Optimizing Gateway Placement in Wireless Mesh Network using Genetic Algorithm and Simulated Annealing

أمثلة مواقع البوابات في الشبكة اللاسلكية المعشقة باستخدام الخوارزمية الجينية ومحاكاة التلدين

BY

AWADALLAH MOHAMMED AHMED ALI

A dissertation submitted in Partial fulfilment of the requirements for the degree of Doctor of Philosophy

in

(Computer Science)

Supervisor

Professor: Aisha Hassan Abdallah Hashim

College of Computer Science and Information Technology

Sudan University of Science & Technology

JANUARY 2016

ABSTRACT

Recently, Wireless Mesh Network (WMN) has gained important roles in current communication technologies. It has been used in several applications and most of them are critical applications such as surveillance, transportation systems and rescue systems. Hence, the WMN attracts a lot of attention from many researchers. WMN consists mainly of mesh clients MCs and mesh routers MRs, some of the latter are supplied by additional functions to serve as Internet gateways (IGs). Thus, most of the network traffic is acting toward IGs. Therefore, the network performance largely depends on the MRs' placement, especially the IGs. There are many research efforts on solving the gateway placement problem (GPP) and it has been proven to be NP-Complete by many researchers. Thus, finding the optimal solution is difficult. Therefore, finding near optimal solution is crucial to improve the network performance. This research proposes a novel approach to solve this problem using Genetic Algorithm (GA) and Simulated Annealing (SA) to achieve a near optimal solution guided by a mathematical model, considering the number of IGs and the number of hops that a packet traverses between the IG and the source / destination MR (MR-IG). The main objective of the proposed approach is to minimize the variation of MR-IG-hop counts (VAR-MR-IG-Hop) among MRs to insure that the IGs are placed in the appropriate positions. Finally, the proposed set of algorithms is evaluated using many generated instances using different parameters (population size, tournament size, crossover type, mutation type) for GA and many parameters for the SA such as the internal temperature, the final temperature and the parameters that have been used to change the internal temperature and in the transition function. Furthermore, a comparison between the two algorithms have been done. The experimental results for GA have shown high convergence rate. Moreover, the algorithm has considerable scalability and robustness to solve the GPP in large and small networks. In addition, SA has shown high convergence rate and fast execution time in comparison with the GA. However, GA has better performance in the small-size network with high scalability opportunities while SA is faster than GA in the large-size network but it has limited chances for further optimization.

مستخلص البحث

في هذه الآونة لعبت الشبكات اللاسلكية المعشقة دوراً هاماً في مجال تكنولوجيا الإتصالات. فقد تم إستخدامها في العديد من التطبيقات الحساسة والمهمة مثل أنظمة المراقبة والمواصلات وأنظمة الأنقاذ. ولهذا فقد جذبت إنتباه العديد من الباحثين. تتكون الشبكة اللاسلكية المعشقة بشكل رئيسي من نوعين من الأجهزة، موجهات وأجهزة عملاء. بعض هذه الموجهات مزودة بمزايا إضافية تجعلها تعمل كبوابات للشبكة العنكبوتية، وبالتالي فإن معظم البيانات يتم توجيهها نحو هذه البوابات، ويعتبر تحديد مواقع البوابات في الشبكة من أعظم المشاكل التي تؤثر في أداء الشبكة. هنالك العديد من البحوث التي قدمت لتعالج مشكلة تحديد مواقع البوابات والمشاكل المتعلقة بها. وقد تم إثبات أن هذه المشكلة من النوع (كثيرة حدود غير قطعية كاملة) وبالتالي من الصعب الحصول على الحل الأمثل. ولذا كان لابد من إيجاد طرق للحصول على حل قريب من الحل الأمثل. هذا البحث يقدم حلاً جديداً للحصول على الحل القريب من الأمثل بإستخدام الخوارزمية الجينية وخوارزمية محاكاة التلدين بإستخدام نموذجاً رياضياً يهتم بعدد النقاط التي تعبرها حزمة البيانات بين بوابات الشبكة العنكبوتية والموجهات الأخرى. يهدف هذا النموذج لتحديد المواقع المثلى للبوابات بهدف الحصول على أداء أفضل وذلك بتقليل التباين بين الموجهات من حيث عدد النقاط التي تعبرها حزمة البيانات من أي موجه إلى أقرب بوابة وقد تم تقييم الخوارزميات المقترحة بإستخدام حالات يتم توليدها بإستخدام العديد من العوامل لكل خوارزمية كحجم المجتمع، طرق تبادل الجينات، التغيير الإحيائي وغيرها من عوامل الخوارزمية الجينية ودرجة الحرارة الداخلية والنهائية وغيرها من عوامل خوارزمية محاكاة التلدين. وقد أظهرت نتائج التقييم على النموذج المقترح أن الخوازمية الجينية ذات إمكانيات جيدة ومتانة ولها القدرة على الحصول على حلول جيدة في شبكات بأحجام مختلفة ولكنها تستغرق زمناً أطول مقارنة بخوارزمية محاكاة التلدين بينما الآخيرة يمكنها الوقوع في الحل القريب والذي قد يبعد الخوارزمية عن الحل الأمثل بينما أظهرت الخوارزمية الجينية بفضل معاملاتها إبتعادها عن هذا الحل والتقدم دائماً نحو الحل الأمثل.

ACKNOWLEDGEMENTS

In the Name of Allah, the Most Gracious, the Most Merciful.

First and foremost, I thank my God (Allah S.W.T) who supplies all my needs. Thank you for the inspiration and perseverance. Thank you, for the incomparable gift of salvation and the eternal hope you offers all who would place their trust in you.

I would like to thank my supervisor, Professor Aisha Hassan, for sharing her knowledge, guidance, patience and support throughout this dissertation.

I truthfully present my appreciation to my beloved brother Mohammed Albarra Hassan for sharing his knowledge on the Genetic algorithm and Optimization with me. I wish to express my gratitude to my brother Mohammed Mahmoud Mantai for supporting me in English language.

Lastly, I would like to thank my parents and my wife for their continued support of my education.

I would like to acknowledge the financial support rendered by the University of Gezira and the Faculty of Mathematical and Computer Sciences.

DECLARATION

I hereby declare that this dissertation is the result of my own investigation, except where otherwise stated. I also declare that it has not been previously or concurrently submitted as a whole for any other degrees at Sudan University of Science and Technology or other institutions.

Awadallah Mohammed Ahmed Ali

Signature_____ Date_____

TABLE OF CONTENTS

LIST OF TABLES

Table No. Page No.

LIST OF FIGURES

Figure No. Page No.

LIST OF ABBREVIATIONS

ACO	Ant Colony Optimization
ADSL	Asynchronous Digital Subscriber Line
AVG-MR-IG-Hop	Average Hop counts between MRs and their nearest IG
BACnet	Building Automation and Control Networks
BWMN	Backbone Wireless Mesh Network
EV	Evolutionary Algorithm
GA	Genetic Algorithm
GPP	Gateway Placement Problem
HCA	Hill Climbing Algorithm
IEEE	Institute of Electrical and Electronics Engineers
IG	Internet Gateway
ILP	Integer Linear Programming
LP	Linear Programming
MAC	Medium Access Control (MAC)
MANET	Mobile Ad hoc Network
MC	Mesh Client
MR	Mesh Router
PC	Personal Computer
PDA	Personal Digital Assistant
PSO	Particle Swarm Optimization
QoS	Quality of Services
RFID	Radio Frequency Identification
SA	Simulated Annealing
TSP	Traveling Salesman Problem
VAR-MR-IG-Hop	Variation of Hop counts between MRs and their nearest IG (Variance)
WMN	Wireless Mesh Network
WN	Wireless Network
WR	Wireless Router

CHAPTER ONE
INTRODUCTION

1.1 OVERVIEW

Obviously, accessing the information has become very important in our life. The Internet is the most important source of information, the users of the internet growing dramatically. Therefore, the demand for satiable network with high performance and low risk of data loss is also increasing and users need to access their data over the internet wherever they live. Thus, the internet access technologies for last mile become very important. In this context, Wireless Mesh Network (WMN) is a useful communication technology, as internet access to serve users at the last mile, especially in rural areas, in the areas where many obstacles exist and also the construction cost is an important factor due to the use of wireless communication, which provides the internet access and other services in affordable cost.

1.2 BACKGROUND

WMN is a communication technology is used as an internet access to the end users and in numerous applications such as neighbourhood networking, surveillance, emergency and rescue systems. Most of the applications that the WMN supports are very critical and sensitive to packet loss and delay, so that the stability is very important. WMN mainly consists of mesh routers (MRs) and mesh clients (MCs), some of the formers have additional functions and features and have external interfaces to connect the internal network with the internet called internet gateways (IGs). Nowadays, the need to access the data over the internet is increasing rapidly [1, 2]. Thus, that most of the network traffic in a network either between the MCs via MRs inside the network or to/from the internet via the IGs and this may create bottleneck points at the IGs due to the huge amount of packets. Therefore, the locations of the IGs are very important to achieve high throughput, minimizing delay and minimizing transmission time as well. If the IGs placed, too far from the MRs this will increase the transmission time by increasing the number of hops that the packets traverse from the source to the destination, which will result in delay. Thus, packet loss may happen, and if they placed

close to MRs (by increasing the number of IGs), the transmission time decreases, but the network cost will increase due to the high cost of the IG installation because of using physical links to be connected to the internet. Therefore, IG placement optimization is essential in WMNs planning and design, especially at the earlier stages of network design, which usually based on topology considerations to minimize overheads of using sophisticated protocols that will be used in the future to overcome the problem of IG placement in high levels of network planning and configuration. Therefore, the network performance depends largely on the optimal placement of MRs especially in the IGs. Many researchers paid their attention and efforts to the WMNs, and considerable research works have been achieved to solve the Gateway Placement Problem (GPP) using different methods and purposes. In addition, many research works dealt with GPP as an optimization problem [3]. Since GPP is considered as NP-Complete problem and computationally can be modelled as a combinatorial optimization problem [4]. Thus, finding the optimal solution is difficult. Hence, some sort of heuristic and metaheuristic methods are required to find the near optimal solution. Recently, Evolutionary Algorithms (EAs) such as Genetic Algorithms and Simulated Annealing (SA) were widely used to solve the optimization problems. In this context, GAs recently have proved their usefulness and efficiency to solve optimization problems especially the combinatorial optimization problems in a reasonable time [1]. This research studies the GPP as it proposes a novel approach to find a near optimal solution for the GPP in WMN.

1.3 PROBLEM STATEMENT AND ITS SIGNIFICANCE

Currently, most of the network traffic move toward the IGs. Therefore, the IGs are potential bottleneck points in the network, which may cause low network performance by reducing the overall network throughput and high bandwidth consumption. Thus, optimizing locations of the IGs is crucial in network planning for high performance [5] [6]. If the IGs are located too far from the sources/destination MRs, then the number of hops that the packet traverses would increase, which cause many problems such as packet loss, delay. Hence, increasing the number of IGs will minimize the number of hop counts, but this will result in higher construction cost due to the cost of physical links that used to connect the IGs to the internet and also may generate interference if the IGs are close to each other [7]. High throughput is required, but the throughput

degrades rapidly with a WMN system as the path length increases [8]. Therefore, if the IGs placed in unsuitable positions, then the MRs those are far away from their IGs will achieve low throughput, while other MRs those are close the IGs achieve high throughput. Therefore, some IGs are overwhelmed with packet traffic, which may cause bottleneck points in the network, whereas some of them with less traffic. Since the Throughput is one of the most important parameters that affect the quality of service of WMN [9]. Hence, optimizing the IGs positions is very important in WMN planning to improve the network throughput. Thus, the number hops between MRs and the nearest IGs should be minimized as much as possible. In this context, one must keep in mind the variations in the number of hops that packet traverses from each MRs to reach the nearest IGs (MR-IG-Hop) and vice versa should be considered early in the planning stage (network topology). Thus, minimizing the variation in MR-IG-Hop is critical for load balancing network. In addition, minimizing the load differences among the IGs also is another issue to guarantee the IGs are distributed in a near optimal position. To minimize the affects the above issues a new approach for IG placement in planning stages (topology) is required.

1.4 RESEARCH OBJECTIVES

Considering the issues, which were stated in the problem definition, this research would be conducted to achieve the following objectives:

- The main goal of this research is to achieve load balancing and enhance the overall network throughput.
    - MRs and IGs locations Optimization.
    - Reduction of the construction cost by optimizing the locations of the IGs rather than increasing the number of IGs based on the user requirements and affordable cost.
- The Detailed objectives are:
    - Development of the IG placement solution, to optimize the resource utilization by computing the MRs and IGs locations.
    - Improvement of the network performance by optimizing the IGs positions according to the positions of MRs, Minimizing the differences in MR-IG-Hop count among MRs.

    - Optimizing the number of MRs associated with each IG, to achieve better load balancing.
- The research outcomes as yelled from the achievement of the above objectives are:
    - A new solution for solving the GPP.
    - A new application (software), which may be used by other researchers in WMN planning.
    - New publications in the research area to increase the knowledge database.
    - A new PhD dissertation.

1.5 RESEARCH QUESTIONS

This research will be conducted to answer the following questions in order to realize the expected objectives:

- What are the design concerns in the proposed solution?
- What is the appropriate method (s) that can be used to deliver the expected solution for the research problem?
- How the proposed solution will be evaluated?

1.6 RESEARCH HYPOTHESISES / PHILOSOPHY

The GPP in WMN is considered to be an NP-Complete problem and it can be modelled as combinatorial optimization problem. Hence, it is difficult to find the optimal solution or it is unlikely to be solvable in a reasonable amount of time [10] [4]. Therefore, methods to find the near optimal solution are needed in this situation and the metaheuristic methods are widely used as resolution methods [10]. There are many (Meta) heuristic methods such as Genetic Algorithm (GA), Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO) and Simulated Annealing (SA). GAs have shown their usefulness in resolution of many computational and combinatorial optimization problems [11]. In addition, to enhance the performance of the WMN, more focus should be given to the network topology rather than only focusing onto routing optimization techniques, which is not sufficient to achieve a good performance, and also to avoid the overhead generated by the routing protocol themselves as much as possible.

1.7 RESEARCH METHODOLOGY AND TOOLS

The following steps have guided the researcher to carry out this research:

1. Investigation of the current research works that dealt with the area of WMN issues and challenges.
2. Determination of the open issues that need to be addressed.
3. Identification and formulation of the research problem.
4. Development of the proposed solution's architecture to address the problem as follows:
    - Design of the proposed solution.
    - Determination of the appropriate methods and tools.
5. Implementation of the algorithms.
6. Evaluation the overall solution

Figure 1.1 illustrates the research framework to deliver the research objectives.

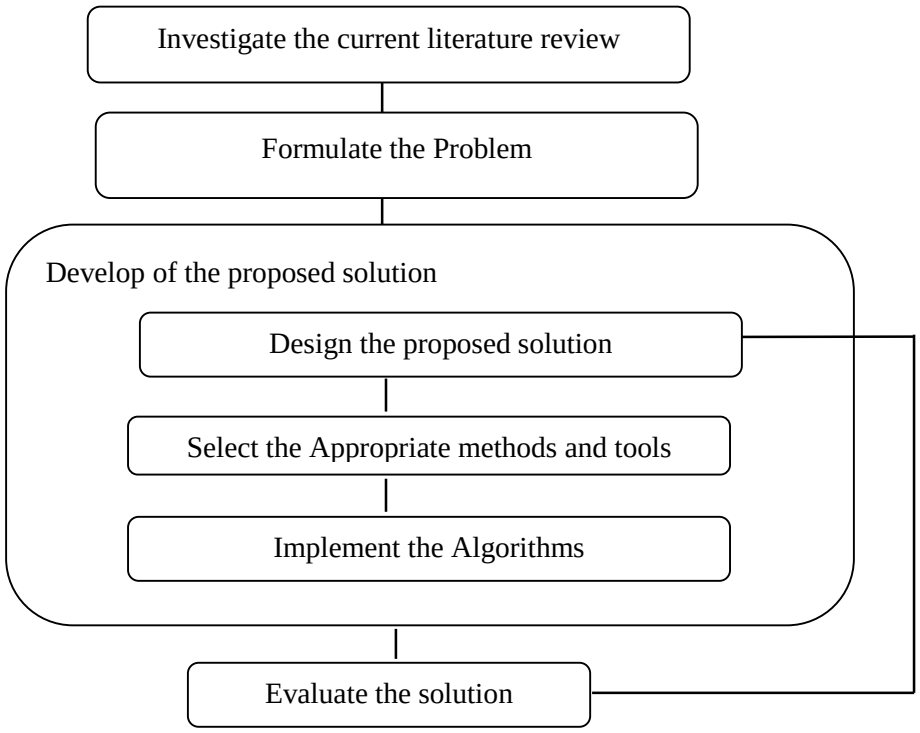

Figure 1.1: the research framework

1.8 DISSERTATION ORGANIZATION

The organization of this dissertation chapters as follows:

Chapter 2: presents the literature review that related to the research topic, gives more details about the research problem, and critically investigates the existing solutions, which were proposed to address the research problem.

Chapter 3: presents the research methodology and the details of the proposed solution.

Chapter 4: presents the implementation and the details of the proposed algorithms.

Chapter 5: presents the results, discussion about the results.

Chapter 6: concludes the dissertation and presents the potential future works.

CHAPTER TWO
LITERATURE REVIEW

2.1 BACKGROUND

Wireless Mesh Network (WMN) is a promising technology for the next generations. However, there are some critical issues related to WMN that need to be addressed for good network sustainability in market [12]. For example, how we can achieve perfect planning in WMNs in order to enhance the network connectivity and coverage, to avoid the bottleneck in the IGs, to avoid large number hops count [13]. What are the optimal number of MRs, IGs, and their optimal locations to increase the network throughput in MRs, IGs and overall network performance [14]. This chapter introduces the basic WMN concepts, its architecture, challenges and applications. In addition, this chapter presents the optimization concepts, especially the combinatorial optimization, optimization problems, the methods that are used to solve these problems besides the basic concepts of the graph theory that are used to solve WMN and the algorithms, which were used with the graph's applications and network problem solutions. Finally concludes with the previous research efforts in WMN planning, especially GPP approaches.

2.2 MESH TOPOLOGY

A network where every node is connected to other nodes on the network through multiple or single hops and some may be connected with more than one hop. In a mesh topology, every node not only sends its own signals but also relays data from other nodes. This type of topology is very expensive as there are many redundant connections, thus it is not mostly used in computer networks. It is commonly used in wireless networks. Flooding or routing technique is used in mesh topology.

Figure 1.1 shows a sample of full mesh topology.

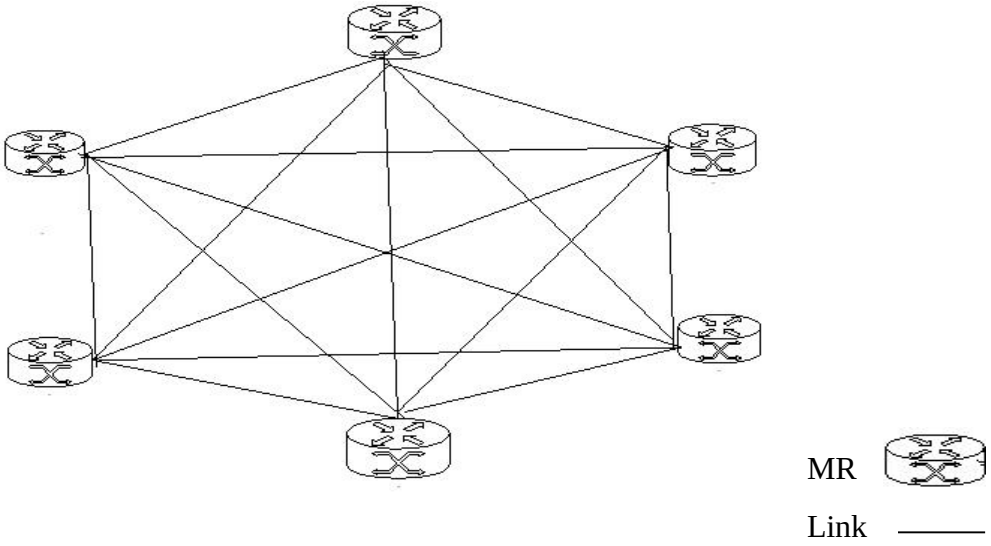

Figure 2.1: Full-mesh topology

2.3 Wireless Mesh Network

WMN is a communication network made up of radio nodes organized in a mesh topology. WMN consists of MRs and MCs as mentioned early in this dissertation. Other than the routing capability for gateway or repeater functions as in a conventional wireless router (WR), MR contains additional routing functions to support mesh networking. For further improve and flexibility of mesh networking, an MR is usually equipped with multiple wireless interfaces, which built on either the same or different wireless access technologies. Compared with a conventional WR, an MR can achieve the same coverage with much lower transmission power through multi-hop communications. In spite of all these differences, the conventional WRs are usually built on a similar hardware platform where MRs can be built on dedicated computer systems (e.g., embedded systems) and look compact. The MRs in the WMN in contrast with the other WN are self-healing and self-organized, self-configured, easily maintainable, highly scalable and reliable service with the nodes in the network, because if a single node goes down, other nodes are available [15]. They also can be built based on general-purpose computer systems (e.g., laptop or PC) [16]. MCs also have the necessary functions for mesh networking, so that may also work as a router in a special type of WMN as we will see later in this chapter. However, IG or bridge (when MR connect different networks with different technologies) functions do not exist in these nodes. In addition, MC usually has only one wireless interface. Consequently, the

hardware platform and the software for MCs can be much simpler than those for MRs can. MCs have a higher variety of device types compared to MRs such as a laptop, desktop PC, pocket PC, Personal Digital Assistant (PDA), IP phone, Radio Frequency Identification (RFID) reader, Building Automation and Control Networks (BACnet) controller, and many other devices [16]. Figure 2.2 shows a demonstration of the basic architecture of WMN.

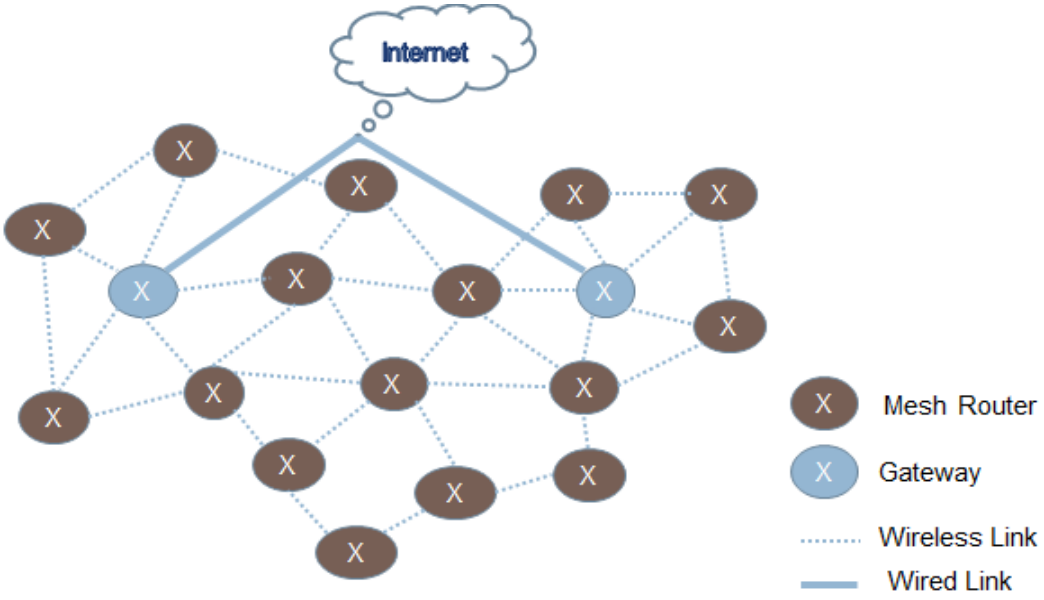

Figure 2.2: Basic WMN Infrastructure.

2.4 WIRELESS MESH NETWORK ARCHITECTURE

WMNs are classified into three different categories based on the network topology: flat WMN, hierarchical WMN, and hybrid. The following three subsections present a brief discussion of these categories [8]. Figure 2.3 shows a sample of the infrastructure/ backbone WMN structure/ backbone WMN [17].

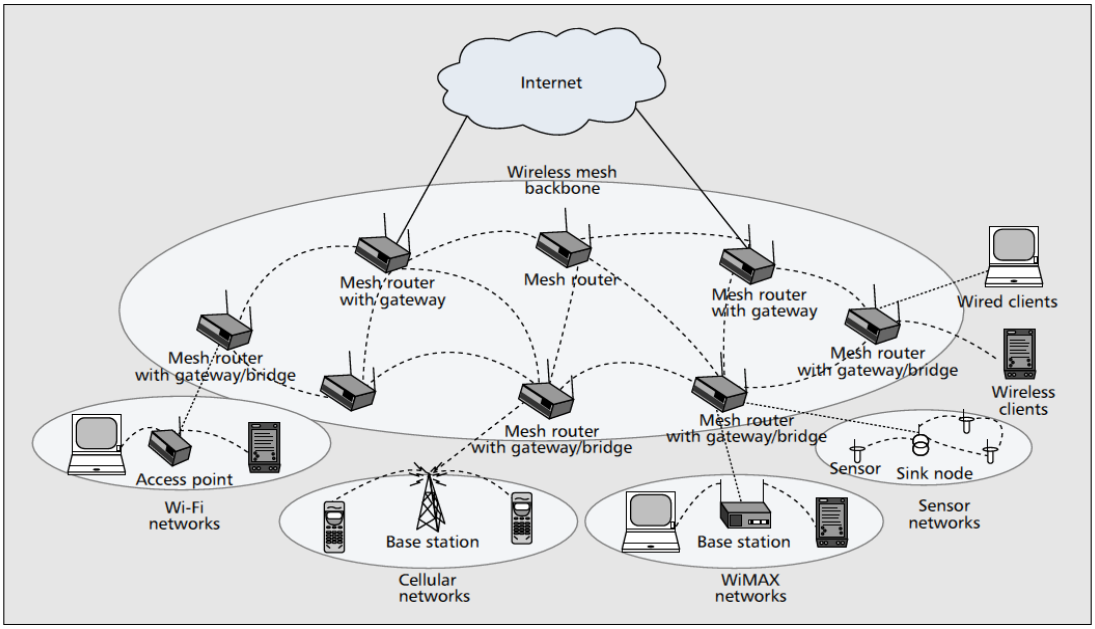

Figure 2.3: infrastructure backbone WMN [8]

2.4.1 Flat Wireless Mesh Network

Here the network consists of MCs that act as both hosts and routers. Thus, the MCs are responsible of routing, network configuration, service provisioning, and other application provisioning, packets forwarding, sending and receiving packets among themselves. Thus, the network architecture is similar to an ad hoc WN [8].

2.4.2 Hierarchical Wireless Mesh Network

In this type, the network consists of multiple levels. The upper levels formed by the MRs, which form the network backbone so, that MRs may not originate or terminate data traffic, but they are responsible of packet forwarding. Thus, in this type of network, the MRs are self-organize and self-healing to provide sustainable network backbone. In addition, as in the other categories, some MRs are connected to the internet acting as IGs to the whole network or a subset of the network. MCs are on the lowest level and communicate via MRs so; MCs originate and terminate the data traffic inside the network [8].

2.4.3 Hybrid Wireless Mesh Network

The network here is a special case of hierarchical WMNs where the WMN utilizes other WNs for communication, such as Wi-Max network, cellular network, mobile ad hoc network (MANET) and a normal backboned WMN which is connected to the internet acting as a backhaul to other networks as shown in Figure 2.4. Here MR can act as IG when it has an interface to the internet, normal router responsible for forwarding the packets or a bridge when connect different networks with different technologies [8].

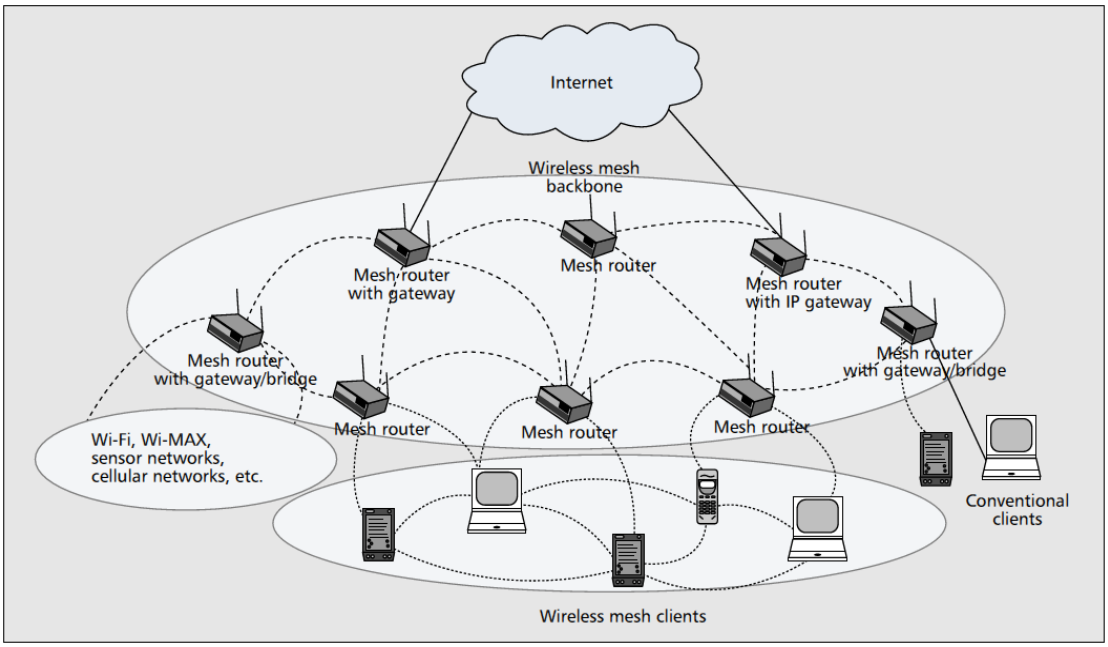

Figure 2.4: Hybrid WMN [17]

2.5 WIRELESS MESH NETWORK APPLICATIONS

WMN is effective alternative technology to offer low cost connectivity. Therefore, WMNs have recently supported numerous applications (Broadband Internet Access for last-mile as backbone, Emergency Networking, Community and Neighbourhood-Networking, Transportation System and Surveillance Systems) better than other types of WNs such as cellular networks, ad hoc networks, wireless sensor networks and standard IEEE 802.11x networks [18]. Hereby, examples of some application scenarios as following:

A. Neighbouring Community Networks

In a community, the usual solution is to deploy ADSL or cable. However, there are some limitations that WMNs can improve as shown in following [18].

- The cable technology delivers the services for houses and therefore many areas between houses will not be covered.
- A broadband IG between different houses cannot be shared and wireless services should be established individually.
- A single path to each neighbour can communicate with the rest of the neighbours or with the outside.

B. Corporative Networks

This scenario corresponds to having a small network in an office or a medium sized network for all offices of a building or even a network to communicate offices located in different buildings. Other similar scenarios include airports, hotels, shopping centres or sports centres.

C. Metropolitan Area Networks

Deploying WMNs in metropolitan areas has a number of advantages. The physical layer provides a higher average transmission to any cellular network and need not depend on a wiring. In addition, deploying such infrastructure is much cheaper than cable or fibre and can be easily and rapidly deployed in areas with few resources, which have never had any network before. Moreover, the following application scenarios can use WMN:

- Transportation Systems: it used to provide information services to passengers, remote monitoring of vehicle safety and communications with the driver and so on.
- Automatic Control Buildings: In buildings, there are several electrical devices to be controlled, including light, elevator, air conditioning, and so on.
- Surveillance: In corporate buildings, shopping malls and stores need broadband data transmission technology (images and videos) for monitoring and surveillance purposes.

2.6 INTERNET GATEWAYS (IGS) PLACEMENT ISSUES

Due to the complex structure of the WMN there are many conflicting requirements that can influence (or will be influenced by) IGs placement approaches. In this subsection, we discuss some of these issues as listed down [19]:

- **Congestion**: In WMNs, most traffic is forwarded through the IGs from the internal network to the internet or vice versa. This may cause some IGs overloaded and others received less traffic, so the IGs should be placed in such a way that making load balancing and no node is over congested.
- **Bandwidth**: While designing gateway placement algorithm one must consider that, the required bandwidth (the rate of data transfer in bits) of MCs should be satisfied.
- **Interference:** interference between gateways can highly influence the network performance. So the gateways should be placed in such a way that the throughput is maximum and interference among gateways is minimum.
- **Distance between nodes (location**): the interference will degrade the throughput when the gateways placed densely and if the distance is too much the signal strength will be affected. Therefore, the distance between gateways (gateway locations) should be optimized.
- **Transmission Delay:** Due to the indirect communication from source to destination in WN, the packet stored in each node traversing and then retransmitted. Thus, the transmission delay time is the storage time and the original transmission time. So if the packet traverse through a long path the transmission delay will increase and therefore the gateways should be placed in such a way that the transmission delay must be minimized by decreasing the number of hops between MRs and IGs (MR-IG-Hop) [19].
- **Cost:** the network performance (more gateways increase the network throughput) will be improved by increasing the number of IGs, but this will increase the network construction cost because the IGs usually use very expensive wired links. Therefore, the number of IGs is also another issue in WMN planning [19] [20].

- **Coverage:** when designing the network, we have to keep in mind that each MR should be covered by more than one gateway, so that if one of the IGs fails then MR may use the backup IG [19].

2.7 WIRELESS MESH NETWORK PLANNING

There are many techniques used in WMNs planning and in other WN types such as WiMAX, WLAN… etc. In contrast, WMN planning is much more complex due to many considerations during network planning stages as mentioned in this chapter and most of these considerations conflict with some others. Hence, there are many research efforts have been done to find the near optimal solution using optimization techniques, especially the metaheuristic methods.

2.8 GRAPH CONCEPTS AND IT'S APPLICATIONS

Graph is a pair of two sets (V, E), where V is the set of vertices, and E is the set of edges. Each element in set E has a multiplicity that is means the vertex can participate more than once [21] . Throughout this research the vertices will be labelled with letters and numbers (for instance, $v_1, v_2, \ldots$). The following list presents some terminologies can be used to describe the graph [21].

1. The two vertices u and v are end vertices of the edge (u, v).
2. Edges that have the same end vertices are parallel.
3. An edge of the form (v, v) is a loop.
4. A graph is simple if it has no parallel edges or loops.
5. A graph with no edges (i.e. E is empty) is empty.
6. A graph with no vertices (i.e. V and E are empty) is a null graph.
7. A graph with only one vertex is trivial.
8. Edges are adjacent if they share a common end vertex.
9. Two vertices u and v are adjacent if they are connected by an edge, in other words, (u, v) is an edge.
10. The degree of the vertex v, written as d (v), is the number of edges with v as an end vertex. By convention, we count a loop twice and parallel edges contribute separately.
11. A pendant vertex is a vertex whose degree is 1.

12. An edge that has a pendant vertex as an end vertex is a pendant edge.
13. An isolated vertex is a vertex whose degree is 0.

There are many graph types such as directed graphs, undirected graphs, weighted graphs, un-weighted graphs, multigraphs and simple graphs. The following subsections discuss some of these types.

2.8.1 Connected Graphs

The G=(V, E), A graph is said to be connected if for every pair of distinct vertices u, v there is a u-v path joining them, otherwise, G is unconnected graph [22]. Figure 2.5 shows a simple connected graph [21].

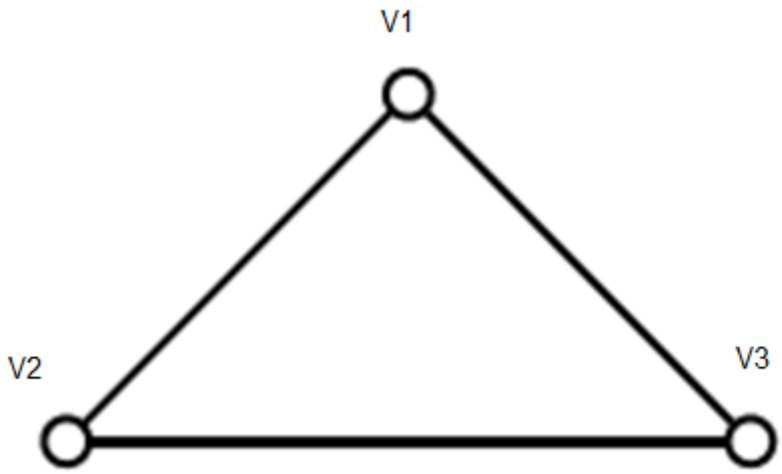

Figure 2.5: Simple connected graph

2.8.2 Directed Graphs

If the vertices connected by directed edges or arcs then the graph is said to be a directed graph or digraph. Figure 2.6 shows a simple directed graph.

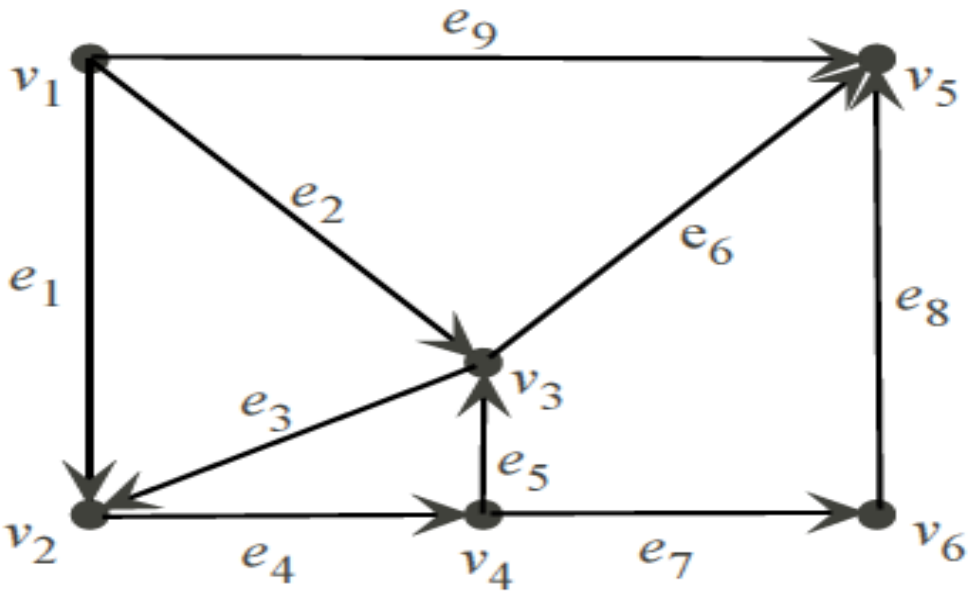

Figure 2.6: Directed Graph [21].

2.8.3 Undirected Graphs

If the direction of the edges is not considered or for u and v vertices in the graph G, the edges (u, v) and (v, u) will be considered as one and the same edge in G. In this case, the graph G is known as undirected graph [22]. Figure 2.7 shows a simple undirected graph.

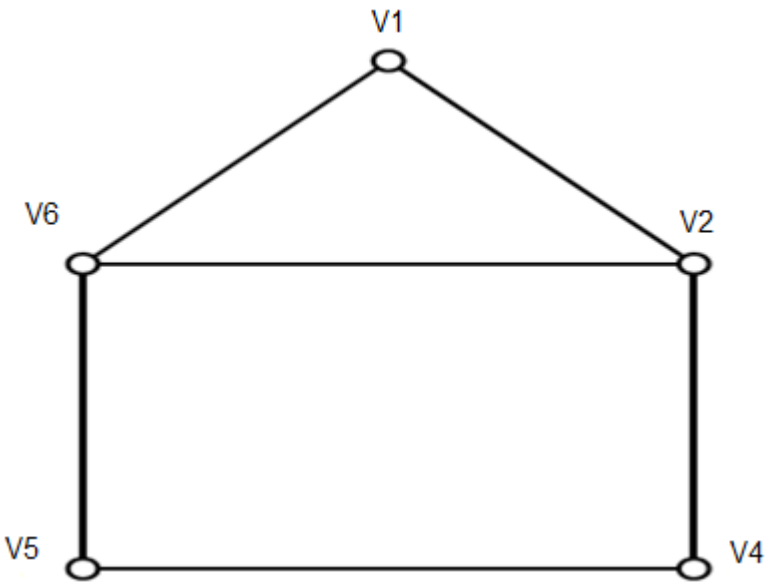

Figure 2.7: Simple undirected graph [22].

2.8.4 Weighted Graphs

Whatever the type of the graph is directed or undirected graph. The edges of the graph may represent real world problems. The vertices may represent cities' map. In this case, the edges represent the distance between the cities. Thus, the graph is a weighted graph otherwise the graph is un-weighted [23]. Figure 2.8 shows a weighted graph.

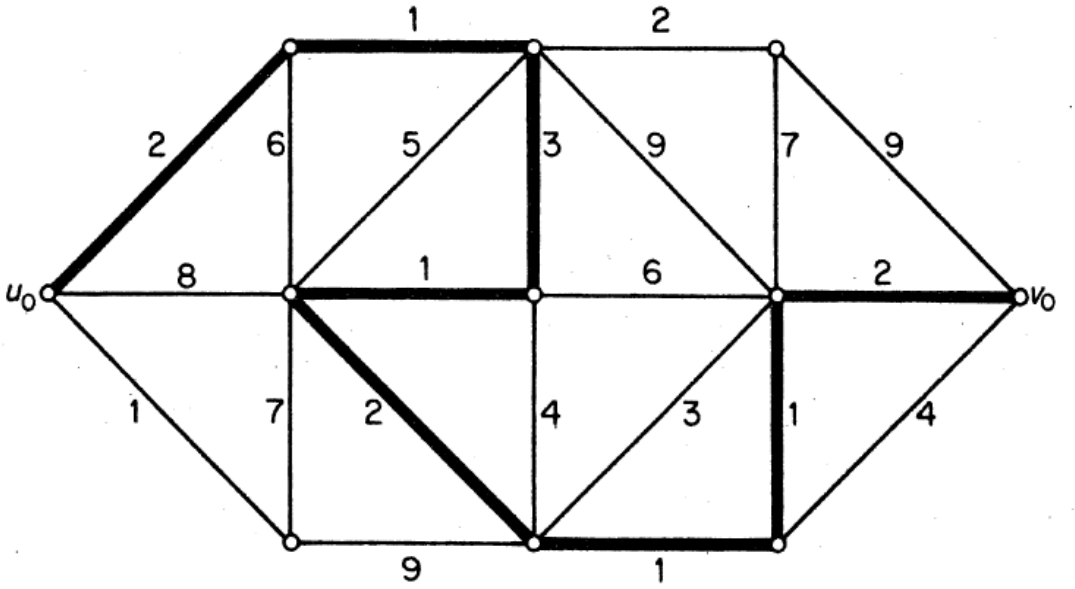

Figure 2.8: A sample of a weighted graph [23]

2.8.5 Paths: Distance and Metrics

If the edge-weighted simple graph G = (V; E; i; h) without negative weight cycles. Here $E \subseteq V^{(2)}, i: E \to V^{(2)}$ is an incidence function. And h: $E \to$ V is an orientation function and W: $E \to$ R is the weight function but if the graph G has not a weight function that means each edge has unit weight. If $v_1, v_2 \in V$ and $P = (e_1, e_2, \ldots, e_m)$ is a $v_1 - v_2$ path (so v_1 is incident to e_1 and v_2 is incident to e_m), the weight of P can be defined to be the sum of the weights of the edges in P [24]:

$$W(P) = \sum_{i=1}^{m} W(e_i)$$

The distance function $d: V \times V \to R\ U\ \{\infty\}\ on\ |G$ is defined by:

$$d(\mathrm{v}_1, \mathrm{v}_2) = \infty$$

If v_1 and v_2 lie in distint connected components of G, and by

$$d(v_1, v_2) = {}^{min}_{P} W(P)$$

If they are not lie in distinct connected component of the graph G or the case of taking the minimum over all paths P from v_1 to v_2. If the graph G has no negative weight cycles, so the minimum can be found as mentioned above. It follows by definition of the distance function that $d(v_1, v_2) = \infty$ if and only if there is no path between u and v [24].

2.8.6 Weight and Distance

A graph is said to be weighted if a numeric label or weight is assigned to each of its edges. Depending on the application, the vertices can represent physical locations and interpret the weight of an edge as the distance separating two adjacent vertices as mentioned before. There might be a cost involved in travelling from a vertex to one of its neighbours, in which case the weight assigned to the corresponding edge can represent such a cost. The concept of weighted digraphs can be similarly defined. When no explicit weights are assigned to the edges of an undirected graph or digraph, it is usually convenient to consider each edge as having a weight of one or unit weight [24].

Based on the concept of weighted graphs, this shows what means for a path to be a shortest path. Let G = (V; E) be a (di) graph with non-negative edge weights $W(e) \in R$ for each edge $e \in E$. The length or distance d(P) of a u-v path P from $u \in V$ to $v \in V$ is the sum of the edge weights for edges in P. Denote by d(u; v) the smallest value of d(P) for all paths P from u to v. When considering edge weights as physical distances, a u-v path that realizes d (u; v) is sometimes called a shortest path from u to v. The above definitions of distance and shortest path also apply to graphs with negative edge weights. If the weight of an edge is not explicitly given, then the edge will be considered to have a unit weight [24].

If the vertices u, v, and w in a graph G, the distance function d on G satisfies the following property [24].

Using the lemma "Path distance as metric function". Let G = (V; E) be a graph with weight function $W : E \to R$ Define a distance function $d : V \times V \to R$ given by

$$d(u,v) = \begin{cases} \infty & \text{if there are no paths from u to v} \\ \min\{ w(W) \mid W \text{ is a } u - v \text{ walk} & , \quad \text{otherwise} \end{cases}$$

Then d is a metric on V if it satisfies the following properties:

1. No negativity: $d(u,v) \geq 0$ with $d(u,v) = 0$ if and only if $u = v$.
2. Symmetry: $d(u,v) = d(v,u)$.
3. Triangle inequality $d(u,v) + d(v,w) \geq d(u,w)$.

The pair (V, d) is called a metric space, where the metric refers to the distance function d. Any graphs when assumed to have finite sets of vertices. For this reason, (V, d) is also known as a finite metric space. The distance matrix $D = [d(u_i, v_i)]$ of a connected graph is the distance matrix of its finite metric space [24].

2.8.7 Adjacency and incidence matrix

The adjacency matrix of a graph G = (V, E) is the V×V matrix A with [25] :

$$A_{u,v} := the\ number\ of\ edges\ connecting\ u\ and\ v\ \in V$$

The incidence matrix, or V×E incidence matrix, of G is the V×E matrix B with [25]:

$$B_{v,e} = \begin{cases} 1 & if\ v \in e\ and\ e\ is\ not\ a\ loop, \\ 2 & if\ v \in e\ and\ e\ is\ a\ loop, \\ 0 & if\ v \notin e. \end{cases}$$

For $v \in V$ and $e \in E$. The transpose of B is called the E ×V incidence matrix of G, or just the incidence matrix, if no confusion is expected [25].

2.8.8 The Shortest path Problem: Dijkstra's Algorithm

Dijkstra's algorithm [26], which is discovered by E. W. Dijkstra in 1959, is a graph search algorithm that solves the single-source shortest path problem for a graph with non-negative edge weights and the graph has no self-loops [22]. The algorithm is a

generalization of breadth- first search. Dijkstra's algorithm can be used to find a shortest route from a fixed city to any other city [22]. It has been used to solve the problem of finding the shortest path in the graph. The edges of a graph or digraph are given non-negative weights. The weight of a path is the sum of the weights of the path traversed [21]. The algorithm returns the shortest path from the source node to the destination node. The shortest path has an important impact on the routing protocol in the network. Is used to represent the hop count in the network when the network represented by a graph.

2.9 OPTIMIZATION

Is the processes of finding an alternative with the most cost effective or highest achievable performance under the given constraints, by maximizing desired factors and minimizing undesired ones. In comparison, maximization means trying to attain the highest or maximum result or outcome without regard to cost or expense. Practice of optimization is restricted by the lack of full information, and the lack of time to evaluate what information is available [27].

2.10 POLYNOMIAL-TIME SOLVABILITY

A polynomial-time algorithm is an algorithm that terminates after a number of steps bounded by a polynomial in the input size. Here a step consists of performing one instruction. Such an algorithm is also called a good algorithm or an efficient algorithm. Thus, the input size is the size of the input, that is, the number of bits that describe the input. We say that a problem is polynomial-time solvable, or is solvable in polynomial time, if it can be solved by a polynomial time algorithm. This definition may depend on the chosen algorithmic model, but it has turned out that for most models the set of problems solvable by a polynomial time algorithm is the same [25].

2.11 THE SETS P AND NP

P, NP, and co-NP are collections of decision problems: problems that can be answered by yes or no, like whether a given graph has a perfect matching or a Hamiltonian circuit [25].

The P is the set of all decision problems that can be solved in polynomial time algorithms. For example, the problem of determining whether a key is present in an array or in a sorted array. Some decision problems have not polynomial-time algorithms also not in P. For example, Traveling Salesman Problem (TSP) because no one has ever created a polynomial-time algorithm to solve TSP and no one has ever proven that it cannot solve with polynomial time algorithm. Therefore, there is a possibility for the TSP to be in P set. In addition, there some problem can or not be in the P set [28].

A polynomial-time nondeterministic algorithm: is a nondeterministic algorithm whose verification stage is polynomial-time algorithm.

NP is the set of all decision problems that can be solved by polynomial-time nondeterministic algorithms [28].

2.12 NP-COMPLETE PROBLEMS

A problem B is called NP-complete if both of the following are true about B [28]:

1. B is in NP set.
2. For every other problem A in NP $A \propto B$, which is means every other problem A in NP can be reduced to B.

2.13 MATHEMATICAL OPTIMIZATION

A mathematical optimization problem, or just optimization problem, has the form [29]

$$Minimize \ \ f_0(x) \qquad 2.1$$

$$Subject\ to \ \ f_i(x) \leq b_i \ , i = 1, \dots, m$$

Here the vector $\mathbf{x} = (\mathbf{x_1}, \dots, \mathbf{x_m})$ is the optimization variable of the problem, the function $f_0 : R^n \ \rightarrow R$ is the objective function, the functions $f_i : R^n \ \rightarrow R \quad i = 1, \dots, m$ are the (inequality) constraint functions, and the constraints $b_1, \dots, b_m$ are the limits, or bounds, for the constraints. A vector x^* is called optimal, or a solution of the problem (2.1), if it has the smallest objective value among all vectors that satisfy the constraints:

for any z with $f_1(z) \leq b_1, \ldots, f_m(z) \leq b_m$ we have $f_0(z) > f_0(x^*)$. We generally consider families or classes of optimization problems, characterized by particular forms of the objective and constraint functions. As an important example, the optimization problem (2.1) is called a LP if the objective and constraint functions $f_0, \ldots, f_m$ are linear [29], which will be discussed later in this chapter.

The optimization problem (2.1) is an abstraction of the problem of making the best possible choice of a vector in R^n from a set of candidate choices. The variable x represents the choice made; the constraints $f_i(x) \leq b_i$ represent firm requirements or specifications that limit the possible choices, and the objective value $f_0(x)$ represents the cost of choosing x. (a one can also think of $-f_0(x)$ as representing the value, or utility, of choosing x.) A solution of the optimization problem (2.1) corresponds to a choice that has a minimum cost (or maximum utility), among all choices that meet the firm requirements [29].

Many practical problems involving decision-making (or system design, analysis, and operation) can be cast in the form of a mathematical optimization problem, or some variation such as a multi-criterion optimization problem. Indeed, mathematical optimization has become an important tool in many areas. It is widely used in engineering, in electronic design automation, automatic control systems, and optimal design problems arising in civil, chemical, mechanical, and aerospace engineering. The Optimization is used for solving many problems in network design and operation, finance, supply chain management, scheduling, and many other areas. The list of applications is still steadily expanding [29].

The previous explanation shows an example of single objective optimization using single objective function. However, the optimization problems can be formulated with mutable objective functions. For instance, if there is a problem with two objective functions f1 and f2. There are two approaches that can be used in the solution to find the optimal solution, namely are hierarchical and simultaneous optimization. In the former, the objectives are classified (sorted) according to their priority. Thus, for the bi-objective case, one of the objectives, say f1, is considered as a primary objective and the other, say f2, as a secondary one [30]. The meaning is that the search method firstly try to optimize f1, and then when no further improvements are possible, it try to

optimize f2 without worsening the best value of f1 [30]. However, the hierarchical optimization is commonly used in the WMNs problem [30].

2.14 COMBINATORIAL OPTIMIZATION

Combinatorial optimization is one of the most active areas of discrete mathematics [31]. It has roots in combinatorics, operations research, and theoretical computer science. Combinatorial analysis is the mathematical study of the arrangement, grouping, ordering, or selection of discrete objects, usually finite in number. Traditionally, combinator lists have been concerned with questions of existence or of enumeration. That is, does a particular type of arrangement exist? On the other hand, how many such arrangements are there? [32].

In 1970s, a new line of combinatorial investigation has gained increasing importance. The question asked is not "Does the arrangement exists?" or "How many arrangements are there? ", but rather, "What is the *best* arrangement?" The existence of a particular type of arrangement is usually not in question, and the number of such possible arrangements is irrelevant. All that matters is finding an optimal arrangement, whether it be one in a hundred or one in an effectively infinite number of possibilities [32]. A large number of combinatorial optimization problems have been generated by research in computer design, the theory of computation, and by the application of computers to a myriad of numerical and non-numerical problems, which have required new methods, new approaches, and new mathematical insights [32].

Combinatorial optimization searches for an optimum object in a finite collection of objects. Typically, the collection has a concise representation such as a graph, while the number of objects is huge or more precisely, grows exponentially in the size of the representation like the problem of finding all matchings or finding all Hamiltonian circuits. [25]

2.14.1 Combinatorial Optimization Problems

Combinatorial optimization problems arise everywhere and certainly in all areas of technology and industrial management. A growing awareness of the importance of these problems has been accompanied by a combinatorial explosion in proposals for their

solution [32]. Thousands of real-life problems that can be modelled as abstract combinatorial optimization problems [33]. Most combinatorial optimization problems can be formulated naturally in terms of graphs (based on the graph theory) and as a linear programing (LP) or integer linear programing (ILP) [33].

Some representative optimization problems:

The problems were listed below involve graphs. These problems are some of applications of a connected undirected graph G, together with a nonnegative length for each arc [32].

- ARC-COVERING PROBLEM

If we have Arc (i, j) that means this arc covers node i and j. The problem here is, how to find the smallest possible subset of arcs that can be chosen, such that each node of G is covered by at least one arc of the subset [32].

- ARC-COLORING PROBLEM

The objective here is to paint the arcs of G various colours, subject to the constraint that not all the arcs in any cycle are painted the same colour. What is the smallest number of colours that will suffice [32].

- MIN-CUT PROBLEM

The objective is to find a subset of arcs (a “cut”) such that when these arcs are removed from G, the graph becomes disconnected. For what subset of arcs is the sum of the arc lengths minimized? [32].

- MAX-CUT PROBLEM

The objective is, to find a minimal cut such that the sum of the arc lengths is to be maximized.

- SPANNING-TREE PROBLEM

The objective here is to find a subset of arcs such that when these arcs are removed from G, the graph remains connected. For what subset of arcs is the sum of the arc lengths maximized? (The complementary set of arcs is a "minimal spanning tree.") [32].

- SHORTEST PATH PROBLEM

What is the shortest path between two nodes in a specific graph G.? This shortest path may be in term of length (the number of nodes between the two nodes) or the minimum cost when the arcs different cost and this may represent the many real-world problem such as the distance between cities, which has been used in famous optimization problem called "Traveling Salesman Problem (TSP)".

- LONGEST PATH PROl3LEM

Is to find the longest path, without repeated nodes, between two specified nodes of G?

2.14.2 Methods of Solution for the optimization Problems

The previous section briefly discussed some optimization problems. One can classify the solution methods into the following categories:

2.14.2.1 Linear programming (LP)

The LP is optimization model, which consists of one linear objective function and any number of equality or inequality constraints. LP is concerned with extermination of a linear objective function subject to linear inequality constraints. From a geometric point of view, the constraints describe a convex polytope. In the simplex, the computation of LP proceeds from one vertex of this polytope to another. One way to solve a combinatorial optimization problem by LP is to formulate a system of linear inequality constraints, which will cause the vertices of the convex polytope to correspond to feasible solutions of the combinatorial problem. Sometimes this results in a relatively small number of constraints, which can be listed explicitly in advance of the computation. Problems for which this is the case include the network flow problems, with the shortest path, min-cut, and assignment problems as special cases. LP is used to

solve many optimization problems such as arc-covering, arc-colouring, and spanning-tree problems as special cases [32].

2.14.2.2 Integer Linear Programming (ILP)

The ILP formulates a set of linear inequalities constraints to describe a convex polyhedron enclosing points (with integer coordinates) corresponding to feasible solutions of the combinatorial problem. A variant of the simplex method is applied and additional inequality constraints are generated as needed during the computation. These additional inequalities or "cutting planes" ordinarily bear little predictable relation to each other or to the original set of constraints [32].

ILP algorithms usually do not exploit any special combinatorial structure of the problem at hand. For this reason, they are sufficiently general to "solve" virtually any combinatorial optimization problem. Nevertheless, there is no possibility of establishing good a priori bounds on the length of computations, and practical experience with these algorithms has been very uneven [32].

2.14.2.3 Recursion and enumeration

Recursion and enumeration methods include dynamic programming and branch-and-bound. Dynamic programming is a technique for determining optimal policies for a sequential decision process. A surprisingly large number of optimization problems can be cast into this form and some of the most useful applications of this technique are in the combinatorial realm. In some cases, dynamic programming can be applied to solve problems with a factorial number of feasible solutions such as TSP and Shortest Path problem [32].

Branch-and-bound methods have been developed in a variety of contexts, and under a variety of names, such as "backtrack programming" and "implicit enumeration." Essentially, the idea is to repeatedly break the set of feasible solutions into subsets, and to calculate bounds on the costs of the solutions contained within them. The bounds are used to discard entire subsets of solutions for further consideration. This simple but effective technique has scored a number of notable successes in practical computations.

However, it is rarely possible to establish good bounds on the length of the computation [32].

2.14.2.4 Heuristics

Heuristics include algorithms whose their justification is based on arguments of plausibility, rather than mathematical proof. Often, these algorithms permit good computational bounds. However, generally speaking, only solutions, which are "close" to optimal or, at best, not the optimal solutions, are obtained [32].

2.14.2.5 Statistical sampling

Statistical sampling means, random generation of a number of solutions from the population of all feasible solutions for making some sort of statistical inference about the closeness of the best solution sampled to the actual optimum [32].

2.14.2.6 Special and ad hoc techniques

The special and ad hoc methods include those techniques, which do not conveniently fall into one of the other categories [32].

2.14.3 Evolutionary Methods (Evolutionary Algorithms)

The evolutionary methods provide a mechanism for accomplishment global and local search simultaneously [34]. The main evolutionary paradigms are genetic algorithm, genetic programming, evolutionary strategies, and evolutionary programming [35].

This section presents a general overview of the evolutionary methods and more detailed about some of these methods.

Evolutionary computing is a family of stochastic search techniques that imitate the natural evolution proposed by Charles Darwin in 1858. In the realm of search techniques, the classification in Figure 2.9 indicates the position of EAs [36]. EAs are stochastic searches and optimization heuristics derived from the classic evolution theory, which are implemented on computers in many cases. The basic idea is that if only those individuals of a population reproduce the next generation, which meet a

certain selection criteria, and the other individuals of the population die, the population will converge to those individuals meet the best selection criteria [37].

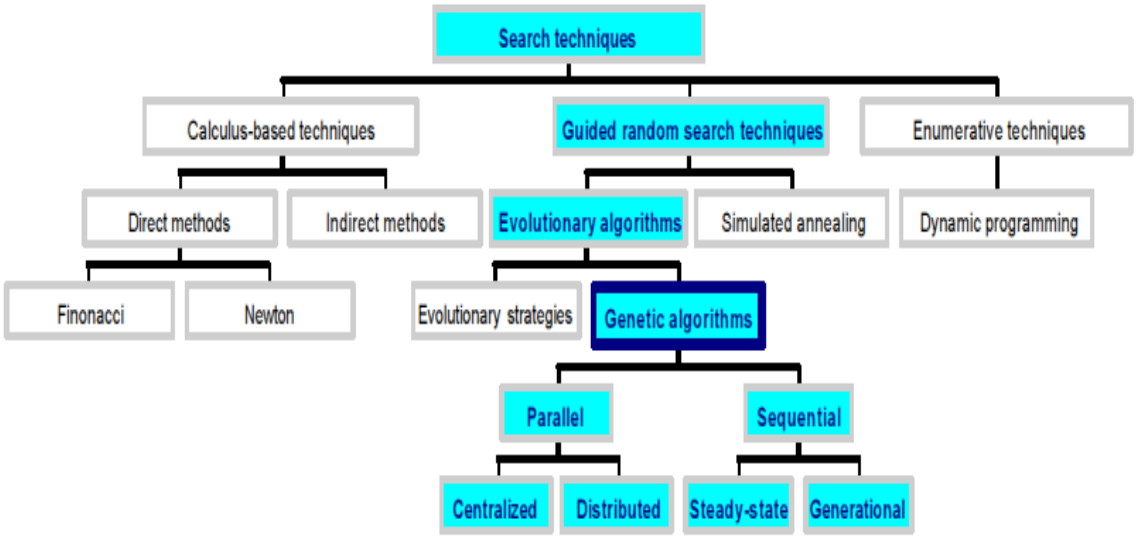

Figure 2.9: Search Techniques [36]

However, if the intelligence is considered as a kind of capability of an entity to adapt itself to ever changing environment, EAs are could be considered as a subdivision of soft computing as shown in Figure 2.10 [36].

The EAs are made of the several iterations of basic Evolution Cycle as shown in Figure 2.11.

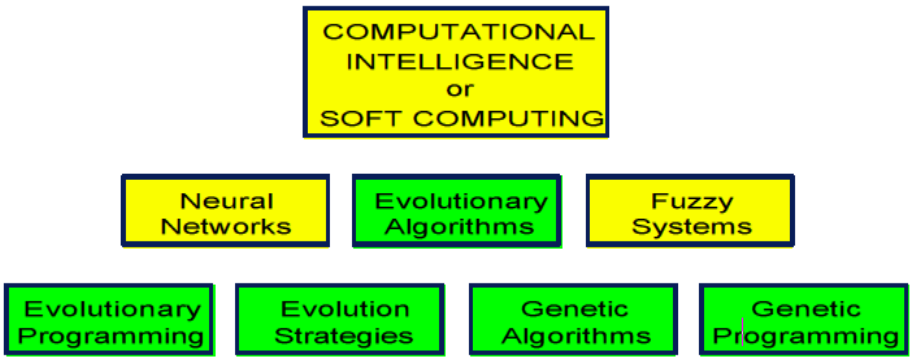

Figure 2.10: Evolutionary Algorithms and Soft computing [36]

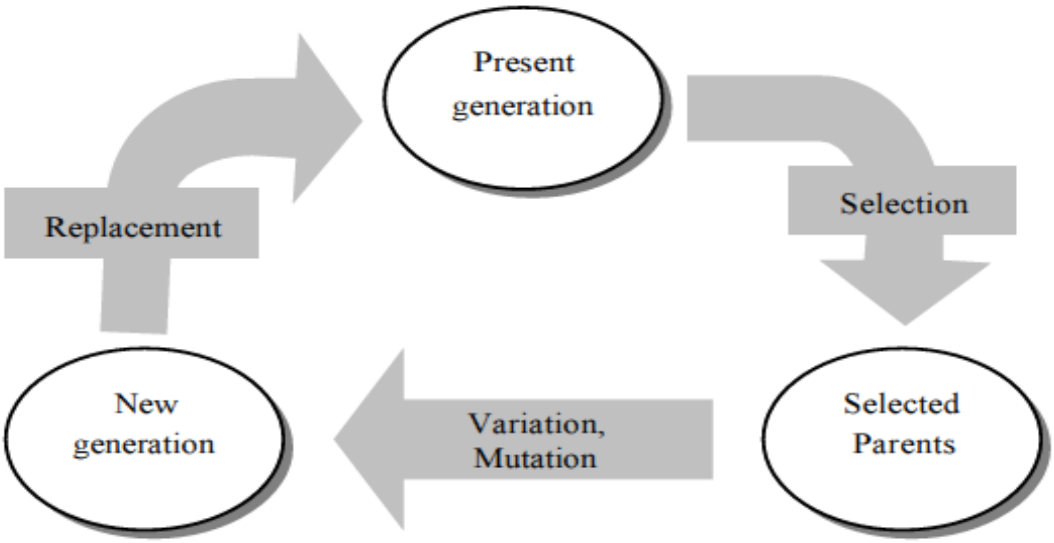

Figure 2.11: Basic Evolution Cycle [36]

Different variations of Evolutionary Computing incorporate the same basic cycle with a different presentations' model or specific combinations of Variation, Mutation, Selection, and Replacement methods. The interesting point in the implementation is the balance between two opposite operations. For example, the Selection operation reduces the diversity of the population while the Variation and Mutation operators try to increase diversity of population. This fact leads to the convergence rate and quality of solution. As an optimization algorithm, EAs should be analysed to find answers to fare questions such as convergence Rate, Quality of evolved solution and Computational requirements [36].

2.14.3.1 Domains of Application

The evolutionary optimization can be used in numerous applications such as:

1. Numerical, Combinatorial Optimization
2. System Modelling and Identification
3. Planning and Control
4. Engineering Design
5. Data Mining
6. Machine Learning
7. Artificial Life

2.14.3.2 Genetic Algorithms

Genetic algorithms (GA) are search algorithms that based on the rules of natural selection and genetics. The bases of genetic algorithm approach are given by Holland [38] and it has been deployed to solve wide range of problems [39]. The GA is a global search heuristic to find exact or approximate solutions for optimization and search problems. It is also, defined as a particular class of evolutionary algorithms (EAs), which is based on an evolutionary biology concepts such as inheritance, selection, mutation, and crossover. The technique inherited from the idea of the natural evolution of the generations to find the near optimal solution by simulating the biological cross of genes. GA is a tool used to solve kinds of computational problems with a high-complexity. In addition to modelling the phenomena occurring in Nature, they help in simulation, modelling, optimization, design and prediction purposes in science, medicine, technology [40]. The process of applying the GA starts by using an initial population, which were randomly generated to act as an initial candidate solution to the problem. GA applies a fitness function to decide, which individual will be kept in the new generation or the best elite depending on a specific criterion to evaluate the quality of each individual. To avoid the local optimal solution GA uses selection, crossover and mutation operators to select individuals that will be used for production and to generate the new generation called offspring. The mutation operator makes small changes in individual by swapping a small number of genes of the individual itself, but the crossover operator generates new generation (child) by cross exchange a number of genes between two individuals (parents). This helps GA to avoid running on local optimum solution and it is a process to find a near optimal solution. Figure 2.12 shows the basic steps of GA programming.

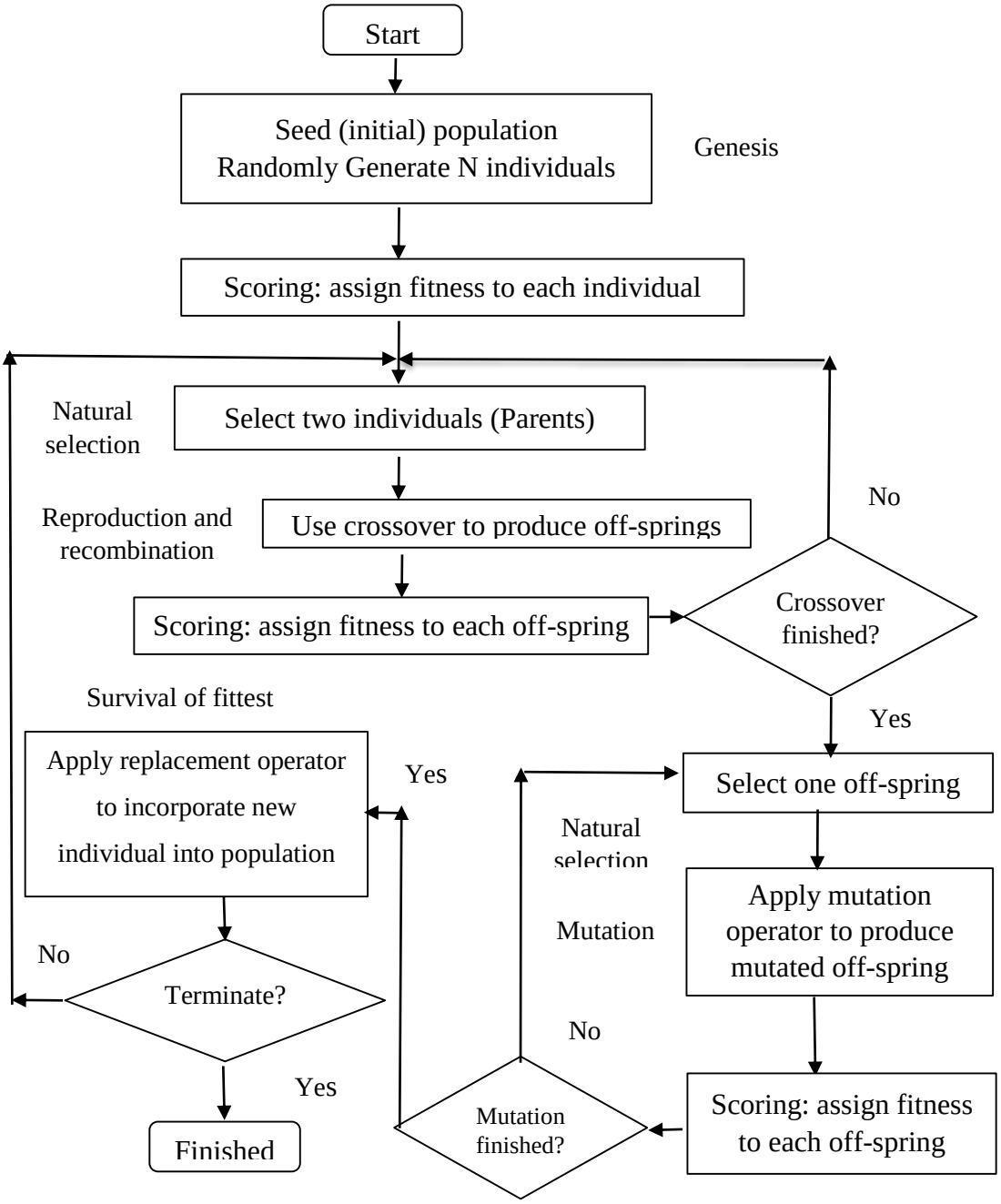

Figure 2.12: The basic GA programming chart

2.14.4 Simulated Annealing (SA)

The idea of SA is inspired from metallurgy science. The annealing is a technique that involves heating and controlled cooling of a material to increase the size of its crystals and lessen their defects. The heat causes the atoms to become unstuck from their initial positions (a local minimum of the internal energy) and wander randomly through states of higher energy; the slower cooling gives them more chances of finding

configurations with lower internal energy than the initial one. The processes of the SA go through a number of steps as follows:

1. Replaces the current solution by a random "nearby" solution, chosen with a probability that depends on the difference between the corresponding function values and on a global parameter *T* (called the *temperature*), that is gradually decreased during the process.
2. The dependency is such that the current solution changes almost randomly when *T* is large, but increasingly "downhill" as *T* goes to zero. The allowance for "uphill" moves saves the method from becoming stuck in local minima, which are the bane of greedier methods.

Figure 2.13 illustrates the basic steps of the SA algorithm.

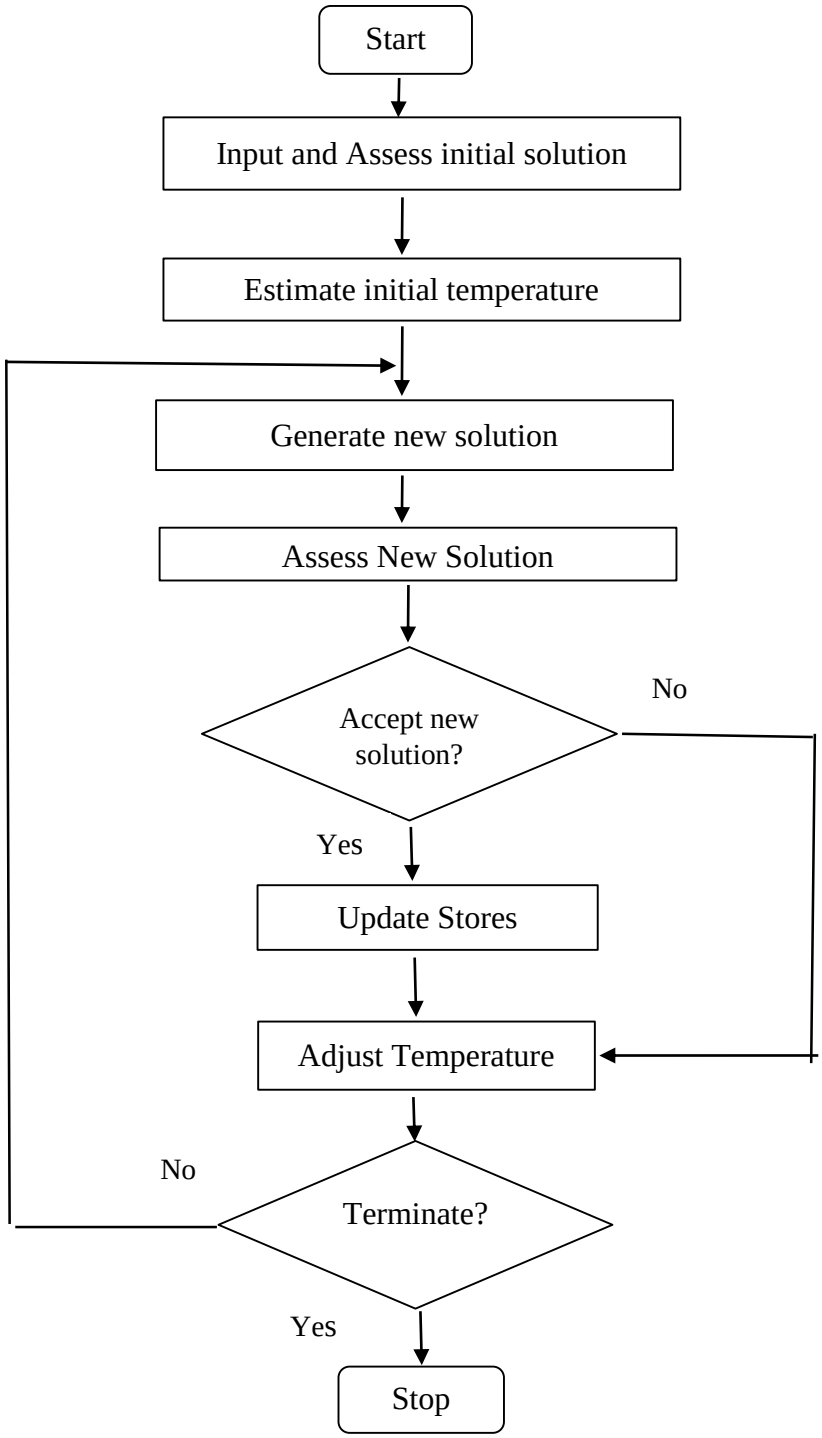

Figure 2.13: The basic steps of Simulated Annealing

2.14.5 Particle Swarm Optimization (PSO)

Practical Swarm Optimization (PSO), which is slightly, differs from GA and GP. PSO is an algorithm from the field of Swarm Intelligence. The algorithm for PSO was conceived based on observations of certain social behaviour in lower class or insects.

In contrast of modifying genetic codes using genetic operations as used in GA, in PSO movement of an individual is determined by the motion of the individual itself and that of the surrounding individuals. [35].

2.14.6 PSO Algorithm

The PSO algorithm simulates the motion of a large number of individuals ("Particles") moving in multi-dimensional space. Each individual stores its own location vector ($\overrightarrow{x_i}$), velocity vector ($\overrightarrow{v_i}$), and the position at which the individual obtained the highest fitness value ($\overrightarrow{p_i}$). All individuals also share information regarding the position with the highest fitness value for the group ($\overrightarrow{p_g}$). As the generations progress, the velocity of each individual is updated using the best overall location obtained up to current time for the entire group and the best locations obtained up to current time for that individual. The efficiency of this type of PSO search is certainly high because focused searching is available near optimal solutions in relatively simple search space. However, the PSO algorithm often gets trapped in local optimum in some sort of optimization problems. GA mutation can be integrated with the PSO to overcome this limitation [35].

2.14.7 Ant Colony Optimization (ACO)

Ant colony optimization was introduced in the early 1990s as a novel technique for solving hard combinatorial optimization problems. ACO belongs to the class of metaheuristics [41] [42] [43], which are approximate algorithms used to obtain good enough solutions to hard CO problems in a reasonable amount of computation time. [44]. The inspiring source of ACO is the foraging behaviour of real ants. When searching for food, ants initially explore the area surrounding their nest in a random manner. As soon as an ant finds a food source, it evaluates the quantity and the quality of the food and carries some of it back to the nest. During the return trip, the ant deposits a chemical pheromone trail on the ground. The quantity of pheromone deposited, which may depend on the quantity and quality of the food, will guide other ants to the food source [44].Thus, indirect communication between the ants via pheromone trails enables them to find shortest paths between their nest and food sources. This characteristic of real ant colonies is exploited in artificial ant colonies in order to solve CO problems [44].

2.15 BASIC NETWORK MODELS

Network design is one of the most important and most frequently encountered classes of optimization problems [45] [46]. It is a combinatory field in graph theory and combinatorial optimization. Many optimization problems in network design come directly from everyday practice in engineering and management: determining shortest or most reliable paths in traffic or communication networks, maximal or compatible flows, or shortest tours; planning connections in traffic networks; coordinating projects; and solving supply and demand problems [46].

The network design is also important for complexity theory, an area in the common intersection of mathematics and theoretical computer science, which deals with the analysis of algorithms [46]. However, there is a large class of network optimization problems for which no reasonable fast algorithms have been developed. And many of these network optimization problems arise frequently in applications. Given such a hard network optimization problem, it is often possible to find an efficient algorithm whose solution is approximately optimal [46]. The genetic algorithm (GA) is one of the most powerful stochastic search and optimization techniques based on principles from evolution theory [46].

2.15.1 Shortest Path Model

Shortest path problem (SPP) is at the heart of network optimization problems. The shortest path can capture most significant ingredients of network optimization problem. Even though it is relatively easy to solve a shortest path problem, the analysis and design of efficient algorithms requires considerable talent [46].

2.15.2 Traditional Methods for solving the Shortest Path Problem

A method to solve SPP is sometimes called a routing algorithm. The common and important algorithms for solving this problem are [46].

2.15.2.1 Dijkstra's algorithm

Solves single source problem if all edge weights are nonnegative. This algorithm can in fact compute the shortest paths from a given start point (source node) to all other nodes (destination nodes) [46].

2.15.2.2 Bellman-Ford algorithm

Is used to compute single-source shortest paths in a weighted digraph (where some of the edge weights may be negative). Dijkstra's algorithm achieves the same problem with a lower running time, but the edge weights should be nonnegative. Therefore, Bellman-Ford is usually used only when there are negative edge weights [46].

2.15.2.3 Floyd-Warshall algorithm

It used to solve the all pairs shortest path problem in a weighted, directed graph by multiplying an adjacency-matrix representation of the graph multiple times [46].

Recently, to address SPP, neural networks (NNs) and GAs (and other evolutionary algorithms) received a great deal of attention regarding their potential as optimization techniques for network optimization problems [46].

2.16 RELATED WORKS

The GPP has been considered to be an NP-Complete problem by many researchers. This section presents some of the research works that dealt with the GPP as an optimization problem. They use different methods for solving the GPP from different aspects and purposes. In addition, this section shows the efficiency of evolutionary methods on solving the GPP. Furthermore, this section discusses the research efforts that are closed the proposed solution.

In [47], the GPP has been studied and formulated as an ILP; two heuristic algorithms were developed in order to minimize the number of IGs while satisfying the MRs' throughput. The very important contribution is that GPP optimization proved as NP-hard problem by reduction from capacitated facility location (CFL) [48]. In [49], the GPP is computationally considered as N-Hard when it can be transformed into

minimum dominating set problem and it has been proven as NP-complete and then adapted a recursive dominating set algorithm to solve the minimum dominating set problem. The proposed algorithm considers the delay, relay load and IG constraints and has better performance than the algorithms in [50] [51] [52].

In [53], two algorithms for load balancing among the clusters as well as satisfying the quality of service constraints have been proposed. The network here has been divided to a number of disjoint clusters and each algorithm aimed to minimize the load difference between these clusters in term of aggregated traffic in the cluster head, which were determined by the gateways. The first algorithm is greedy one named GA-LBC, the second is combination of greedy, and GA formed new algorithm named HA-LBC. The results showed that the Hybrid algorithm outperformed the greedy algorithm due to the ability of GA to solve multiple objective problems.

In [54], a GA based solution has been proposed to solve the GPP. The approach aimed to integrate the locations of the existing gateways that based on the physical links (wired cable) as well as minimizing the number of extra gateways that are required to satisfy the users' demands to enhance the network capacity. The extra gateways based on Hybrid-FSO/RF that may use either Free Space Optical (FSO) or RF to form new clusters. The GA based solution is used to find the near optimal solution. However, the evaluation result has shown that the proposed solution achieved the optimum solution in small network made of up to 50 APs, which compared to result generated by the ILP formulation. Furthermore, the result has shown the feasibility of the approach in a relatively large network.

In [55] , a new solution has been proposed using GA to optimize the planning of WMN backbone focusing on routing and channel assignment. However, the proposed algorithm provides a good solution when dealing with large-scale WMN in relatively small computation time. The experimental results show the effectiveness of GA operators [55] but didn't consider the locations of end users and also dealt with ready network topology and may suffer from routing operations overhead [56].

In [57], a mixed approach of GA and LP methods has been proposed to optimize the WMN planning at the early stages. The approach considers two issues related to the

WMN that affect the network performance, the approach used to solve the Channel Assignment (CA) using GA and Multi-Channel Routing problems (MCR) using LP methods. The main idea is to find the optimal CA configuration with corresponding MCR schedule in the network in order to increase the overall network capacity. However, the genetic algorithm based approach achieved good result in solving this problem.

In [56], a new scheme has been proposed for planning and optimizing WMN, GA used for planning the location of IGs and MRs and as well as for routing and channel allocation optimization. However, the experimental results show that the proposed algorithms outperform the introduced greedy algorithm.

In [58], a configuration model for a fixed WMN has been proposed for determining the maximum and the optimal throughput depending on fixed wireless nodes with fixed locations and data flows generated in a logical manner, and to determine how the network can be configured to achieve the optimum throughput. They developed and investigated optimization framework to define the optimal throughput and to set the network configuration. They used an enumerative method to get numerical results in different situations of interest and to get different insights about the network structure considering the optimal routes, schedules and physical layer parameters. The proposed model helps in determining the achievable throughput in correspondent scenario. The main drawback of the proposed solutions in [59] [58] is that their algorithms based on ready network.

In [52], a new solution has been proposed to solve the GPP using clustering technique in the following four stages: select cluster heads, assign each node to an identified cluster satisfying the delay constraint, break down the clusters that do not satisfy the relay load constraint or the gateway capacity constraint, and finally select gateways to reduce the maximum relay load [60]. However, the algorithm does not have competitive performance because of the following two reasons: first, when identifying cluster heads and assigning MRs to the identified cluster heads, the algorithm does not make use of global information about the BWMN; second, splitting a cluster without considering re-assigning those MRs to existing clusters may create some unnecessary clusters and therefore increases the number of clusters significantly [60].

In [48], the researchers studied the Node placement problem, and then they proposed an algorithm based on mathematical programming. In addition, they proposed another two heuristic algorithms ("greedy algorithm and k-median based algorithm") in order to upgrade the highly congested nodes to IGs aiming to improve the packet delivery ratio.

In [61], the WMN planning for load balancing has been studied and a new model to optimize the load balancing in a multicast network called "Path-MR-Gateway load balancing (PMRGLB)". The model developed based on the PSO to minimize the all following four factors: (1) network's cost; (2) the length of the path; (3) interference in the path; (4) load variation among the gateways. However, the proposed model achieved better on minimizing the path length and load balancing as well.

In [62], a new approach based on EA has been to address the GPP. The proposed approach aimed to maximize the network throughput. In this approach, the gateways randomly distributed, and then the fitness calculated to find the near optimal solution. However, the numerical results have shown that the proposed approach achieved better than the "Multihop Traffic-flow Weight (MTW)" algorithm proposed in [63]. The remarkable point here MTW algorithm has been evaluated in [63] and achieved better than "Random Gateway Placement (RDP)", "Regular Placement (RGP)" and "Busiest Router Placement (BRP)" algorithms. This shows us the superiority of the evolutionary approaches over the random and the other conventional approaches in solving the problem gateways placement.

In [64], the efficiency of the PSO, ACO and GA in solving the problem of gateway placement has been studied and a comparative study between these algorithms has been presented. The evaluation results show that all of these algorithms outperformed the MTW. Moreover, the results show that the algorithms based on PSO and ACO showed better proprieties than the one that is based on the GA in small networks.

In [56], a new scheme for planning and optimizing the gateway locations in WMN has been proposed. The authors proposed two algorithms to optimize the location of the gateways, an algorithm based on the simple GA, and the other based on proposed improved GA. However, the experimental results show that both of the two algorithms

achieved better than the developed greedy algorithm and improved GA achieved better than the simple one.

In [65], a GA based approach to find the near optimal solution for MRs and the node placement problem has been proposed. The proposed approach aimed to maximize the size of giant network's component as a primary objective and the number of users to be converted as second objective. The approach has been evaluated based on a number of generated instances using different statistical distribution methods that were used in estimating the users' locations. The results show the usefulness of the GA on solving the placement problem of nodes and MRs on different network sizes, but the user coverage mainly depends on how the users were distributed in the specific area. In [66], further discussion about the approach proposed in [65] and the analysis results shows the efficiency of the GA on solving this problem.

In [67], the design of network topology and the gateway placement issues have been studied in order to minimize the network construction cost. Two algorithms have been proposed to address this problem; the first algorithm is the Predefined Gateway Set Algorithm (PGSA); the second algorithm is the Self-Constituted Gateway Algorithm (SCGA), also enhanced Dijkstra's algorithm has been proposed that is used with the GA in the main two algorithms to find network configuration with low-cost constraints such as delay and link capacity. However, the evaluation results show that PGSA has less computation time in comparison with SCGA, but the SCGA can achieve better results when we are concerned with the network construction cost but it requires more time.

In [5], the WMN design problem has been studied to minimize the network construction cost by optimizing the number of gateways. Two algorithms have been proposed using GA and SA. However, the proposed algorithms proved their ability to solve the network design problem. Furthermore, the results show SA is faster than GA in a large-size network, But GA achieved better than SA in the small-size network.

In [18], a comprehensive comparison study of GA and Hill Climbing Algorithm (HCA) considering the MRs placement optimization problem in WMN has been done and the

results proved that the solutions based on GA outperform the solution that's based on the HCA.

In [2], grid-based gateway deployment method has been proposed using cross-layer throughput optimization. LP-Flow-Throughput ("linear programming based") used as an evaluation tool. The evaluation shows that, the method exploits the available resources effectively and it performs better than the random and fixed deployment methods.

In [68], a heuristic model to find an optimal position for IG has been proposed, which it considers a single IG in WMN; they used the proposed model in [58] to generate multiple scenarios and then compare their relative performance in terms of the network throughput. However, the proposed solution can achieve a good performance by a achieving the optimum through, but the solution can be used in networks with single gateway only.

In [1] new approach has been proposed for load balancing considering the number of IGs, the average MR-IG-hop count and the load in balance in the IGs.

In [69], a gateway placement approach to minimize the number of IGs and to guarantee the bandwidth requirements in MRs and The Problem is formulated as a network flow problem. A max-flow min-cut based algorithm is developed for IG selection. An MR may be attached to multiple IGs through multiple paths. In this approach, the path length between MR and IGW has not considered as an optimization parameter and thus long paths may be selected [70].

In [71], the GPP has been confirmed as NP-Hard problem. The deployment of IGs has been solved using heuristic algorithms and formulated as LP model. The problem here was formulated as multi-objectives model to minimize the number of IGs and to minimize IG-MR-Hop count in affordable computation complexity. The model has distributed the IGs in clusters (IG as a cluster head) and each MR may has one more path to reach the IG.

2.17 SUMMARY

From the previous sections in this chapter, many researchers have paid their attentions to the WMN, especially the GPP. Thus, a considerable number of research works have dealt with the GPP. But, some of them based on a fixed or ready network (pre-established network), which take in the very limited chances in the enhancement, since the enhancement became in a later phases of network design. Some of them based on LP methods that have been proved that have restricted capabilities in larger size networks. However, the research works that are very close to our model in [1] and [71]. Nevertheless, the main difference is, our model aims to minimize the variation in the number of hop count between the MRs and their nearest IGs (VAR-MR-IG-Hop) among MRs in the whole network to insure that the IGs are placed in the appropriate positions, whereas the model that has been proposed in [1] is aimed to minimize the AVG-MR-IG-Hop and model that has been proposed in [71] is aimed to minimize the MR-IG-Hop.

CHAPTER THREE
GATEWAY PLACEMENT SOLUTION

3.1 BACKGROUND

WMN planning is an optimization problem as mentioned in the previous chapters. There are many issues that should be considered when designing WMNs, such as the area that will be covered, MCs' distribution in the specific area, bandwidth demanded by all clients, capacity of links and channels that should serve MR, the location of the IGs in the specific area, the cost of the IGs, which require high cost physical links compared with the wireless links used inside the network, the number of hops that the packet travers from the source to the destination (the distance between source and destination), especially the number of hops between each MR and the nearest IGs, the interference in WNs so that the wireless nodes should be placed in such a way that the interference is minimum so that to maximize the network throughput and the delay or the total time for packet transfer. Hence, the WMN planning is an optimization problem due to the conflicting requirements as mentioned before. For example, we can increase the network through and decreasing the distance between source and destination easily by increasing the number of IGs, but this will increase the construction cost due to the high cost of physical links used in IGs construction, so the optimization in planning the network is very important and its continuous challenge.

The main objective of this dissertation is to develop a new solution for IG placement in WMN considering the number of hops that the packet traverse from each MRs to the nearest IG (MR-IG-Hop) and the number of MRs associated with each IG to achieve better load and traffic balance in the network. This research, proposes two algorithms using GA and SA, both of the two algorithms aiming to minimize the variation in MR-IG-Hop among the MRs in the whole network as well as minimizing the variation among the IGs since each MR is associated with a specific IG. The problem is formulated as a mathematical model, whereas the network is represented by an undirected connected graph. The reset of this chapter presents and discusses the network representation, problem formulation and the details of the proposed algorithms.

3.2 NETWORK MODEL

The network is represented by undirected, connected and weighted graph, the edges of the graph is wireless links between the MRs including IGs inside the network. Since we are only concerned with the number of hop counts between the MR and IG (path length), the edges have the same weight with a value one. The IGs use the physical link to provide the Internet service for the network. Figure 3.1 shows a simple network sample, which consists of nine MRs (eight normal MRs and the ninth is IG). The vertices in the graph represent the network (MRs and IGs) and the edges represent the links between the MRs.

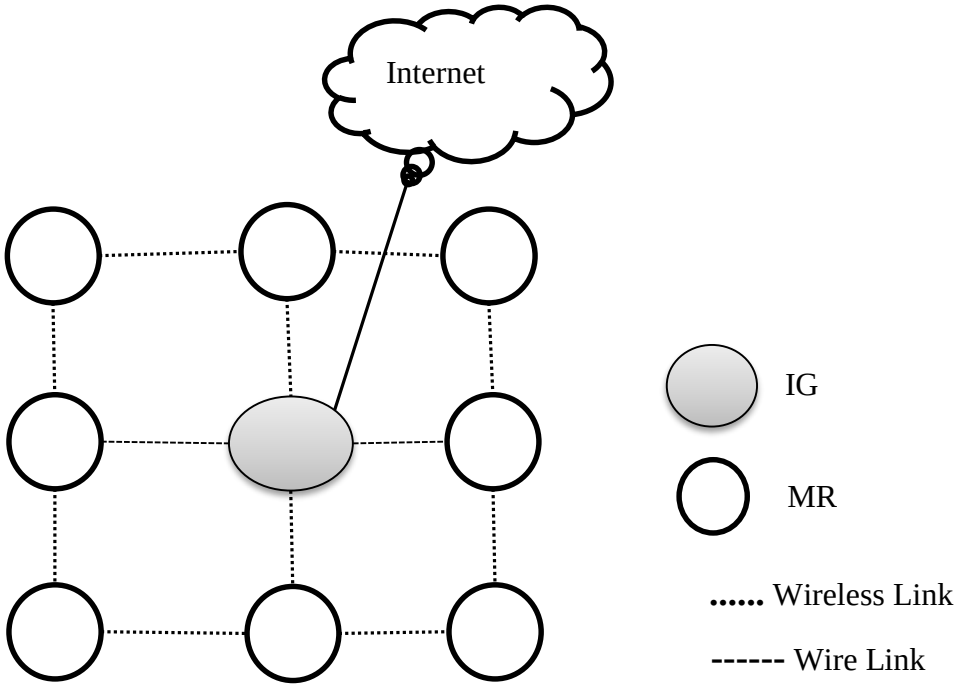

Figure 3.1: A Simple Network Example

The following figure shows the graph representation of the network model. The network here consists of 25 MRs and 4 IGs.

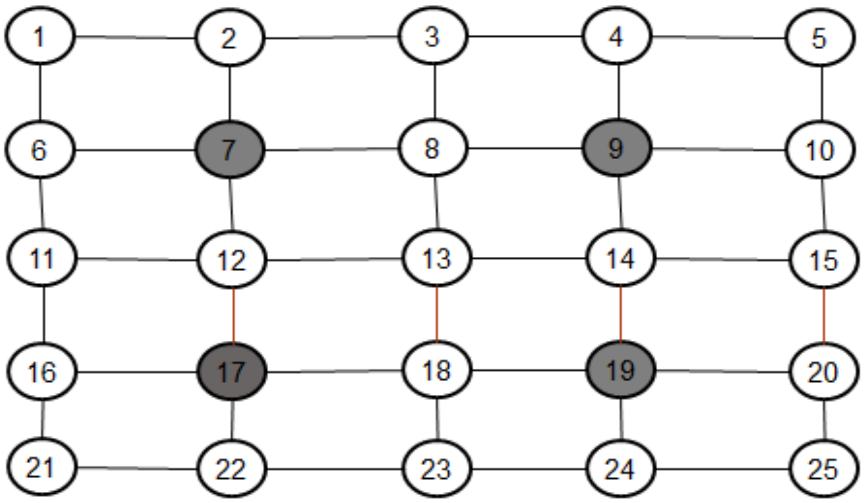

Figure 3.2: Network sample in undirected graph.

3.3 THE PROBLEM FORMULATION

The network here is defined as a set R= $\{R_1, R_2, \ldots, R_r\}$ of r MRs that construct the network, a subset N = $\{ n_1, n_2, \ldots, n_n\}$ of n MRs that act as none gateways nodes, a subset Gn= $\{ g_1, g_2, \ldots, g_g \}$ of g IGs connected the internet as $G \subseteq R$ where $1 < g <$ n, and a set E of m edges that connecting MRs denoted as $e_{ij} \subseteq E$ where e_{ij} represents the link between node i and node j. Thus, the network can be denoted as a graph G(R, E), r=|R| is the number of MRs including the IGs. The number of IGs is g=r-n where (1<=g<r). and the graph here is undirected graph and all its edges have the same weights of the value one because we consider only the number of nodes between each node and the nearest IG. The previous description will be used during all the next processes of the proposed solution and the notations that will be employed in the problem formulation and the other model stages are presented in the following table:

Table 3.1: The notations and symbols used in the problem formulation

Symbol	Description
R	The set of all MRs
r	The number of all MRs.
N	The set of none IG nodes.
n	The number of none IG nodes.
E	The set of all links between each MRs.
Gn	The set of IGs.
g	The number of IGs

l_{ij}	Indicator with a value 1 if the path between the MR i and the IG j is the smallest path between the MR i and each other IGs otherwise the indicator should be 0.
D_{ij}	The distance between the MR i and the IG j.
g_{nj}	The number of all MRs associated (all the MRs that their nearest IG is IG j) with the IG j.
y_j	The average hop count that the packet traverses between the IG j and every MR associated with this IG.
x_i	The distance between the MR i and the nearest IG.
$\overline{y}$	The average hop count among IGs.
$\overline{x}$	The average hop count among all MRs.
σ	The variance between all MRs in term of hop count to the nearest IG (VAR-MRs-IGs-Hop)
σ_{IGs}	The variance of hop count of the among the IGs (MRs associated with each IG)

The network design problem is mathematically formulated as follows:

$$\text{Min } \sigma = \left(\frac{1}{n}\sum_{i=1}^{n}(x_i - \overline{x})^2\right) \quad , where\ n \geq 1 \tag{3.1}$$

$$\text{Min } \sigma_{IGs} = \left(\frac{1}{g}\sum_{j=1}^{g}\left(y_j - \overline{y}\right)^2\right) , where\ g \geq 1 \tag{3.2}$$

Such that

$$x_i = \min_{j=1,\dots,g}\left(D(n_i, g_j)\right), \text{where } i = 1, \dots, n \tag{3.3}$$

$$D(i,j) = \begin{cases} 1 \text{ , if } i \neq j \text{ and if i and j are adjecent} \\ 0 \text{ , if } i = j \\ > 1 \quad otherwise \end{cases} \tag{3.4}$$

$$\text{where } i = 1, \dots, n\ ;\ j = 1, \dots, g \ \text{ and } \ 1 \leq g < n$$

$$l_{ij} = \left\{ \begin{array}{ll} 1 & , \text{if } x_i = \min(D(n_i, g_j)) \\ 0 & \text{otherwise} \end{array} \right\} \tag{3.5}$$

$$\text{where } i = 1, \dots, n \; ; \; j = 1, \dots, g \;\; \text{and} \;\; 1 \leq g < n$$

$$y_j = \frac{\left(\sum_{i=1}^{n} x_i * \; l_{ij} \;\right)}{\sum_{i=1}^{n} l_{ij}} \;\; , \text{where } i = 1, \dots, n \; ; \; j = 1, \dots, g \text{ and } 1 \leq g < n \tag{3.6}$$

$$\overline{y} = \frac{\sum_{j=1}^{g} y_j}{g} \;\; , \text{where } j = 1, \dots, g \;\; and \;\; g \geq 1 \tag{3.7}$$

$$\overline{x} = \frac{1}{(n)} \sum_{i=1}^{n} x_i \;\; , \text{where } i = 1, \dots, n \;\; \text{and } n \geq 1 \tag{3.8}$$

$$\left(\sum_{i=1}^{n} l_{ij} \right) \geq 1 \qquad , \text{where } i = 1, \dots, n \text{ and } j = 1, \dots, g \tag{3.9}$$

$$\forall \; n_i, g_j \in R \tag{3.10}$$

$$\forall \; n_i \; \in N \; and \; \forall \; g_i \; \in Gn \tag{3.11}$$

$$\nexists \; n_i \; \in Gn \;\; \text{and} \nexists \; g_j \in N \tag{3.12}$$

The objective function (3.1) means minimizing the variation of hop count between MRs and their nearest IGs (VAR-MR-IG-Hop), which denoted by MR-IG-Hop. The objective function (3.2) means minimizing the variation of hop count between each IG and its associated MRs to guarantee that, each IG is placed in the near optimal position to serve a group of selected MRs that were selected based on the shortest paths between the MRs and the IGs. Constraint (3.3) is used to calculate the shortest paths between the corresponding MR and all IGs in the network using Dijkstra's algorithm. Then, the shortest path will be returned and this MR will be associated with the IG that satisfies this constraint. Constraint (3.4) defines a rule for adjacent MRs as well as each MR with itself. Constraint (3.5) returns the value one when the IG satisfies constraint (3.3)

otherwise return the value zero. Equation (3.6) is used to calculate the average hop counts among the MRs that associated with the corresponding IG, which subjected to a constraint (3.3). Equation (3.7) is used to calculate the average among the IGs that to be used by the objective function (3.2). Equation (3.8) returns the average hop count among the MRs (AVG-MR-IG-Hop) in the network to be used by the objective function (3.1).

3.4 PROPOSED ALGORITHMS

In order to deliver the final solution for GPP, two algorithms have been proposed. The first algorithm based on the GA, the second algorithm based on the SA. More discussion and details about these algorithms will be presented in chapter four and how these algorithms will be implemented to solve the research problem that were guided by the proposed mathematical model. Finally, the discussion about experimental results will be presented in chapter five.

3.5 THE EVALUATION METHOD

Clearly, from literature review, GA and SA are evolutionary algorithms (EA). The EA should be analysed to answer one of the following questions [36].

1. The convergence rate.
2. The quality of the evolved solution.
3. The computational requirements.

So far, no general analysis framework has been proposed to analyse the general form of evolutionary algorithms, but some specific variations or implementations could be focused along two lines of investigations: theoretical and empirical. The theoretical approach attempts to discover mathematical truths of algorithms that will hold in a reasonably broad domain of applications. However, the empirical approach attempts to assess the performance of an implementation in a specific domain of application. Both methods have advantages and disadvantages, and in practice they should be used as complementary means to design and tune the specific instance of an algorithm. Therefore, only the proposed algorithms will be evaluated and analysed to answer the first two questions (the convergence rate and the quality of the evolved solution)

3.6 SUMMARY

This chapter has presented a brief discussion about the research problem (GPP) and the proposed network model using undirected graph. In addition, the chapter has presented and explained the mathematical formulation of the research problem and its solution besides a brief overview about the proposed algorithms that will be used to solve the research problem based on the proposed network model and mathematical formulation to achieve the research objectives, which are also have been represented in the mathematical with the objective functions. Moreover, the chapter has presented and discussed the evaluation method that will be used to measure the quality of the proposed algorithms.

The next chapter presents more details about the proposed algorithms and how they will be implemented to solve the GPP.

CHAPTER FOUR
THE PROPOSED ALGORITHMS

This chapter presents in details how the proposed algorithms will be implemented, as well as presenting the idea behind the tools and functions, which have been used to carry out the proposed approaches. Furthermore, the chapter discusses the development environment that has been used to implement the proposed algorithms. Finally, the chapter presents a brief conclusion summarizing what have been done in the implantation of the proposed algorithms.

4.1 NETWORK MODEL

The network model determines the network representation on the decoding stage. This model simulates the real network configuration in mathematical method. The solution represented by an undirected graph with one-unit weights edges as mentioned in the previous chapter. The graph here is represented by $n \times n$ matrix where n is the number of MRs in the network including the IGs. The matrix here represents the graph and edges connect the vertices (MRs). Two vectors represent the network configuration. The first one is $1 \times g$, which represents the IGs and the second one is $1 \times (n - g)$ vector, which represents the normal MRs. Both of the two vectors store the node IDs to be in consistency with the matrix that represents the network. Figure 4.1 shows a network configuration of nine MRs and one of them is an IG. The matrix shows the edges between nodes in the graph, the first vector shows the normal MRs nodes and the second vector shows the IGs nodes.

$$\begin{pmatrix} 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \qquad \begin{Bmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 6 \\ 7 \\ 8 \\ 9 \end{Bmatrix} \qquad \{5\}$$

Figure 4.1: Network representation (edges' matrix and configuration vectors)

4.2 THE GA-BASED APPROACH

The following subsections present the detail of the GA based algorithm (the GA representation for the GPP). The stages of applying GA to solve GPP those we will discuss are firstly, the network representation (chromosome) or network encoding in the real world. Secondly, the crossover operator and here we will present three crossover types namely: uniform, single point and Two-point crossover. Thirdly, the mutation operation and here we will present only inversion mutation type. Fourthly, the important parts of our GA, the fitness function and here we will present three different fitness functions that are used separately. Fifthly, the selection operator that will be used in the proposed approach and finally: a novel repair procedure that have been used to keep the number of the IGs constant in the network.

4.2.1 Network Encoding (Chromosome Representation)

The encoding process is the first and the core process of GA to represent the real-world problem. In this research, all MRs are labelled from 1 to n where n is the number of MRs including those were chosen as IGs. Therefore, the binary string has been used to represent the chromosome as shown in Fig. 1, the order of gene in the chromosome determines the node ID in the network, and the value of this gene determines either this node is selected as an IG, which represented by 1 or selected as a normal MR, which represented by 0.

The binary string representation is used to represent the network configuration in encoding stage of the GA as shown in Figure 4.2. Then, this representation will be inverted in the decoding stage to simulate the real network based on graph's representation, which have been discussed in section 4.1

Node ID	1	2	3	4	5	6
Node type	0	0	1	0	0	1

Figure 4.2: Chromosome's representation

4.2.2 Fitness Function

The GA uses the fitness function to evaluate the quality of each individual in the current population based on the objectives defined in equations (3.1, 3.2) guided by the decision

variables defined in the equations 3.3 through 3.8 and subject to constraints 3.9 through 3.18 The individual with the highest fitness value will be selected as elite to be kept in the new population, while the remaining individuals of the new population will be generated using crossover and mutation operators. Three different fitness functions have been developed based on the combination of the objective functions for further optimization opportunities as follows.

4.2.2.1 MRs-VAR Fitness Function

This function considers the objective (3.1) so, that the aim is, to minimize the VAR-MRs-IGs only whatever the value of the objective (3.2).

4.2.2.2 IGs-VAR Fitness Function

This function considers the objective (3.2) so, that the aim is, to minimize the VAR-MRs-IGs among the MRs associated with each IG only whatever the value of the objective (3.1).

4.2.2.3 VAR-MRs-IGs-Hop Fitness Function

This function considers the objectives (3.1, 3.2) in the evaluation process. The idea is, the objective 3.1 ($f1$) is the primary objective while objective 3.2 ($f2$) is the secondary objective in the optimization processes. Here, we minimize $f1$ until no improvement will happen, then minimizing $f2$ without worsening the value of $f1$.

4.2.3 Selection Operator

An ideal selection strategy should be such that it is able to adjust its selective pressure and population diversity so, as to fine-tune GA search performance. The tournament selection strategy provides selective pressure by holding a tournament competition among the selected individuals on the tour [72]. The best individual from the tournament is the one with the highest fitness, which is the winner of the tour. Therefore, the algorithm uses the tournament selection to select a number of individuals from the current generation based on a specific probability called the tournament probability denoted by Tp. Then, the selected individuals will be ranked according to their fitness

values and the individual with the highest quality used to reproduce the offspring by applying crossover and mutation operators.

4.2.4 Crossover Operator

Three types of crossover, single point, two point, and uniform have been used alternatively for further optimization.

4.2.4.1 Single Point Crossover

The single-point crossover has been used to generate the new offspring's, which is done by selecting a single point within the parents. Copies the genes before this point from the first parent in the corresponding positions of the child (offspring) and then fill the remaining genes of the child from the second parent with the genes in the positions after the selected point. Figure 4.3 shows how to perform the single-point crossover.

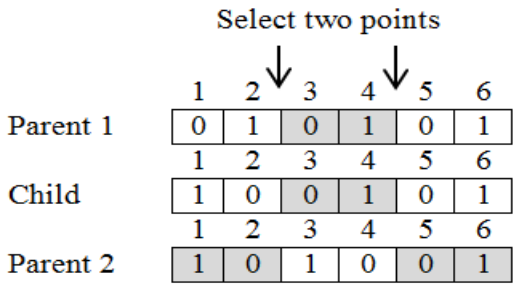

Figure 4.3: Single-point crossover operator

4.2.4.2 Two-Point Crossover

The two-point crossover has been used to generate the new offspring's, which is done by selecting two points within the parents, then copy the genes between these points from the first parent in the corresponding positions of the child and fill the remaining genes of the child from the second parent as illustrated in Figure 4.4.

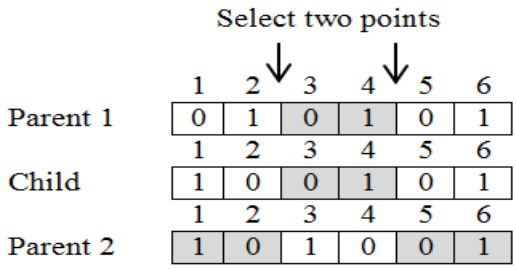

Figure 4.4: Two-point Crossover operator

4.2.4.3 Uniform Crossover

The uniform crossover has been used to generate the new offspring's based on a fixed mixing ratio between two parents. Unlike single and two-point crossover, the Uniform Crossover enables the parent chromosomes to contribute the gene level rather than the segment level. Therefore, this type has been used for further optimization and to compare the results of different crossover types. Here, the genes' exchange operation uses a specific probability called crossover probability and denoted by P_c then, a random mask of 1s and 0s will be generated with a chromosome length as follows:

- Generate a random number P_R per each gene in the chromosome.
- Generate the random mask according to the following formula:

$$corresponding\ mask\ gene = \begin{cases} 1 & if\ P_R < P_c \\ 0 & ,otherwise \end{cases}$$

- The final mask will be as shown in Figure 4.5:

Mask	0	0	1	0	0	0	1	0

Figure 4.5: Mask Sample in Uniform Crossover

- The one value indicates that, the corresponding gene in the child will be filled from the first parent, where the zero value from the second parent. Figure 4.6 illustrates these steps.

Parent 1	1 0 1 1 0 0 1 1
Parent 2	0 0 0 1 1 0 1 0
Mask	1 1 0 1 0 1 1 0
Child 1	1 0 0 1 1 0 1 0
Child 2	0 0 1 1 0 0 1 1

Figure 4.6: Uniform Crossover

4.2.5 Mutation Operator

The swap mutation operator has been used to modify the individual by selecting two positions within the individual randomly and swaps the genes in these positions to produce a new individual and prevent GA from falling on the local optimum solution as shown in

Figure 4.7.

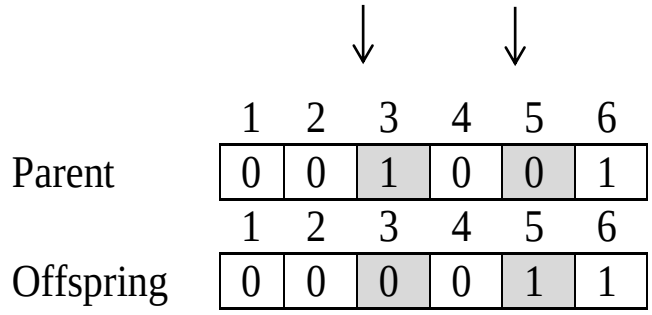

Figure 4.7 : Mutation Operator

4.2.6 Repair Procedure

The number of IGs is constant (the number of IGs is determined before the starting of the GA process) in this solution and we use designated parameter to keep the number of IGs in the network but due to the genetic operator, especially the crossover operator ,which causes continuous changing within the individuals this number may either increase or decrease. Therefore, this procedure has been added to the GA processes in order to alter the individual that violates the constraint of IGs number.

The repair process runs as follows:

- Count the number of the genes that have the value 1, which represents the IGs in the real network (in the decoding stage).
- If the number of IGs is greater than the desired number then do the repair as follows:
    - Generate a random number R between zero and the chromosome size (the total number of MRs in the network).
    - If the value of the gene at the position r is, one then this value will be changed to zero to reduce the number of ones. Else if the gene at position r is zero, then a new random number r will be generated
- Repeat these three until the required number of ones (IGs number) is met.

4.2.1 The initial population

Firstly, the r number of MRs will be distributed in the area that we want to cover. Then, the IGs will be selected randomly among the r MRs, where the number of IGs z is considered as designated parameter. The previous description will be used to generate each individual in the initial population P_0 to be used in the reproduction of the next generation through the GA operators (selection, crossover, and mutation besides the repair procedure to maintain the number of IGs).

4.2.2 The Algorithm Template

Algorithm 1: GA-Based algorithm procedure (pseudocode)

Input:

Set i=0 ‘ iteration counter

Set initial parameters: Termination condition: T_c, Population Size: **Z**, tournament probability: T_p

Step 1:

Generate the initial population P_i With the size Z.

Step 2: Evaluate P_i

If P_i Pass the Evaluation then

Step 3:

Do while not T_c

Select **Elite** = the best individual P_i According to fitness value

For j =2 to z

Selects two parents (Pr_1, Pr_2) using **selection operator** with probability T_p

‘Generate the offspring using crossover on the selected individuals using following formula:

Offspring=crossover(Pr_1, Pr_2)

‘Checks the validity of the offspring if not passes, then apply repair procedure.

If not checkIndividual (offspring) then

RepairGenes(offspring)

End if

‘Add offspring to the crossover population

$\mathbf{P_i^c}$.Add (offspring)

Next j

‘Apply mutation using the following formula:

$\mathbf{P_i^m = mutate(P_i^c)}$.

Create a new generation $\mathbf{P_{i+1}}$ From the **Elite** the individuals in $\mathbf{P_i^m}$

i=i+1

Loop

Output:

Return the best solution

Figure 4.8: GA-Based Algorithm pseudocode

4.2.3 Illustration of the GA processes

The following subsection presents a part of GA processes to optimize the network configuration by optimizing the locations of the IGs according to the constraints defined in the problem formulation. Figure 4.9 shows an example of an individual in the encoding stage (chromosome) and Figure 4.10 shows the correspondent network in the decoding stage (the real world). The network will be represented here consists of 25 MRs and four IGs. In addition, the figures show how the genetic operators change and optimize the network’s configuration.

1. Individual in the encoding and decoding stages.

Figure 4.9 : sample of individual in the encoding stage

Figure 4.10 shows the corresponding network configuration of the above individual. The node IDs 7, 8, 15, and 25 represent the IGs.

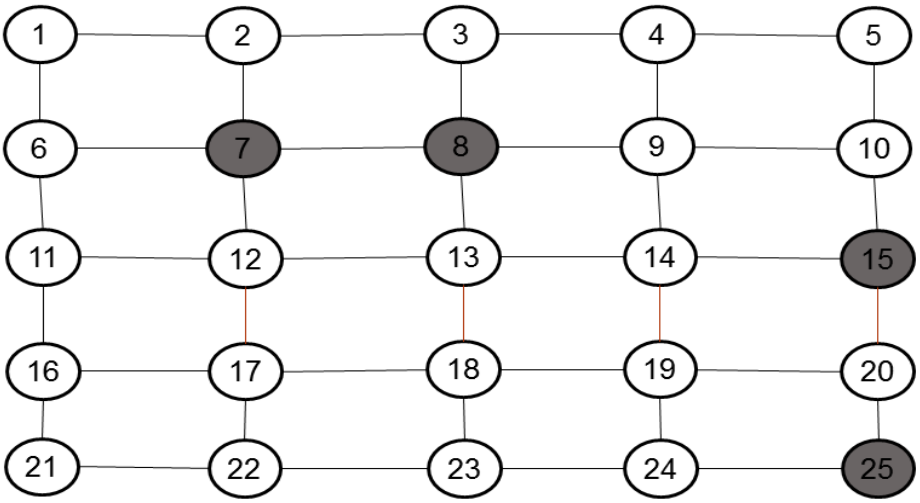

Figure 4.10: a sample of network configuration (Decoding stage)

2. The population

   Figure 4.11 shows a sample population in the GA. The population consists of ten individuals.

0	0	0	0	0	0	1	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1
0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	1	0	0
0	0	1	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0
1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0
0	1	1	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0
1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0	0	0	0	0	1	1	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
0	0	1	0	0	1	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	1	0	1	0	0	1	0	0	0	1	0	0	0	0	1	0	0	0	1
0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	1	0

Figure 4.11: Population’s sample of ten individuals

3. Crossover process

The best individual from the current generation will be selected as an elite based on the result of the fitness function as mentioned before in this chapter. The elite individual will be kept in the new generation. The remaining individuals of the new generation will be generated using the current generation through the processes of crossover and mutation operators. From each population a number of individuals will be selected using the selection operator to be used in the reproduction to generate the new generation (crossover’s generation) using the crossover operator. Figure 4.12 shows the crossover process.

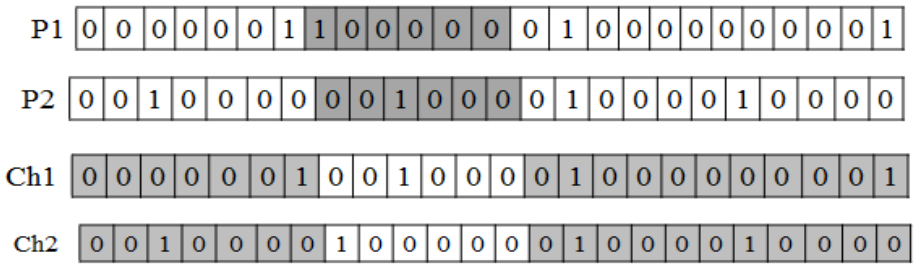

Figure 4.12: crossover process at the encoding stage

Figure 4.13 shows the corresponding network configuration of the first and the second parents and the child before and after decoding stage.

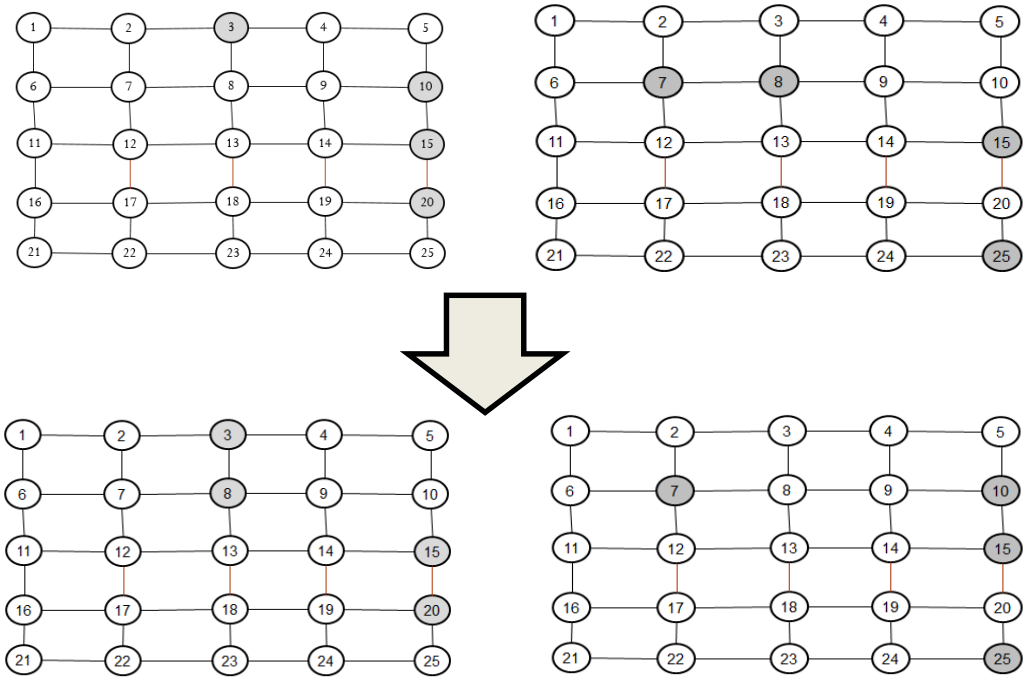

Figure 4.13: network sample generated by the crossover process

4. Mutation Operator

The mutation will be applied to the new generation that has been generated by the crossover operator (crossover's generation). In this example, the mutation will be applied to the child1 and child2 that have been shown in Figure 4.13 . Figure 4.14 and Figure 4.15 show the effect of mutation operator in the encoding and decoding stages.

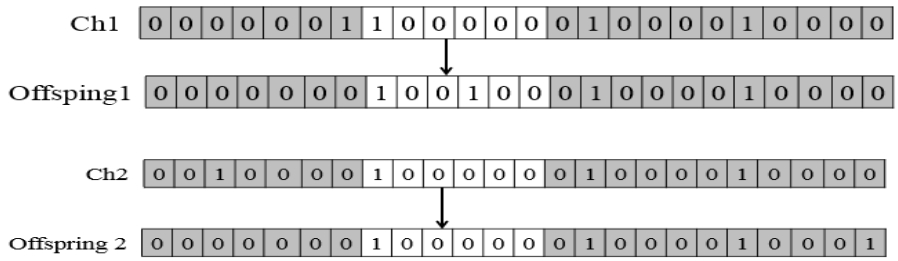

Figure 4.14: mutation process (in the encoding stage)

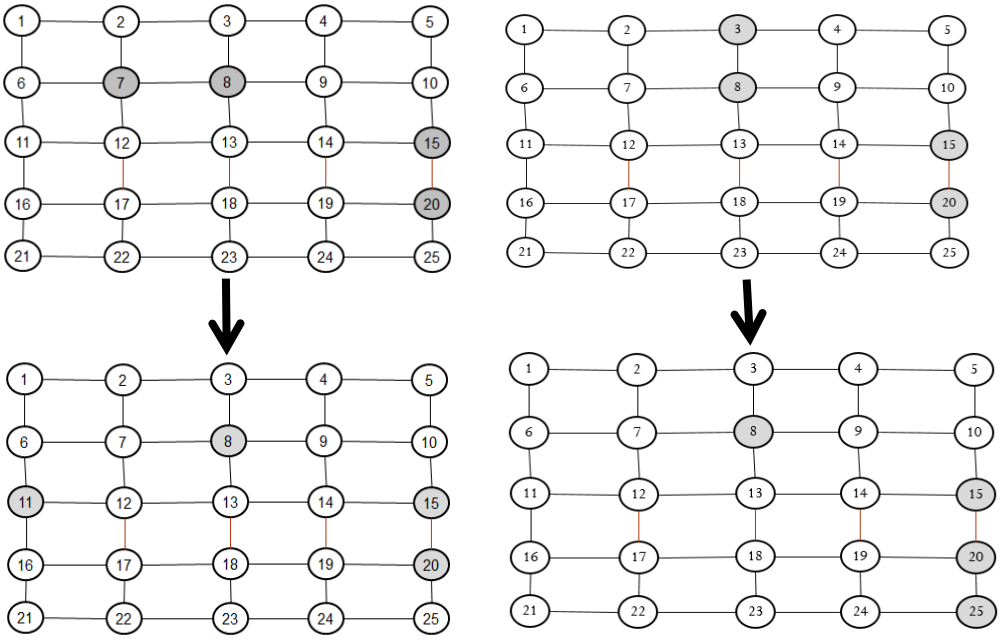

Figure 4.15: two networks generated by the mutation processes (decoding stage)

4.3 THE SA-BASED APPROACH

The SA algorithm works iteratively while the termination condition is false to find the best solution. SA starts from initial solution at starting temperature (internal temperature) denoted by T, which randomly generated and denoted here by S_0 in each iteration, new solution (current solution) S_w will be generated from the previous one (Preceding solution) based on a specific procedure. Then the best solution S_b between S_0 and S_w depends on a specific measure, according to the objective functions defined in equations (3.1 and 3.2). If the new solution is better than the old one, then the new solution will be chosen as S_b to replace the previous S_b. If not, then the replacement will be done based on probability denoted by P forming a new function named transition function. This function uses two parameters α and T to determine whether to accept this

solution as S_b or to keep the old S_b. While the iterations continue, and keeping the best solution at each iteration, the T value is decreasing (cooling operation) until the final temperature is reached, which denoted here by T_f and it is used as a termination condition. Table 4.1 shows the symbols that have been used in the SA processes.

Table 4.1: The symbols that have been used in SA processes.

Symbol	**Description**
T	The internal temperature
T_f	The final temperature
S_0	The initial solution
S_w	The working solution
S_b	The best solution
E_w	The energy of the S_w.
E_b	The energy of the S_b.
A	Is used to alter the temperature value.
Δ	The energy difference between S_w and S_b $(E_b - E_w)$
R	Random number
A_c	The acceptence condition of the S_w as the S_b
I	The temperature incremental counter
G	The IGs number
N	The of MRs including the IGs

The following subsections present the details of the SA algorithm.

4.3.1 Network Encoding (Representation)

It is similar to network representation in the GA based algorithm. The algorithm uses a binary string to represent the network solution in the encoding stage. The solution consists of n bits represent the MRs in the network. The bits of value one represent the IGs where the bits with the values zeroes represent the normal MRs. The order of the bit in the string determines the node ID in the network, which is used to differentiate between the nodes. The node IDs are used in both encoding and decoding stages. The latter has been discussed in section 4.1. Figure 4.16 shows the solution representation in the binary string.

Figure 4.16: the solution representation in SA in the encoding stage

4.3.2 The initial solution

The initial solution will be generated randomly with the consideration of the IGs number. The solution length is equal to the number of the MRs in the network as mentioned in the previous subsection. The number of the IGs will be kept also in designated parameter in the same process of the GA solution. The solution here is a single network configuration corresponding to the individual in the GA algorithm representation. As shown in the Figure 4.16.

4.3.3 The fitness function

The quality of the solution is called the solution energy in the SA concepts, which is inherited from the metallurgy science as mentioned before in chapter 2. The fitness uses the same criterion in the GA solution based on the objective functions (3.1 and 3.2) and the constraints and equations defined in the mathematical formulation of the problem as follows:

4.3.3.1 MRs-AVG Fitness Function

This function considers the objective (3.1) so, as to minimize the variation in AVG-MRs-IGs only whatever the value of the objective (3.2).

4.3.3.2 IGs-AVG Fitness Function

This function considers the objective (3.2) so, as to minimize the variation in AVG-MRs-IGs among the MRs associated with each gateway only whatever the value of the objective (3.1).

4.3.3.3 MRs-IGs-AVG Fitness Function

This function considers the objectives (3.1, 3.2) in the evaluation process. The idea is, the objective 3.1 ($f1$) is the primary objective while objective 3.2 ($f2$) is the secondary objective in the optimization process. Here, the aim is to minimize $f1$ until no

improvement happened for a long time, then minimizing $f2$ without worsening the value of the $f1$.

4.3.4 The best solution

There are always two solutions during the SA processes, the current solution, which is the best solution and working solution, which is the solution undergoes the evaluation of the fitness function. If S_w has a better fitness value (higher energy) than the current S_b, then S_b will replaced by S_w, Otherwise, the S_b will be determined by a function called "transition function."

4.3.5 The transition function

This function is used to determine the best solution if the current working solution has less energy than the best current solution. It uses probability transition for better chances toward the optimal solution or tries to prevent the algorithm from not to stuck on the local optimum solution. The processes of this function use two parameters α and T to determine the acceptance or the rejection of the working solution as the best solution. The function returns logical values true or false in comparison with a random number. The true value indicates the acceptance and false indicates the rejection. The following illustration shows the details of the transition processes.

1. Generate the random number R
2. Calculate δ value using the following formula: $\delta = E_b - E_w$
3. The acceptance of S_w condition is: $A_c = \begin{cases} true & ,if\ (e^{\left(-\frac{\delta}{T}\right)} > R) \\ false & ,otherwise. \end{cases}$
4. If $A_c = true$, then $S_b = S_w$

4.3.6 Cooling Control and Stopping Condition

The algorithm processes start with high internal temperature (T) and final temperature T_f. The parameter α is used to decrease the T value at each iteration as follows:

$$T = T * \alpha$$

The iterations will continue until the following condition (stopping condition) is true:

$$T = T_f$$

4.3.7 The Algorithm Template (Pseudo code)

Algorithm 2: SA Procedure

Input:

Initialize T, T_f, α, g, n

S_0= Initial Solution ()

S_w= Evaluate (S_0)

$S_b = S_w$

While $T > T_f$ Do

While I < Iterations_At_Temperature do

S_w=Generate(S_w,T)

$E_w = ComputeEnergy(S_w)$

E_b=ComputeEnergy (S_b

A_c=Evaluate (T, E_w, E_b)

If A_c=True then

S_b=S_w

End if

I = I+1

End while

T = T * α

End while

Output:

Return S_b

Figure 4.17: SA-Based algorithm pseudocode.

4.4 SUMMARY

This chapter has presented the data structure of the graph representation and network configuration using matrix and vectors. In addition, the detailed discussion about the proposed algorithms has been done in this chapter. The first algorithm based on the GA and the second based on the SA. Furthermore, the chapter has presented an illustration about how these algorithms work to solve the research problem (GPP). The illustration covered the representation of the network in the encoding stage, which they used by the proposed algorithms to simulate the real network as well as the corresponding network configuration in the decoding stage (Graph's representation). The next chapter will present and discuss the numerical results of the proposed algorithms and a comparison between these algorithms as well.

CHAPTER FIVE
RESULTS ANALYSIS AND EVALUATION

This chapter presents the numerical results of the proposed algorithms (GA and SA), a comparison between the two algorithms and deep insights in each algorithm to present further optimization opportunities in different situations and parameters in the algorithms. According to the best of our knowledge, currently there is no algorithm can be found in the literature use the same parameters to solve the GPP. Therefore, we have tested only our algorithms in the experiments and a comparison between them has been done as well. We have tested the algorithms using 25 MRs and 4 IGs. We run the application on Hewlett-Packard HP 2000 Notebook PC with Intel core-i3 2.40GHz processor and 4.00GB of RAM and tested under Windows 7 (32-bit) operating system. Microsoft Visual Studio 2010 (Visual Basic) has been used as a programming language to implement the proposed algorithms. Two different applications have developed for each algorithm, the first one uses the windows application, which based on graphical user interface for ease of use, and the second one based on console application for less memory consumption that is required by the graphical user interface. Finally, this chapter presents a discussion about the two applications' development environment.

5.1 THE EVALUATION OF THE GA-BASED ALGORITHM

The population evolution has been investigated to demonstrate the effectiveness of the GA. In order to prove the correct functionality of the GA the growth of the fitness must be observed over the new generations, which are produced by the genetic operators. Thus, the effects of these operators have been investigated to show the effectiveness of the GA. Therefore, we evaluated the algorithm using different conditions such as the effects of population size, crossover type, mutation type and the tournament size. The evaluation process is aimed to show the convergence rate of the proposed algorithm as well as showing the robustness and the scalability of the proposed algorithm, which may be used in high intensity situation. Moreover, the evaluation also aimed to show the significance of the objective function on solving the GPP besides how it is different from the existing research works that aimed to minimize the AVG-MR-IG-Hop to

enhance the network performance. The following subsections present and discuss the generated results.

Figure 5.1 shows the generated result using population size of 100, two-point crossover type, inversion mutation and tournament size of 10. The number of MRs is 100 with 10 IGs and we ran the algorithm for four rounds for only ten generations per each round, the figure below shows that the algorithm always get good results, and this indicates that the algorithm will get better result in the next generations. In addition, the results show a positive convergence rate at each round.

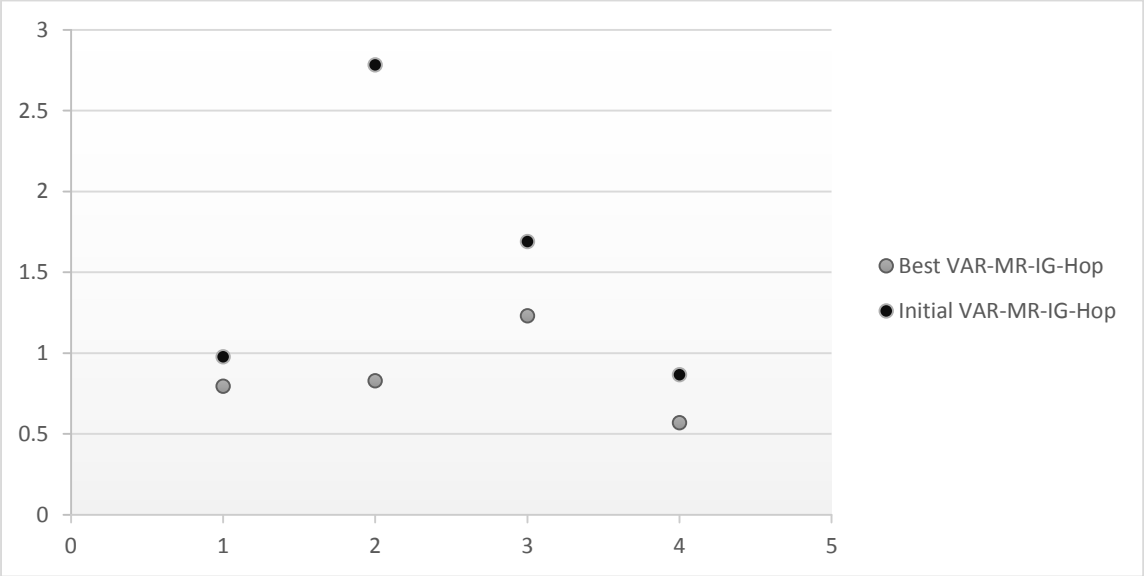

Figure 5.1: Initial GA results for four rounds and ten generations

5.1.1 The effect of population size on convergence rate

The effect of population size on the convergence rate has been studied, and the algorithm parameters that were used in the experiments as shown in Table 5.1.

Table 5.1: Parameters used to evaluate the effect of population size

Parameter	Value
Crossover type	2-points
Mutation type	Inversion
Tournament size	10
Number of MRs	25
Number of Gateways	4

Population sizes	50,100,150
Maximum generation No.	6000

Figure 5.2 shows the VAR-MR-IG-Hop of the best solution of the initial population and the current generation when the population size is 100. Figure 5.3 shows the convergence rate using different population sizes: 50,100,150 and the results show that the algorithm has a good convergence rate at different population size but the best population size with the parameter values shown in Table 5.1 is 100, which shows the lowest variation (highest convergence rate).

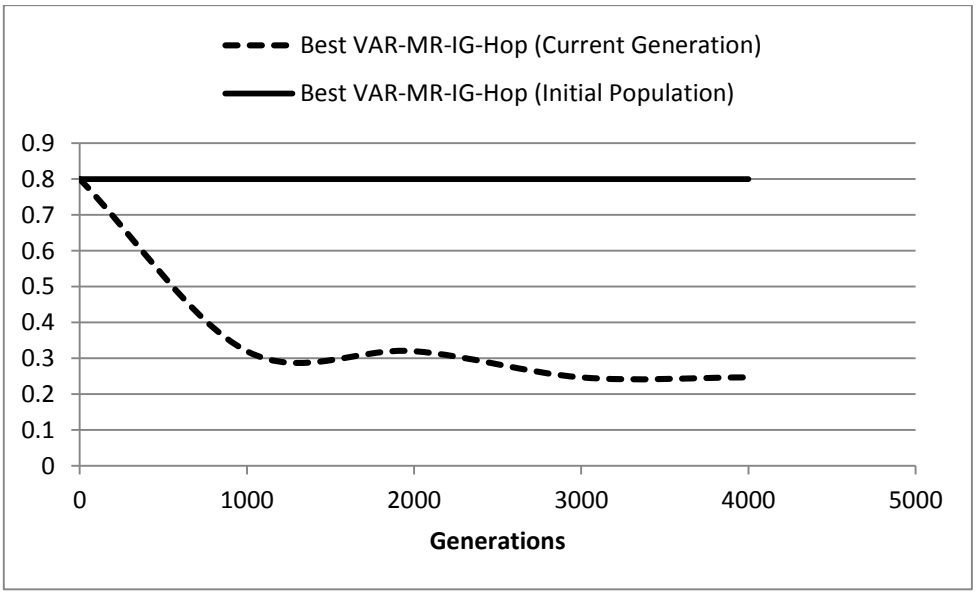

Figure 5.2: VAR-MR-IG-Hop when the population size=100

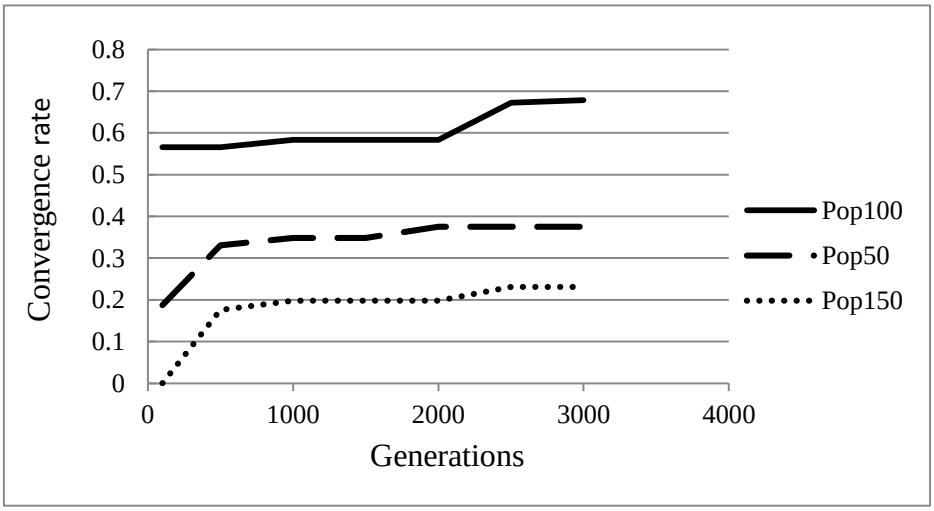

Figure 5.3: Convergence rate using different population sizes

The algorithm has been ran three times using the population sizes 50,100,150 and the fitness values of each instance have been compared at 3000 generations. The result, as shown in Figure 5.4.

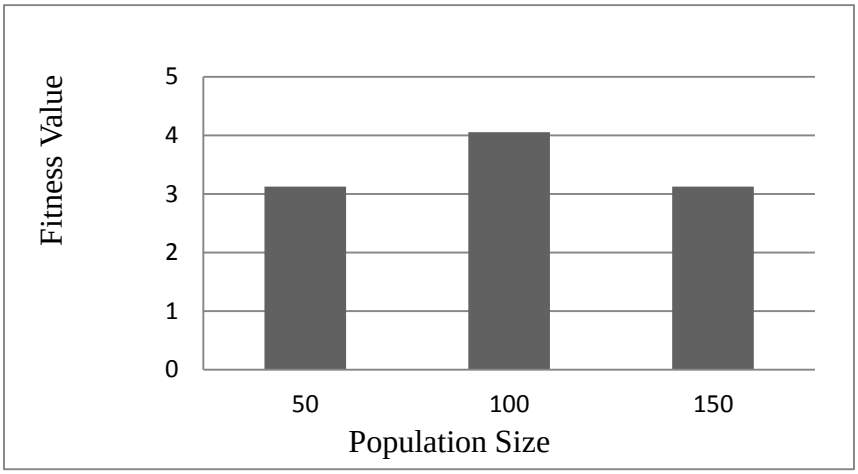

Figure 5.4: Fitness values at the 3000 generation

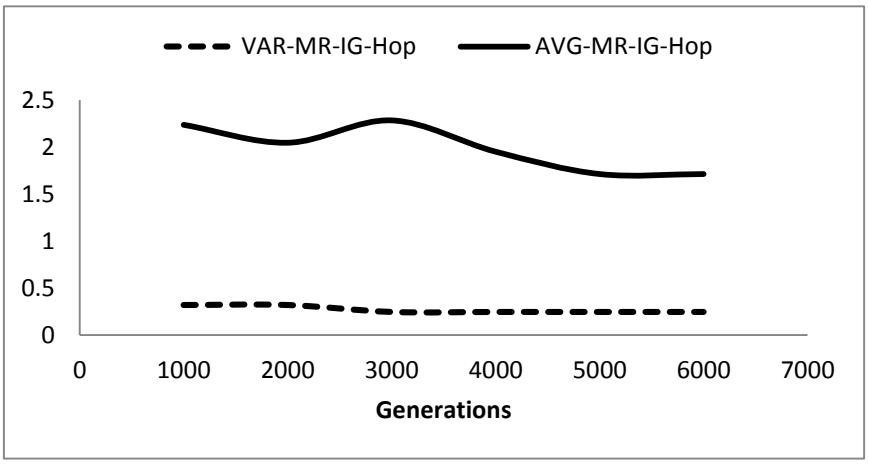

Figure 5.5: AVG-MR-IG-Hop and VAR-MR-IG-Hop when the population size=100

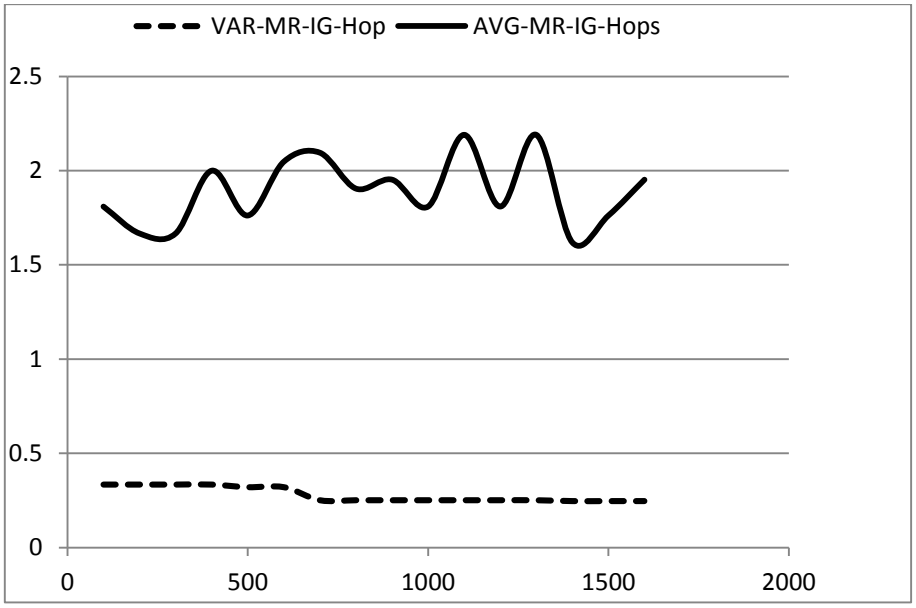

Figure 5.6: The relationship between AVG-MR-IG-Hop and VAR-MR-IG-Hop

Figure 5.5 and Figure 5.6 show the relationship between the AVG-MR-IG-Hop and VAR-MR-IG-Hop counts. Other algorithms for IG placement aimed to minimize the AVG-MR-IG-Hop in order to minimize the bandwidth consumption, delay, the transmission time and maximize the network throughput. However, depending only on AVG-MR-IG-Hop does not guarantee that the all MRs positioned in near equal distance from their IGs. From the result shown in Figure 5.5 while the AVG-MR-IG-Hop keeps decreasing through the generations, the VAR-MR-IG-Hop has a constant value for a long period with different VAR-MR-IG-Hop values. The result presented in Figure 5.6 shows that the stability of the VAR-MR-IG-Hop value and the variety of AVG-MR-IG-Hop. This result supports the philosophy behind using the VAR-MR-IG-Hop in our algorithms rather than depending on AVG-MR-IG-Hop.

5.1.2 The effect of tournament size on the convergence rate

The algorithm has been tested using different tournament sizes and the other GA's parameters as shown in Table 5.2.

Table 5.2: Parameter' settings to evaluate the effect of tournament size

Parameter	Value
Crossover type	2-points
Mutation type	Inversion

Tournament size	6,10,15,20,30
Number of MRs	25
Number of Gateways	4
Population size	100
Maximum generation No.	6000

Figure 5.7 shows the convergence rates of four instances of the algorithm for a number of generations using different tournament sizes shown in Table 5.2. The results show the highest convergence rate at 30-tournament size for 2000 generations, which means 30% of the population.

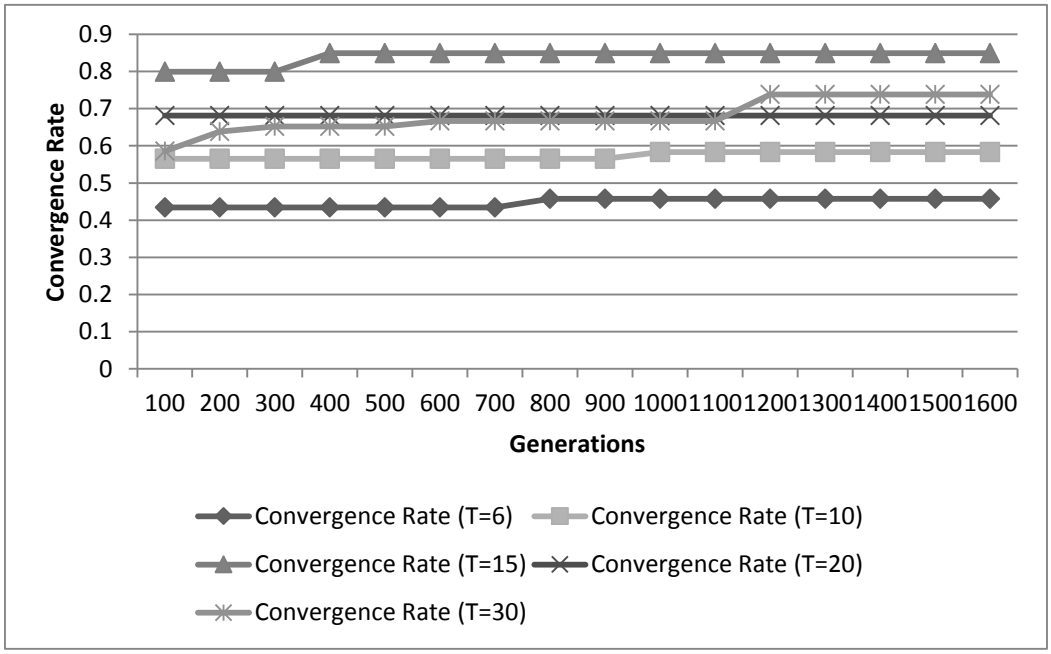

Figure 5.7: Convergence Rate using different tournament sizes

5.1.3 The effect of crossover type on the convergence rate

From the previous results the best population size is 100 and the best tournament size, is 30% of the population where the total number of MRs is 25 (individual size) with four IGs. The previous tests have been done using the 2-point crossover. In the following section presents the effect of using different crossover types and the other GA's parameters as shown in Table 5.3.

Table 5.3: The parameter settings to evaluate the effect of crossover type

Parameter	Value
Crossover type	Single point, 2-points, Uniform
Mutation type	Inversion
Tournament size	30
Number of MRs	25
Number of Gateways	4
Population size	100
Maximum generation No.	1600

Table 5.4: Effect of different crossover types

	Single Point crossover		**2-Points Crossover**		**Uniform Crossover**	
Generation No.	Convergence Rate	AVG-MR-IG-Hop	Convergence Rate	AVG-MR-IG-Hop	Convergence Rate	AVG-MR-IG-Hop
100	0.5805	1.8095	0.5857	1.8095	0.3482	2.0476
200	0.5805	1.6667	0.6381	1.8571	0.3482	2.8571
300	0.5805	1.6667	0.6524	2.0000	0.3482	1.8095
400	0.5805	2.0000	0.6524	1.9524	0.3482	2.2381
500	0.5977	1.7619	0.6524	2.4762	0.3482	1.9048
600	0.5977	2.0476	0.6667	2.2381	0.3482	1.9524
700	0.6839	2.0952	0.6667	1.9524	0.3482	2.4286
800	0.6839	1.9048	0.6667	1.9524	0.3482	1.7619
900	0.6839	1.9524	0.6667	2.0952	0.3482	1.8571
1000	0.6839	1.8095	0.6667	2.1429	0.3482	1.8571
1100	0.6839	2.1905	0.6667	2.0952	0.3482	1.9524
1200	0.6839	1.8095	0.7381	1.7619	0.3482	2.2381
1300	0.6839	2.1905	0.7381	1.8571	0.3482	1.9048
1400	0.6897	1.6190	0.7381	1.7619	0.3482	2.3810
1500	0.6897	1.7619	0.7381	2.0952	0.3482	2.1905
1600	0.6897	1.9524	0.7381	2.1905	0.3482	1.8571

Table 5.4 shows the convergence rate of the proposed GA algorithm and AVG-MR-IG-Hops. The results show that, the highest convergence rate with 2-points crossover through generations. Figure 5.8 shows the convergence rate of the three crossover types.

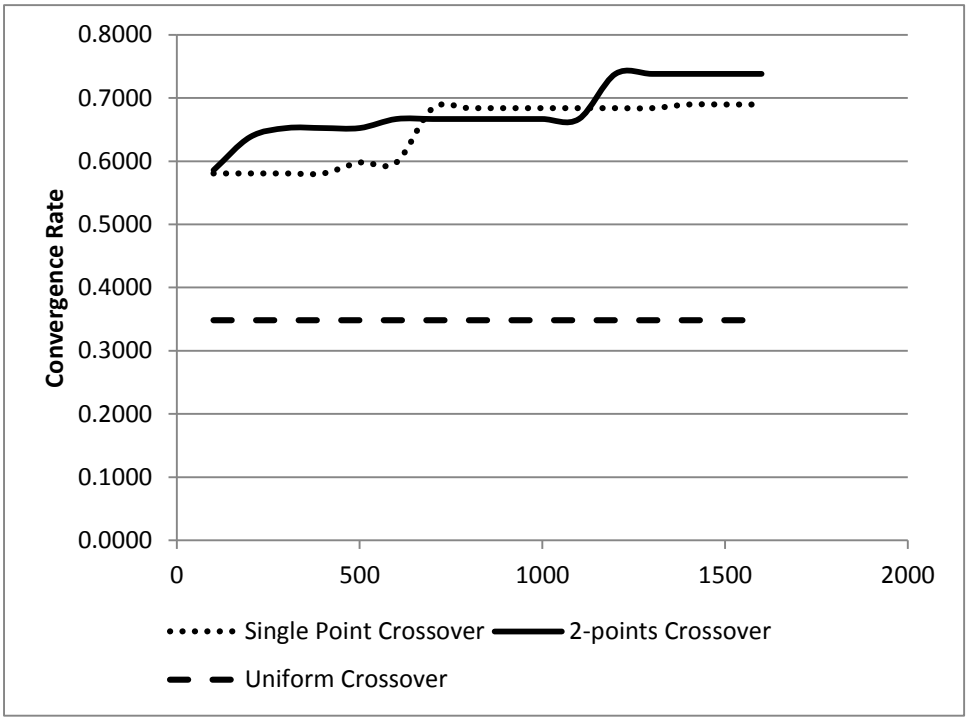

Figure 5.8: Convergence rate of the three crossover types

5.2 THE EVALUATION OF THE SA-BASED ALGORITHM

The algorithm has been ran using different values of internal temperature, iterated the execution of the algorithm ten times for each temperature value and then the best solution among them has been chosen. Table 5.5 describes the parameters used in the experiments. Figure 5.9: VAR-MR-IG-Hop and Internal Temperature shows the VAR-MR-IG-Hop of the best solutions.

Table 5.5: The Parameter' settings of the experiment

Parameter	Value
Alpha	0.99
Final Temperature	0.01
Internal Temperature	100, 200, 300, 400, 500, 600, 700, 800, 900, 1000
Number of MRs	25
Number of IGs	4

The experiments have shown that, the best results in internal temperature values of 200 and 500. However, in the optimization the time is very important. Thus, we can say that, the best result is at internal temperature 200 because increasing the internal temperature will increase the execution time. Figure 5.10: Execution time at different values of internal temperature shows the execution time for at the different values of the internal temperature at the best iteration among the ten iterations.

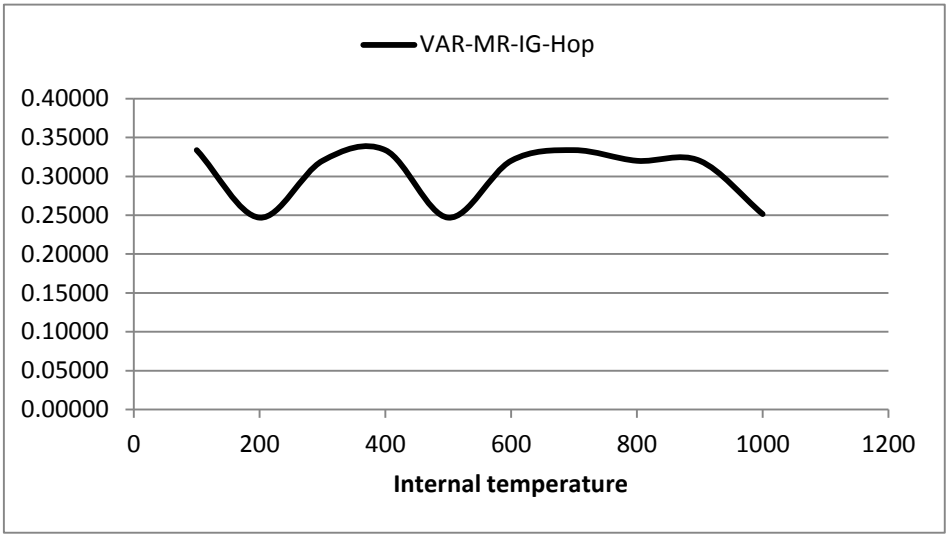

Figure 5.9: VAR-MR-IG-Hop and Internal Temperature

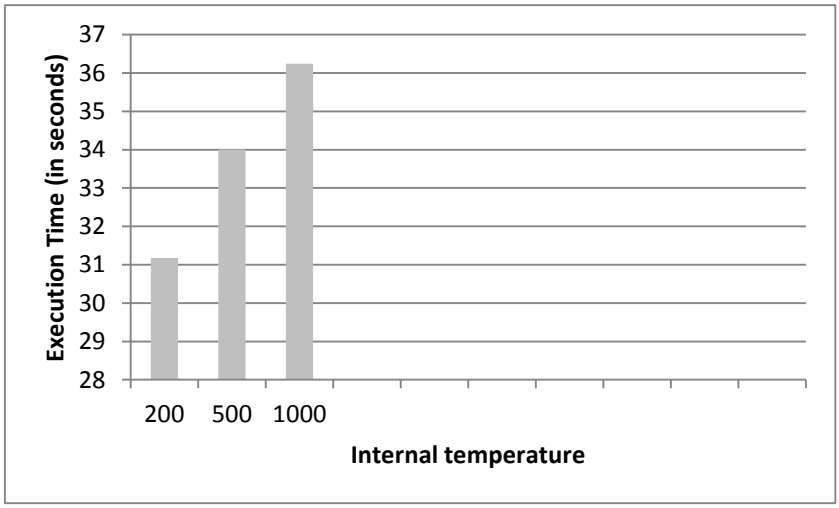

Figure 5.10: Execution time at different values of internal temperature

5.3 SUMMARY

This chapter has presented the evaluation of the proposed algorithms (the GA-Based and SA-Based algorithms). The GA-Based algorithm has been evaluated using different parameters (genetic operators) and have shown their usefulness to solve the GPP at

different situations. The SA-Based algorithm has been evaluated using different value for the algorithm's operators, especially the internal temperature, which has considerable effect on the algorithm execution time. However, both of the two algorithms have ability to deliver an acceptable solution for the GPP.

CHAPTER SIX
CONCLUSION AND FUTURE RECOMMENDATIONS

6.1 CONCLUSION

In this research, the GPP in WMN has been studied and addressed; a novel solution has been proposed to solve this problem. Two algorithms have been developed based on GA and SA to find the near optimal solution for the GPP. The proposed approach aimed to minimize the VAR-MR-IG-Hop count among MRs in the network to insure that each MRs were placed in a near optimal position from their nearest IG as well as to insure the MRs were distributed equally among the IGs. The problem has been formulated as a mathematical model and the network was represented as an undirected graph of a one-unit weights. The Dijkstra's algorithm has been used to calculate the shortest path between the MRs and the IGs. The GA and the SA have been used to find the near optimal solution based on the objective functions in the mathematical model. The two algorithms have been evaluated based on generating instances to show the convergence rate, the scalability, and the robustness of the algorithms. The experimental results have shown good results for both algorithms. Further optimization has been done for both algorithms using different parameters that formed, the processes these algorithms and size of the networks to test the algorithms in high and low intensity situations. For the GA the parameters that were considered in the test are population size, tournament size and crossover type. For the SA the parameters that were considered in the experiments are internal and final temperature besides the probability and the parameters that were used in the transition function. The results have shown that the algorithms could achieve good results in different situations in high intensity and low intensity. Hence, the algorithms have considerable scalability and robustness to solve the GPP in large and small networks. Moreover, the results have shown the positive significance of VAR-MR-IG-hop in comparison with the AVG-MR-IG-hop on enhancing the network performance by exploiting the resources in a way that cloud insures the load balance among MRs. Finally, a comparison between GA and SA have been done. The results have shown the GA achieved better than SA in the small-size networks and it has better opportunities for further optimization through many generations. Nevertheless, the SA

can achieve better in large-size networks in small execution time and it is better than GA when the time is an important issue but less quality in comparison with the GA has.

6.2 DISSERTATION CONTRIBUTION

This dissertation aimed to optimize the gateway placement in WMN. A new solution has been developed to solve the GPP. The problem has been formulated as undirected weighted and connected graph and guided by a mathematical model. Two algorithms have been developed to find a near optimal solution for the research problem, the first algorithm based on genetic algorithm where the second based on simulated annealing algorithm. The evaluation results showed that, both of the algorithms have good results.

The following conference papers are presented at international conferences, these papers are entitled as follows:

- Gateway Placement Approaches in Wireless Mesh Network: Study Survey, IEEE International Conference – Khartoum Aug 2013.
- Investigation of Gateway Placement Approaches in Wireless Mesh Networks using Genetic Algorithms, ICCCE2014, IIUM, Kuala Lumpur, Sept 2014 (published in IEEE explore).

Two journal papers have been published in international journal, these papers are entitled as follows:

- Metaheuristic Approaches for Gateway Placement Optimization in Wireless Mesh Networks: A Survey, International Journal of Computer Science and Network Security, 2014
- A Genetic Approach for Gateway Placement in Wireless Mesh Networks, International Journal of Computer Science and Network Security, 2015

6.3 FUTURE WORKS

For further enhancement, more objectives can be added to the existing solution's objectives such as the network throughput or any performance metrics.

REFERENCES

[1] Wu, Wenjia, Junzhou Luo, and Ming Yang, "Gateway placement optimization for load balancing in wireless mesh networks," in *Computer Supported Cooperative Work in Design. CSCWD 2009. 13th International Conference on*, 2009.

[2] Li, Fan, Yu Wang, and Xiang-Yang Li, "Gateway placement for throughput optimization in wireless mesh networks," in *Communications. ICC'07. IEEE International Conference on.*, 2007.

[3] Awadallah M Ahmed, Aisha Hassan A Hashim, "A Genetic Approach for Gateway Placement in Wireless Mesh Networks," *International Journal of Computer Science and Network Security (IJCSNS)*, vol. 15, no. 7, p. 11, 2015.

[4] Talay, Ahmet Cagatay, "A gateway access-point selection problem and traffic balancing in wireless mesh networks," *Applications of Evolutionary Computing. Springer Berlin Heidelberg*, pp. 161-168, 2007.

[5] Moheb R. Girgis, Tarek M., Bahgat A., Ahmed M. Rabie, "Solving the Wireless Mesh Network Design Problem using Genetic Algorithm and Simulated Annealing Optimization Methods," *International Journal of Computer Applications*, vol. 96, no. 11, p. 0975 –8887, June 2014.

[6] Awadallah M. Ahmed, Aisha Hassan A. Hashim, "Metaheuristic Approaches for Gateway Placement Optimization in Wireless Mesh Networks: A survey," *IJCSNS International Journal of Computer Science and Network Security*, vol. 14, no. 12, December 2014.

[7] Awadallah M. Ahmed, Aisha Hassan. A. Hashim, and Wan Haslina Hassan, "Investigation of Gateway Placement Optimization Approaches in Wireless Mesh Networks using Genetic Algorithms," in *Computer and Communication Engineering (ICCCE), 2014 International Conference on*, Kuala Lumpur, Malaysia, 2014.

[8] Yang Zhang,Jijun luo, Honglin Hu, Wireless Mesh Networking: Architectures, Protocols and Standards (Wireless Networks and Mobile Communications), 2006.

[9] Huyao Dac-Nhuong Le, Nhu Gia Nguyen, Nghia Huu Dinh, Nguyen Dang Le, and Vinh Trong Le, "Optimizing Gateway Placement in Wireless Mesh Networks based on ACO Algorithm," *International Journal of Computer and Communication Engineering,* vol. 2, no. 2, pp. 143--147, 2013.

[10] Oda, Tetsuya, et al., "Effects of population size for location-aware node placement in WMNs: evaluation by a genetic algorithm--based approach," *Personal and ubiquitous computing,* no. 18.2, pp. 261-269, 2014.

[11] Barolli, Admir, et al., "Performance Evaluation of WMN-GA System for Node Placement in WMNs Considering Exponential and Weibull Distribution of Mesh Clients and Different Selection and Mutation Operators," in *Complex, Intelligent, and Software Intensive SysteSystems (CISIS), 2013 Seventh International Conference on. IEEE,* 2013.

[12] Jun, Jangeun, and Mihail L. Sichitiu, "The nominal capacity of wireless mesh networks," *Wireless Communications, IEEE*, vol. 10.5, pp. 8-14, 2003.

[13] R. Draves, J. Padhye, and B. Zill, "Routing in multi-radio, multi-hop wireless mesh networks," in *ACM International Conference on Mobile Computing and Networking (MobiCom)*, 2004.

[14] A. Raniwala and T. Chiueh, "Architecture and algorithms for an IEEE 802.11-based multi-channel wireless mesh network," in *IIEEE Conference on Computer Communications (INFOCOM)*, 2005.

[15] Das, Banani and Roy, Sudipta, "Load Balancing Techniques for Wireless Mesh Networks: A Survey," in *Computational and Business Intelligence (ISCBI), 2013 International Symposium on*, 2013.

[16] Akyildiz, Ian F., Xudong Wang, and Weilin Wang, "Wireless mesh networks: a survey," *Computer networks* , vol. 47.4 , pp. 445-487, (2005).

[17] Akyildiz, I.F. and Xudong Wang, "A survey on wireless mesh networks," *Communications Magazine, IEEE,* vol. 43, no. 9, pp. S23-S30, Sept 2005.

[18] Oda, Tetsuya, et al., "A Comparison Study of GA and HC for Mesh Router Node Placement in Wireless Mesh Networks," in *Network-Based Information Systems (NBiS), 2013 16th International Conference on. IEEE*, 2013.

[19] Srivastava, Smriti, Anant Kumar Jaiswal, and Paramjeet Rawat., "Gateway Placement Approaches: A Survey," *International Journal of Engineering and Innovative Technology* , no. 1.4, pp. 306-309, 2012.

[20] Ahmed, Awadallah M., Aisha H. Abdalla, and Ismail El-Azhary, "Gateway placement approaches in Wireless Mesh Network: Study survey," in *Computing, Electrical and Electronics Engineering (ICCEEE), 2013 International Conference on. IEEE*, 2013.

[21] Keijo Ruohonen (Translation by Janne Tamminen, Kung-Chung Lee and Robert Piché), GRAPH THEORY, 2013.

[22] Joyner, David and Van Nguyen, Minh and Phillips, David, Algorithmic Graph Theory and Sage, Version 0.8-r1991, 2013, p. 304.

[23] Bondy, John Adrian and Murty, Uppaluri Siva Ramachandra, Graph theory with applications, vol. 6, Macmillan London, 1976.

[24] David Joyner, Minh Van Nguyen, Nathann Cohen, Algorithmic Graph Theory, Version 0.7-r1843, 2011.

[25] Schrijver, Alexander, Combinatorial Optimization. Algorithms and Combinatorics, Polyhedra and efficiency, vol. 24, Springer, 2004.

[26] Dijkstra, Edsger W, "A note on two problems in connexion with graphs," *Numerische mathematik*, vol. 1, no. 1, pp. 269--271, 1959.

[27] "www.businessdictionary.com," [Online]. Available: http://www.businessdictionary.com/definition/optimization.html. [Accessed 26 January 2015].

[28] Richard E. Neapolitan, Kumarss Naimipour, Foundations of Algorithms Using Java Pseudocode, Mississauga, Lonadon: Jones and Bartlett Publishers (Canada, international-UK), 2004.

[29] Boyd, Stephen and Vandenberghe, Lieven, Convex optimization, Cambridge university press, 2009.

[30] Xhafa, Fatos and Bravo, Albert and Barolli, Admir and Takizawa, Makoto, "An Interface for Simulating Node Placement in Wireless Mesh Networks," in *Network-Based Information Systems (NBiS), 2012 15th International Conference on*, 2012.

[31] Korte, Bernhard, and Jens Vygen, "Combinatorial optimization. Algorithms and Combinatorics," vol. 21, pp. 434-443, 2000.

[32] Lawler, Eugene L, Combinatorial optimization: networks and matroids, Courier Dover Publications, 1976.

[33] Bernhard Korte, Jens Vygen, Combinatorial optimization: theory and algorithms, 2 ed., Springer, 2002.

[34] Nikolaev, N. Y. and Iba, H., Adaptive Learining of Polynomial Networks Genetic Programming, Backpropagation and Bayesian Methods, New York: Springer, 2006.

[35] Hitoshi Iba, Nasimul Noman, New Frontier in Evolutionary Algorithms: Theory and Applications, London: Imperial College Press, 2012.

[36] Dianati, Mehrdad and Song, Insop and Treiber, Mark, "An introduction to genetic algorithms and evolution strategies," Technical report, University of Waterloo, Ontario, N2L 3G1, Canada, 2002.

[37] Streichert, Felix, "Introduction to Evolutionary Algorithms," *paper to be presented Apr*, vol. 4, 2002.

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

[39] Abdullah, B and Abd-Alghafar, I and Salama, Gouda I and Abd-Alhafez, A, "Performance evaluation of a genetic algorithm based approach to network intrusion detection system," in *13th international conference on aerospace sciences and aviation technology, Military Technical College, Kobry Elkobbah, Cairo, Egypt*, 2009.

[40] Paszkowicz, Wojciech, "Genetic algorithms, a nature-inspired tool: Survey of applications in materials science and related fields," *Materials and Manufacturing Processes*, vol. 24, no. 2, pp. 174--197, 2009.

[41] Blum, Christian and Roli, Andrea, "Metaheuristics in combinatorial optimization: Overview and conceptual comparison," *ACM Computing Surveys (CSUR)*, vol. 35, no. 3, pp. 268--308, 2003.

[42] Glover, Fred and Kochenberger, Gary A, Handbook of Metaheuristics, Kluwer Academic Publishers, Springer Science \& Business Media, 2003.

[43] Hoos, Holger H and St{\"u}tzle, Thomas, Stochastic Local Search: Foundations and Applications, Elsevier, 2004.

[44] Dorigo, Marco, and Christian Blum, "Ant colony optimization theory: A survey.," *Theoretical computer science*, vol. 344, no. 2, pp. 243-278, 2005.

[45] R. a. M. T. a. O. J. Ahuja, *Networks flows*, Prentice Hall, Englewood Cliffs, New Jersey, 1993.

[46] Gen, Mitsuo and Cheng, Runwei and Lin, Lin, Network models and optimization: Multiobjective genetic algorithm approach, Springer Science \& Business Media, 2008, pp. 1--47.

[47] He, Bing, Bin Xie, and Dharma P. Agrawal, "Optimizing deployment of internet gateway in wireless mesh networks," *Computer Communications*, no. 31.7, pp. 1259-1275, 2008.

[48] Hu, Jie., "Gateway Node Selection for Improving Traffic Delivery Ratio in Wireless Mesh Networks," in *International Conference on Computer, Networks and Communication Engineering (ICCNCE 2013). Atlantis Press*, 2013.

[49] B. Aoun, R. Boutaba, Y. Iraqi, and G. Kenward, "Gateway placement optimization in wireless mesh networks with QoS constraints," *Journal on Selected Areas in Communications*, vol. 24, no. 11, p. 2127– 2136, November 2006.

[50] J. Wong, R. Jafari, and M. Potkonjak, "Gateway placement for latency and energy efficient data aggregation," in *in Proceedings IEEE LCN*, 2004.

[51] R. Chandra, L. Qiu, K. Jain, and M. Mahdian, "Optimizing the placement of Internet TAPs in wireless neighborhood networks," in *in Proceedings IEEE ICNP*, 2004.

[52] Y. Bejerano, "Efficient integration of multihop wireless and wired networks with QoS constraints," *IEEE/ACM Transactions on Networking*, vol. 12, no. 6, p. 1064–1078, December 2004.

[53] Zeng, Feng, and Zhigang Chen., "Load balancing placement of gateways in wireless mesh networks with QoS constraints," in *Young Computer Scientists, 2008. ICYCS 2008. The 9th International Conference for. IEEE*, 2008.

[54] Smadi, Mohammed N., et al., "Gateway placement in wireless mesh networks using free space optical links," *Computer Communications and Networks, 2008. ICCCN'08. Proceedings of 17th International Conference on. IEEE*, 2008.

[55] Pries, Rastin, et al. , "A genetic approach for wireless mesh network planning and optimization," in *Proceedings of the 2009 International Conference on Wireless Communications and Mobile Computing: Connecting the World Wirelessly. ACM*, 2009.

[56] Pries, Rastin, et al., "Genetic algorithms for wireless mesh network planning," in *Proceedings of the 13th ACM international conference on Modeling, analysis, and simulation of wireless and mobile systems. ACM*, 2010.

[57] Lin, Ting-Yu, Kai-Chiuan Hsieh, and Hsin-Chun Huang., "Applying genetic algorithms for multiradio wireless mesh network planning," *Vehicular Technology, IEEE Transactions on*, no. 61.5, pp. 2256-2270, 2012.

[58] Karnik, Aditya, Aravind Iyer, and Catherine Rosenberg, "Throughput-optimal configuration of fixed wireless networks," *ACM Transactions on Networking (TON)*, vol. 16.5 , pp. 1161-1174, 2008.

[59] Ge, Zhi-hui, and Tao-shen Li., "A novel gateway load-balance algorithm for wireless mesh network," *The Journal of China Universities of Posts and Telecommunications* , vol. 18, pp. 75-78, 2011.

[60] Maolin, T. A. N. G, "Gateways placement in backbone wireless mesh networks," *Int'l J. of Communications, Network and System Sciences 2.1*, pp. 44-50, 2009.

[61] Aljober, Mijahed Nasser, and R. C. Thool., "Multi-Objective Particle Swarm Optimization for Multicast Load Balancing in Wireless Mesh Networks".

[62] Le, Dac-Nhuong, and Nhu Gia Nguyen, "A New Evolutionary Approach for Gateway Placement in Wireless Mesh Networks," *International Journal of*

Computer Networks and Wireless Communications (IJCNWC) , no. 2.5, pp. 550-555, 2012.

[63] Ping, Zhou, Wang Xudong, and Rao Ramesh., "On optimizing gateway placement for throughput in wireless mesh networks," *EURASIP Journal on Wireless Communications and Networking*, 2010.

[64] Le, Dac-Nhuong., "A Comparatives Study of Gateway Placement Optimization in Wireless Mesh Network using GA, PSO and ACO," *International Journal of Information and Network Security (IJINS)* , no. 2.4, pp. 292-304, 2013.

[65] Xhafa, Fatos, Christian Sánchez, and Leonard Barolli., "Genetic algorithms for efficient placement of router nodes in wireless mesh networks," in *Advanced Information Networking and Applications (AINA), 2010 24th IEEE International Conference on. IEEE*, 2010.

[66] Xhafa, Fatos, et al., "Evaluation of genetic algorithms for mesh router nodes placement in wireless mesh networks," *Journal of Ambient Intelligence and Humanized Computing*, no. 1.4, pp. 271-282, 2010.

[67] Hsu, Chun-Yen, et al., "Survivable and delay-guaranteed backbone wireless mesh network design," *Journal of Parallel and Distributed Computing*, no. 68.3, pp. 306-320, 2008.

[68] Muthaiah, Skanda N., and C. Rosenberg, "Single gateway placement in wireless mesh networks," in *Proceedings of 8th international IEEE symposium on computer networks, Turkey*, 2008.

[69] R. Chandra, L. Qiu, K. Jain, M. Mahdian, "Optimizing the placement of integration points in multi-hop wireless networks," in *Proceedings of IEEE ICNP*, Berlin, 2004.

[70] Gumel, M. I., N. Faruk, and A. A. Ayeni, "Investigation and Addressing Unfairness in Wireless Mesh Networks," *Journal of Emerging Trends in Computing and Information Sciences 2.*

[71] He, Bing and Xie, Bin and Agrawal, Dharma P, "Optimizing the internet gateway deployment in a wireless mesh network," in *Mobile Adhoc and Sensor Systems, 2007. MASS 2007. IEEE Internatonal Conference on*, 2007.

[72] Sivanandam, SN and Deepa, SN, Introduction to genetic algorithms, Springer Science \& Business Media, 2007.

[73] Jun, Peng, and Zhou QiangQiang, "Gateways Placement Optimization in Wireless Mesh Networks," in *Networking and Digital Society. ICNDS'09. International Conference on*, 2009.

[74] Ding, Jingzhi, Jianxiao Xu, and Zhifeng Zheng, "Gateway Deployment Optimization in Wireless Mesh Network: A Case Study in China," in *Service Operations, Logistics and Informatics. SOLI'09. IEEE/INFORMS International Conference on.* , 2009.

[75] Johnson, David S and Garey, Michael R, Computers and intractability: A guide to the theory of NP-completeness, New York: NY, USA: W. H. Freeman & Co, 1990.

[76] Waharte S., Boutaba R., Iraqi Y., Ishibashi B., "Routing protocols in Wireless Mesh Networks: Challenges and design considerations," *Springer,Multimedia Tools Applications.,* pp. 285-303, 2006.

[77] Lei W., Landfeldt B., "The problem of Placing Mobility Anchor points in Wireless Mesh Networks," in *In proceedings of 6th ACM International Symposium on Mobility Management and Wireless access, MobiWac'08.*, 2008.

[78] Xhafa, Fatos, et al., "A simulated annealing algorithm for router nodes placement problem in Wireless Mesh Networks," *Simulation Modelling Practice and Theory,* no. 19.10, pp. 2276-2284, 2011.

[79] Lin, Chun-Cheng., "Dynamic router node placement in wireless mesh networks: A PSO approach with constriction coefficient and its convergence analysis," *Information Sciences* , no. 232, pp. 294-308, 2013.

[80] Neumann, Frank and Witt, Carsten, "Combinatorial Optimization and Computational Complexity," in *Bioinspired Computation in Combinatorial Optimization*, Berlin Heidelberg, Springer, 2010, pp. 9-19.

[81] Hansen, Nikolaus and Arnold, Dirk V and Auger, Anne, "Evolution Strategies," *Handbook of Computational Intelligence. Springer,* 2013.

[82] Cornuejols, Gerard and T{\"u}t{\"u}nc{\"u}, Reha}, Optimization Methods in Finance, Citeseer, 2005.

[83] Abdullah, B and Abd-Alghafar, I and Salama, Gouda I and Abd-Alhafez, A, "Performance evaluation of a genetic algorithm based approach to network intrusion detection system," in *13th international conference on aerospace sciences and aviation technology, Military Technical College, Kobry Elkobbah, Cairo, Egypt*, 2009.

[84] Blum, Christian and Roli, Andrea, "Metaheuristics in combinatorial optimization: Overview and conceptual comparison," *ACM Computing Surveys (CSUR)*, vol. 35, no. 3, pp. 268--308, 2003.

[85] Yagiura, Mutsunori and Ibaraki, Toshihide, "On metaheuristic algorithms for combinatorial optimization problems," *Systems and Computers in Japan*, vol. 32, no. 3, pp. 33--55, 2001.

[86] Chih-Hao Lin and Pei-Ling Lin, "A New Non-dominated Sorting Genetic Algorithm for Multi-Objective Optimization," in *Modeling Simulation and Optimization - Focus on Applications, ISBN: 978-953-307-055-1*, S. C. (Ed.), Ed., InTech, 2010.

[87] Angelova, Maria and Pencheva, Tania, "Tuning genetic algorithm parameters to improve convergence time," *International Journal of Chemical Engineering*, 2011.

[88] Grotschel, M., and L. Lovász., Combinatorial optimization. "Handbook of combinatorics 2", 1995, pp. 1541-1597.

[89] Champion, Brett and Strzebonski, Adam, "Constrained optimization," *USA. Champain: Wofram Research Inc,* 2008.

[90] Kumar, Rajeev and Rockett, Peter, "Improved sampling of the Pareto-front in multiobjective genetic optimizations by steady-state evolution: a Pareto converging genetic algorithm," *Evolutionary computation*, vol. 10, no. 3, pp. 283--314, 2002.

[91] Goldberg, David E and Deb, Kalyanmoy, "A comparative analysis of selection schemes used in genetic algorithms," *Urbana*, vol. 51, pp. 61801--2996, 1991.

[92] Das, Banani and Roy, Amit Kumar and Khan, Ajoy Kumar and Roy, Sudipta, "Gateway-level Load Balancing Techniques for WMN: A Comparative Study,"

in *Communication Systems and Network Technologies (CSNT), 2014 Fourth International Conference on*, 2014.

[93] Liu, Chun-yan and Fu, Bo and Huang, He-Jiao, "Delay minimization and priority scheduling in wireless mesh Networks," *Wireless Networks*, vol. 20, no. 7, pp. 1955--1965, 2014.

[94] Mountassir, Tarik and Nassereddine, Bouchaib and Haqiq, Abdelkrim and Bennani, Samir, "Wireless Mesh Networks Topology Auto Planning," *International Journal of Computer Applications*, vol. 52, no. 2, pp. 27--33, 2012.

[95] Le, Dac-Nhuong and Nguyen, Nhu Gia and Le, Nguyen Dang and Dinh, Nghia Huu and Le, Vinh Trong, "ACO and PSO Algorithms Applied to Gateway Placement Optimization in Wireless Mesh Networks," *International Proceedings of Computer Science \& Information Technology*, vol. 57, 2012.

[96] Jahromi, Abdolhamid Eshraghniaye and Rad, Zohreh Besharati, "Optimal topological design of power communication networks using genetic algorithm," *Scientia Iranica*, vol. 20, no. 3, pp. 945--957}, 2013.

[97] M. Barbehenn, "A Note on the Complexity of Dijkstra's Algorithm for Graphs with Weighted Vertices," *IEEE Transactions on Computers*, vol. 47, no. 1, p. 263, 1998.

[98] Javaid, Muhammad Adeel, "Understanding Dijkstra's Algorithm," *Available at SSRN 2340905*, 2013.

[99] Bertsekas, Dimitri P and Scientific, Athena, "Network Optimization: Continuous and Discrete Models," *INTERFACES-PROVIDENCE-INSTITUTE OF MANAGEMENT SCIENCES-*, vol. 28, pp. 73--75, 1998.

[100] Zhang, Xu and Qian, Zhi-Hong and Guo, Yu-Qi and Wang, Xue, "An efficient hop count routing protocol for wireless ad hoc networks," *International Journal of automation and computing*, vol. 11, no. 1, pp. 93--99, 2014.

[101] Vo, Hung Quoc and Hong, Choong Seon, "Hop-count based congestion-aware multi-path routing in wireless mesh network," in *Information Networking, 2008. ICOIN 2008. International Conference on*, 2008.

[102] Tehuang Liu and Wanjiun Liao, "Location-Dependent Throughput and Delay in Wireless Mesh Networks," *Vehicular Technology, IEEE Transactions on*, vol. 57, no. 2, pp. 1188-1198, March 2008.

[103] David F. Barrero, David Camacho, and Mar´.a D. R-Moreno, "A Framework for Agent-Based Evaluation of Genetic Algorithms," in *Intelligent Distributed Computing III*, Berlin, Springer, 2009, pp. 31--41.

[104] Coello, Carlos A Coello and Van Veldhuizen, David A and Lamont, Gary B, Evolutionary algorithms for solving multi-objective problems, vol. 242, Springer, 2002.

[105] Abraham, Ajith and Jain, Lakhmi, Evolutionary multiobjective optimization, Springer, 2005.

[106] Korte, Bernhard, Vygen, Jens, Combinatorial Optimization Theory and Algorithms 5th edition, Springer, 2012.

[107] Caillouet, Christelle, Stéphane Pérennes, and Hervé Rivano., "Framework for optimizing the capacity of wireless mesh networks," *Computer Communications*, vol. 13, no. 34, pp. 1645-1659, 2011.

[108] Govil, Swati and Rawat, Paramjeet, "Comparative Analysis of Gateway Placement Approaches for Wireless Mesh Network," *International Journals of Computer Techniques*, vol. 3, no. 2, 2016.

APPENDIX A

The following visual basic (VB.Net) code defines the population, individual, Fitness calculation (CalcFitness) and the main GA class besides the general module that defines mutation, crossover repair function / procedure.

1. Population Class

```
Public Class Population
Friend individuals() As Individual
Public Sub New(ByVal populationSize As Integer, ByVal initialise As Boolean)
  individuals = New Individual(populationSize - 1) {}
 'Initialise population
 Dim NonePassed As Integer
 If initialise Then
    ' Loop and create individuals
   For i As Integer = 0 To size() - 1
     Dim newIndividual As New Individual()
     'NewIn:
      newIndividual.generateIndividual()
      saveIndividual(i, newIndividual)
      NonePassed = 0
      NonePassed = ChenkGenes(Me.getIndividual(i))
      If NonePassed <> 0 Then
        RepairGenes(Me.getIndividual(i), NonePassed)
      End If
    Next i
 End If
 End Sub
 ' Getters
Public Overridable Function getIndividual(ByVal index As Integer) As Individual
       Return individuals(index)
End Function
Public Overridable ReadOnly Property GetFittest As Individual
  Get
```

```
    Dim fittest As Individual = individuals(0)
    ' Loop through individuals to find fittest
    For i As Integer = 0 To size() - 1
     If fittest.getFitness >= getIndividual(i).getFitness Then
       fittest = getIndividual(i)
     End If
    Next i
    Return fittest
  End Get
End Property
   'Public methods
   'Get population size
Public Overridable Function size() As Integer
      Return individuals.Length
End Function
   'Save individual
Public Overridable Sub saveIndividual(ByVal index As Integer, ByVal indiv As
Individual)
      individuals(index) = indiv
End Sub
End Class
```

2. Individual class

```
Imports System.Math
Imports System.Random
Public Class Individual
   Friend Shared defaultGeneLength As Integer = IndividualSiZe
   Public genes(defaultGeneLength - 1) As Integer
   Dim R As New Random
   'Cache
   Dim fitness As Double = 0
   Friend GetMaxFitness As Object
   'Create a random individual
```

```
    Public Overridable Sub generateIndividual()
        For i As Integer = 0 To size() - 1
          genes(i) = "0"
        Next i
        Dim inx As Integer
        For i = 1 To GatewaysNo
          inx = CInt(R.Next(0, size() - 1))
          genes(inx) = "1"
        Next i
    End Sub
    'Getters and setters
‘Use this if you want to create individuals with different gene lengths
Public Shared WriteOnly Property SetDefaultGeneLength As Integer
    Set(ByVal length As Integer)
          defaultGeneLength = length
        End Set
End Property
Public Overridable Function getGene(ByVal index As Integer) As SByte
    Return genes(index)
End Function
Public Overridable Sub setGene(ByVal index As Integer, ByVal value As
SByte)
    genes(index) = value
    fitness = 0
End Sub
'Public methods
Public Overridable Function size() As Integer
    Return genes.Length
End Function
Public Overridable ReadOnly Property getFitness As Double
    Get
    Dim NewF As New FitnessCalc
     If fitness = 0 Then
```

```
            fitness = NewF.getFitness(Me)
          End If
          Return fitness
        End Get
      End Property
      Public Overrides Function ToString() As String
       Dim geneString As String = ""
       For i As Integer = 0 To size() - 1
        geneString &= getGene(i).ToString
       Next i
       Return geneString
      End Function
     End Class
```

3. Fitness calculation class

```
Imports Microsoft.VisualBasic
Imports System.Math
Public Class FitnessCalc
   Public Function getFitness(ByVal individual As Individual) As Double
      Dim Gateways(GatewaysNo - 1) As Integer
      Dim Nodes(IndividualSiZe - GatewaysNo - 1) As Integer
      Dim x1 As Integer
      Dim x2 As Integer
      x1 = 0
      x2 = 0
      Dim NonePassed As Integer = 0
      NonePassed = ChenkGenes(individual)
      If NonePassed <> 0 Then
         RepairGenes(individual, NonePassed)

      End If
      For i = 0 To individual.size() - 1
         If individual.getGene(i).ToString = "1" Then
            Gateways.SetValue(i + 1, x1)
```

```
            x1 = x1 + 1
        ElseIf individual.getGene(i).ToString = "0" Then
            Nodes.SetValue(i + 1, x2)
            x2 = x2 + 1
        End If
    Next
    " Calculate the standard deviation
    Dim v As Double
    v = GetVariant(Gateways, Nodes)
    Return v
End Function
Private Function GetVariant(ByVal Gateways() As Integer, ByVal Nodes() As
Integer) As Double
 Dim NodeNo As Integer
 Dim NodeGateDistance(Nodes.Length - 1, 1) As Double
 Dim NearGateway As Integer
 Dim NodeDistance As Integer, ShortestDistance As Integer
 Dim NodesAverge As Double
 Dim NodesVar As Double
 Dim NodesCost As Integer
 NodeNo = Nodes.GetUpperBound(0)
 NodesCost = 0
 For i As Integer = 0 To NodeNo - 1
  ' Loop throuth all nodes
  NodeDistance = FrmDrawGraph.GetShortestPath(Nodes(i),Gateways(0))
  NearGateway = Gateways(0)
  For j As Integer = 1 To GatewaysNo - 1
   Application.DoEvents()
  ' Loop through all gateways
  ShortestDistance  =  FrmDrawGraph.GetShortestPath(Nodes(i),Gateways(j))
  If ShortestDistance < NodeDistance Then
     NodeDistance = ShortestDistance
     NearGateway = Gateways(j)
```

```
 End If
Next j
 If NodeDistance > 0 Then
   NodeGateDistance(i, 0) = NearGateway
   NodeGateDistance(i, 1) = NodeDistance
 End If
 NodesCost = NodesCost + NodeDistance
Next i

If NodeNo > 0 Then
   NodesAverge = NodesCost / NodeNo
End If
Dim SumDiff As Double
SumDiff = 0
For n As Integer = 1 To NodeNo
  SumDiff = SumDiff + System.Math.Pow(Abs((NodeGateDistance(n - 1, 1)-
NodesAverge)),2)
Next n
If NodeNo > 1 Then
   NodesVar = SumDiff / (NodeNo - 1)
End If
AerageHopCount = NodesAverge
Return NodesVar
End Function

Private Function GetVariantold(ByVal gateways() As Integer, ByVal Nodes() As
Integer) As Double

 Dim NodeNo As Integer
 Dim NodeGateDistance(Nodes.Length - 1, 1) As Double
 Dim NearGateway As Integer
 Dim NodeDistance As Integer, ShortestDistance As Integer
 NodeNo = Nodes.GetUpperBound(0)
```

```
 For i As Integer = 0 To NodeNo - 1
  ' Loop throuth all nodes
  NodeDistance = FrmTestGA.GetShortestPath(Nodes(i),gateways(0))
  NearGateway = gateways(0)
  For j As Integer = 1 To gateways.GetUpperBound(0)-1
    Application.DoEvents()
    ' Loop through all gateways
ShortestDistance=FrmTestGA.GetShortestPath(Nodes(i),gateways(j))
     If ShortestDistance < NodeDistance Then
        NodeDistance = ShortestDistance
        NearGateway = gateways(j)
     End If
  Next j
   If NodeDistance > 0 Then
     NodeGateDistance(i, 0) = NearGateway
     NodeGateDistance(i, 1) = NodeDistance
   End If
Next i

Dim NodePerG As Integer
Dim GateSum As Integer
Dim GateAvg As Double
Dim SumGateAvgs As Double
SumGateAvgs = 0
Dim avg As Double
Dim v As Double
Dim GatewayAverage(GatewaysNo - 1, 1) As Double
For g As Integer = 0 To GatewaysNo - 1
  NodePerG = 0
  GateSum = 0
  GateAvg = 0
  For node As Integer = 0 To NodeNo - 1
   If NodeGateDistance(node, 0) = g + 1 Then
```

```
    NodePerG = NodePerG + 1
    GateSum = GateSum + NodeGateDistance(node,1)
   End If
  Next node
  If NodePerG > 0 And GateSum > 0 Then
    GateAvg = GateSum / NodePerG
  Else
    GateAvg = 0
  End If
  GatewayAverage(g, 0) = g + 1
  GatewayAverage(g, 1) = GateAvg
  SumGateAvgs = SumGateAvgs + GateAvg
 Next g
 If GatewaysNo > 0 And SumGateAvgs > 0 Then
    avg = SumGateAvgs / GatewaysNo
 Else
    avg = 1000
 End If
 Dim SumDiff As Double
 SumDiff = 0
 For r As Integer = 0 To GatewaysNo - 1
   SumDiff = SumDiff + Math.Abs((GatewayAverage(r, 1) - avg)) *
Math.Abs((GatewayAverage(r, 1) - avg))
 Next r

 If GatewaysNo > 1 And SumDiff > 0 Then
   v = (SumDiff / GatewaysNo - 1)
 Else
   v = 200
 End If
 Return 0
End Function
End Class
```

4. GA class

```
Public Class Algorithm


 'Create a New Population
 Public Shared Function evolvePopulationNew(ByVal pop As Population) As
Population
   Dim newPopulation As New Population(pop.size(), False)
   Dim R As New Random
   ' Keep our best individual
   If elitism Then
     newPopulation.saveIndividual(0, pop.GetFittest)
   End If

   ' Crossover population
   Dim elitismOffset As Integer
   If elitism Then
     elitismOffset = 1
   Else
     elitismOffset = 0
   End If
   ' Loop over the population size and create new individuals with
   ' crossover

   Dim Inx1 As Integer
   Dim inx2 As Integer
   For i As Integer = elitismOffset To pop.size - 1
     Inx1 = R.Next(elitismOffset, pop.size() - 1)
     inx2 = R.Next(elitismOffset, pop.size() - 1)
     If Inx1 = inx2 Then
       Do Until Inx1 <> inx2
         inx2 = R.Next(elitismOffset, pop.size() - 1)
       Loop
     End If
```

```
    Dim newIndiv As Individual
    newIndiv = crossover(pop.getIndividual(Inx1), pop.getIndividual(inx2), 2)
    newPopulation.saveIndividual(i, newIndiv)
  Next i

    ' Mutate population
  Dim IndvToMutate As New Individual
  For j As Integer = elitismOffset To newPopulation.size() - 1
        mutate(newPopulation.getIndividual(j))
  Next j

  Return newPopulation
End Function

‘Evolve a population

Public Shared Function evolvePopulation(ByVal pop As Population) As Population
 Dim newPopulation As New Population(pop.size(), False)
 Dim RSel As New Random
 ' Keep our best individual
 If elitism Then
   newPopulation.saveIndividual(0, pop.GetFittest)
 End If

 ' Crossover population
 Dim elitismOffset As Integer
 If elitism Then
       elitismOffset = 1
  Else
       elitismOffset = 0
  End If
     ' Loop over the population size and create new individuals with
     ' crossover
```

```
For i As Integer = elitismOffset To pop.size() - 1
  Dim indiv1 As Individual = tournamentSelection(pop)
  Dim indiv2 As Individual = selectIndividual(pop)
  Dim newIndiv As Individual = crossover(indiv1, indiv2, 2)
  newPopulation.saveIndividual(i, newIndiv)
Next i

' Repair Genes
Dim NonePassed As Integer
NonePassed = 0
For k As Integer = elitismOffset To newPopulation.size() - 1
  NonePassed = ChenkGenes(newPopulation.getIndividual(k))
  If NonePassed <> 0 Then
    RepairGenes(newPopulation.getIndividual(k), NonePassed)
  End If
Next k
   "
' Mutate population

 For i As Integer = elitismOffset To newPopulation.size() - 1
  mutate(newPopulation.getIndividual(i))
 Next i
 Return newPopulation
End Function

  Private Shared Function tournamentSelection(ByVal pop As Population) As
  Individual
 ' Create a tournament population
  Dim tournament As New Population(tournamentSize, False)
  ' For each place in the tournament get a random individual
  For i As Integer = 0 To tournamentSize - 1
```

```
        Dim randomId As Integer = CInt(Math.Truncate((New
Random(1)).NextDouble() * pop.size()))
        tournament.saveIndividual(i, pop.getIndividual(randomId))
      Next i
      ' Get the fittest
      Dim fittest As Individual = tournament.GetFittest
      Return fittest
  End Function

Private Shared Function selectIndividual(ByVal pop As Population) As Individual
   Dim tournament(tournamentSize - 1) As Individual
   Dim tournamentFitness(tournamentSize - 1) As Double
   For i As Integer = 0 To tournamentSize - 1
    Dim index As Integer = CInt(Math.Truncate((New Random(1)).NextDouble() *
pop.size()))
    tournament(i) = pop.getIndividual(index)
    tournamentFitness(i) = pop.getIndividual(index).getFitness '
  Next i

  Dim bestIndividual As Individual = tournament(0)
  Dim bestFitness As Double = tournamentFitness(0)
  For i As Integer = 1 To tournamentSize - 1
    If tournamentFitness(i) < bestFitness Then
      bestIndividual = tournament(i)
      bestFitness = tournamentFitness(i)
    End If
  Next i

  Return bestIndividual
  End Function
End Class
```

5. General module (general settings, functions and procedures)

```
Module GeneralFunction
 Public Const uniformRate As Double = 0.5
 Public tournamentSize As Integer = 10
 Public Const mutationRate As Double = 0.015
 Public Const elitism As Boolean = True
 Public IndividualSiZe As Integer = 30
 Public GatewaysNo As Integer = 3
 Public CurrentIndividual As String
 Public IndividualParent1 As String
 Public IndividualParent2 As String
 Public TeriminationCondition As Boolean = False
 Public AerageHopCount As Double = 0
 Public R1 As New Random
 Public Function crossover(ByVal indiv1 As Individual, ByVal indiv2 As Individual,
ByVal CrossOverType As Integer) As Individual
 Dim Child As New Individual()
 Dim intCrossoverPoint As Integer
 Dim intCrossoverPoint2 As Integer
 Dim IntX As Integer
 ' Loop through genes
 Dim R As New Random
 If CrossOverType = 1 Then
   intCrossoverPoint = R.Next(uniformRate * indiv1.size, indiv1.size - 1)
   For i As Integer = 0 To indiv1.size() - 1
     If i < intCrossoverPoint Then
       Child.setGene(i, indiv1.getGene(i))
      Else
       Child.setGene(i, indiv2.getGene(i))
      End If
    Next i
    ElseIf CrossOverType = 2 Then
    For i = 0 To indiv2.size() - 1
       Child.setGene(i, indiv2.getGene(i))
```

```
      Next i
      Do
        intCrossoverPoint = R.Next(0, indiv1.size - 1)
        intCrossoverPoint2 = R.Next(0, indiv1.size - 1)
      Loop Until intCrossoverPoint <> intCrossoverPoint2
      If intCrossoverPoint > intCrossoverPoint2 Then
         IntX = intCrossoverPoint
         intCrossoverPoint = intCrossoverPoint2
         intCrossoverPoint2 = IntX
       End If

       For j As Integer = intCrossoverPoint To intCrossoverPoint2
          Child.setGene(j, indiv1.getGene(j))
       Next j
        End If

        Return Child
    End Function

    Public Sub mutate(ByVal indiv As Individual)
     ' Loop through genes
     Dim R As New Random
     Dim x1 As Integer
     Dim x2 As Integer = 0

      x1 = R.Next(IIf(indiv.size() Mod 2 = 0, indiv.size() / 2, (indiv.size()+1) / 2),
indiv.size() - 1)

s1:
     x2 = R.Next(0, (indiv.size() - 1))
     If indiv.getGene(x2) = indiv.getGene(x1) Then
       GoTo s1
     End If
```

```
    Dim gene As SByte
    gene = indiv.getGene(x1)
    indiv.setGene(x1, indiv.getGene(x2))
    indiv.setGene(x2, gene)

  End Sub

 Public Function ChenkGenes(Indiv As Individual) As Integer
        Dim Passed As Integer
        Dim NoOfOnes As Integer = 0
        For i As Integer = 0 To Indiv.size() - 1
          If Indiv.getGene(i) = "1" Then
            NoOfOnes = NoOfOnes + 1
          End If
        Next i
        Passed = NoOfOnes - GatewaysNo
        Return Passed
    End Function
Public Sub RepairGenes(ByVal indiv As Individual, ExtraGateway As Integer)
Dim i As Integer
If ExtraGateway > 0 Then
      Dim ArrayOnes(GatewaysNo + ExtraGateway - 1) As Integer
      For k = 1 To ExtraGateway
            ReDim ArrayOnes(GatewaysNo + ExtraGateway - 1)
            i = 0
            For j As Integer = 0 To indiv.size() - 1
                  If indiv.getGene(j) = "1" Then
                  ArrayOnes(i) = j
                  i = i + 1
            End If
            Next j

       i = R1.Next(0, ArrayOnes.Length - 1)
```

```
            i = ArrayOnes(i)
            indiv.setGene(i, "0")
          Next k
 Else
          Dim Ix As Integer
       For n As Integer = ExtraGateway To -1
s1:
          Ix = R1.Next(0, indiv.size() - 1)
            If indiv.getGene(Ix) = "0" Then
                indiv.setGene(Ix, "1")
              Else
                GoTo s1
              End If
          Next n
 End If

 Dim PassedGenes As Integer = ChenkGenes(indiv)
 If PassedGenes <> 0 Then
          RepairGenes(indiv, PassedGenes)
 End If
End Sub
Public Sub GetGateways(ByVal indiv As Individual)
Dim Gateways(GatewaysNo - 1) As Integer
Dim x As Integer
For i As Integer = 0 To indiv.size - 1
          If indiv.getGene(i).ToString = "1" Then
                  Gateways.SetValue(i + 1, x)
                  x = x + 1
          End If
Next i
FrmDrawGraph.ListBox1.DataSource = Gateways
End Sub
End Module
```

APPENDIX B

The following vb.net code defines Dijkstra's algorithm including Graph class and its components (nodes and edges)

```
Namespace Dijkstra

    ''' Represents a collection of Vertex objects and Edge objects.
    Public Class Graph

        '//fields
        Private needsCalculate As Boolean = True

        '//properties
        Private _verticies As VertexCollection
        ''' Gets the verticies associated with this graph.
        Public ReadOnly Property Verticies() As VertexCollection
            Get
                If Me._verticies Is Nothing Then
                    Me._verticies = New VertexCollection(Me)
                End If
                Return Me._verticies
            End Get
        End Property

        ''' Gets a value that indicates whether all the verticies in the graph are visited.
        Public ReadOnly Property Finished() As Boolean
            Get
                Return Me.Verticies.Finished
            End Get
        End Property

        Private _edges As EdgeCollection
        ''' Gets the edges associated with this graph's verticies.
        Public ReadOnly Property Edges() As EdgeCollection
            Get
                If Me._edges Is Nothing Then
                    Me._edges = New EdgeCollection(Me)
                End If
                Return Me._edges
            End Get
        End Property
        '//methods
        Private Sub Dijkstra(ByVal initialVertex As Vertex)
            If initialVertex Is Nothing Then
                Throw New ArgumentNullException("initialVertex")
            End If
```

```
        If Not Me.needsCalculate Then
            Return
        End If
        '//initialize starting values
        For Each vertex As Vertex In Me.Verticies
            vertex.SetDistance(Double.PositiveInfinity)
            vertex.SetVisited(False)
        Next
        initialVertex.SetDistance(0.0R)
        Try
            '//calculate shortest paths
            Me.Dijkstra(Me, initialVertex)
        Catch ex As Exception
            Throw New AlgorithmException("The graph, vertex, or edges are invalid.
Either they are not all connected, or there are edges missing.", ex)
        End Try
    End Sub

    Private Sub Dijkstra(ByVal graph As Graph, ByVal current As Vertex)
        '//loop each neighboring vertex
        For Each neighbor In current.Neighbors
            If Not neighbor.Visited Then
                '//vertex has not been visited yet
                Dim edge = graph.Edges(current, neighbor)
                '//get the distance between the verticies
                Dim distance = (current.Distance + edge.Distance)
            '//check if the distance is smaller than it's previous distance
                If distance < neighbor.Distance Then
                    neighbor.SetDistance(distance)
         '//sets the vertex that you would follow to get to this neighboring one
                    neighbor.PreviousVertex = current
                End If
            End If
        Next
        '//mark Vertex visited
        current.SetVisited(True)
        If Not graph.Finished Then
            '//graph has unvisited verticies
            Me.Dijkstra(graph, graph.FindShortestVertex())
        End If
        Me.needsCalculate = False
    End Sub

    Private Function FindShortestVertex() As Vertex
        Dim result As Vertex = Nothing
        Dim min = Double.PositiveInfinity
        '//loop all unvisited verticies to find the vertex that has the smallest distance
        For Each vertex As Vertex In Me.Verticies
            If Not vertex.Visited Then
```

```
                If vertex.Distance < min Then
                    '//set the current smallest vertex
                    min = vertex.Distance
                    result = vertex
                End If
            End If
        Next
        '//after all distances are evaluated return result
        Return result
    End Function
    ''' Adds a Vertex to the end of the collection.
    ''' <param name="key">The key used as an identifier for the Vertex. Can be
null.</param>
    Public Function AddVertex(ByVal key As String) As Vertex
        Dim vertex = New Vertex(Me, key)
        Me.Verticies.Add(vertex)
        Return vertex
    End Function

    ''' Removes the specified Vertex from the collection.
    ''' <param name="vertex">The Vertex to remove.</param>
    Public Sub RemoveVertex(ByVal vertex As Vertex)
        Me.Verticies.Remove(vertex)
    End Sub

    ''' Adds an Edge to the end of the collection.
    ''' <param name="first">The first Vertex for this Edge.</param>
    ''' <param name="second">The second Vertex for this Edge.</param>
    ''' <param name="distance">The distance between the two vertices.</param>
    Public Function AddEdge(ByVal first As Vertex, ByVal second As Vertex,
ByVal distance As Double) As Edge
        Dim edge = New Edge(Me, first, second, distance)
        Me.Edges.Add(edge)
        Return edge
    End Function

    ''' Adds an Edge to the end of the collection.
    ''' </summary>
    ''' <param name="firstKey">The key of the first Vertex for this Edge.</param>
    ''' <param name="secondKey">The key of the second Vertex for this
Edge.</param>
    ''' <param name="distance">The distance between the two vertices.</param>
    Public Function AddEdge(ByVal firstKey As String, ByVal secondKey As
String, ByVal distance As Double) As Edge
        Dim edge = New Edge(Me, Me.Verticies(firstKey), Me.Verticies(secondKey),
distance)
        Me.Edges.Add(edge)
        Return edge
    End Function
```

```
        ''' Removes the specified Edge from the collection.
        ''' <param name="edge">The Edge to remove.</param>
        Public Sub RemoveEdge(ByVal edge As Edge)
            Me.Edges.Remove(edge)
        End Sub

        ''' Calculates the Graph using the specified Vertex as the starting point.
        ''' <param name="initialVertex">The starting vertex to calculate from.</param>
        Public Sub Calculate(ByVal initialVertex As Vertex)
            Me.Dijkstra(initialVertex)
        End Sub

        ''' Resets the Graph by clearing all verticies and edges.
        Public Sub Reset()
            '//clear all contents
            Me.Verticies.Clear()
            Me.Edges.Clear()
            '//notify graph that it will need to recalculate
            Me.NotifyRecalculate()
        End Sub

        ''' Serializes this Graph to a file.
        ''' <param name="fileName">A string that contains the name of the file.</param>
        Public Sub Save(ByVal fileName As String)
            '//create the document
            Dim doc = <?xml version="1.0" encoding="utf-8"?>
                      <root vertexCount=<%= Me.Verticies.Count %> edgeCount=<%=
Me.Edges.Count %>></root>
            '//loop all verticies
            For Each vertex As Vertex In Me.Verticies
                doc.Root.Add(<vertex>
                                 <key><%= vertex.Key %></key>
                             </vertex>)
            Next
            '//loop all edges
            For Each edge As Edge In Me.Edges
                doc.Root.Add(<edge>
                                 <firstKey><%= edge.First.Key %></firstKey>
                                 <secondKey><%= edge.Second.Key %></secondKey>
                                 <distance><%= edge.Distance %></distance>
                             </edge>)
            Next
            '//saves the file
            doc.Save(fileName)
        End Sub

        ''' Creates a new Graph from a file.
        ''' <param name="fileName"></param>
```

```
        ''' <remarks></remarks>
        Public Shared Function Load(ByVal fileName As String) As Graph
            '//try to read the Xml file
            Try
                '//load the Xml and create the new graph
                Dim doc = XDocument.Load(fileName)
                Dim graph = New Graph()

                '//get all verticies
                For Each node In doc.<root>.<vertex>
                    graph.AddVertex(node.<key>.Value())
                Next

                '//get all edges
                For Each node In doc.<root>.<edge>
                    graph.AddEdge(node.<firstKey>.Value(), node.<secondKey>.Value(), _
                            Convert.ToDouble(node.<distance>.Value()))
                Next
                Return graph
            Catch ex As Exception
                Return Nothing
            End Try
        End Function

        ''' Gets the shortest distance between two verticies in the graph.
        ''' <param name="initialVertex">The starting vertex to calculate from.</param>
        ''' <param name="endingVertex">The ending Vertex to calculate to.</param>
        Public Function GetDistance(ByVal initialVertex As Vertex, ByVal
endingVertex As Vertex) As Double
            Me.Dijkstra(initialVertex)
            Return endingVertex.Distance
        End Function

        ''' Gets a string representation of the shortest path's order of verticies to follow.
        ''' <param name="initialVertex">The starting vertex to calculate from.</param>
        ''' <param name="endingVertex">The ending Vertex to calculate to.</param>
        Public Function GetPath(ByVal initialVertex As Vertex, ByVal endingVertex
As Vertex) As String
            Return Me.GetPath(initialVertex, endingVertex, False)
        End Function

        ''' Gets a string representation of the shortest path's order of verticies to follow.
        ''' <param name="initialVertex">The starting vertex to calculate from.</param>
        ''' <param name="endingVertex">The ending Vertex to calculate to.</param>
        ''' <param name="reverse">Indicates whether the direction of the string should
start at the end.</param>
        Public Function GetPath(ByVal initialVertex As Vertex, ByVal endingVertex
As Vertex, ByVal reverse As Boolean) As String
            Me.Dijkstra(initialVertex)
```

```
            Dim verticies = Me.GetVerticies(initialVertex, endingVertex, reverse)
            Dim builder = New Text.StringBuilder()
            For vertex = 0 To verticies.Count - 2
                builder.AppendFormat("{0} -> ", verticies(vertex).Key)
            Next
            builder.Append(verticies(verticies.Count - 1).Key)
            Return builder.ToString()
        End Function

        ''' Gets a System.TimeSpan object that represents the time it takes to calculate the
current instance.
        ''' <param name="initialVertex">The Vertex to start the calculation
from.</param>
        Public Function GetElapsed(ByVal initialVertex As Vertex) As TimeSpan
            '//notify graph that it will need to recalculate
            Me.NotifyRecalculate()
            '//create the watch object
            Dim watch = Stopwatch.StartNew()
            '//run the algorithm
            Me.Dijkstra(initialVertex)
            '//return results
            watch.Stop()
            Return watch.Elapsed
        End Function

        ''' Gets an array of the verticies that are used for the shortest path.
        ''' <param name="initialVertex">The starting vertex to calculate from.</param>
        ''' <param name="endingVertex">The ending Vertex to calculate to.</param>
        Public Function GetVerticies(ByVal initialVertex As Vertex, ByVal
endingVertex As Vertex) As Vertex()
            Return Me.GetVerticies(initialVertex, endingVertex, False)
        End Function

        ''' Gets an array of the verticies that are used for the shortest path.
        ''' <param name="initialVertex">The starting vertex to calculate from.</param>
        ''' <param name="endingVertex">The ending Vertex to calculate to.</param>
        ''' <param name="reverse">Indicates whether the order of the verticies should be
from end to start.</param>
        Public Function GetVerticies(ByVal initialVertex As Vertex, ByVal
endingVertex As Vertex, ByVal reverse As Boolean) As Vertex()
            Me.Dijkstra(initialVertex)
            Dim current = endingVertex
            Dim verticies = New List(Of Vertex)()
            Do Until current Is Nothing
                verticies.Add(current)
                current = current.PreviousVertex
            Loop
            If Not reverse Then
                verticies.Reverse()
```

```
            End If
            Return verticies.ToArray()
        End Function

        Friend Sub NotifyRecalculate()
            Me.needsCalculate = True
        End Sub

    End Class
End Namespace
```

APPENDIX C

The following vb.net code defines the main SA-Based algorithm class including the settings of its parameters besides all the classes that are required to build the solution such as solution, fitness calculation, Dijkstra's algorithm (including the graph definitions) classes. Moreover, this code defines the main function that initializes and runs the algorithm.

```
Imports Microsoft.VisualBasic
Imports System
Imports System.Collections.Generic
Imports System.Random
Imports System.Math
Public Class MySA
 Private Const INITIAL_TEMPERATURE As Double = 100
 Private Const FINAL_TEMPERATURE As Double = 0.01
 Private Const ALPHA As Double = 0.99
 Private Const ITERATIONS_AT_TEMPERATURE As Integer = 1
 Private Shared currentSolution As New Solution()
 Private Shared workingSolution As New Solution()
 Private Shared bestSolution As New Solution()
 Private Const TARGET As Double = 0.02 ' correct answer.
 Public Shared BestSol As String
 Public Shared InitialSol As String
 Public Shared InitialSolutionEnergy As Double
 Private Shared Sub simulatedAnnealingAlgorithm()
 Dim solution As Boolean = False
 Dim useNew As Boolean = False
 Dim accepted As Integer = 0
 Dim temperature As Double = INITIAL_TEMPERATURE
 currentSolution = New Solution()
 workingSolution = New Solution()
```

```
 bestSolution = New Solution()
 initializeSolution()
 currentSolution.computeEnergy()
 'Keep initial Solution Engergy
 InitialSolutionEnergy =currentSolution.solutionEnergy()
Console.WriteLine("Fitness:"& currentSolution.solutionEnergy())
bestSolution.solutionEnergy(currentSolution.solutionEnergy())
workingSolution.Equals(currentSolution)
 Do While temperature > FINAL_TEMPERATURE
  accepted = 0
  For i As Integer = 0 To ITERATIONS_AT_TEMPERATURE - 1
    useNew = False
    workingSolution.randomChange()
    workingSolution.computeEnergy()
Console.WriteLine("Fitness:"& workingSolution.solutionEnergy())
    If workingSolution.solutionEnergy() <= currentSolution.solutionEnergy() Then
      useNew = True
    Else
     Dim test As Double = (New Random()).NextDouble()
        'Get random value between 0.0 and 1.0
     Dim delta As Double = workingSolution.solutionEnergy() -
currentSolution.solutionEnergy()
     Dim calc As Double = Math.Exp(-delta / temperature)
     If calc > test Then
      accepted += 1
      useNew = True
     End If
    End If
  If useNew Then
   useNew = False
   currentSolution.Equals(workingSolution)
   If currentSolution.solutionEnergy() < bestSolution.solutionEnergy() Then
    bestSolution.Equals(currentSolution)
```

```
      solution = True
    End If
  Else
    workingSolution.Equals(currentSolution)
  End If
  Console.WriteLine("Current Solution Energy:" & currentSolution.solutionEnergy())
  Console.WriteLine("Working Solution Energy: " &
workingSolution.solutionEnergy())
  Console.WriteLine("Best Solution Energy: " &  bestSolution.solutionEnergy())
  Next i
  temperature *= ALPHA
  Console.WriteLine("Temperature: " & temperature)
Loop
If solution Then
   BestSol = ""
   For j As Integer = 0 To IndividualSiZe - 1
      Console.Write(bestSolution.data(j) & ", ")
      BestSol = BestSol.ToString + bestSolution.data(j).ToString
   Next j
   Console.Write(ControlChars.Lf)
   If bestSolution.solutionEnergy() <= TARGET Then
     Console.WriteLine("Best solution is: Correct")
   Else
     Console.WriteLine("Best solution is: Not Correct")
   End If
End If
Return
End Sub
Private Shared Sub initializeMap()
Console.WriteLine("Enter No of Mesh Routers Per Square Side:")
'NodePerSide = Console.ReadLine()
  NodePerSide = 5
  IndividualSiZe = NodePerSide * NodePerSide
```

```
 Console.WriteLine("The individual Size will be:")
 Console.WriteLine(IndividualSiZe.ToString)
 Console.WriteLine("Enter No of Gateways:")
 'GatewaysNo = Console.ReadLine()
  GatewaysNo = 4
 EdgNo = 2 * NodePerSide * (NodePerSide - 1)
 ReDim EdgeMatrix(EdgNo - 1, 1)
 Filledges(NodePerSide)
End Sub
Private Shared Sub Filledges(SideNode As Integer)
Dim x1 As Integer, x2 As Integer, indx As Integer
For i As Integer = 1 To SideNode
  For j As Integer = 1 To SideNode - 1
    x1 = (i - 1) * SideNode + j
    x2 = (i - 1) * SideNode + j + 1
    EdgeMatrix(indx, 0) = x1
    EdgeMatrix(indx, 1) = x2
    indx = indx + 1
  Next j
Next i
For j As Integer = 1 To SideNode
  If i < SideNode Then
    x1 = (i - 1) * SideNode + j
    x2 = (i * SideNode + j)
    EdgeMatrix(indx, 0) = x1
    EdgeMatrix(indx, 1) = x2
    indx = indx + 1
  End If
Next j
End Sub
Private Shared Sub initializeSolution()  ' Done
 'Initial setup of the solution.
 For i As Integer = 0 To IndividualSiZe - 1
```

```
    currentSolution.data(i, 0)
  Next i
  Dim inx As Integer
  Dim R As New Random
  For i = 1 To GatewaysNo
s1:
    inx = CInt(R.Next(0, IndividualSiZe - 1))
    If currentSolution.data(inx) = 1 Then
      GoTo s1
    End If
    currentSolution.data(inx, 1)
  Next i
  ' Randomly perturb the solution.
  For i As Integer = 0 To IndividualSiZe - 1
    currentSolution.randomChange()
  Next i
  'Keep initial Solution for convergence
  InitialSol = ""
  For i As Integer = 0 To IndividualSiZe - 1
    InitialSol = InitialSol.ToString + currentSolution.data(i).ToString
  Next i
  Return
End Sub
Private Shared Function getExclusiveRandomNumber(ByVal high As Integer,
ByVal except As Integer) As Integer
  Dim done As Boolean = False
  Dim getRand As Integer = 0
  Do While Not done
    getRand = (New Random()).Next(high)
    If getRand <> except Then
      done = True
    End If
  Loop
```

```
  Return getRand
End Function
Private Class Solution
  Private mSolutionEnergy As Double = 0.0
  Private mData() As Integer = Nothing
  Public Sub New()
    mData = New Integer(IndividualSiZe - 1) {}
  End Sub
  Public Sub New(ByVal that As Solution)
    mData = New Integer(IndividualSiZe - 1) {}
    For i As Integer = 0 To IndividualSiZe - 1
      Me.mData(i) = that.data(i)
    Next i
    Me.mSolutionEnergy = that.mSolutionEnergy
  End Sub
Public Overridable Sub Equals(ByVal that As Solution)
  For i As Integer = 0 To IndividualSiZe - 1
     Me.mData(i) = that.data(i)
  Next i
  Me.mSolutionEnergy = that.mSolutionEnergy
  Return
End Sub
Public Overridable Sub data(ByVal index As Integer, ByVal value As Integer)
  Me.mData(index) = value
  Return
End Sub
Public Overridable Function data(ByVal index As Integer) As Integer
  Return Me.mData(index)
End Function
Public Overridable Sub solutionEnergy(ByVal value As Double)
   Me.mSolutionEnergy = value
   Return
End Sub
```

```
Public Overridable Function solutionEnergy() As Double
  Return Me.mSolutionEnergy
End Function
Public Overridable Sub randomChange() ' done
  Dim temp As Integer = 0
  Dim x As Integer = 0
  Dim y As Integer = 0
  'Get two different random numbers.
 x = (New Random()).Next(IndividualSiZe - 1)
 y = getExclusiveRandomNumber(IndividualSiZe - 1, x)
 temp = Me.mData(x)
 Me.mData(x) = Me.mData(y)
 Me.mData(y) = temp
 Return
End Sub
Public Overridable Sub computeEnergy()
  Me.mSolutionEnergy = 0.0
  'Find the round-trip distance.
  Me.mSolutionEnergy = getFitness()
  Return
End Sub
'Calculate Eneragy
Public Function getFitness() As Double
  Dim Gateways(GatewaysNo - 1) As Integer
  Dim Nodes(IndividualSiZe - GatewaysNo - 1) As Integer
  Dim x1 As Integer
  Dim x2 As Integer
  x1 = 0
  x2 = 0
  For i = 0 To IndividualSiZe - 1
    If Me.data(i) = 1 Then
      Gateways.SetValue(i + 1, x1)
      x1 = x1 + 1
```

```
    ElseIf Me.data(i) = 0 Then
      Nodes.SetValue(i + 1, x2)
      x2 = x2 + 1
    End If
  Next i
 'Calculate the standard deviation
  Dim v As Double
  v = GetVariant(Gateways, Nodes)
  Return v
End Function
Private Function GetVariant(ByVal Gateways() As Integer, ByVal Nodes() As
Integer) As Double
 Dim NodeNo As Integer
Dim NodeGateDistance(IndividualSiZe - GatewaysNo - 1, 1) As  Double
 Dim NearGateway As Integer
 Dim NodeDistance As Integer, ShortestDistance As Integer
 Dim NodesAverge As Double, NodesVar As Double
 Dim NodesCost As Integer
 NodeNo = IndividualSiZe - GatewaysNo
 NodesCost = 0
 Dim g As Graph
 For i As Integer = 0 To NodeNo – 1  ' Loop throuth all nodes
   g = New Graph
   ShortestDistance = 0
   NodeDistance = g.GetShortestPath(Nodes(i), Gateways(0))
   NearGateway = Gateways(0)
   For j As Integer = 0 To Gateways.GetUpperBound(0) - 1
    'Loop through all gateways
     ShortestDistance = g.GetShortestPath(Nodes(i), Gateways(j))
     If ShortestDistance < NodeDistance Then
       NodeDistance = ShortestDistance
       NearGateway = Gateways(j)
     End If
```

```
    Next j
    If NodeDistance > 0 Then
      NodeGateDistance(i, 0) = NearGateway
      NodeGateDistance(i, 1) = NodeDistance
    End If
    NodesCost = NodesCost + NodeDistance
  Next i
  If NodeNo > 0 Then
    NodesAverge = NodesCost / NodeNo
  End If
   Dim SumDiff As Double
   SumDiff = 0
   For n As Integer = 0 To NodeNo - 1
    SumDiff = SumDiff + System.Math.Pow(Abs((NodeGateDistance(n, 1) -
NodesAverge)), 2)
   Next n
   If NodeNo > 1 Then
     NodesVar = SumDiff / (NodeNo)
   End If
   AerageHopCount = NodesAverge
   'Calc Variance between Gateways
   Dim NodesPerG As Integer, SumG As Integer
   Dim AvgG(GatewaysNo) As Double, SumAges As Double, AvgAll As Double,
SumDiffAll As Double, Vall As Double
   SumDiffAll = 0
   Vall = 0
   NodesPerG = 0
   SumAges = 0
  For gcounter As Integer = 0 To Gateways.GetUpperBound(0) - 1
    NodesPerG = 0
    For n As Integer = 0 To NodeGateDistance.GetUpperBound(0)-1
      If NodeGateDistance(n, 0) = Gateways(gcounter) Then
        NodesPerG = NodesPerG + 1
```

```
            SumG = SumG + NodeGateDistance(n, 1)
          End If
        Next n
        If NodesPerG > 0 Then
          AvgG(gcounter) = SumG / NodesPerG
          SumAges = SumAges + (SumG / NodesPerG)
        End If
      Next gcounter
      AvgAll = SumAges / GatewaysNo
      For Gcounter As Integer = 0 To GatewaysNo - 1
        SumDiffAll = SumDiffAll + Pow((AvgG(Gcounter) - AvgAll), 2)
      Next Gcounter
      Vall = SumDiffAll / (GatewaysNo)
      Return NodesVar
    End Function
End Class ' Solution class
Public Shared Sub Main(ByVal args() As String)
  For i As Integer = 1 To 10
    initializeMap()
    Static start_time As DateTime
    Static stop_time As DateTime
    Dim elapsed_time As TimeSpan
    start_time = Now
    simulatedAnnealingAlgorithm()
    stop_time = Now
    elapsed_time = stop_time.Subtract(start_time)
    Console.WriteLine("Total Execution Time is (In seconds):" &
elapsed_time.TotalSeconds.ToString("0.000000"))
    Console.WriteLine("Total Execution Time is (In Minutes):" &
elapsed_time.TotalMinutes.ToString("0.000000"))
    Console.WriteLine("Total Execution Time is (In Hours):" &
elapsed_time.TotalHours.ToString("0.000000"))
```

```
    SendDataToExcel(IndividualSiZe, GatewaysNo, INITIAL_TEMPERATURE,
ALPHA, FINAL_TEMPERATURE _
,ITERATIONS_AT_TEMPERATURE, bestSolution.solutionEnergy(),
elapsed_time.TotalSeconds.ToString("0.000000") _
,elapsed_time.TotalMinutes.ToString("0.000000"),
elapsed_time.TotalHours.ToString("0.000000"), BestSol.ToString,
InitialSol.ToString, InitialSolutionEnergy)
Next i
Return
End Sub
End Class
```