. It attaches several additional attributes to each vertex in the graph. The colour of each vertex $(u \in V)$ in the attribute (u.colour) and the predecessor of u in the attribute ($u.\pi$). If u has no predecessor (for example, if $u = s$ or u has not been discovered), then $(u.colour = Nil)$. The attribute $(u.d)$ holds the distance from the source s to vertex u computed by the algorithm. The algorithm also uses a first-in, first-out queue Q to manage the set of gray vertices.

BFSA pseudo code by Cormen, T. H, et al. (2009)

```
1)  for each vertex u ∈ G.V − {s}
2)          u.colour = WHITE
3)          u.d = ∞
4)          u.π = NIL
5)  s.color = GRAY
6)  s.d = 0
7)  s.π = NIL
8)  Q ≠ Ø
9)  ENQUEUE. (Q, s)
10) while Q = Ø
11)         u =  DEQUEUE (Q)
12)         for each v ∈ G:Adj[u]
13)                 if v.color == WHITE
14)                         v.color = GRAY
15)                         v.d = u.d + 1
16)                         v. π = u
17)                         ENQUEUE (Q, v)
18)     u.colour =  BLACK
```

C.2 Shortest Path Algorithm

Dijkstras's shortest path algorithm, which is named after the Dutch Scientist Edsger Dijkstra, has been used extensively in transportation engineering to determine the shortest path between any pair of nodes within a transportation network. The algorithm is briefly discussed in this work due to the significant role it plays in the route generation stage of the model An investigation of literature reveals that there are two classes of algorithms used to compute the shortest path between nodes in a transit network. The two are known as label setting and label correcting algorithms, see Gallo G and Pallotino S, (1998). The former are known to be more efficient for computing the shortest path between one source node and one destination node (one to one) networks, while the latter is considered more efficient for getting all the shortest path between one source and many destinations (one to many) networks. The main difference between the two is in the way they converge to the optimal shortest path distances. While the label setting algorithm scans and permanently labels destination node after each iteration, the label correcting algorithm labels all nodes as temporary until all are scanned and permanently labelled in the final iteration as in Zhan F.B. and Woon C.E, (2000). Djikstras's algorithm is considered a label setting algorithm in that it associates nodes with labels. Permanent labels are assigned to nodes that have been reached while temporary ones are assigned to the nodes that are yet to be reached. The main idea is to change the temporary labels to permanent as the node is added to the shortest path tree. The permanent label of a node, represents the shortest path distance from a given source to that node. For a mathematical statement of Dijkstra's Shortest Path Algorithm, consider a network composed of a set of nodes $N$and a set of arcs A which is represented as a weighted directed graph. $G= (N, A)$ with $n = |N|$ for nodes and $a = |A|$ for the arcs. Each link (i, j) is represented as an ordered pair of nodes ie node (i) and node (j). Assume C_{ij} is the non-negative cost (distance) associated with each link. For an arbitrary source node P and another arbitrary destination node Q. at each iteration, the algorithm chooses the node $i \in \bar{P}$ with the least temporary distance, and makes it permanent, records its ancestors index and update the temporary values of all nodes $j \in A(i)$. Repeat this procedure till all nodes becomes permanent.

Dijkstra's Algorithm adopted from Fan and Machemehl, (2004)

begin

d (i) = ∞ and p(i) = -1 for each node i∈ N ;

d (s) = 0 and p(s) = 0;

P = Φ and P = N;

while |P| < N **do**

begin

Select a node i of smallest d (i) = min {d (j): j∈ $\overline{P}$ }

Make node i permanent and delete i from $\overline{P}$, P = P ∪ {i} $\overline{P}$ = $\overline{P}$ - {i}

for each (i, j) ∈ A(i) **do**

if d(j) > d(i) + C_{ij} **then**

begin

Do distance update

d (j) := d(i) + C_{ij} and pred (j) := i;

end;

end;

end;

Where:

A (i) = arc adjacency list of node i

P = set of all nodes with permanent labels

$\overline{P}$ = set of all nodes with permanent labels

∞ = distance from source to all other nodes

C.3 All-or Nothing Assignment.

This assignment model assumes all trips will be made from any origin to any destination by a singular least cost path (shortest path), hence it loads all trips on that path. A criticism of this traffic assignment method is that it doesn't reflect realistic passenger behavior since it does not consider link volume capacity. Another criticism of the assignment technique arises from its apparent neglect of other least cost path which might be similar to the shortest path. However, here is also some support for the usefulness of the All-or-Nothing assignment, which describes the model's result as an indicator of the preferred or desired paths. This information is very helpful for transit authorities to help them prioritize route investment alternatives, see (Mathew and Krishna, 2006). Since the thrust of this work was route network design and not frequency setting, the All-or-Nothing assignment was suitable for the computation of route frequencies, which were used to evaluate the objective function.

C.4 Pseudo code for the Crossover operator from Fan and Machemehl, (2004)

See details in section 4.3.3.1.2

Step 1: Set i = 1;

Step2: Generate a random number r within the range [0, 1);

Step 3: If c r ≤ P, then select chromosome i for crossover;

Step 4: i = i +1;

Step 5: Repeat the above steps until i > pop _ size.

C.5 Mutation Pseudo code for the Mutation operator from Fan and Machemehl, (2004)

See details in section 4.3.3.1.3

Step 1: Set i = 1;

Step 2: Generate a random number r ∈ [0, 1) for each bit;

Step 3: If m r ≤ p, mutate the bit;

Step 4: i = i +1;

Step 5: Repeat the above steps until i > m· pop _ size.

Appendix D: Python Codes for the Bus Route Network Design Algorithm (BRNDA)

D.1 '''BUS ROUTE NETWORK GENERATION ALGORITHM (BRNGA)'''

```
def generate_routes(random, args):
    """
     This function generates the initial individual bus network. It
     iteratively selects random terminal nodes (A and B), and gets
     the bus route between them through the implementation of
     Dijkstra's shortest path algorithm. This is done subject to
     user determined parameters like min/max route length as well
     as route number. Developed by Obiora .A. Nnene (2014)
    """

    # Declaration of global variables
    global shortest_path_length, edge, rail_edge
    individual = []
    shortest_path_length = []

    # Randomly select terminal nodes A and B, then evaluate the
    shortest path between A and B.
    for i in xrange(number_of_routes):
        while True:
            try:
                A = int(rand.choice(road_pos.keys()))
                B = int(rand.choice(road_pos.keys()))
                if A == B or A in rail_pos or B in rail_pos:
                    break
                else:
                    path = nx.dijkstra_path(X, A, B,
                    weight='length')
                    pathlength = round(nx.dijkstra_path_length(X, A,
                    B, weight='length'), 2)
                if min_len <= pathlength <= max_len:
                    individual.append(path)
                    shortest_path_length.append(pathlength)
```

```
                break
            else: pass
        except: pass

# Read road network link data
rail = open('rail_edges1.txt', 'r').read()
rail_edge = literal_eval(rail)
X.add_edges_from(rail_edge, speed=rail_speed)

# Calculate traveltime on rail links and add them to the link or
arc attributes
for r2 in rail_edge:
    r2[2]['traveltime'] = r2[2]['length'] / rail_speed
    r2[2]['speed'] = rail_speed

# Make the enlarged network link dictionary by merging road and
rail sub networks
edge = rail_edge + road_edge
return individual

# Loop through pos and find shortest path node positions then
append their dictionary values to empty list sp_node_positions
    sp_node_pos = []
    for path in cs1:
        temp_list = []
        for nodes in path:
            if nodes in pos:
                temp_list.append(pos[nodes])
        sp_node_pos.append(temp_list)

  # Add offset values equivalent to (sum of penalties for using
  bus network) to sp_node_positions. Append the bus node
  positions to the bus_node_pos list
    bus_node_pos = []
    alight_pen = 10
```

```
        offsets = [(-(30 / 2 + alight_pen), -(30 / 2 + alight_pen))]
        xy = sp_node_pos[:]
        for i,lst in enumerate(sp_node_pos):
            list_temp = list()
            for j, item in enumerate(lst):
                xy[i][j] = (sp_node_pos[i][j][0] + offsets[0][0],
sp_node_pos[i][j][1] + offsets[0][1])
                list_temp.append(xy[i][j])
```

 # Make a new dictionary (bus_route_nodes_dict) of bur route nodes and their positions in the format {B : (X1, Y1)} where (X1, Y1) are tuples Update the initial walking network nodes data in line 30 with the bus route nodes

```
        bus_route_nodes_dict =
dict(izip(chain.from_iterable(bus_route_nodes),
chain.from_iterable(bus_node_pos)))
        pos.update(bus_route_nodes_dict)
```

 # Make bus route edge tuples of lagging and leading bus route nodes:

```
        bus_route_edge = []
        for line in bus_route_nodes:
            temp1 = []
            for i in range(len(line) - 1):
                temp1.append((line[i], line[i + 1]))
            bus_route_edge.append(temp1)

        bus_route_edge1 = []
        for line in cs3:
            temp2 = []
            for i in range(len(line) - 1):
                temp2.append((line[i], line[i + 1]))
            bus_route_edge1.append(temp2)
```

```
# Get attributes of the newly created bus route links

    a = {(item[0], item[1]): item[2] for item in edge}

    bus_route_edge1 = [item for sublist in bus_route_edge1 for
item in sublist]

    bus_route_attributes = [a[item] for item in bus_route_edge1
if item in a]


# Create a new list of bus route link data from bus_route_edge
and bus_route_attributes,

    bus_route_edgelist = [(k1[0], k1[1], k2) for k1, k2 in
zip(chain.from_iterable(bus_route_edge), bus_route_attributes)]


# Make bus routes access (boarding and alighting)links

    board = []

    alight = []

    board1 = []

    alight1 = []

    for i,access_routes in enumerate(cs1):

        tem1 = []

        tem2 = []

        board = board + (list(tuple(zip(cs2[i],
bus_route_nodes[i]))))    # boarding direction

        alight = alight + (list(tuple(zip(bus_route_nodes[i],
cs2[i]))))

        tem1 = tem1 + (list(tuple(zip(cs2[i],
bus_route_nodes[i]))))

        tem2 = tem2 + (list(tuple(zip(bus_route_nodes[i],
cs2[i]))))

        board1.append(tem1)

        alight1.append(tem2)

    alighting = [alit + ({'traveltime': 0.0005, 'length':
0.25},) for sub in alight1 for alit in sub]

    boarding = [brd + ({'traveltime': 0.0005, 'length': 0.25},)
for sub in board1 for brd in sub]
```

```
    # Populate bus route list with access routes

        bus_route_edgelist1 = bus_route_edgelist + alighting +
boarding


    # Populate study network links with bus routes

        edge1 = edge + bus_route_edgelist1


    # Write the network matrix that will be used for the traffic
assignment to an external csv file

        graf = open('graf.csv', 'w')

        print >> graf, 'Link,O,D,Impedance'

        index = 1

        for e in edge1:

            print >> graf, index, ',', e[0], ',', e[1], ',',
e[2]['traveltime']

            index += 1

        graf.close()
```

D.2 '''BUS ROUTE NETWORK ANALYSIS PROCEDURE (BRNAP)'''

```
def evaluate_routes(candidates, args):

    """
          This function assigns traffic to the bus routes generated
          in the BRNGA. It then computes their objective function
          expression by evaluating the objective function
          expression. The raw objective function values are
          transformed to non-negative fitness values using the
          transform function (V - (Oi * P) / sum (Oi)). A Breadth
          First Search Algorithm (BFSA) is also implemented to
          determine the potential demand for each bus route
          network. Developed by Obiora .A. Nnene (2014)
    """

    # Define Variables
    user = []
    avr_obj = []
    fitness = []
    ntwk_vol = []
    obj_rslt = []
    unsat_vol = []
    operations = []
    bus_demand = []
    best_objective = []
    user_cst = iter(user)
    b_supply = iter(ntwk_vol)
    op_cst = iter(operations)
    b_demand = iter(bus_demand)
    obj2 = iter(best_objective)
    unsat_cst = iter(unsat_vol)

    for cs in candidates:
        cs1 = deepcopy(cs)
```

```
    cs3 = [l for l in cs1 if type(l) == list]
    cs2 = [map(lambda x: float(x), l) for l in cs3]
    bus_route_nodes = [map(lambda x: str(float(x)), l) for l in
       cs3]


# performs an All-or-Nothing assignment
    def main():
        """
            This functions performs an All-or-Nothing assignment
            on the network
        """
         graph = 'graf16.csv'
         matrix = 'matrix_16.csv'
         dire_data = os.getcwd()
         dire = os.getcwd()

        # Load the data
         graph = np.loadtxt(graph, delimiter=',', skiprows=1)
         j = np.loadtxt(matrix, delimiter=',', skiprows=1)

        # Characteristics of the graph
         zones = int(np.max(j[:,0:2]) + 1)
         links = int(np.max(graph[:,0]) + 1)

# Builds the matrix on a sparse format because the matrix is too big
and too sparse
         matrix = csr_matrix((j[:,2], (j[:,0].astype(int),
j[:,1].astype(int))), shape=(zones, zones), dtype="float64")

        #Prepares arrays for assignment
         L = np.zeros(links, dtype="float64")
         demand = np.zeros(zones, dtype="float64")

         for z in range(int(zones)):  # We assign all zones
```

```
                if matrix.indptr[z + 1] > matrix.indptr[z]:
demand.fill(0)


demand[matrix.indices[matrix.indptr[z]:matrix.indptr[z + 1]]] =
matrix.data[matrix.indptr[z]:matrix.indptr[z + 1]]  # Fill it in
with the new information

                    zones, loads = AoN.AllOrNothing(graph, demand,
z)

                    L = L + loads

            w = open('RESULT16.CSV', 'w')

            print >> w, 'Link,NodeA,NodeB,LOAD'

            for lin in range(links - 1):

                if L[lin] > 0:

                    print >> w, lin, ',', graph[lin, 1], ',',
graph[lin, 2], ',', L[lin]

            w.flush()

            w.close()

            #print "TRAFFIC ASSIGNMENT DONE"

            #print


        if __name__ == '__main__':

            main()
```

 # Assign link volumes from the result of the All-or-Nothing assignment

```
        with open('RESULT16.csv') as ff:

            for row in csv.DictReader(ff):

                for nodeA, nodeB, attrDict in bus_route_edgelist1:

                    if float(row['NodeA'].strip()) == float(nodeA)
and float(row['NodeB'].strip()) == float(nodeB):

                        attrDict.update({'volume':
float(row['LOAD'].strip())})

                    elif float(row['NodeA'].strip()) == float(nodeB)
and float(row['NodeB'].strip()) == float(nodeA):

                        attrDict.update({'volume':
float(row['LOAD'].strip())})
```

```
                for nodeA, nodeB, attrDict in sht1:
                    if float(row['NodeA'].strip()) == float(nodeA)
and float(row['NodeB'].strip()) == float(nodeB):
                        attrDict.update({'volume':
float(row['LOAD'].strip())})
                    elif float(row['NodeA'].strip()) == float(nodeB)
and float(row['NodeB'].strip()) == float(nodeA):
                        attrDict.update({'volume':
float(row['LOAD'].strip())})
        ff.close()
```

 # Implement the Breadth First Search Algorithm (BFSA)

```
        sht1, sht2, sht3 = [], [], []
        for c in cs3:
            tem = []
            for bfs, p in enumerate(c):
                sht = nx.shortest_path_length(X, p)
                for k, v in sht.iteritems():
                    if v == 1:
                        tem.append((p, k, {'length': v}))
                        sht1.append((p, k, {'length': v}))
            sht3.append(tem)
```

 # Sum demand link volumes from the result of the All-or-Nothing assignment

```
        demand_vol = sum([sh[2]['volume'] for sh in sht1 if 'volume'
in sh[2]])
        bus_demand.append(demand_vol)
```

 # Sum Feeder link volumes from the result of the All-or-Nothing assignment

```
        vv = []
        for bsrt in bus_route_edgelist1:
            if 'volume' in bsrt[2]:
                vv.append(bsrt[2]['volume'])
```

 # Create a tuple of bus route edges that will be used to get volumes in the next step

```
        global spd, temp_rou_len, traveltime,bus_route_dict
        br_links = []
        br_links1 = []
        spl = iter(shortest_path_length)
        bus_route_dict = {}

        for i,brn in enumerate(bus_route_nodes):

            temp_rou_len = spl.next()
            headway = rand.randint(2, 30)
            spd = rand.choice(speeds)
            traveltime = temp_rou_len / spd
            trv_time = round(traveltime, 3)
            vehicle_type = rand.choice(vehicle)
            br_links.append(zip(brn[0:], brn[1:]))
            route_name = str(brn[0]) + ' ' + '-' + ' ' + str(brn[-
1])
            bus_route_dict[i] = {'Name':route_name,
'Headway':headway, 'Length':temp_rou_len, 'speed': spd,
'traveltime':trv_time, 'vehicle_type':vehicle_type}  #
{'links':zip(brn[0:], brn[1:]), 'Route Number': route_number}

        for i,brtn in enumerate(bus_route_nodes):
            br_n = br_links[i] + board1[i] + alight1[i]
            br_links1.append(br_n)
```

 # Get the sum of the bus link volumes by looping through the list of bus route links(bus_route_edgelist)

```
        vol = []
        vol_sum = []
        for i, elem in enumerate(br_links1):
            try:
                vol.append([])
```

```
                for sub_e in elem:
                    for l in bus_route_edgelist1:
                        if l[0:2] == sub_e:
                            l[2]['speed'] =
bus_route_dict[i]['speed']
                            l[2]['traveltime'] =
bus_route_dict[i]['traveltime']
                        if l[0:2] == sub_e and 'volume' in l[2]:
                            vol[i].append(l[2]['volume'])


                vol_sum.append(sum(vol[i]))
            except: Exception
        unsat = demand_vol - sum(vol_sum)
```

 # Assign and Print the operating characteristics of the bus routes eg speed, vehicle type, headway, path, route number.

```
        hdw_list, spid_list, route_objective = [],[],[]
        trnsfr_pen = 25
        vs = iter(vol_sum)
        hds = iter(hdw_list)
        wait_time = 5
        spid = iter(spid_list)
        spl = iter(shortest_path_length)

        for brn in cs:
            try:
                route_len = spl.next()
                route_vol = vs.next()
                headway = rand.randint(2, 30)
                hdw_list.append(headway)
                spd = rand.choice(speeds)
                spid_list.append(spd)
                vehicle_type = rand.choice(vehicle)
                headways = hds.next()
```

```
                spiid = spid.next()

#Compute Objective function values for the individual routes using:

                route_user_cost = (weight_usercost * route_vol *
((route_len / spiid) + wait_time + trnsfr_pen) * value_of_time)

                route_opcost = (weight_opcost * (60 / headways) *
(route_len / spiid) * value_of_time)


                route_unsat_cost = (weight_unsat * unsat)


                route_obj = route_user_cost + route_opcost +
route_unsat_cost

                route_objective.append(route_obj)


            except:

                Exception

        avr_route_obj = (sum(route_objective) /
len(route_objective))

        hds = iter(hdw_list)

        spid = iter(spid_list)


# Compute aggregate network characteristics gotten by taken the mean
of individual characteristics that make up each network

        network_headway = sum(hdw_list) / len(hdw_list)

        network_length = sum(shortest_path_length)

        network_volume = sum(vol_sum)

        network_speeds = sum(spid_list) / len(spid_list)


#Compute Objective function values for the network using:

        user_cost = (weight_usercost * network_volume *
(((network_length / network_speeds) + wait_time + trnsfr_pen)

                    * value_of_time))


        op_cost = (weight_opcost * ((60 / network_headway) *
(network_length / network_speeds) * opcost * network_length
```

```
                    + ((6505 / number_of_routes) * (network_length /
network_speeds) * (60 / network_headway))))


        unsat_cost = (weight_unsat * unsat * value_of_time)


        obj = user_cost + op_cost + unsat_cost
        obj_rslt.append(obj)
        user.append(user_cost)
        operations.append(op_cost)
        unsat_vol.append(unsat_cost)
```

 # Compare the network objective function values and get the best

```
        best_obj = route_objective[:]
        best_obj.append(obj)
        best_obj.sort()
        best_obj_value = best_obj[-1]
        best_objective.append(best_obj_value)
        ntwk_vol.append(network_volume)
    avr_obj.append(sum(best_objective))
    avr_total_obj = sum(avr_obj) / len(avr_obj)
    sum_obj.append(avr_total_obj)
    for i,cs in enumerate(candidates):
```

 # Compute the fitness values by transforming the objective function values

```
        print "Network", i + 1
        obj_i = obj2.next()
        temp6 = op_cst.next()
        temp7 = user_cst.next()
        temp3 = b_supply.next()
        temp5 = b_demand.next()
        temp4 = unsat_cst.next()
```

```
        fit = 1000 - ((obj_i * popul_size) / sum_obj[0])
        fitness.append(fit)
        print 'Relative fitness Value:', fit
        print
    return fitness
```

D.3 '''BUS ROUTE NETWORK SEARCH ALGORITHM (BRNSA)'''

```
def route_mutator(random, candidates, args):


    """
           This function performs a mutation operation on the
           generated offspring. Developed by Obiora .A. Nnene (2014)
    """


    children = inspyred.ec.variators.n_point_crossover(random,
candidates, args)
    mutants = []


    for ind, child in enumerate(children):


        if random.random() > mut_rate:
            pass
        else:
            c = []
            shortest_path_lengths = []
            while True:
                try:
                    A = int(rand.choice(pos.keys()))
                    B = int(rand.choice(pos.keys()))


                    if A == B: # or A in rail_pos or B in rail_pos:
                        break


                    else:
                        path = nx.dijkstra_path(X, A, B,
weight='length')
```

```
                        pathlength =
round(nx.dijkstra_path_length(X, A, B, weight='length'), 2)
                    if min_len <= pathlength <= max_len:
                        children[ind][ind] = path
                        shortest_path_lengths.append(pathlength)
                        break
                    else:pass
                except:pass

    return children
```

Genetic Algorithm

```
import time
rand.seed(int(time.time()))
es = ec.ES(rand)
es.observer = [inspyred.ec.observers.stats_observer,
inspyred.ec.observers.file_observer]
es.selector = inspyred.ec.selectors.default_selection
es.replacer = inspyred.ec.replacers.plus_replacement
es.variator = route_mutator
es.terminator = inspyred.ec.terminators.generation_termination

my_seed = int(time.time())
seedfile = open('randomseed2.txt', 'w')
seedfile.write('{0}'.format(my_seed))
seedfile.close()
prng = rand.Random()
prng.seed(my_seed)

import logging
logger = logging.getLogger('inspyred.ec')
logger.setLevel(logging.DEBUG)
file_handler = logging.FileHandler('inspyred.log', mode='w')
```

```
file_handler.setLevel(logging.DEBUG)
formatter = logging.Formatter('%(asctime)s - %(name)s -
%(levelname)s - %(message)s')
file_handler.setFormatter(formatter)
logger.addHandler(file_handler)
final_pop = es.evolve(generator=generate_routes,
                      evaluator=evaluate_routes,
                      pop_size=60,
                      maximize=False,
                      max_generations=100,
                      crossover_rate=0.70,
                      num_crossover_points=2,
                      mutation_rate=0.15)

# Sort and print the best individual, who will be at index 0.
final_pop.sort(reverse=True)
current_time = datetime.datetime.now().time()
print "End time :", current_time.isoformat()
print(final_pop[0])
best_indiv = final_pop[0]
f = open('bestindividual.txt', 'w+')
f.write(str(final_pop[0]))
f.close()
current_time = datetime.datetime.now().time()
```